PHENIX Technical Note

Birdseye: An Interactive Event Visualization for PHENIX

by
Jeffery T. Mitchell (Brookhaven National Laboratory)
and
Paul Nilsson (Lund University)

9/27/01

Abstract

This Technical Note will describe the details of an OpenGL-based event display for PHENIX data and simulations that runs as a stand-alone interactive visualization.

1. Introduction

The high particle multiplicity in relativistic heavy ion collisions can make for some very impressive event displays with hundreds of particles spraying out from the event vertex. These event displays can be used for Public Relations purposes to illustrate the difficult task that we face in extracting information from a collision, but it can also be used to help root out problems in the detectors, their calibrations, or in the event reconstruction algorithms. This Technical Note will describe an event visualization for PHENIX named *Birdseye*. Birdseye is meant to serve primarily as an interactive Public Relations visualization package ideal for demonstrations on almost any computer running UNIX. The package relies entirely on open source software, OpenGL, with its accompanying GLUT utility libraries. The detector geometry and the hit and track information from an event are stored and read in from separate ASCII files. With this set-up, it is extremely simple for anyone to install the Birdseye package on any UNIX platform and run the display independently. An early version of Birdseye served as a pad chamber monitor during the Year 2000 run. The Birdseye display has been shown in many of the recent RHIC Open Houses at Brookhaven National Laboratory and has been seen by thousands of people.

2. The P.O.V. Analysis Package for generating Birdseye display files

Birdseye reads in reconstructed hit and track information from an ASCII file. The format of this ASCII file will be described later in the document. The software necessary for generating a Birdseye display file from either real or simulated data can be found in the PHENIX repository within the P.O.V. package (PHENIX Offline Visualization) under */offline/analysis/POV*. Up-to-date detailed information on the POV package, which can simultaneously generate POV-ray display files, can be found on the web at the URL *http://www.phenix.bnl.gov/WWW/software/luxor/uti/birdseye/*.

Classes are provided in the POV package to produce Birdseye visualization files from PHENIX DSTs (using the class named *dstPOV*) and from PISA output ROOT files (using the class named *simPOV*). The package contains numerous flags and options for choosing which detectors and what information to display. For example, it is possible to display straight line track projections for field off events, or display reconstructed track connections directly, or even display track projections from the track model for both field on and field off events. For both real data and simulated data, the general procedure to produce a display is the same (see the next section for more details on how to generate a picture). You must first have a DST or PISA file available. Then you must execute the corresponding ROOT macro (within ROOT) that will run the package on the event number that you specify. The package will produce a Birdseye display file containing any specified hit and track information. Finally, you must run the Birdseye program with this file as its input display file. You will find that generating the Birdseye display file within ROOT is nearly instantaneous once the requested event is read.

3. Producing a Display File From a DST File

This section outlines a step-by-step guide describing how to go from a PHENIX raw data file to an interactive Birdseye session.

- Obtain the P.O.V. package by typing *cvs checkout offline/analysis/POV* in your working directory.

- Use the standard PHENIX procedure to build the *POV* package libraries. Make sure that your *LD_LIBRARY_PATH* environment variable contains the location of the libraries.

- Produce or locate a DST file.

- Select the event number (in sequence on the DST) which you wish to display. You can use the POV package to do this for you by running over a large number of events (via *eventNumber*) and setting *Verbose* to 1 in order to have the number of reconstructed tracks in each event printed to the screen.

- Generate a Birdseye display file from the DST file (default name: *birdseye.dat*). In order to do this, follow the following steps:

    - Edit *povData.C* (in the macros directory of the package) in your running directory. This is the ROOT macro that will run the *dstPOV* methods on the DST file. Set the input parameters to the desired values (including the event number and the name of the output Birdseye file). Make sure that *outputMode* is set to 1 or 2. See the web documentation for updated descriptions of the input parameters.

    - Run ROOT.

    - Type *.x povData.C()* to execute the macro within ROOT. The macro will skip to the event number you have selected and then analyze that event before stopping.

- Type .quit to get out of ROOT.

- Your Birdseye file will be ready to go. You must repeat this for each event you wish to visualize.

- Run Birdseye on the output Birdseye display file as outlined below.

4. Producing a Display From a Simulation (PISA) File

This section outlines a step-by-step guide describing how to go from a PHENIX PISA output ROOT file to a Birdseye visualization.

- Obtain the P.O.V. package by typing *cvs checkout offline/analysis/POV* in your working directory.

- Use the standard PHENIX procedure to build the *POV* package libraries. Make sure that your *LD_LIBRARY_PATH* environment variable contains the location of the libraries.

- Produce or locate a PISA ROOT file.

- Select the event number (in sequence on the DST) which you wish to display. You can use the POV package to do this for you by running over a large number of events (via *eventNumber*) and setting *Verbose* to 1 in order to have the number of reconstructed tracks in each event printed to the screen.

- Generate a Birdseye display file from the PISA file (default name: *birdseye.dat*). In order to do this, follow the following steps:

    - Edit *povGEANT.C* (in the macros directory of the package) in your running directory. This is the ROOT macro that will run the *simPOV* methods on the DST file. Set the input parameters to the desired values (including the event number and the name of the output Birdseye file). Make sure that *outputMode* is set to 1 or 2. See the web documentation for updated descriptions of the input parameters.

    - Run ROOT.

    - Type *.x povGEANT.C()* to execute the macro within ROOT. The macro will skip to the event number you have selected and then analyze that event before stopping.

    - Type .quit to get out of ROOT.

- Your Birdseye file will be ready to go. You must repeat this for each event you wish to visualize.

- Run Birdseye on the output Birdseye display file as outlined below.

5. Visualization Implementation Details

This section will describe the details of the implementation of the visualization of the hits and tracks for both the simulation and the data.

For visualization of simulated (GEANT) data, only the GEANT hits and tracks information is used. For each detector, the 3-dimensional Cartesian coordinates of each recorded GEANT hit are used in the display. Activation of the hits display for a given detector is handled through the user-controlled data members of the *simPOV* class, which can be set within the ROOT macro. Hit associations into tracks are facilitated by using the utilities in the *cge* (**C**entral Arm **G**lobal **E**valuations) package. Documentation on *cge* can be found in *PHENIX Technical Note 384* and on the web at the URL *http://www.phenix.bnl.gov/WWW/software/luxor/cge/package/index.html*. The *cge* utilities return GEANT hit-to-track relation information, which is then written to the output Birdseye display file upon user request.

For visualization of analyzed DST data, the situation is complicated by the fact that the drift chamber and the time expansion chamber (TEC) both are projective detectors and provide only 2-dimensional information for their recorded hits. For drift chamber hits, it is assumed that the z information for that hit can be estimated by the drift chamber track reconstruction. If the z information for a drift chamber hit is not available, Birdseye will not plot it, nor will it attempt to estimate it. The same applies to drift chamber tracks. If the drift chamber track does not contain z information, it is not plotted. For TEC tracks, the PC3 reconstructed clusters are queried and the TEC tracks are assigned the z-coordinate of the closest PC3 cluster to the TEC track in phi. The PC3 cluster z-coordinate must be consistent with the sign of the TEC track z coordinate to be used in the assignment. If a z-coordinate cannot be assigned, the TEC track is not plotted. For the remaining detectors, the 3-dimensional information is obtained directly from the DST quantities.

6. Obtaining, building, and launching the Birdseye Program

The Birdseye program is contained within the POV analysis package in the sub-directory *birdseye*. This contains all source code, make files, and the default PHENIX geometry file, *beGeo.dat*. To compile this on any UNIX machine, first edit the *Makefile.am* file to point to the OpenGL (or Mesa libraries for older LINUX distributions). Then type the following commands in order: "*automake –copy –add-missing*", "*autoconf*", "*./configure*", "*make*". The result will be the Birdseye executable, *birdseye* in that directory.

The Birdseye program requires that *beGeo.dat* and the Birdseye display file, *birdseye.dat*, be present in the running directory. It is recommended that soft links be made to *birdseye.dat* from the Birdseye files that you generate. To run, simply type *"./birdseye &"*.

7. Interacting with Birdseye

This section will describe some of the controls implemented in the early stages of Birdseye development.

The Birdseye display can be manipulated using only the mouse. In general, dragging with the left mouse button will rotate the display in the direction dragged. In rotation mode, doing this will impart a rate of spin to the display that is proportional to the speed of the drag. In flying mode, the rotation will stop when the drag stops. Dragging with the middle mouse button (or both the left and right simultaneously for a two-button mouse) will zoom in and out in both the rotation and flying modes. Clicking and holding the right mouse button will bring up the in-window menu.

The menu bar contains tabs labeled "*Movement*", "*Display*", and "*Quit*". The *movement* menu has the following options: *Reset view* - if you get lost, this will bring you back to the original viewpoint; *Polar Move* will reset the viewpoint and start the rotation mode where the entire display is rotated about the origin; *Flying Move* will not reset the viewpoint, but will start flying mode whereby the view will move in a straight line through the detector. Steer by using left button on the mouse. The *Display* options in the menu are to be supported in the future. The *Quit* option will end the program.

8. Birdseye in Pictures

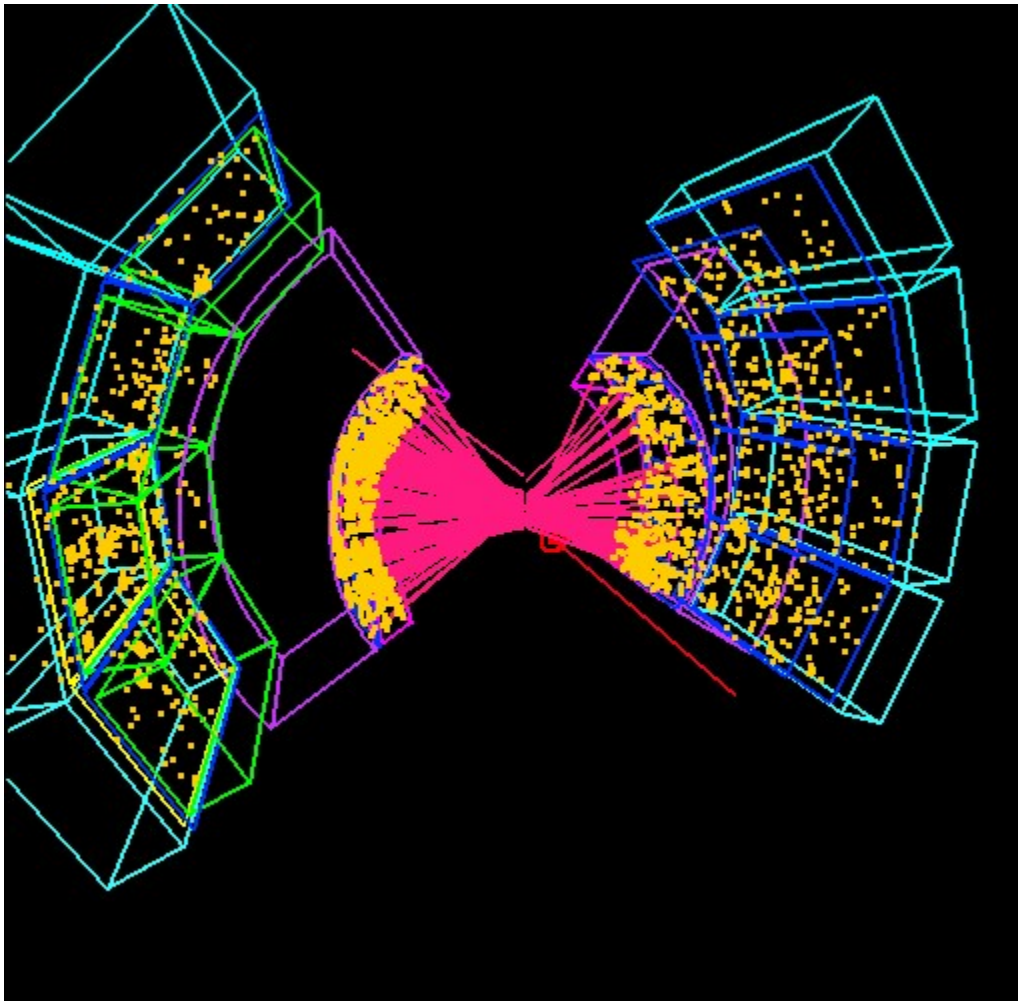

Figure 1: A Birdseye picture from an event taken during the Year 2001 run showing reconstructed drift chamber tracks from both arms. This is a B=0 run. The lines emanating from the drift chamber are straight line projections of the drift chamber tracks as calculated by the POV package. Also included are detector information from the pad chambers, time-of-flight, TEC, and calorimeters.

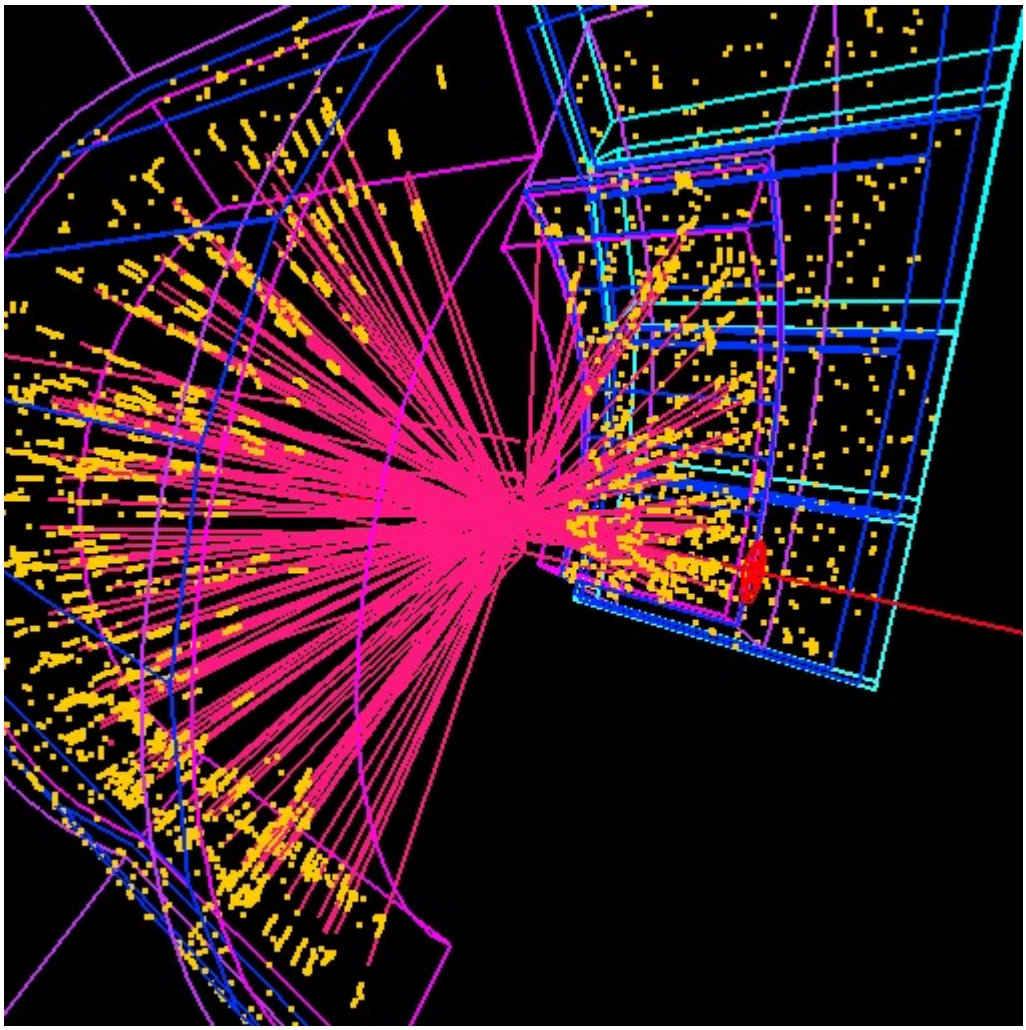

Figure 2: The same event shown in Figure 1, only viewed from a different angle and zoomed in towards the event vertex. The individual drift chamber hits are now visible.

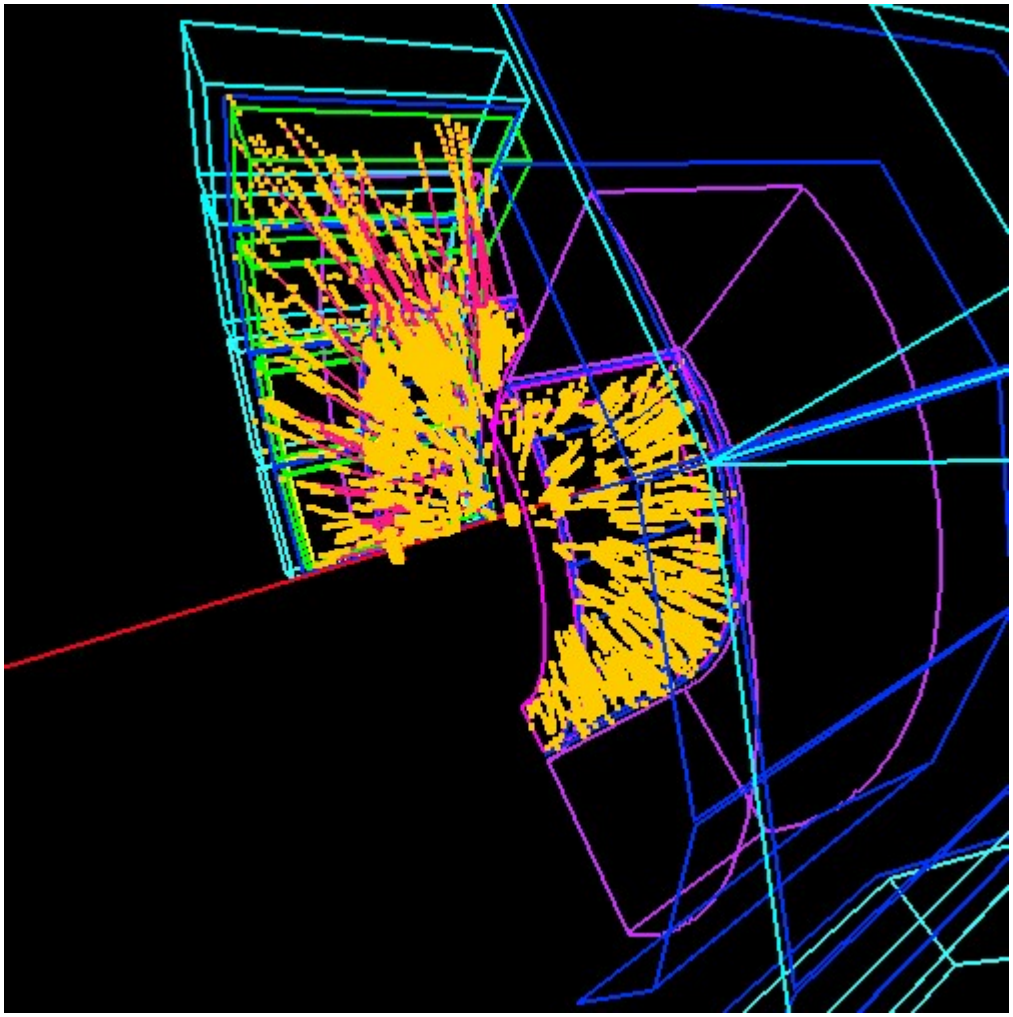

Figure 3: A simulated event with PHENIX in the Year 2000 run detector configuration. The lines emanating from the far (east) drift chamber connect drift chamber information to PC3 hits.

9. Birdseye Geometry File Specification

The Birdseye geometry (and hits, and tracks) are all defined as displayable C++ objects in the program. The geometry objects are defined from an ASCII file *(default: beGeo.dat)* that contains a description of the detector geometry. This section will describe the specification of this file.

Comments can be inserted into the geometry file by placing # or // at the beginning of a line. A geometry object is specified on several lines, always starting with a line containing the type of object and the detector it belongs to (pc1, dch, beam, etc.). The following lines specify the location and dimensions of the geometry object.

The currently supported geometry objects for Birdseye are: *cylinder, plane, box, and cylsect*.

- cylinder:

  cylinder detname
  x1 y1 z1
  x2 y2 z2
  ax ay az
  radius
  r g b t

  x1,y1,z1 are one endpoint. *x2,y2,z2* are the other endpoint. *ax,ay,az* specify the rotation angles. *radius* is the radius of the cylinder. *r,g b t* define the color and transparency of the object.

- plane:

  plane detname
  x1 y1 z1
  x2 y2 z2
  ax ay az
  r g b t

  x1,y1,z1 is the center of the plane. *x2,y2,z2* are the lengths of the sides of the plane. *ax,ay,az* specify the rotation angles. *r,g b t* define the color and transparency of the object.

- box:

  box detname
  x1 y1 z1
  x2 y2 z2
  ax ay az
  radius
  r g b t

  x1,y1,z1 is the center of the box. *x2,y2,z2* are the lengths of the sides of the box. *ax,ay,az* specify the rotation angles. *r,g b t* define the color and transparency of the object.

- cylsect:

  cylsect detname
  x1 y1 z1
  phi1 phi2 dz
  r1 r2 n
  r g b t

x1,y1,z1 is the center of the cylinder section. *phi1 and phi2* are the beginning and ending azimuthal angles of the cylinder. *dz* is the z length of the cylinder. *r1 and r2* specify the inner and outer radius of the section. *n* specifies the number of steps to use in drawing the section. *r,g b t* define the color and transparency of the object.

10. Birdseye Display File Specification

The Birdseye hits and tracks (and geometry) are all defined as displayable C++ objects in the program. The hit and track objects are defined from an ASCII file *(default: birdseye.dat)* that contains a description of each hit and track. This section will describe the specification of this file.

Comments can be inserted into the hits file by placing # or // at the beginning of a line. A hit/track object is specified on several lines, always starting with a line containing the type of object and the detector it belongs to (pc1, dch, beam, etc.). The following lines specify the location and dimensions of the geometry object.

- hit:

  hit detname
  x y z

  x,y,z are the coordinates of the hit.

- track:

  track detname
  x1 y1 z1
  x2 y2 z2

  x1,y1,z1 and *x2,y2,z2* are the endpoints of the track.