Development of a Two-Dimensional Event Display for Tracking Mode of Q_{weak}

Derek Jones

The George Washington University, Washington, DC

Thomas Jefferson National Accelerator Facility
Newport News, VA

July 30, 2010

Prepared in partial fulfillment of undergraduate physics research requirements at The George Washington University. Conducted under support by a grant from the National Science Foundation with Wouter Deconinck and Allena Opper as advisers.

Participant: _____
Signature

Project Advisor: _____
Signature

TABLE OF CONTENTS

Abstract

In nuclear physics research, several different types of apparatus are used to detect scattered particles. The computer programs that are often developed to analyze data taken from the detectors are frequently very complex and use so much computing time that the acquisition of new data is reduced or even lost. The Q_{weak} Collaboration at Jefferson Lab uses gas electron multipliers, horizontal drift chambers, vertical drift chambers, and trigger scintillators for calibration. The purpose of this project was to develop a graphical user interface that displays Q_{weak} tracking mode data in analyzable projections with scaled geometry to show hit patterns in real time for each event. By plotting the triggered elements in the tracking hardware for each event, collaborators may be able to derive a viable interpolation of the particle track(s) for each event by using the drawn patterns. The new macro was created with C++ primarily using the ROOT library. Geant4 Monte Carlo simulated data and raw experimental data were used to troubleshoot the program. Gathering data for beam production analysis was controlled by the CEBAF Online Data Acquisition system and stored into ROOT files to be read by the event display. Orthographic projections and tracking data were organized to move between events and beam-line regions to depict hit patterns of the particles for each event. Through effective collaboration and testing, hit patterns were successfully displayed with practical features in a user-friendly environment. The macro may also be used by collaborators to provide hardware diagnostics during production. It may be run independently or within the larger Q_{weak} Data Analysis Graphical User Interface.

Introduction

Q_{weak} Collaboration Overview

The Q_{weak} Collaboration was created to conduct a very precise study to challenge the Standard Model in the search of new physics. Parity-violating electron scattering on the proton (shown in Figure 1) at low Q^2 (a momentum factor) will provide a precision measurement of the proton's weak charge Q_W^P. This sensitivity to the weak interaction is shown in the following formula:

$$Q_w^P = 1 - 4\sin^2\theta_W$$

The measurement of the weak charge of the proton is expected to be conducted with about a 4% total combined error. With this, the weak mixing angle $\sin^2\theta_W$ will be derived within 0.3% error. This weak interaction factor describes the tendency of quarks in a proton to exchange the Z-boson with electrons rather than the more often exchange of a photon via the Coulomb interaction as shown in Figure 2. This collaboration is in contrast to research at the Large Hadron Collier at the European Organization for Nuclear Physics (CERN) where the search for new physics focuses on high energy interactions [1].

Tracking Systems

The Q_{weak} tracking system was developed into three regions that have a specific set of hardware to follow the track of scattered electrons. Region 1 incorporates a gas electron multiplier (GEM) that uses charged strips called traces within a chamber of ionized gas. The output of a GEM is essentially a point on a grid. Region 2 incorporates horizontal drift chambers (HDCs) that use an array of charged wire planes in separate chambers. As an electron passes through a chamber of ionized gas, the charge of the wire and the electron interact to cause the electron to drift toward the wire. The time it takes from the gas to get excited until the wire gets excited is recorded as the drift time. From this, a drift distance from the wire can be calculated for tracking [2]. Region 3 incorporates similar hardware

called vertical drift chambers (VDCs), but these are tilted at an angle so that the electrons will pass through close to a 45 degree angle. Region 3 also contains a trigger scintillator equipped with photomultiplier tubes (PMTs) that can detect where the particle passed based on the amount of charge signal transmitted to the PMTs. The trigger scintillator is also used to tell the main detector when to gather data [2]. A drawing of the entire Qweak apparatus is shown in Figure 3.

The Qweak apparatus has two coordinate systems defined—local and global. This is because the detectors have the ability to rotate to eight positions to measure asymmetry. Each individual octant would be difficult to provide understandable coordinates with global coordinates, so only local coordinates are used in the event display. The Z direction is defined as the direction that the electron beam travels, the Y direction perpendicular to the right of that direction, and the X direction goes perpendicular in the upward direction [3].

There was an original concept for the event display to link the regions in tracking mode by Marcus Hendricks at The George Washington University in 2009 [4]. The current version of the event display initially used the earlier version as a reference, but ultimately strove for a new and more efficient renovation of the graphical user interface.

Data Acquisition Systems

The data acquisition system (DAQ) used for the Q_{weak} Collaboration follows a standard procedure at Jefferson Lab for taking information from detectors and converting them into viable data files. First, a trigger tells the tracking hardware to collect data. This can be a scintillator or the independent tracking apparatus. Next, the signals are digitized from individual tracking channels to be transported from the accelerator site to a farm of computers. Common devices that do this are analog-to-digital converters and time-to-digital converters [5]. Before being stored at a Jefferson Lab farm, a protocol called CEBAF Online Data Acquisition or CODA (CEBAF stands for Continuous Electron-Beam Accelerator Facility) formats the signals. CODA is a software toolkit that standardizes and

transports event data by minimalizing commercial software dependance and providing specialized support for common Jefferson Lab detector systems [6]. Data from CODA is stored as a LOG file that is then processed by the QwTracking analyzer to produce a ROOT file. The QwTracking executable organizes the raw information into readable files that can be used by user-created software such as the event display. Figure 4 shows at a glance how Q_{weak} DAQ follows this path.

Materials and Methods

Using the ROOT Library

The Q_{weak} 2D Single Event Display was created as a graphical user interface macro in the C++ computer programming language. More specifically, the program was predominantly made with a C++ library called ROOT that was developed by CERN. CERN created this library to provide functionality to systems that need to analyze a large amount of data in a short amount of time [7]. Several classes were created that specifically deal with nuclear physics and tracking systems. An example of this is the *GEANT* class, an acronym for "geometry and tracking" with Geant4 being the latest installment, that is used to describe detector geometry and to simulate particle collision events in tracking systems [7]. A Q_{weak} package using this class was used to test the Q_{weak} tracking systems and is described in the next section.

The most useful class for the event display is called *GUI* and was created specifically for developing graphical user interfaces. A prototype developing program called TRootGuiBuilder was used to create an initial mock-up of the event display. By using the available widgets to control frames, labels, boxes, and more, the display was able to exhibit a neat aesthetic. ROOT also contains a powerful tool called CINT, which stands for C++ Interpreter. By using CINT, the TRootGuiBuilder was able to define which elements inherited each other (such as the labels inside the frames) and output

C++ code that would create the exact mock-up shown on the screen [7]. After a few touch ups through variable name changes and code formatting to personal style, the shell of the event display was ready to have functional logic inserted.

Programming Logic

After drawing the initial graphical interface framework, the program essentially takes the data stored in a ROOT file by DAQ and outputs that information into a variety of useful analytical features. A partial logic tree of the event display is shown in Figure 5. The macro first opens the ROOT file and finds the *hits_tree* tree and sets *fQwHits* as the branch to import data from. The data leaves are collected and placed in a ROOT hit list called *fHitRootList*. While this is happening, the leaf containing the current event number is used to print the current event number in the "Current Event" group frame. Also, the leaf containing the octant position (proposed quantity) is used to draw the appropriate boxes in the "Octant Identification" box. The ROOT hit list is then converted to *fHitList*, a standard Q_{weak} class in the Q_{weak} tree event buffer to organize data.

fHitList is then separated and stored into several different hit containers that correspond to each region and a direction of the elements in that region. These are gathered as *Hits_R1x*, *Hits_R1y*, *Hits_R2x*, *Hits_R2u*, *Hits_R2v*, *Hits_R3u*, and *Hits_R3v* where the region number is proceeded by the element direction. The information for each container is stored into a character array called *fHitBuffer* that allows that inherited data to be printed in the "Wire Hit Information" list box.

Then, individual embedded canvases are called corresponding to the region of each container. Each of these canvases is noted as XY (front), YZ (top), or XZ (side) views referring to local coordinates of the tracking system. The appropriate canvas is selected as the current pad and made available to edit. A TLine is created called *Line* that is used to store in associated vectors for each hit container the drawn elements. The process of drawing the elements is different for each of the tracking regions, but the method of storing them is consistent.

Generally, there are three types of elements that are drawn: full length, left of full length, and right of full length (wires more specifically as traces are the same length). These are separated by the first and last full length elements that are taken from the geometry and saved as constants in the header. This system decides how to draw the elements (using trigonometry) so that they have the correct length in the chambers. The elements are drawn and centered with respect to the chamber in ROOT local coordinates (coordinates with respect to the canvas rather than the whole frame). A scaling factor was also multiplied with all the Geant4 detector descriptions to convert the true length of a centimeter to the ROOT local coordinates of the canvas. Lines are drawn with colors following a key displayed at the bottom of the display. To create a standardized convention, the element numbers go from right to left and bottom to top while the chambers are numbered front to back. After drawing the elements in each canvas, editing of the canvas is disabled.

For Region 1, a variable called *fTrace* is created to reflect a triggered element. In a simple algorithm, the Region 1 traces are drawn to scale using the chamber data taken from the simulation file QwSimGEM.cc [8]. Each drawn line is pushed to the back of the *Line* vector and drawn in the canvas. This occurs for both the X and Y directions in all three views.

Region 2 output goes through a more intricate process. Several variables are created including *fWire* (gets wire number), *fPlane* (gets plane number), *fPShift* (to shift wires in the prime planes half of the drift cell distance), *fXShift* and *fYShift* (that are used to shift the wires based on plane position), and *fShiftTrim* (accounts for wires near the edge that are shifted "outside" of the chamber). First, a switch gathers the planes in the prime chambers (every other chamber after the first) and adds a *fPShift* and a *fShiftTrim* if the wire is a full wire. Then by using a conditional statement, if the plane number is greater than 12, the *Line* vectors are told to be drawn in the 2nd arm region frame "Region 2—HDC (5-18)". The wire numbers are separated with respect to the full length wires, pushed back in the *Line* vector, and drawn. An example of this for the Region 2 U Plane XY View is shown in Figure 6. The

fPlane switch is used in additional ways with the top and side views as it uses *fXShift* and *fYShift* to draw the wires in the correct chamber and with *fPShift* if necessary. The projections are drawn to scale with data taken from the simulation file QwSimHDC.cc [8].

Region 3 output has a similar process as Region 2. The variables that are created are *fWire* (gets wire number), *fPlane* (gets plane number), and *fXShift* and *fYShift* (that are used to shift the wires based on plane position). Another *fPlane* switch separates the wire planes into the appropriate chambers using *fXShift* and *fYShift*. Since there are no prime planes like Region in 2, these shifts are purely for aesthetics so that the chambers may be easily seen. Also, all of the chambers fit into one region frame, so there was no need to add another if statement to send the drawings to another frame. The wire numbers are separated with respect to the full length wires, pushed back in the *Line* vector, and drawn. The top and side views are drawn similarly. The projections are drawn to scale with data taken from the simulation file QwSimVDC.cc [8].

Using Q_{weak} Geant4 Monte Carlo Simulations

While developing the programming code for the event display, Geant4 Monte Carlo simulations were used to test and troubleshoot the program. Q_{weak} Geant4 Monte Carlo was customized with supervision under Klaus Grimm, Michael Gericke, and John Leacock at Jefferson Lab. This package used Monte Carlo analysis and Q_{weak} hardware geometry to provide realistic hit patterns for simulated events. They were stored in LOG files and analyzed under a similar program to QwTracking called QwSimTracking to create ROOT files that could be run by analysis programs in Q_{weak} [3]. The hardware geometry listed in the package was also used as a reference to gather the dimensions and layout of the chambers in each region and the number and layout of the elements within those chambers.

The event display was initially constructed with the assumption that these simulations were accurate, but many inconsistencies became apparent such as the U and V directions and where to begin

counting elements. The simulations also did not take into account the octant position and ignored the second arms of chambers. Other errors are listed in the *Known Errors* section later in the Results chapter. With these drawbacks in mind, the event display was completed with caution while using Q_{weak} Geant4 Monte Carlo as a testing medium.

Results

Event Display Features

The graphical user interface was separated into two main sections: event boxes and region frames. The three event boxes on the upper section displayed specific data for each event. The five region frames in the lower section were controlled by composite frame tabs and displayed orthographic projections—in local coordinates—of each event with scaled geometry taken from Q_{weak} simulation files. The event display application is shown in Figure 7.

The three event boxes were titled “Event Counter”, “Wire Hit Information”, and “Octant Identification”. The event counter box displayed the current event number and provided control of the GotoEvent() function that allowed the user to choose the current event number. The wire hit information box contained a list box that included the region, element, plane, and drift distance for each hit in each event. The octant identification box displayed the current octant position and orientation for each event. The dark orange box represented the lower chamber numbers (i.e. one through four for Region 2) while the light orange box represents the higher chamber numbers (i.e. five through eight for Region 2).

The five event frames were conveniently labeled “Region 1—GEM”, “Region 2—HDC (1-4)”, “Region 2—HDC (5-8)”, “Region 3—VDC”, and “Region 3—TS”. The first draws the tracking data for the GEM. The next two draw the tracking information for the HDCs where the first displays

chambers one through four and the second displays chambers five through eight. The final two draw the tracking information for Region 3 where the first displays all of the VDC chambers (one through four) in a single view and the second displays the trigger scintillators.

There were also several color-coded keys positioned around the display to easily differentiate directions and chambers. At the bottom of the window, the definition of the local coordinate system was described with the X, Y, U, and V directions in red, violet, green, and blue respectively. For Regions 2 and 3 with multiple chambers, keys were positioned directly into the region frames including the name or number of the chambers. A dotted line was included in the Region 3 canvases to separate the two arms at an off-scale distance. For all region frames, the orientation of each projection was labeled at the top. Regions 1 and 2 were oriented with standard orthographic projections while Region 3 was oriented with respect to the tilt angle of the chambers.

The event display was built with several control features for easy access between events and region views. The first element was a menu bar that uses two pop-up menus titled “File” and “Help”. The file menu included open and close options for working with ROOT files. The help menu included options for a link to the tutorial documentation [9] and to print a text about the event display. Scroll bars were included to allow the user to move up and down in the wire hit information list box to view the data for all of the hits. The GotoEvent() function features included a number entry and a button. This allowed users to select which event to display by typing that event number in an entry box and clicking on the button to move to the corresponding event to the displayed number in the entry box. Tabs were included to connect to the composite frames for each region. These allowed simple switching between the region frames because all three views for each tab changed with a single click. As for general control of the current event, a “Previous Event” and a “Next Event” button were included to move to the preceding or following event, respectively, from the current event number when clicked. Finally, an “Exit” button was added to terminate the application when clicked.

Event Display Analysis

A simple analysis of the drawn lines in the region frames provided an interpolation of the particle track(s) in each view. As the information in the event boxes included more supplemental information, the region frames displayed the hit patterns for analysis. It was found that each of the tracking hardware required its own technique to analysis.

The track in Region 1 was found with the intersection between the X and Y traces in the front view. The top and side views depicted the track angle if any. Region 2 tracks were found with the intersection of the X, U, and V wires in the front view. The red points of the X wires in the side view were also useful in describing the pitch angle of the track. Region 2 analysis is the same for both the "HDC (1-4)" tab and the "HDC (5-8)" tab. An example of this is shown in Figure 8. The Region 3 track was found at the intersection of the U and V wires in the front view, but was more difficult to interpolate as the track came at an oblique angle. It was best to use the intersections between the U and V wires in all the views. Since there was no trigger scintillator data in the simulation files, the trigger scintillator frame was not completed.

The drift distance information was not taken into account for any of the element drawings. However, this data may have been used for error analysis if necessary. Also, it was found that it was important to look at *all* of the views in each region because individual views may be misleading (as with the nature of orthographic projections) even when suggested to look at a certain view.

Known Errors

Some features of the event display contain incomplete information or bugs. For the first drawing of an event each time the program is opened, certain wire hits did not appear event though they were extracted from the hit container. A simple fix for this was found by going to another event then returning to the first desired event. The GEM did not receive any X trace data from simulation files while some of the simulated data did not provide projections with crossing wires at all. There was

no trigger scintillator data in the event display at the time of completion. Also, the octant identification did not work because of the want of a variable to describe octant position.

Discussion and Conclusion

There are some aspects of the event display that are still not fully developed. At the time of the creation of this paper, there had been no usable test data for the GEMs or the trigger scintillators. Another reason for this is that the *hits_tree* tree did not include accurate Region 1 data or any trigger scintillator data at all. There were also no GEMs installed at this time. If updated in the future, the Region 1 frame display will most likely be similar, but the trigger scintillator will display the point where a particle entered and PMT signals on a colored scale.

The event display is used to draw the triggered elements to depict patterns in the chambers so that collaborators may interpolate the tracks manually. This is not a completely full-proof method, so in the future, the event display might be developed to include computer drawn fit lines for the particle tracks. These lines might be drawn using an algorithm that combines the triggered elements and the drift distance data. For the octant identification, a new variable would need to be created that describes the current position of the chambers in the octant system.

The macro was easily implemented into the Q_{weak} Data Analysis Graphical User Interface by copying the source code from QwEventDisplay.cc into QwGUIEventDisplay.cc (and QwEventDisplay.h into QwGUIEventDisplay.h) in the Extensions trunk of the Qweak repository. Only a few variables and functions needed to be edited allowing the display to become a valuable subsystem for analyzers. The menu bar was removed from the event display because the analysis graphical user interface already contained one with similar features. The event display may still be used to as an independent program using the test executable QwEventDisplayTest.

The event display was also found to be useful for tracking hardware diagnostics as originally predicted. When live run data was used with the event display, it was easy to see that some element channels may fire with every event and some not at all showing channeling problems. In some cases, cosmic ray particles contributed to triggered elements skewing the hit patterns. The wire hit information list box data was not able to distinguish these hits, but a glance at the region frames often allowed collaborators to see the elements triggered in error.

The display also serves as a compliment to the Q_{weak} 3D Single Event Display developed by Juan Carlos Cornejo at William and Mary. The 3D display provides a more detailed look at the detector system as a whole, while the 2D display provides a quick and simple snapshot of each region chamber.

Overall, the Q_{weak} 2D Single Event Display was successful in providing viable reconstructions of hit patterns in scaled geometry with an interface that includes many useful viewing and control features.

Acknowledgments

This project was conducted at the Thomas Jefferson National Accelerator Facility, which is supported by the United States Department of Energy. The advisers for this project were Wouter Deconinck and Allena Opper. The members of the Q-weak Collaboration, especially the analysis team, were instrumental in the completion of this project. Strong contributions were made by Marcus Hendricks (original GUI design) and Ramesh Subedi (technical support). Additional support was given by John Leacock (horizontal drift chambers), John Leckey (vertical drift chambers), Katherine Myers (trigger scintillators), and Juan Carlos Cornejo (collaboration with his 3D event display). This project was supported by a grant from the National Science Foundation.

References

[1] Qweak Collaboration, Thomas Jefferson National Accelerator Facility, “Qweak”, 6 May 2008, http://www.jlab.org/qweak/.

[2] Mestayer, Mac, “Wire Chambers,” (presented at Thomas Jefferson National Accelerator Facility, Newport News, VA, June 15, 2010).

[3] Qweak Collaboration, Thomas Jefferson National Accelerator Facility, “Qweak Wiki”, May-July 2010, https://qweak.jlab.org/wiki/index.php.

[4] Hendricks, Marcus, “QwEventDisplay.cc,” (Qweak repository revision 1110, July 2009.)

[5] Abbott, David, “Data Acquisition Systems,” (presented at Thomas Jefferson National Accelerator Facility, Newport News, VA, July 2, 2010).

[6] Heyes, Graham, “CODA,” (presented at Thomas Jefferson National Accelerator Facility, Newport News, VA, 24 July 2003).

[7] ROOT Development Team, European Organization for Nuclear Research, “ROOT Reference Guide,” May-July 2010, http://root.cern.ch/drupal/content/reference-guide.

[8] Qweak Collaboration, Thomas Jefferson National Accelerator Facility, “QwGeant4 Trunk,” (Qweak repository revision 1573, July 2009.)

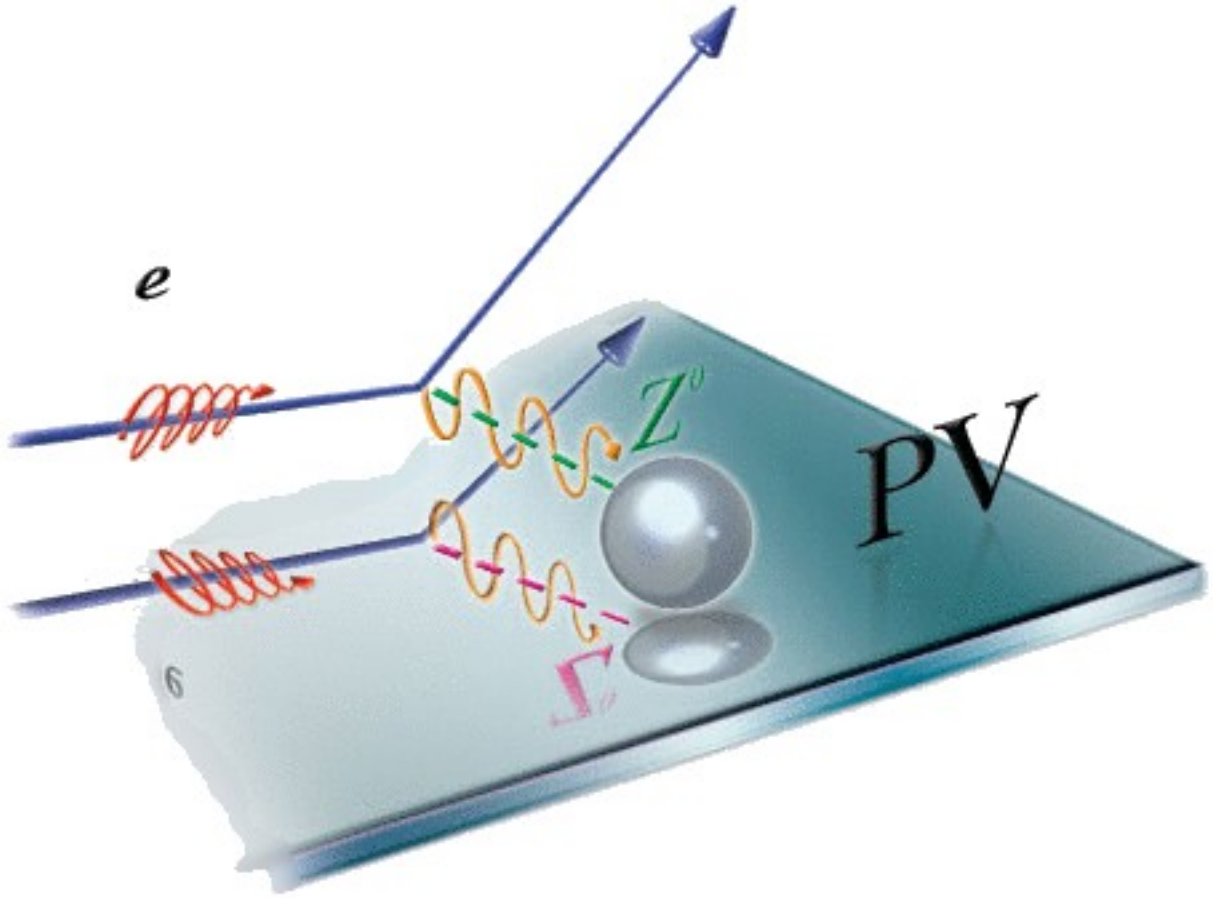
[9] Jones, Derek, “QwEventDisplay_Tutorial,” 28 July 2010, https://qweak.jlab.org/wiki/images/QwEventDisplay_Tutorial.pdf.

[10] Paschke, Kent, “Qweak” (Image), 20 August 2007, http://people.virginia.edu/~kdp2c/.

[11] TriX Collaboration, Accelerator Institute of the University of Groningen, “Atomic Parity Violation” (Image), 21 March 2008, http://www.kleinekruisstraat1.nl/kvi/index.php?option=com_content&task=view&id=14&Itemid=31.

Appendix A

Figure 1. Weak Interaction of the Proton [10]



Appendix B

Figure 2. Three-Step Diagrams of Parity Violation (left) and Coulomb Interaction (right) [11]

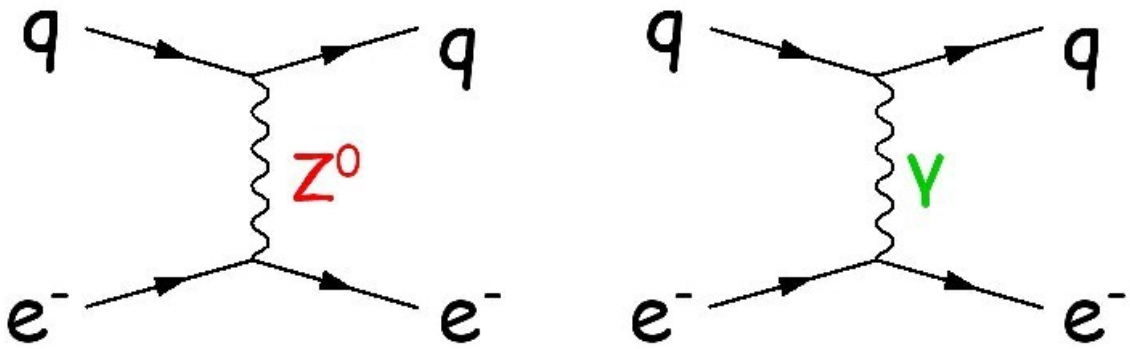
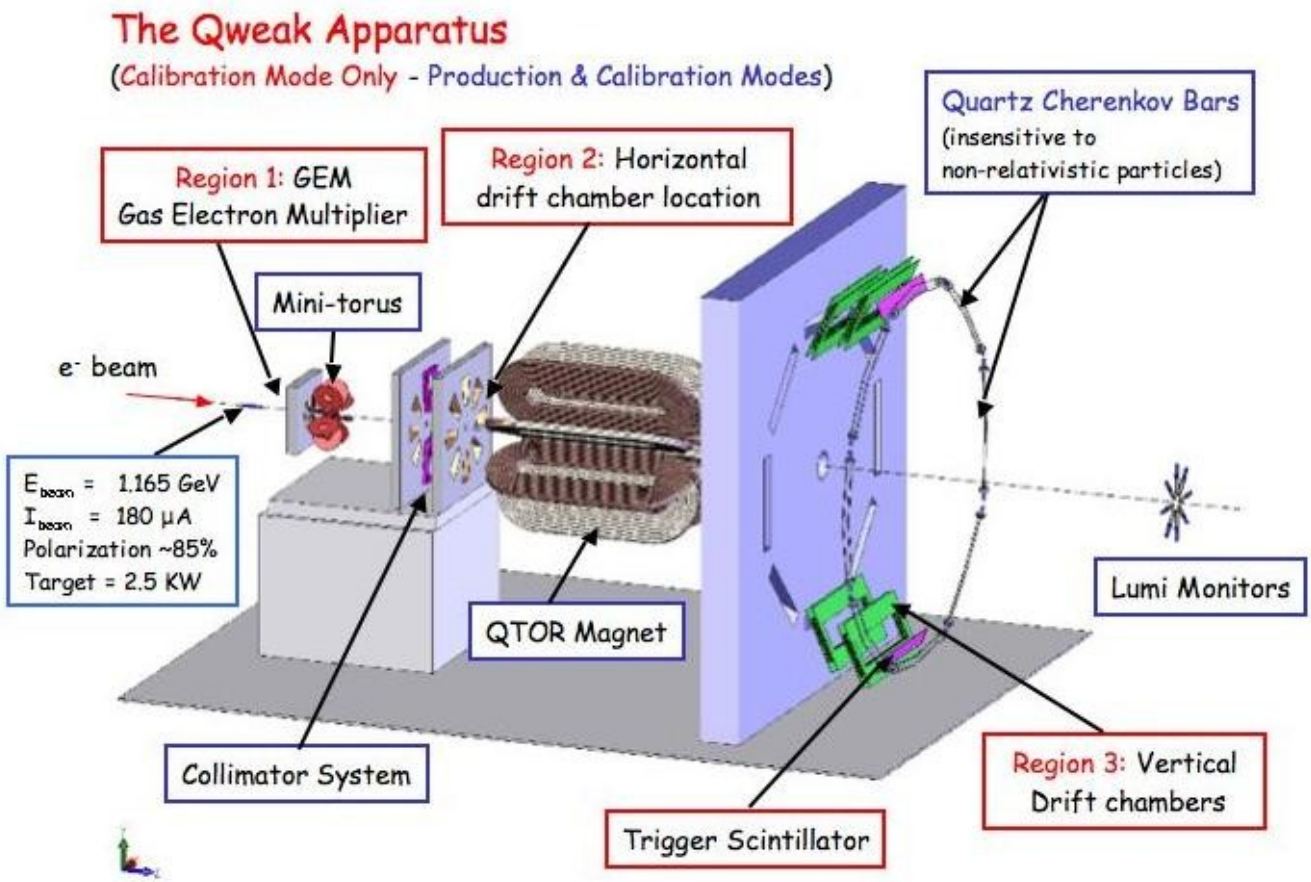


Figure 3. The Q_{weak} Apparatus [1]



Appendix C

Figure 4. Flow Chart of the Q_{weak} Data Acquisition System

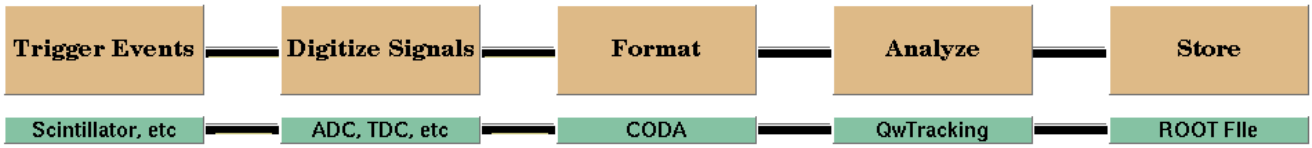
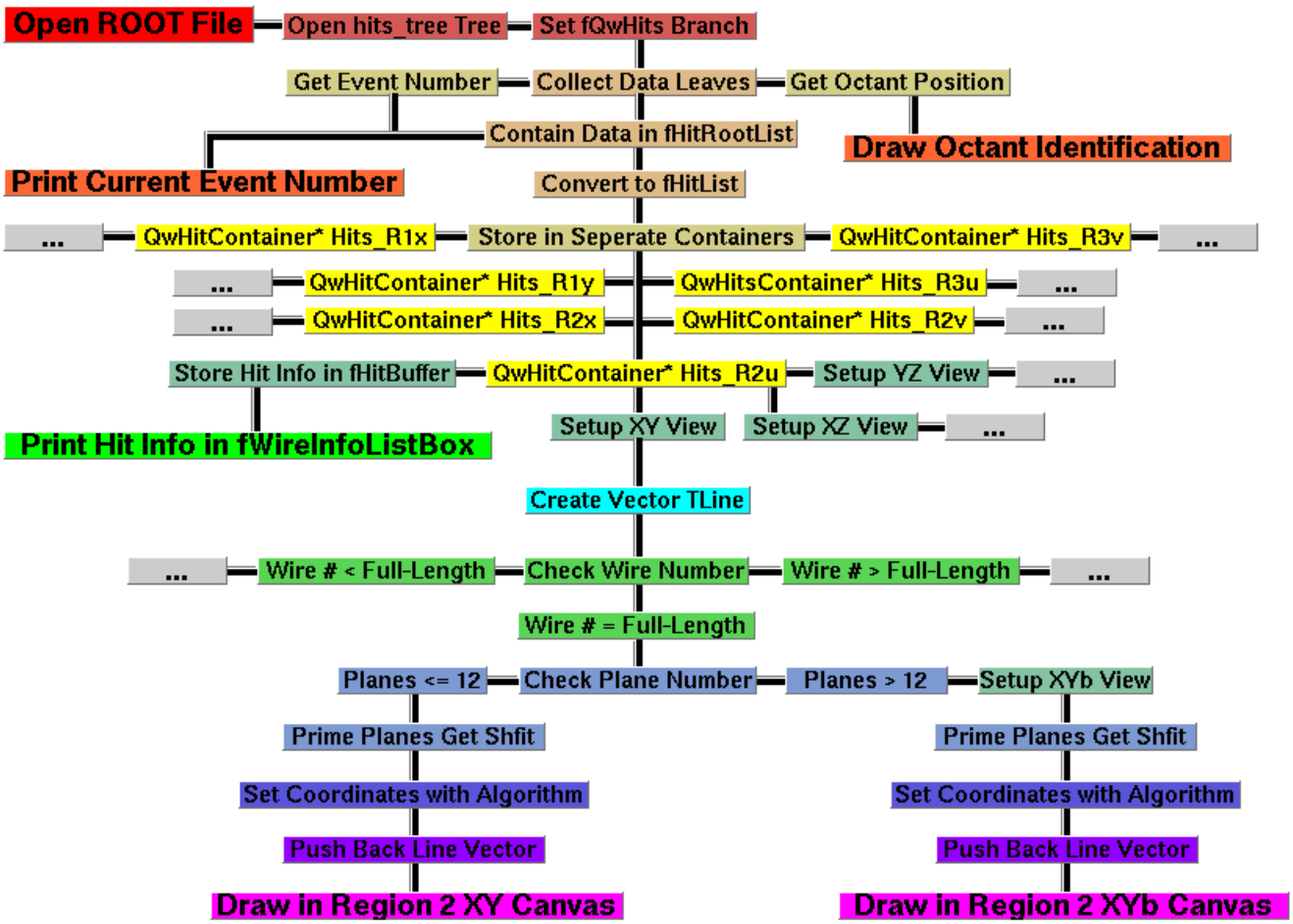


Figure 5. Partial Event Display Programming Logic



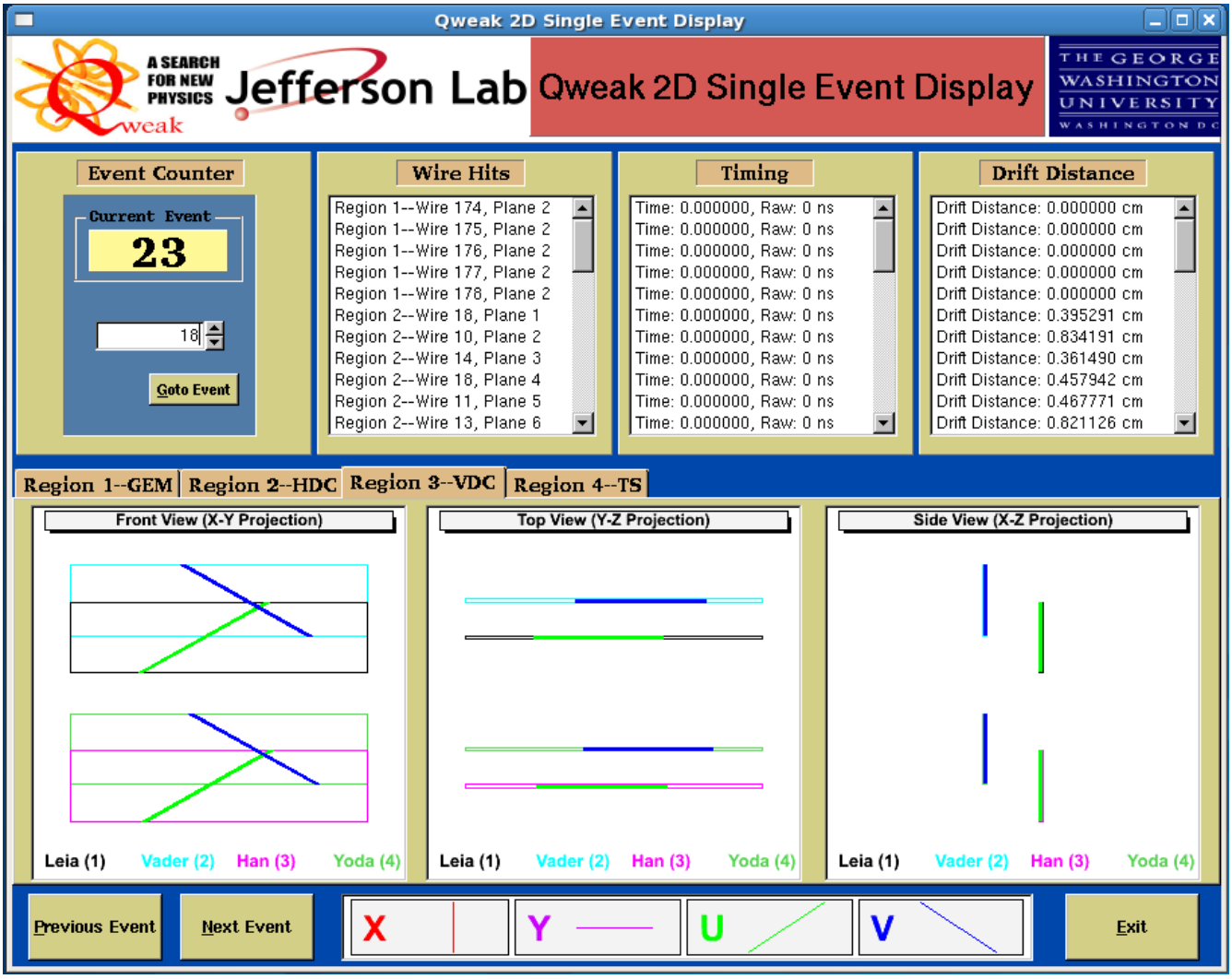
Appendix D

Figure 6. Code Snippet of Region 2 U Plane XY View Logic

```
// Region 2 U wires (29 total)
for(QwHitContainer::iterator fHit = Hits_R2u->begin(); fHit != Hits_R2u->end(); fHit++){
  TLine Line;
  int fWire = fHit->GetElement(); // Wire number
  int fPlane = fHit->GetPlane();  // Wire plane
  int fShiftTrim = 0;             // Used to correct shifted prime wires from going outside the frame
  double fPShift = 0;             // Used to shift the wires on the prime planes (ROOT coordinates)
  switch (fPlane){
  case 4:
  case 5:
  case 6:
  case 10:
  case 11:
  case 12:
  case 16:
  case 17:
  case 18:
  case 22:
  case 23:
  case 24:
    fPShift = .5*R2_UVDIST*R2_CM; // Half drift cell distance
    if (fWire == R2_FULLWIRE1 || fWire == R2_FULLWIRE2)
      fShiftTrim = 1;
    break;
  }
  if (fPlane > 12) // If on second arm, draw in appropriate tab
    fRegion2bXY->GetCanvas()->cd();
  if (fWire < R2_FULLWIRE1){
    Line.SetX1(.5 + (R2_WIDTH*R2_CM*.5) - (R2_UVDIST*R2_CM*fWire) - fPShift);
    Line.SetY1(.5 - (R2_LENGTH*R2_CM*.5));
    Line.SetX2(.5 + (R2_WIDTH*R2_CM*.5));
    Line.SetY2(.5 - (R2_LENGTH*R2_CM*.5) + (((R2_UVDIST*R2_CM*(fWire - fShiftTrim)) + fPShift)*tan(R2_ANGLE*TMath::DegToRad())));
  }
  else if (fWire < R2_FULLWIRE2){
    Line.SetX1(.5 + (R2_WIDTH*R2_CM*.5) - (R2_UVDIST*R2_CM*fWire) - fPShift);
    Line.SetY1(.5 - (R2_LENGTH*R2_CM*.5));
    Line.SetX2(.5 + (R2_WIDTH*R2_CM*.5) - (R2_UVDIST*R2_CM*(fWire - R2_FULLWIRE1)) - fPShift);
    Line.SetY2(.5 + (R2_LENGTH*R2_CM*.5));
  }
  else{
    Line.SetX1(.5 - (R2_WIDTH*R2_CM*.5));
    Line.SetY1(.5 + (R2_LENGTH*R2_CM*.5) - (((R2_WIDTH*R2_CM) - (R2_UVDIST*R2_CM*(fWire - R2_FULLWIRE1)) - fPShift)*tan(R2_ANGLE*TMath::DegToRad())));
    Line.SetX2(.5 + (R2_WIDTH*R2_CM*.5) - (R2_UVDIST*R2_CM*(fWire - R2_FULLWIRE1)) - fPShift);
    Line.SetY2(.5 + (R2_LENGTH*R2_CM*.5));
  }
  Line.SetLineColor(kGreen);
  Line_R2u.push_back(Line);
  Line_R2u.back().Draw();
}
```

Appendix E

Figure 7. Screen Shot of the Q_{weak} 2D Single Event Display



Appendix F

Figure 8. Screen Shot of Region 2 Analysis