Preliminary CLAS12 Simulation Analysis and Optimization

Dergachev, K., Gilfoyle, G.P.
University of Richmond, VA
April 11, 2008

Abstract: Jefferson Laboratory (JLab) is undergoing an upgrade to increase beam energy from 6 GeV to 12 GeV to more clearly understand hadronic structure and quantum chromodynamics. The existing detector in Hall B is being upgraded to take advantage of the new physics opportunities. The new detector, CLAS 12, is in the design phase and we are developing a new simulation package, Sim12. The new code uses at least eight specially configured software packages to run, requiring extended compilation and configuration times. This installation procedure was optimized and documented resulting in a far shorter installation time. Sim12 is resource intensive, hence optimization was performed to decrease initialization times, accommodate the large amount of vertices necessary to represent the detector, and to decrease processing times of simulations. To accommodate event generators, which are necessary for experimentally relevant simulations, utilities to input event files were created. Finally, preliminary analysis was performed on specific data.

Introduction:

Jefferson Laboratory and CEBAF: The facility at Jefferson Laboratory (JLab) stands at the frontier of precision nuclear physics. Unlike other existing higher-energy facilities such as Fermilab, JLab serves the primary purpose of determining and understanding nuclear structure as precisely as possible. The present specific focus of JLab is the interaction between quarks and the gluons that bind them together. Quantum Chromodynamics (QCD) is the current theory used to explain these interactions, but to confirm the validity of QCD, a higher energy beam of 12 GeV from the original 6 GeV is necessary. In doing so, the probing wavelength will be greatly decreased, increasing the resolution of particle detection. The Continuous Electron Beam Accelerating Facility (CEBAF) is the heart of the JLab complex, and it is what provides an extremely precise electron beam to the three current detector halls. CEBAF is a 7/8 mile tunnel, 25 feet underground, capable of accelerating electrons to between 2 and 6 GeV[3]. Thermoionic emission creates electrons, which are then accelerated along super cooled linear portions of the accelerator path. The electrons can travel up to five times around the accelerator to the point of gaining the full 6 GeV. The 12 GeV upgrade for CEBAF (Fig. 1) will mainly encompass the addition of ten cryomodules used for the acceleration, a drastic increase in refrigeration capacity, and upgraded magnets in the arc areas to handle the higher energy.

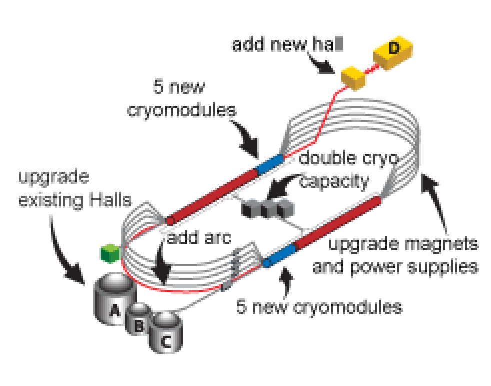


Figure 1: CEBAF-Specific Upgrades[5]
In this image, most of the significant new components of CEBAF are visible. Two sets of 5 cryomodules are at the end of the LINAC sections. The magnets are shown to be physically in the same area, but will be more powerful. Additionally, the refrigeration units are also shown in their original location but will be upgraded.

Physics Motivation: Current detection means used at JLab have reached their limit in studying internal nucleon dynamics. Hence, in order to farther probe the accuracies of the currently best working theory, QCD, higher resolution imaging is necessary, which basically requires an electron beam of higher energy. This upgrade should bring the ability to better study a large repertoire of topics including general parton distributions, deeply virtual Compton scattering, and deeply virtual Meson production[1]. However, to successfully study those issues, the detectors need to accept the higher beam energy, so each of the detectors will require reconstruction.

CLAS: The CEBAF Large Acceptance Spectrometer (CLAS) is currently used to detect electrons, protons, neutrons, pions, photons, and other subatomic particles from Hall B at JLab. Aside from being a 50 million dollar detector weighing 45 tons, CLAS is also unique in being a 4π detector allowing particle detection in any direction. There are six major sections that compose CLAS consisting of three drift chamber layers, Cerenkov counters, scintillators, and calorimeters[4] (Fig. 2). The information from these components is used to determine particle momentum and particle identification.

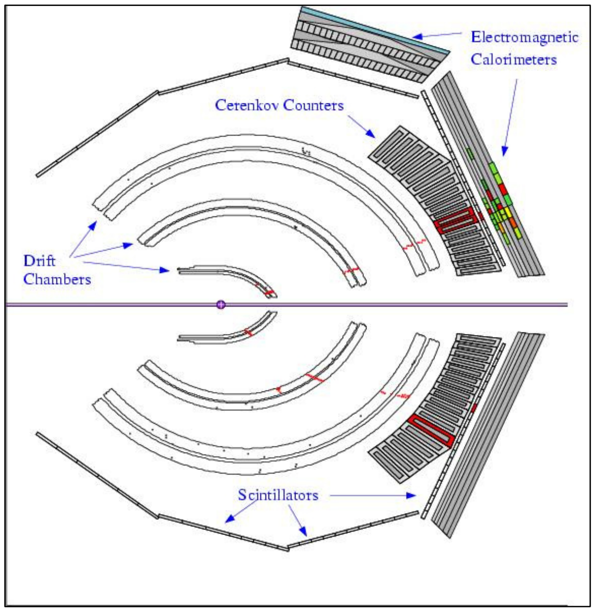


Figure 2: CLAS[4]
An event path is shown here in red. It begins in the center of the detector, moving through the drift chambers, followed by the Cerenkov counters, then going through the scintillators, and finally through the calorimeters.

The drift chambers compose the first three layers of what any post-reaction particles encounter. Between the chambers are massive toroidal magnets of known strength that alter the path of any charged particle. These chambers are mainly made up of 32,000 wires of 30 µm diameter. When a particle encounters one of these wires, an electric current is made, travels down the wire, and finally reaches a detection sensor which transmits the information to a computer. Given the delay in the transmission of the signal, which is determined by the time of flight via the scinitillators, the computer can accurately calculate where exactly on the wire a hit took place. This information is then used from each of the three chambers to reconstruct the specific trajectory that any given particle takes.

After the drift chambers, Cerenkov counters are present, though only for forward-angle particles. The counters basically detect Cerenkov light created when particles at the speed of light in a vacuum enter the gas filled chambers, slowing down to the speed of light in the new medium, emitting light in the process. Photomultiplier tubes within the Cerenkov layers actually detect the miniscule amount of light that is produced. Cerenkov light is quite useful for particle identification, especially for the differentiation between pions and electrons. Though the two particles have very similar energies, pions are significantly more massive, and hence travel slower through space. Such slow speeds are far below the speed of light, which is necessary for Cerenkov light to be exhibited. Hence, questionable identifications for particles can usually be dismissed from being electrons by looking at whether a Cerenkov hit was seen or not.

Scintillators make up the fourth layer of detection and serve to determine the time of flight for any given particle. They are basically made of plastic that shines when hit by a particle, and that light is captured by photomultiplier tubes sensitive down to one photon. Knowing the specific time of flight for a particle is important as that information allows velocity calculations from the known path length. Additionally, this information is used to aid in pinpointing where a particle hit a wire in a drift chamber. In conjunction with momentum calculations made from the drift chambers and calorimeters, this can be used to determine the mass of the particle, and ideally to identify it precisely.

The final layer in the detector is composed of alternating layers of scintillator sheets and lead, forming a calorimeter. Any given particle will travel through a certain number of lead layers before it stops. Based on how many layers the particle travels through, energy values can be calculated. This information, combined with trajectory data as well as time of flight information, can all be brought together to fully characterize the particle via the identity of the particle, its energy, momentum, and velocity.

Currently, CLAS is an extremely sensitive detector that has proved useful throughout its lifetime. However, once the 12 GeV upgrade is complete to CEBAF, a significantly more powerful beam will enter the detector at higher luminosity, and allowing for higher resolution detection. This poses several problems to CLAS in terms of kinematics, data acquisition, and the need for even higher accuracy in particle detection. These problems were addressed by designing and now building CLAS 12.

CLAS 12**:** CLAS requires reconstruction in order to be compatible with the more powerful beam, and will soon become known as CLAS 12 (Fig. 3). Essentially, each piece of the detector was redesigned to make the detector more sensitive and accurate to support higher resolutions, and to have the ability to continuously record the huge volumes of data that will be produced[3].

There are many key improvements from reconstruction that combine to make CLAS 12 a more robust detector. The first of these is the ability to support a 10-fold higher luminosity for hydrogen targets. Luminosity is basically the frequency of events, which along with the new beam energy, will be much higher. Additionally, the kinematics of having a higher energy beam result in significantly more events occurring closer to the beam line, so the area of events will be even more concentrated. CLAS would not have been able to keep up data logging at such luminosity for two reasons, the first of which is the lack of enough computing power to process the large amount of data. This is of course easily remedied by building a larger data pipe, and adding more processing power. More importantly, CLAS would also not have enough time to return from any kind of signal state to a ready state. Essentially, this stems from the fact that the currents present in a wire need to be discrete, and hence continuous bombardment would simply produce a single extended signal, especially in the drift chambers and scintillators. To alleviate this problem, the detector will be segmented even farther, so that the probability of two events occurring at the same time on one wire or detection element would be greatly decreased.

In addition to allowing for higher luminosity, there will be improved detection for low angle particles with characteristic high momentum, aided by the solenoidal magnet in the central detector. The central detector will also aid in the filtering of Moller electrons, which are formed when high energy particles knock off low energy electrons from any atoms in the detector. The Moller electrons are usually not associated with any given event, and hence are a source of vast noise. High angle detection will also be improved to aid in identifying and characterizing recoil baryons.

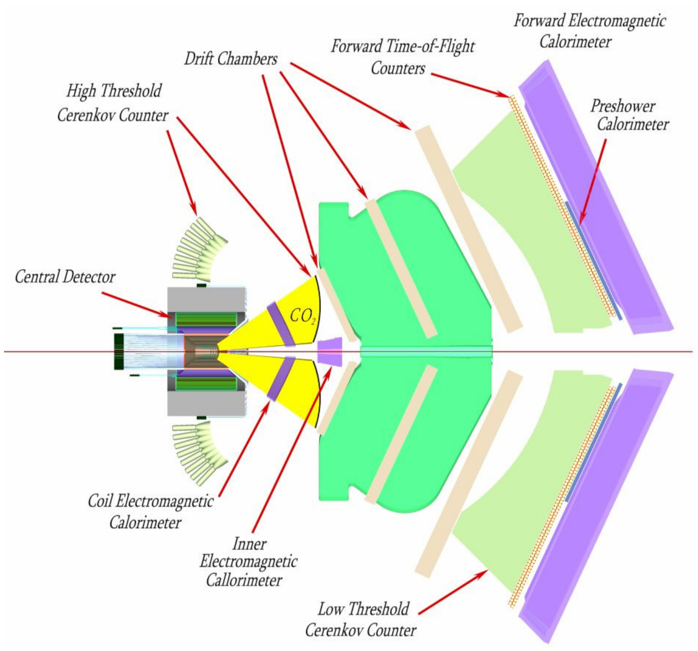

Forward Electromagnetic Calorimeter
Drift Chambers
Forward Time-of-Flight Counters
High Threshold Cerenkov Counter
Preshower Calorimeter
Central Detector
CO_2
Coil Electromagnetic Calorimeter
Inner Electromagnetic Callorimeter
Low Threshold Cerenkov Counter

Figure 3: CLAS 12[3]
All of the major new CLAS 12 components are shown here. The additional calorimeter, Cerenkov counters, and central detector are labeled.

Sim12: Once CLAS 12 becomes operational alongside the more powerful 12 GeV beam, new physics experiments will be performed and new exciting results will be obtained. However, of utmost importance is the accuracy of those results, which is why simulation is essential to the CLAS 12 upgrade. By performing simulations, results that are obtained can be confirmed to be true physics effects, as opposed to random occurrences due to detector imperfections or similar issues. Additionally, a rough estimate of the results of certain experiments can be obtained by preemptively running the experiment through the simulation saving money, and most importantly, beam time. Previously, the package known as GSIM was used in order to simulate CLAS. However, for CLAS 12, a completely new simulation front end has been designed known as Sim12. Sim12 is a robust solution allowing for not only mathematical simulations and output to data files, but also visual representations of reaction events, detector hits, and particle charge[1]

(Fig. 4). In contrast to GSIM, Sim12 provides a much more user friendly interface due to the improved visualizations, but it is also a more efficient system as it utilizes more object oriented features of GEANT4. Additionally, Sim12 utilizes XML to import any kind of detector data, meaning it is much simpler to modify and kind of detector features. A majority of the backend utilities that power Sim12 are in fact inherited from GSIM. However, Sim12 certainly brings new features to the table, of which one of the most interesting and useful is the ability to incorporate custom built plugins.

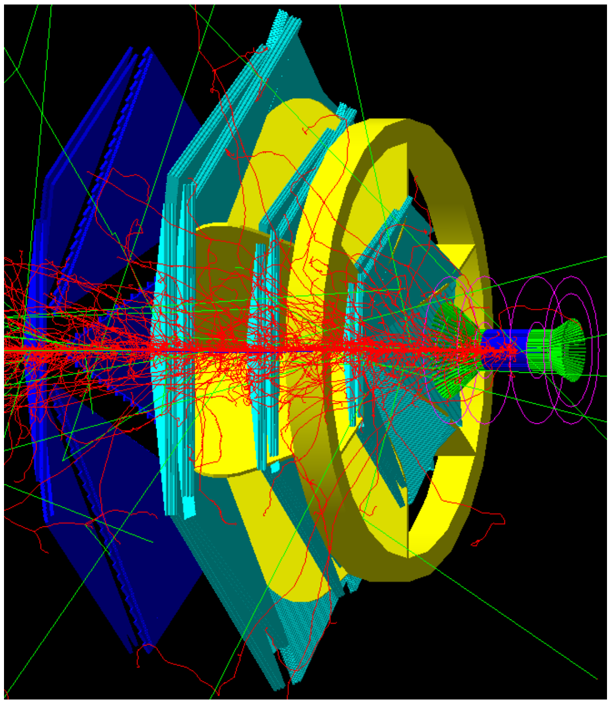


Figure 4: Sim12 Visual Output
This image consists of an experiment performed in Sim12 with 1000 events of random nature. Red lines show negatively charged particle paths. Green shows neutral particle paths. Blue shows positive paths. It can be seen that while the majority of particles are not significantly deflected, some certainly do follow drastically different trajectories from the original, implying interesting collisions have occurred.

***Installation Optimization*:** As mentioned, Sim12 is the tip of a large iceberg of software packages, all of which are necessary for proper function. This brings a large hurdle to the efficient distribution of the software since all users must run a similar computing environment and meet all of the software prerequisites for successful simulation. Additionally, since the software is in the development stage, the user must be able to properly integrate the software between all of the packages. Finally, the software can take a significant amount of time to compile, so optimizations in the form of some sort of binary package distribution are necessary.

Package Collection: A total of eight major software utilities are accessed by Sim12, each of which is distributed and supported by a different group. This software includes CLHEP, Geant4, Xerces-C, Motif, Dawn, EVIO, HDDS, and JGeant4. Additionally, it is possible to extract ROOT files from Sim12, so that software could potentially be required. Though most of these packages are independently distributed, there are also very strict interdependencies on other packages making installation more difficult. Essentially there is a very specific order that the packages must be installed in, as otherwise dependencies would not be met, and the software would not run correctly or fail to install.

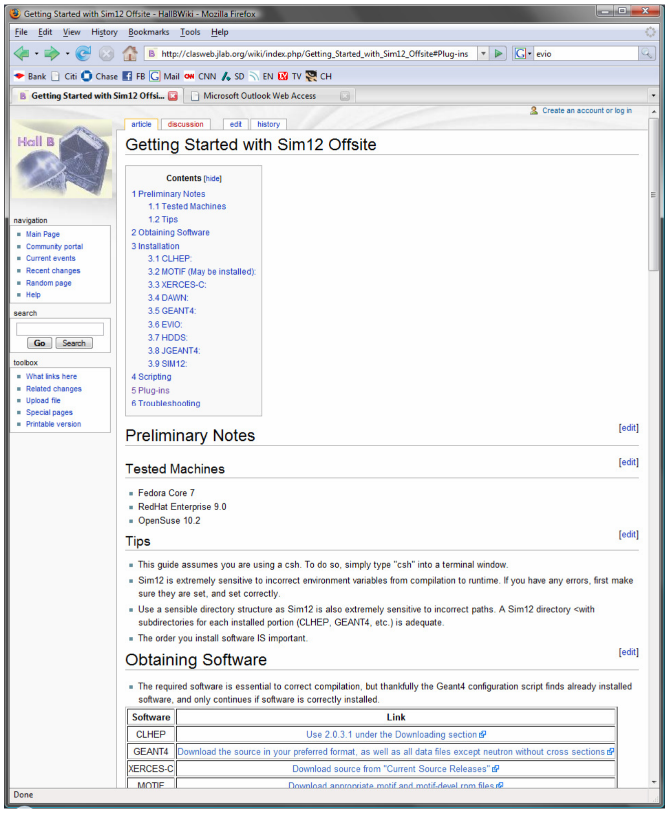
CLHEP (Class Library for High Energy Physics)[6] is a library utilized within high-energy physics simulation and analysis. It is written for and in C++, and is developed by CERN (European Organization for Nuclear Research). However, the package is now in a maintenance stage in which bug fixes are addressed, but new functionality is not being added. CLHEP serves as the ultimate backbone for the majority of the other packages. Hence it is the first installed package, followed by Geant4.

Geant4 (Geometry and Tracking)[7] is a Monte Carlo simulation package designed to track the passage of particles through matter. Monte Carlo simulations function by using random numbers and probabilities to determine whether a certain event will or will not happen. The predecessors to Geant4 were initially distributed by CERN, though Geant4 is currently supported by the Geant4 collaboration. Though CLHEP is at the root of the Sim12 software tree, GEANT4 really provides the majority of functionality to

Sim12. It is capable of geometry analysis, tracking, detector response analysis, and run management. Of major significance is the fact that it is also the first platform in the Geant series to be based on object oriented programming. This allows for much simpler code maintenance, and more resources can be allocated to development.

The rest of the necessary packages each perform minor duties in comparison to CLHEP and Geant4. Xerces-C is an XML parser used for input of detector parameters. Motif is a visualization package as is Dawn. EVIO is the third most important base package as it is responsible for data input and output, though it works together with JGeant 4. Ultimately, it is quite clear that each of these packages can be very unique, and that to a large extent, development teams do not interact with each other in adding features. Hence, bringing all of the packages together and making them work without fail can be a very arduous task.

Documentation: Given the significant interdependencies between all of the packages necessary for Sim12 runtime, proper documentation is absolutely essential for any kind of distribution with the expectation of users installing and configuring all of the software. Additionally, this documentation must be easily accessible and understandable. If these conditions are not met, users will basically ignore the software, and hence any kind of usefulness the software potentially had would not be realized. In the course of this development project, a Wiki (Fig. 5) was made to hold of the instructions necessary for not only Sim12, but almost all aspects of CLAS 12. Such a repository is excellent for documentation due to the simplistic nature of adding, editing, and accessing information placed on the Wiki. Nonetheless, there was still significant work that had to be done in order to account for all of the packages.


Getting Started with Sim12 Offsite - HallBWiki - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://clasweb.jlab.org/wiki/index.php/Getting_Started_with_Sim12_Offsite#Plug-ins
evio
Bank Citi Chase FB Mail CNN SD EN TV CH
Getting Started with Sim12 Offsi...
Microsoft Outlook Web Access
Create an account or log in
article discussion edit history
Hall B
navigation
Main Page
Community portal
Current events
Recent changes
Random page
Help
search
Go Search
toolbox
What links here
Related changes
Upload file
Special pages
Printable version
Done

Getting Started with Sim12 Offsite

Contents [hide]

1 Preliminary Notes
- 1.1 Tested Machines
- 1.2 Tips

2 Obtaining Software

3 Installation
- 3.1 CLHEP:
- 3.2 MOTIF (May be installed):
- 3.3 XERCES-C:
- 3.4 DAWN:
- 3.5 GEANT4:
- 3.6 EVIO:
- 3.7 HDDS:
- 3.8 JGEANT4:
- 3.9 SIM12:

4 Scripting

5 Plug-ins

6 Troubleshooting

Preliminary Notes [edit]

Tested Machines [edit]

- Fedora Core 7
- RedHat Enterprise 9.0
- OpenSuse 10.2

Tips [edit]

- This guide assumes you are using a csh. To do so, simply type "csh" into a terminal window.
- Sim12 is extremely sensitive to incorrect environment variables from compilation to runtime. If you have any errors, first make sure they are set, and set correctly.
- Use a sensible directory structure as Sim12 is also extremely sensitive to incorrect paths. A Sim12 directory <with subdirectories for each installed portion (CLHEP, GEANT4, etc.) is adequate.
- The order you install software IS important.

Obtaining Software [edit]

- The required software is essential to correct compilation, but thankfully the Geant4 configuration script finds already installed software, and only continues if software is correctly installed.

Software	Link
CLHEP	Use 2.0.3.1 under the Downloading section
GEANT4	Download the source in your preferred format, as well as all data files except neutron without cross sections
XERCES-C	Download source from "Current Source Releases"
MOTIF	Download appropriate motif and motif-devel rpm files

Figure 5: Sim12 Installation Wiki[2]
A Wiki allows for very simple editing, addition, and removal of information. Links can automatically be created, and formatting is automatic. Hence, it is the perfect platform for any kind of documentation.

Documenting the installation and configuration of these packages presented several issues due to the developmental nature of Sim12. Firstly, users can span several environments, which in this case were distributions of Linux. In order to accommodate this, testing had to be performed on the major distributions to make sure any discrepancies were accounted for. Fedora Core 7, RedHat Enterprise, and OpenSuse 10.2 were the three tested systems. Luckily, no major installation discrepancies were found, so additional testing was not done. For a production environment, testing would be done to assure compatibility. Gentoo and Ubuntu were two distributions that were not tested.

In the realm of environments, users also have their own general ways of setting up how their computer functions. For someone attempting to write a generic document, this is of course was not a welcome reality. Almost every user will certainly have some sort of variation that could conflict with a certain step. To solve this issue, the written documentation must incorporate clear direction for any kind of directory structure creation or environment use. In this particular case, it was assumed that the user would be using csh as their terminal. Additionally, all files had to be placed in the users' ubiquitous home directory. In doing so, environmental variables could be set properly, and the chances of errors due to environment issues were minimized.

Of course, the final and major point of documentation is to lay out a clear set of instructions to accomplish a given task. To satisfy this, the written documentation for Sim12 was written to be extremely user friendly. Essentially, any direct commands were always written in red, and instruction steps were clearly numbered. Finally, any kind of responses to configuration questions were discovered and documented as well. Utilizing the Wiki created for Sim12, most power users should be able to follow the instructions to achieve a correctly configured system. This documentation was also tested via third parties attempting to follow the instructions, and their feedback was utilized to improve the final product.

Pre-compilation: Though exceptional documentation would likely lead most users to a successful installation of the software, there is one aspect that even superb documentation could not drastically improve upon, which is time. As written, almost all of the packages including Sim12 required compilation on the user's computer before becoming useful. All of this compilation added up to installation times of several hours. Such long durations were unacceptable in almost all cases. Hence installation time had to be decreased in some fashion.

The only way to eliminate the huge time penalty of users having to compile their own software is to compile the software ahead of time. This is generally done by compilation and then creation of binary packages such as .rpm or .deb files. However, this was done for Sim12 at a basic level consisting of archived tar.gz files. To make the tar.gz package, a specific system was considered which in this case was Fedora Core 7. After selecting the system, all of the packages were compiled. Finally, all of the binary files were added to the archive, which could then be distributed. When a user received such an archive, all that was necessary was to unpackage it, which took less than 5 minutes, rather than several hours. The main difficulty was that an individual package needed to be created for every different system out there that was used, so resources had to be managed. Additionally, the binary packages generally had to be placed into specific directories, so it was essential users unpacked packages in those directories, while requiring that they had proper access rights.

***Run-Time Optimization*:** Once installation time was decreased to reasonable levels, Sim12 was tested for run-time efficiency. Initially, it was found that basic operations took far too long for daily use. Additionally, certain operations, namely file input, were not functional at all. Hence, it was essential that the code for Sim12 was analyzed and optimized. The process was made slightly less rigorous because the slow processes only took place in the visualization portions of the code, and file input was easily added as a plug-in.

Though the quality of images and animations produced by Sim12 was decent and acceptable from the beginning, it was the unacceptably low frame rate that directed the need for visual optimization. In order to do this, the visualization code was analyzed and it was determined that there were two areas where improvements could clearly be made, which were in changing to a stored vertex paradigm from an immediate paradigm, as well as removing unnecessary redraw commands.

OpenGL is the language in Sim12 used to make any kind of graphics. It is a universally accepted language which is used throughout the gaming, movie, and simulation industries. Everything in OpenGL is represented by vertices, which are basically points on a polygon. These vertices exist within a given three dimensional space inside of memory, so that manipulating them is a fairly natural process. However, in order to draw the vertices on the screen, it is necessary to convert them from the three dimensional system they are stored in, to the two dimensional system of a computer monitor. Within OpenGL, there are two major modes of drawing images on screen, which are known as immediate and stored. Within immediate mode, any time an image is redrawn, the location of every single vertex in the object to be visualized is recomputed. Essentially, a computation of where a particular vertex should be is made, and that result is discarded as soon as the pixels are drawn. However, there is also a stored methodology that utilizes a display list to store images. Hence, if an object does not actually change between various frames, the vertices that make up the image do not have to be recomputed for display, and are accessed via the display list containing post-processed objects.

It is clear that in any situation with any kind of image or object repetition, the stored method will significantly improve graphical performance. This was certainly the case for Sim12 since the majority of the CLAS 12 detector never moved, but was composed of thousands of vertices. Hence, an attempt was made to convert the image generation to the stored method. To reiterate, prior to this, any kind of translational, rotational, or zooming that was requested in Sim12 took approximately 10 seconds to complete. Additionally, it was clear that the image was getting redrawn several times. After the conversion to the stored method, the same transformations only took a few brief flickers to complete. The reason for the flickering was thought to be due to unnecessary points of redrawing the image. Once the code was analyzed, it was found that certain functions called the redraw command within themselves, yet code for redraw appeared afterwards. Hence, certain instances of redraw commands within the code for Sim12 were removed. Once this was done, transformations occurred instantaneously and without delay.

Plug-Ins : After a core software program is written and distributed, any updates, critical or not, are difficult to distribute if the program is large and requires long recompilation times. Users do not expect to have to perform such lengthy maintenance, especially if only one simple feature is updated. Plugins, on the other hand, can be extremely easy to implement by a user, often involving a single download into a specific directory as the only necessary step to gain or improve functionality. Sim12 will eventually have full support for plugins for purposes currently known, and uses that will be discovered in the future. The main priority at this time is to develop the infrastructure necessary for easy plugin integration. Once that is in place, and the documentation is present to ease the development process of future plugins, useful additional features to the core program can be considered. In the process of developing a plugin infrastructure, a plugin must be written and in this case it enabled event file input functionality.

File-Input: The majority of users for CLAS 12 simulations have particular data that they wish to simulate in terms of particles with given momenta and origin. To prepare this data, there are many different event generators written by groups, individuals, and organizations. Of course, attempting to incorporate all of these generators into Sim12 would have been a futile task. However, what these event generators do share is a very limited set of output data. Almost all of them export particle ID, momentum, and origin. Hence, a way to allow event generator incorporation was to simply allow their output files to be read. In order to do this, separate code was written to parse two file types. They were then compiled as plug-ins, meaning Sim12 and any other packages would not have to be recompiled for the additional functionality. The first file type was a generic file containing only necessary data in a specific form (Fig. 6). The second file type was the most popular documented format known as LUND, which is a descriptive

```
n_part: 1
id: 2112 x: 0 y: 0 z: -11.0504 px: 8 py: 0.2 pz: 1
n_part: 2
id: 2112 x: 0 y: 0 z: -11.0504 px: 8 py: 0.2 pz: 1
id: 2112 x: 0 y: 0 z: -11.0504 px: 8 py: 0.2 pz: 1
```

Figure 6: Representation of Generic Input File. This representation shows the basic layout of the generic event generator output. "n_part:" refers to how many particles are in the following event. "id:" refers to the Geant particle ID which here are neutrons. "x:", "y:", and "z:" refer to the origin of the event in centimeters. "px:", "py:", and "pz:" refer to initial momentum

format containing not only momentum and origin information, but also parent-child particle relationships as well as other data.

In making these plug-ins and determining that they were in fact functional, another task was accomplished which was the testing of the plug-in system. These pieces of code were separately combined and worked without issue in the Sim12 system implying that other software could just as easily be written to accept more file formats, or directly feed an event stream to Sim12 as it was generated.

Preliminary Simulation Testing: Given the amount of time that was taken to research and document Sim12 set-up, as well as the time necessary for plug-in development, only brief testing was performed to see if Sim12 produced reasonable data. Essentially, the testing that was done consisted of running a 5,000 event series composed of the exact same momenta and origin values with the toroidal magnets turned off. The results that were analyzed consisted of how many particles ended at given points on an x-y plane perpendicular to the beam line. In doing so, it was expected that a majority of particles would follow the same trajectory and hence end up at a very similar final position. The clear result was that a vast majority of the 5000 events ended in a very specific area on the plane (Fig. 7). The remaining hits were likely Moller electrons, as seen by the vast even dispersion and very low hit counts. Taking into consideration that the magnets that would normally alter a particle trajectory were turned off, this preliminary result confirms that given a certain momentum, the detector will not usually adversely affect the path of a particle. This was exactly the kind of result that was desired, as more in depth testing could now be performed with magnets, varying momenta, and varying origins, without having to consider the potential effects of simulation error.

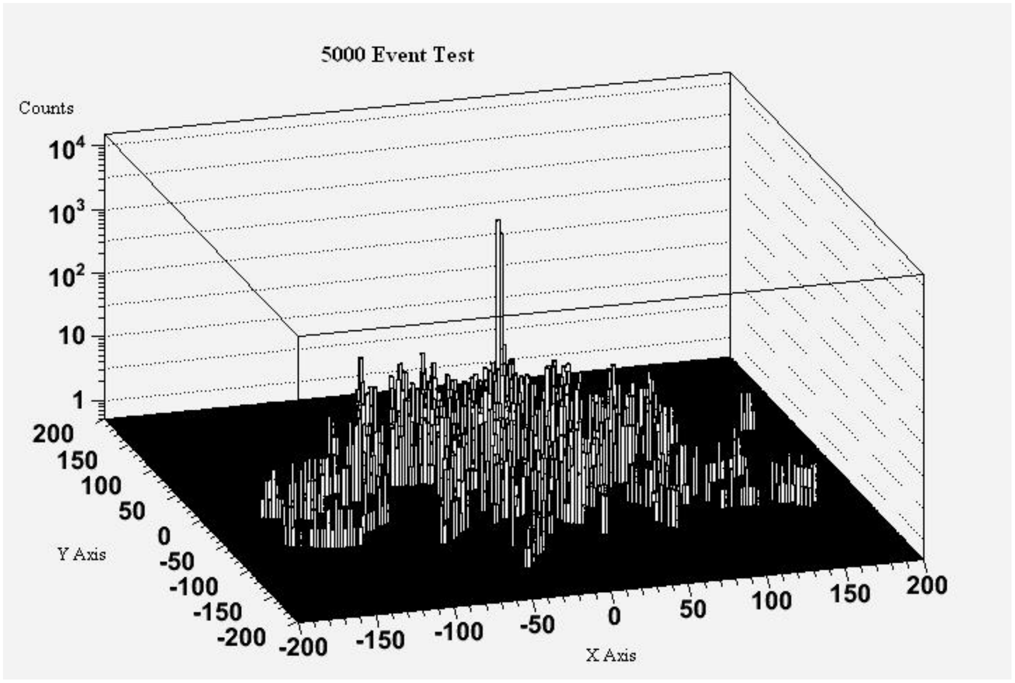


Figure 7: Results of 5000 Event Test in Terms of Particle Position

Conclusions: The 12 GeV upgrade will without a doubt bring exciting new physics results and experiments to the community. However, even though JLab was designed to be cost-effectively improved from the beginning, the upgrade is no small task requiring massive amounts of development in terms of both structures and software. During this development stage, many hurdles have been overcome including the creation of adequate documentation and binary packaging to allow for user release. Additionally, the software was optimized to run at usable speeds via a paradigm switch to stored graphics and the elimination of unnecessary commands. Finally, brief simulation testing was performed and it was found that at least in the most basic cases, the results are appropriate. The project is still without a doubt in the development stage, though the focus should be soon shifting to testing rather than functionality research.

Future Work: Simulation software is integral to the proper analysis of CLAS 12 data, and hence it must be accurate. CLAS 12 itself will become a far more sensitive device after the upgrade, and data will have less uncertainty within it. Hence, once all of the functional aspects of the software are complete, the main focus will be to assure that the results from simulations performed by Sim12 will be equally certain and usable for data analysis.

References

1. Dergachev, K. Gilfoyle, G.P. *CLAS12 SIMULATION ~ 12 GeV Upgrade~* University of Richmond, 2007.

2. Dergachev, K. Gilfoyle, G.P. *Getting Started with Sim12 Offsite*. 10.08.2007 <http://clasweb.jlab.org/wiki/index.php/Getting_Started_with_Sim12_Offsite>

3. Dergachev, K. Gilfoyle, G.P., *Jefferson Laboratory – 12 GeV Upgrade*. University of Richmond, 2007.

4. Gill, K., Gilfyole, G.P. *Investigating Asymmetries in the D(e,e'p)n Reaction* University of Richmond, January 10, 2008

5. Lung, Allison. *12 GeV Upgrade Technical Scope*. 01/12/2007 <http://www.jlab.org/12GeV>

6. Pfeiffer, A. *CLHEP – A Class Library High Energy Physics.* 15/11/2006 <http://proj-clhep.web.cern.ch/proj-clhep>

7. Wikipedia, *Geant4*. 02/01/2008 <http://en.wikipedia.org/wiki/Geant4>