DSV - A General Visualization Tool for HEP Event Data

D.J. Prindle and T.A. Trainor

Nuclear Physics Lab. 354290
University of Washington
Seattle, WA 98195
USA

Abstract

DSV is an event visualizer and data filter package which offers a powerful means to deal with difficult imaging and diagnostics problems encountered with high-energy heavy-ion collisions. This Xlib-based general-purpose graphics tool is currently used by experiments at CERN and RHIC and has been applied to MC simulations from several popular event generators. DSV images have appeared in various news media in connection with the CERN and RHIC heavy-ion programs.

Keywords: graphics,display,event vizualization,heavy ions,images

1 Introduction

With the advent of very high multiplicity collision events at RHIC and LHC visual representation of single collision events has become very challenging. Single-event data can easily contain 10^6 separate objects. Efficient use of visual media is essential, including effective incorporation of color, motion and data filtering. Fast response, platform independence and convenient reconfiguration of data maps are highly desirable attributes. DSV is a Tcl/Motif-based event visualization package designed for very high multiplicity environments. Applications have focused on display of Monte Carlo event-generator output and TPC tracking data for heavy-ion collisions in a variety of contexts. The underlying software uses Tcl as a scripting language and GUI interface manager. This public-domain software is platform independent and insures portability of the DSV system.

DSV is designed to display data objects in an arbitrarily defined space, to color code and hide objects based on their properties and to indicate relationships among data objects visually. In addition to coloring and hiding, high object density is also accomodated by providing a sense of perspective through parallax developed by rapid rotation/zoom/pan of display objects on the canvas. Ability to explore relationships among data is greatly improved by the possibility dynamically to modify mappings between input data and graphical objects and change their appearance on the display canvas..

2 DSV

DSV (data set viewer) has several major components: a command-line interpreter (TkCon), a GUI control panel for data I/O, drag-drop mapping from input data objects to display objects and filtering data objects (Data View), and a display canvas for presentation and manipulation of display objects controlled by menu and pointing device (QCDisplay). DSV uses dynamically loaded Tcl packages for data access. Data I/O packages can be analysis specific (ascii or other specific interface format). Drag-drop interfaces are provided for simple mapping of data to display. DSV can be script driven, controlled with a pointing device via drag-drop operations and pull-down menus, Tcl command-line driven or any combination of these. Tcl scripts can be invoked within DSV for moderately complicated manipulation. This approach has advantages over C++/Java because Tcl is designed to be a command-line interface, and its simple syntax does not require extensive programming expertese. A C/C++ interface is provided for more complex manipulation of data (e.g., calculating particle trajectories). C/C++ packages can be wrapped with SWIG (http://www.swig.org) to provide a Tcl interface. Input data can also be displayed in tabular form using TkCon or with GUIs invoked by pointer actions on the canvas (picks).

2.1 TkCon: command-line interface

DSV is based on the Tcl scripting language, with Tcl/Tk as a toolkit. Configuration of DSV after launch can be carried out entirely from pulldown menus and drag-drop operations. However, dynamic interaction and modification of DSV itself can also be carried out using Tcl commands with the TkCon console. For more complex operations and space definitions predefined Tcl procedures can be invoked in TkCon, or DSV can be completely configured during launch using a Tcl script. DSV can also be launched in pre-established custom configurations from a scripting system such as Perl or from a unix shell command. This launch mode is valuable for data monitoring tasks performed by operators during production runs as an example. TkCon provides for on-line help, command recognition/continuation and multi-level error diagnostics.

2.2 Data-View: data mapping and I/O control panel

Data-View controls data I/O and mapping. Input data objects in a variety of formats, including ascii text files and XDF files, can be mapped to displayed objects in a visual space by several methods, including Tcl procedures and drag-drop operations, or combinations of these. Mapping is compeletely general, associating any combination of data objects through arbitrarily complex algebraic expressions, including vectors, to graphical objects in an arbitrary space reduced to three-dimensions and projected on the 2-D canvas. Procedures, expressions and vectors can be utilized. Procedures can be Tcl procedures or compiled C/C++ routines loaded as a shared library and used in the same manner as a Tcl procedure.

2.3 QCDisplay: viewing canvas and data manipulation

QCDisplay is the graphical display component of DSV. It draws data objects consisting of point sets and track (curve) sets on a canvas. Graphical objects of both types can be dynamically created and mapped from input data objects. QCDisplay uses Xlib, a widely available package, for drawing tasks. Xlib is a very fast drawing package which provides convenient and responsive rotation, zoom, dilation and panning. QCDisplay menus and dialogues are implemented with Motif. (which can be replaced by Lesstif on Linux platforms).

Selected objects or collections can be hidden to improve drawing speed, to reduce density of displayed objects or to isolate special objects for close examination. By direct Tcl intervation or by drag-drop one can impose cuts on graphical objects (points or tracks) based on object properties or correlations within the data system. Data objects can be selected on the QCDisplay canvas with a pick mode. If points/tracks are related by a pointer system included with the input data they will highlight together if any one element is picked. For instance, specific groups of points can be associated with correponding track fits. This feature can clarify the connectivities within the input data set.

Further QCDisplay attributes include the following:

- Flag maps: flags included with input data can be mapped to colors, and thereby used to hide data depending on predefined flag states.
- Zooms, dilates, pans and rotates: the canvas image can be controlled by pointing device to provide rapid data motion and scale changes. Zoom implies changed point-of-view in perspective mode. Dilations are changes in scale of the canvas image without POV change, always true in orthogonal draw mode.
- Picks: pointing device can be enabled to select and change the color state of a display object or a collection of objects related by pointers. It can be used to launch a table browser associated with the picked object, or attach a local size reference scale to the object.
- Views: Scale, POV and orientation of display objects may be saved and restored to/from view files, insuring exact reproduction of a previous view and the ability to relocate specific display objects in a complex event.
- Hides: display objects can be hidden depending on color attributes. Color is assigned based on corresponding data object attributes and maps. This provides a method for cuts on data attributes. Set visible alternatively permits hiding classes of objects such as point or track types, or a wireframe or set of axes. Clip planes can hide all display objects on one side of a clip plane defined on the projection space.
- Configurations control panel: Detachable GUI contains hidden controls for several QCDisplay features, including POV, origin, axes labels, independent axis scales, point size, pick mode and draw mode.
- Draw mode: The canvas can be changed from perpective (adjustable point of view) to orthogonal ('infinite-distance' point of view) draw mode.
- Color editor: color palettes can be dynamically defined and palette definition files saved. Provision is made for nine color values each for two point and two track object types plus canvas background, wireframe and axes. Data attributes can then be mapped to colors.
- Table browsers: Table browsers can be launched either from the TkCon window or by pick mode from the QCDisplay canvas. Data tables containing selected input data objects are displayed with full scroll capability and the ability to follow pointers by launch of further table browsers.
- Save/restore: QCDisplay can save and restore color palettes, viewpoints, mappings and overall DSV state to/from files with corresponding file extensions.. In addition, use of a launch script provides for reproducible restoration of a DSV state for repeated analysis sessions. The QCDisplay image state can easily be saved to postscript output file.

3 Applications

Two DSV applications are presented, an investigation of MC event-generator events for RHIC Au-Au collisions and an investigation of TPC tracking anomalies.

3.1 Monte Carlo events

In Fig. 1 is shown the configuration space freezout distribution for a single Au-Au collision event at $\sqrt{s} = 200\ GeV/u$ as simulated by NEXUS [1]. The view is along the symmetry

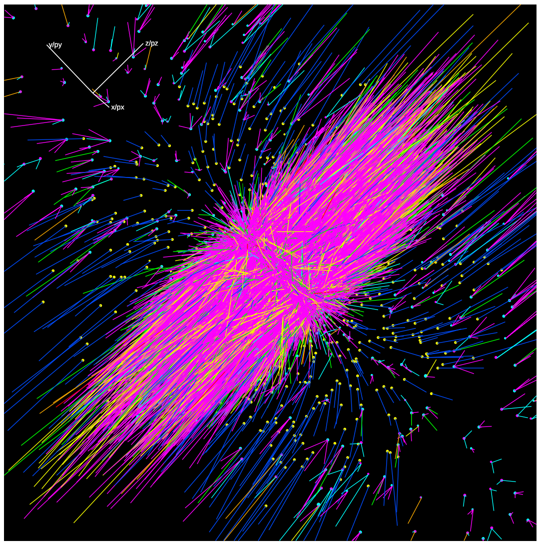

Figure 1: Configuration-space rendering of Au-Au collision simulated by Nexus MC program [1]. Use of 'rapidities' as coordinates and vector components permits very compact visualization.

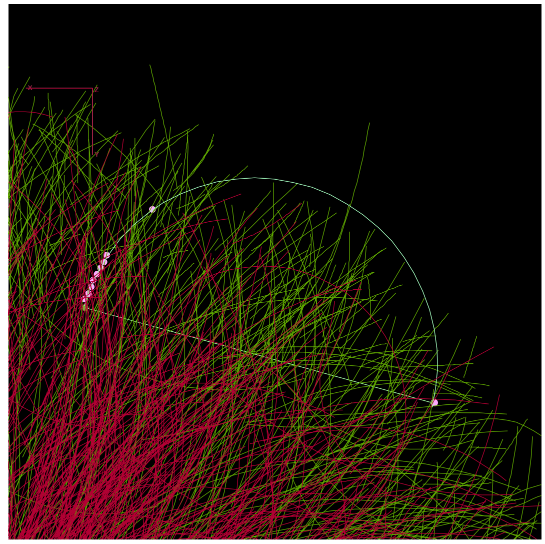

Figure 2: Example tracking diagnostic study showing a specific tracking error located in the high-density tracking environment. Visual inspection quickly indicates the source of difficulty in the tracking algorithm.(STAR collaboration)

plane of the collision. Attached to each point is a corresponding momentum vector. Point and vector colors indicate particle type – baryons, mesons of various kinds, with or without strangeness or charm, leptons, etc. The definition of the coordinates provides for very compact visualization. All coordinates are defined as 'rapidities.' That is, if x is a linear coordinate and x_o is a characteristic size for that coordinate then $Y_x \equiv log\{\sqrt{1+(x/x_0)^2}+x/x_o\}$ is the 'rapidity' for that coordinate. This definition is consistent with the standard rapidity defined for longitudinal momentum p_l with $x_0 \rightarrow m_t$. A similar treatement for all configuration and momentum space coordinates or vector components is illustrated in Fig. 1.

3.2 TPC tracking diagnostics

In Fig. 2 is shown a small fraction of a RHIC Au-Au event (VENUS simulation). A cut system was formed on momentum and other track properties which identified a small anomalous track class, of which one instance is highlighted in this figure. The reason for the tracking anomaly is immediately identified using DSV as inclusion of a random incorrect endpoint far from the end of a track, producing displacement of this track class to smaller p_t. Here a few 10s of objects out of 8000 were the focus of the study. With visual identification of the source of the momentum-space distortion the tracking algorithm can easily be corrected.

References

1 K. Werner et al., University of Nantes, private communication