AIAA

A Virtual Testing Environment for Electric Propulsion

L.Brieda, J. Pierru, R. Stillwater, J. Wang
Computational Advanced Propulsion Laboratory
Virginia Polytechnic Institute and State University
Blacksburg, VA 24060-0203

39th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit
July 20-23, 2003/Huntsville, AL

A Virtual Testing Environment for Electric Propulsion

L.Brieda*, J. Pierru†, R. Stillwater‡, J. Wang§
Computational Advanced Propulsion Laboratory
Virginia Polytechnic Institute and State University
Blacksburg, VA 24060-0203

Electric Propulsion provides an appealing alternative to chemical boosters for long duration missions and for station keeping purposes due to its high Isp and low propellant consumption. However, electric thrusters lack sufficient flight heritage, resulting in a costly and time-consuming experimental testing of new concepts. Currently 3D computer particle simulations are increasingly being used in the design and testing of electric propulsion devices. Various simulation models exist (Particle-In-Cell, Direct Simulation Monte Carlo, etc...), but all tend to produce large three-dimensional datasets. The development of the CAPLab Virtual Testing Environment (capVTE) was initiated by the need to effectively visualize and interpret such results. The software creates a virtual environment, through which the user can move and investigate the results from various angles, as well as study the time dependent behavior. A full integration with the CAVE immersed environment will produce a virtual vacuum tank, allowing multiple interested parties to walk through the 3D dataset, witness the flow of the particles in real time and obtain a better understanding of flow properties by defining isosurfaces and cutting planes through the computational domain. The software is currently used to visualize results from the Deep Space 1 NSTAR ion thruster plume and NSTAR ion optics.

Introduction

The flight of NASA's Deep Space 1 mission, which relied on an NSTAR ion engine for its main source of thrust, demonstrated that electric propulsion is a feasible alternative to chemical boosters. However, this technology lacks the flight heritage of chemical rockets, and extensive testing is required before it can reach a more widespread application. Experimental testing is however very inefficient, as it requires both the manufacture of the model, as well as the lengthy run in a vacuum chamber. Hence, large scale 3D computer plasma simulations are being used more frequently in the design and testing stages of the thruster development. However, such simulations usually create a large set of three dimensional data that is difficult to comprehend without the help of powerful visualization tools.

This paper outlines the visualization tools being employed by the Virginia Tech's Computational Advanced Propulsion (CAP) Laboratory. CAP Lab specializes in computer simulation of ion thrusters, and the visual processing of the respective data. Visualization is performed on a wide variety of hardware platforms, ranging from high-end PCs, through SUN and SGI machines to the 3D projection system called the CAVE. The bulk of this paper, however, concentrates on the software side of the visualization process. Two visualization packages are described. First of these, Diverse, is a scalable framework which can handle rendering and data input on a wide variety of hardware platforms, thanks to the dynamically loaded shared objects. Its data processing capabilities are very limited, which led to the development of capVTE, CAP Lab Virtual Testing Environment. This program as well as two examples demonstrating its use are described here in a greater detail.

Visualization Tools

CAVE

The CAVE is an immersed, walk-in 3D visualization tool. From the physical perspective, it is a "cube" missing the front and the top face, with a 73.5x84 inch footprint and a 104-inch high ceiling. The three side faces along with the floor act as screens, on which an image is projected from four Electrohome Marquis 8000 projectors. The false-color images are shifted such that, when viewed using the CrystalEyes goggles, they create a perception of the visualized object floating in the center of the cube.

The CAVE can accommodate several people at one time, however, only one user wears the tracking headset. Its position and view angle is tracked by an array of sensors. The central computer then adjusts the projections to create a realistic image. Further control can be accomplished using a hand-held wand. Its translation as well as rotation are both tracked, and thus the

*Undergraduate Research Assistant; Undergraduate AIAA student member

†Undergraduate Research Assistant; Undergraduate AIAA student member

‡Graduate Research Assistant; Graduate AIAA student member

§Associate AIAA Fellow

wand can act as a virtual pointer. A small joystick located on the wand allows the user to travel greater distances that allowed by the limited physical size of the CAVE.

Diverse

Diverse[1] (Device Independent Virtual Environment Reconfigurable Scalable Extensible) is a collection of software, Application Programming Interfaces (APIs) developed at Virginia Tech by the University Visualization and Animation Group. It is written in C++ and runs on GNU/Linux and SGI IRIX systems. Diverse is a free software and is distributed under the licenses LGPL for the libraries and GPL for the software. Diverse can display programs in the CAVE, Immersa Desk, Head Mounted Display (HMD) and desktop or a laptop. It consists of three major components:

- Diverse graphics interface for Performer or DPF, uses OpenGL Performer to implement 3D Virtual Environment applications.
- Diverse GL or DGL, supports rendering with the OpenGL programming interface.
- Diverse ToolKit or DTK, allows DPF and DGL to access local and networked interaction devices.

The flexibility of Diverse comes from its use of dynamically-loaded shared objects(DSO's). The Diverse program provides just the general framework, onto which additional objects can be loaded as needed. Several DSO's exist, but these are mostly helper applications, adding features such as a wand flashlight to the CAVE. Diverse can display 3D objects defined as a polygonal structure, but it does not provide any support for data processing. Although it would be possible to write a DSO managing data processing, a simpler solution was writing a visualization application from scratch, but utilizing an already existing data processing library. This led to the creation of capVTE, which is described in a greater detail in the following section.

CapVTE

The primary data output produced by a computer plasma simulation is often the distribution of some variable, such as ϕ or ρ, through the domain. Such an output is usually incomprehensible in its raw form, and must be processed using some visual tools into a more human-friendly format. The large variety of available hardware resources at Virginia Tech inevitably led to a great variety of available supporting software. The software would commonly be available only on one platform, and each piece would be able to perform only a specific task. As such, Amtec's Tecplot was used on Windows based PC's. PW Wave, running under

[1]Available at http://diverse.sourceforge.net

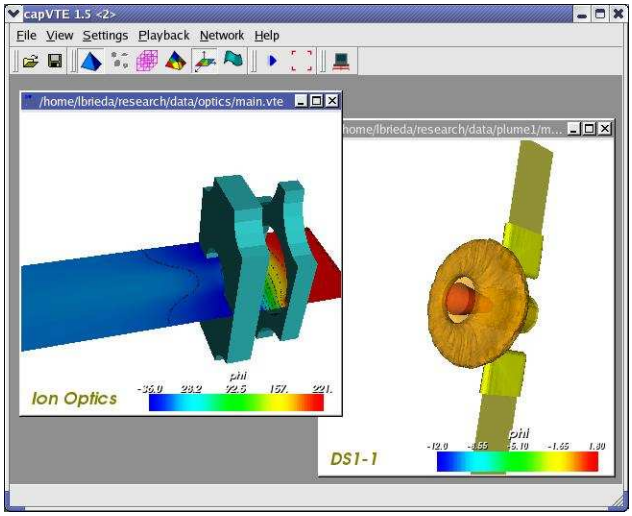

Fig. 1 capVTE user interface, showing two loaded datasets

SUN Solaris would generate HDF files, which could then be loaded into Diverse, running in the CAVE. Slicer/Dicer, running on Apple Macintosh, could also generate HDF files, as well as QuickTime movies of the cutting plane moving through the dataset. On-line collaboration was possible by loading Inventor files into the CAVE Collaborative Toolkit.

The large variety of visualization tools is of course very time inefficient. If, during the on-line collaboration, a certain party was interested in seeing a different aspect of the dataset, the original data had to be loaded into the appropriate software, the requested isosurface or cutting plane had to be extracted, the output had to be saved in a file and finally loaded back into the collaboration console. Resources were wasted as well, since due to either licensing issues, or the mere lack of support of a particular package for a certain platform, we could not always utilize of the fastest machines available.

Furthermore, we were interested in visualizing all the aspects of plasma simulation, and not be limited to displaying just the grid-contained scalar values. Plasma simulation solves the flow of particles around a particular geometry, and we needed a software which could easily visualize the geometry as well as the particle flow.

Therefore, the CAP Lab Virtual Testing Environment (capVTE) project was initiated. The aim was to create a single, easy to use piece of software capable of visualizing the results generated by a CFD or a PIC code. As such, the program had to be able to display the geometry, grid data and particles. The process had to be real-time, allowing the user to "immerse" himself into the data. The "extract plane, set camera view, render, wait" procedure common to many other visualization tools was not acceptable. The program had to be able to process multiple data frames, and play them back to display time-dependent trends.

Sharing of results had to be as simple as possible, with a fully integrated network support. Lastly, it was desired for the source code to be platform independent, resulting in a fast release to any required platform.

CapVTE, shown in Figure 1, is a multi-window, GUI application. The user navigates through each dataset in real-time using the mouse. Menus and toolbars are used to select the active tools and to change their properties. The data input consists of a single ASCII file which contains the appropriate header, followed by program commands and data blocks. However, the *INSERT* command allows for the dataset to span over several physical files, in a fashion similar to that of C/C++'s *#include* directive. As such, the main input file can be a just a short collection of *INSERT* statements pointing at a geometry file created from an output of a 3D modeling application, and the grid and particle files, generated by the PIC code.

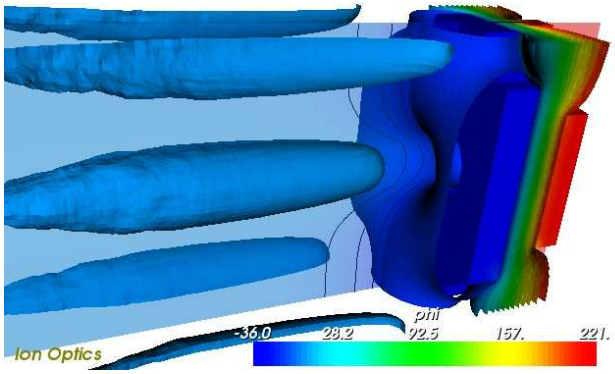

Fig. 2 View of a cutting plane, isosurfaces and the geometry scalar gradient

The geometry is specified by its nodes and their respective connectivity. Each node can be individually colored, resulting in a more attractive presentation. Support for transparency is also included, making it possible to generate windows and other see-through sections. The geometry serves two roles. First, the addition of geometry simplifies the task of visually deducing results from the data set. Changes in the contour lines on a cutting plane, for instance, can be more easily translated into the corresponding effect by observing their relationship to the surface geometry. Secondly, capVTE supports a real-time scalar gradient mode. In this mode, the user specified coloring of the geometry is replaced by color-values extrapolated from the surrounding grid nodes. As such, every surface of the geometry acts as an individual cutting planes.

The spatial distribution of one or more variables is described by the grid block. The Cartesian mesh on which the data is defined can have a variable spacing and a variable number of nodes. The number of scalar components is also limited only by the available memory. The distribution of the active component can

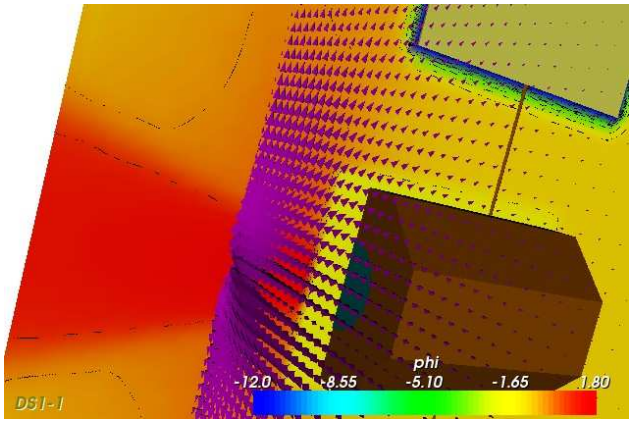

Fig. 3 View of glyphs, a cutting plane and an object geometry

be studied using cutting planes and isosurfaces, shown in Figure 2. The user can specify the minimum and maximum extent of the data contained in each isosurface, as well as the number of contour levels and their respective values. In conjunction with the geometry block, the grid data can be used to generate a scalar gradient along the geometry surface.

The glyphs block specifies small graphical objects that can be placed anywhere in the domain. The geometry representing each glyph can be one of the many predefined types, ranging from point clouds through wedges and cubes to cones and spheres of varying resolution. Each glyph can be scaled and oriented according to its vector data. The number of glyphs can vary from frame to frame. As such, glyphs have many uses, among which is the representation of the macroparticles used in the PIC simulation, or addition of vectors onto the grid nodes. An example of the use of glyphs and a cutting plane is given in Figure 3.

The grid and glyphs are loaded as a collection of frames. In the playback mode, the program automatically advances the current frames. Cutting planes and isosurfaces are automatically updated and glyphs are moved to their new position. The playback rate and the level of detail available is of course limited by the graphical capabilities of the used computer, however, modern, off-the-shelf PCs are more than capable of sustaining playback of several frames per second, with cutting planes and isosurfaces generated on a grid with several thousand nodes without compromising the real-time navigation. The record mode exports a bitmap image at a chosen interval. These pictures can then be converted into a movie using software such as Adobe Premier or Flash MX.

capVTE allows a data export in three formats. First, static snapshots can be saved as JPEG images. Second, the VRML (Virtual Reality Markup Language) is a commonly used method to capture 3D objects. Unlike the JPEG format, VRML captures the polygons making up the currently seen isosurfaces, cutting planes, glyphs, geometry, and so forth. VRML

is commonly used to publish 3D data on the Internet, and plugins are available for the popular browsers. Lastly, the Inventor format also captures the three-dimensional structure, but is compatible with Diverse, software used to run the CAVE.

The graphic engine used by capVTE is the open-source Visualization Toolkit (VTK) library, produced by Kitware. Not only has the use of this package drastically reduced the development time, VTK is also platform independent. It interacts with the OpenGL package, and versions of VTK are available for Windows, UNIX/Linux and System X. Similarly, the GUI is created using Trolltech's QT. QT is also platform independent and takes care of the many headaches associated with a cross-platform development, such as interaction with windows, user input, and the creation of timers. Therefore, deployment of capVTE for a particular platform involves only linking the source code with the VTK and QT libraries made for that specific platform.

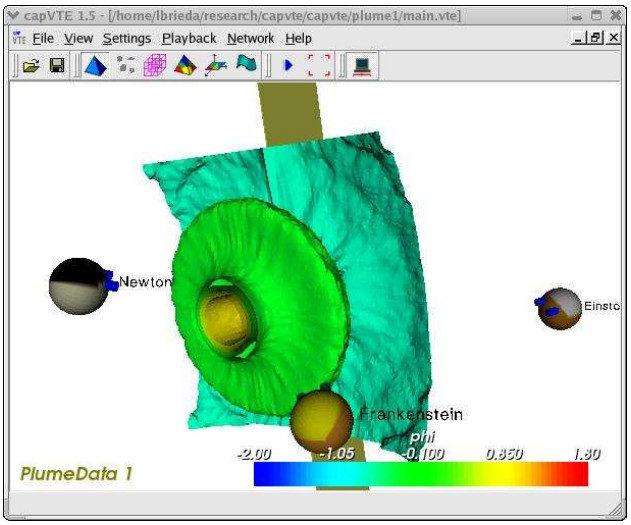

Fig. 4 Network collaboration showing three connected clients

The real-time data immersion offered by capVTE is an important tool in the study of large 3D data sets. However these days scientific projects are lead by groups composed of universities and research laboratories throughout the world. The network module, based on a client/server, TCP/IP architecture, simplifies the sharing and studying of the data across multiple labs. The user specifies whether a new server should be established, or whether the program should connect as a client to an already running collaboration. Before connecting, each user specifies an "avatar", a small graphical object, which is located and oriented according to that user's viewport. Thus, every member of the online collaboration can see what part of the dataset their colleagues are looking at, as illustrated in Figure 4. The network is also partially synchronized. When one client toggles cutting planes, for instance, cutting planes will be toggled for the other clients as well. Thus, all members share the same virtual space, with the only difference being the position from which the view is made. Some synchronization problems can occur, however, if the hardware performance of the used machines varies widely. This problem will be corrected shortly, as described in the *Future Work* section.

Examples

The Deep Space 1 Spacecraft

One of the first applications of capVTE was the visualization of CEX expansion around the DS1 satellite. The dataset was generated by Dr. Joseph Wang and is further described in XXXXX. The data consisted of the steady state potential and charge density distribution throughout the domain, as well as macro-particle velocity averaged onto the grid nodes taken at several time steps. The spacecraft model was symmetric on both the x and y axis and thus only one quarter of the satellite was analyzed. It was embedded in a 43x43x71 grid, averaging approximately 4000 macro-particles per cell.

The simulation results for the quarter-sized satellite were first replicated accordingly to the other three quarters. We were interested in displaying the results in conjunction with the satellite geometry. Since the simulation was performed using an analytically defined object, we had to generate the surface mesh matching the analytical description. The computational domain captured only the portion of the solar panels close to the spacecraft body, however, they were extended for visualization purposes. This extension is easily visible in Figure 5, as the sheath around the solar panel comes to an abrupt stop at the grid boundary.

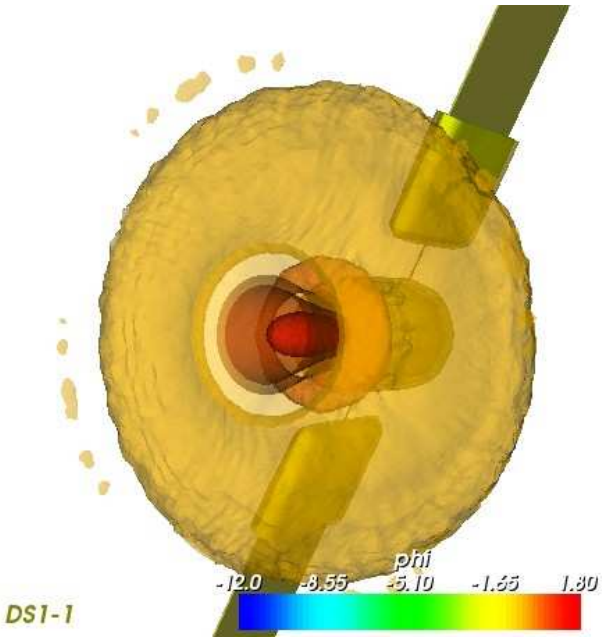

Fig. 5 DS1 ϕ isosurfaces

Figure 5 shows the visualized product. The number of isosurface levels was set at 25, and they were

generated at evenly spaced intervals. Each isosurface is shown at only a 50 % opacity, making the region of high potential near the thruster exit easily visible. A more simplified view of the plume expansion can be obtained using the cutting plane tool (see figure). However, we have found that the three-dimensional characteristic of isosurfaces aid greatly in the visual study of the results. Unlike cutting planes, which limit the result to a two dimensional space, the isosurface tool connects equal scalar values through the entire domain. It is possible to miss an important feature of the result if one doesn't choose a cutting plane passing through that particular region. The possibility of this happening with isosurfaces, although existing, is much smaller.

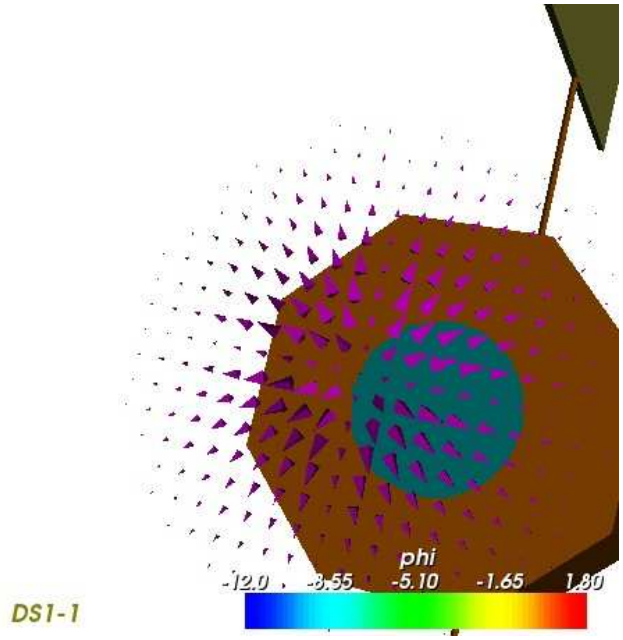

Fig. 6 Averaged flow velocity, early stage

Visualizing the actual flow of macro-particles would be a daunting task, since approximately 5 million particles were present in the simulation at a steady rate REF XXXX. Although a small subset could be chosen, a more accurate description of flow can be achieved by averaging the flow velocity onto the grid nodes. Scaled and oriented glyphs can then be used to indicate the flow direction and magnitude. Further, only a single plane of the grid was chosen to reduce the visual clutter of the resulting animation. The chosen plane corresponded to z=55, or 11 computational cells downstream from the thruster exit. This section roughly corresponds to the region of maximum plume expansion. Several frames of the resulting animation are shown in figures 6, 7, 8. Figure 6 was taken shortly after the particle injection process began, while the flow velocity in figure 8 is approaching the steady state. The flow velocity is the highest close to the thruster exit, as can be seen from the larger size of the glyphs. The radial expansion of the plume is visible from the outward orientation of the glyphs.

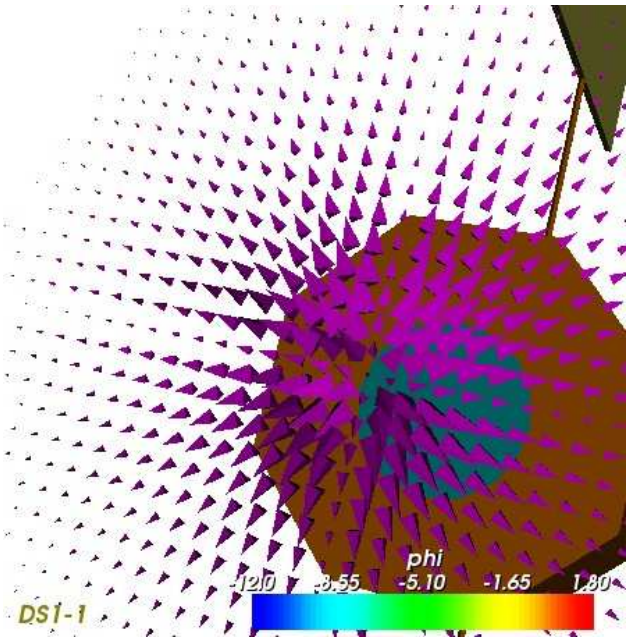

Fig. 7 Averaged flow velocity, medium stage

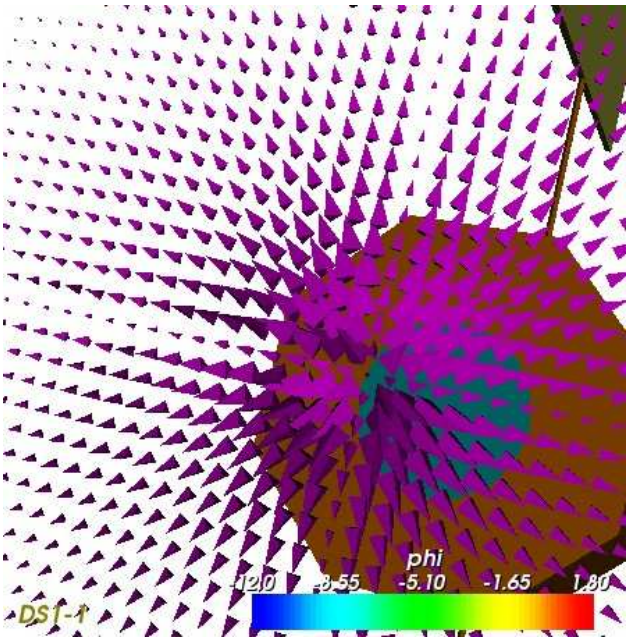

Fig. 8 Averaged flow velocity, steady-state

A full three dimensional data immersion was accomplished by exporting the geometry/isosurface combination of Figure 5 to an Inventor file. The Inventor file was the loaded into the CAVE using Diverse. The plume expansion data is a great subject for a CAVE study, as it is possible to walk in between the isosurfaces of the ϕ distribution. The full 3D immersion results in a wider range of visible detail than can be perceived through a 2D computer monitor.

The DS1 Ion Optics

The ion optics grid is susceptible to erosion by impingement of charge exchange ions. An effort is being made to improve the grid design, predict thruster service life and understand failure modes using computer simulation.

The particle flow around the DS1 NSTAR ion optics was simulated using three PIC codes developed by Dr. Joseph Wang. The first code handled the motion of the ion beamlets, the second the neutrals, and the third handled the charge exchange ions. The simulation geometry was chosen such as to represent the minimum domain accounting for all the 3D effects of the hexagonally laid out grid, and can be seen as one quarter of the geometry seen in figure 9. No assumptions were made about the state up or downstream of the optics. As such, the computational was chosen large enough to fully resolve the beamlet neutralization and backflow of the charge exchange ions. The results presented in figure 9 were generated using a 30x30x400 grid, with about 1.8 million macro-particles used in the beamlet simulation and 9 million particles used in the charge-exchange ion simulation. The results were found to agree with measurements from a long duration test at JPL.

Figures 2 and 9 show two different views of the data set with different types of visual tools. Figure 2 has almost all of the visual features of capVTE toggled on. The number of isosurfaces was set to 25 but the data displayed were shifted by omitting the part of the data before the screening grid so that the data downstream could be seen. Indeed the values for the electrostatic potential upstream of the screening grid are so large that when capVTE displays evenly spaced isosurfaces, not enough isosurfaces render the complex structures occurring at much lower potentials. The shape of those isosurfaces describes the pattern of the flow of ions both through and at the exit of the grids. The scalar gradient was applied to the surface, thus the two grids have two different colors, the screening grid is red to indicate the high electrostatic potential on it while it is blue on the accelerating grid for it has a low relative electrostatic potential. Finally a plane is set up along the flow cutting the grids orthogonally to the y-axis and passing through the center of the geometry. The plane has a 50% opacity and shows contours lines of the electrostatic potential downstream of the grids. Figure 2 is a much lighter version of figure 2 showing the same data set except that this time the isosurfaces have a 50% opacity. Using a low opacity for the isosurfaces help at seeing the geometry that those isosurfaces wrap, however it slows down the rendering.

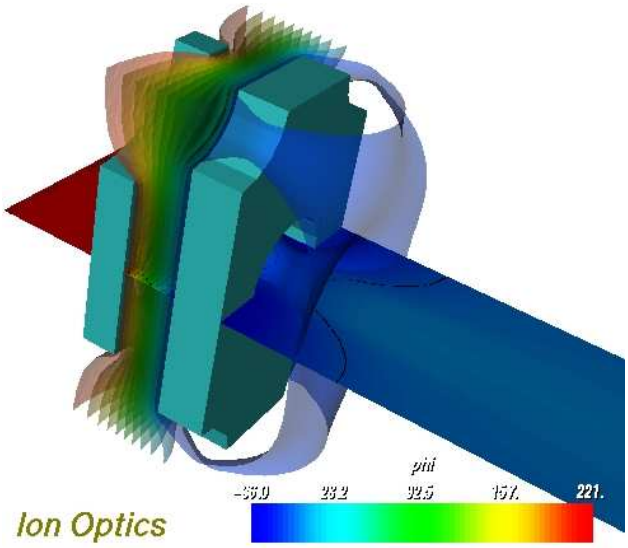

Fig. 9 Deep Space 1 Ion Optics.

Future Work

Although the desktop version of capVTE contains most of the desired features, several issues still remain to be resolved. First of all, the program requires a very extensive testing to verify its stability in the cross-platform environment. Secondly, the network interface, which is the newest addition to the project, needs to be optimized. Currently, the central server relays the messages pertaining to the state of each client as received. This results in a large number of network traffic, and high number of screen redraws. Instead, the server should collect the state information, and communicate it at a preset interval in a single network packet. This will also improve the overall synchronization of the clients. Further, the network communication relies on open sockets. Open sockets can be potential security holes, and could prevent clients behind a strong firewall from connecting to other capVTE users. Thus, the possibility of implementing a secured connection using the OpenSSL package is being investigated. Dialog window will be added allowing users to send text messages across the network, as well to "jump" into somebody else's head, and thus share the view of the dataset with their collegeau.

After the work on the desktop version of capVTE is finalized, the work will begin on porting the code to the CAVE. Successful completion of this project will result in an unsurpassed visualization experience. The user will be able to physically walk through the data. The CAVE will serve as a physical vacuum tank. The user will be able to see the thruster geometry, with the macro particles being emitted at real time. Generation of cutting planes will be as simple as waving the wand along the direction of the cut. Of course, porting the code to the CAVE presents some challenges, most of these coming from the different architecture between the PC, containing a single graphics pipe, and the four-

pipe CAVE. Thus, a version of VTK for the CAVE is not directly available. However, several packages exist which interface, at varying levels of success, VTK's calls to OpenGL, to the CAVE graphic libraries.

Conclusion

As the field of electric propulsion advances, the possible applications of such a propulsion system grows both in number and in feasibility. Due to the fact that electric propulsion is still a relatively new field many more extensive ground-based and space-based tests are required before electric propulsion becomes a staple of the spacecraft propulsion industry. The price tag for such a test can be sizeable, which is why computer simulations are such an attractive prospect. A computer simulation can enhance the knowledge about the system and help guide the research and development in the right direction at a much lower cost. In order to fully comprehend the results of the simulation it is often necessary to use a visualization software package. CapVTE is a user-friendly package that requires minimal training to be able to use. This virtual environment program allows the user more functionality than many currently available visualization packages and the networking capability allows colleagues to share their research much more easily. Computer simulations are becoming more common in many fields, and like electric propulsion they will need visualization software. CapVTE can be a valuable tool in increasing the knowledge gained through simulation research.

References

1. Arsenault, L.E. et al., "DIVERSE: a Software Toolkit to Integrate Distributed Simulations and Heterogeneous Virtual Environments", Joint Aerospace Weapon Systems: Support, Sensors, and Simulations Symposium Exhibition, San Diego, CA, July 22-27, 2001

2. Brieda, L. and Pierru, J., "Development of a Virtual Testing Environment and Applications to Spacecraft Electric Propulsion", AIAA Regional Student Conference, University of Maryland, MD, 2003

3. Virginia Tech Cave. "What is the Cave?". 1998.<http://www.sv.vt.edu/future/vt-cave/whatis/> (June 25, 2003)

4. Wang, J., et al., "Three Dimensional Particle Simulations of Ion Optics Plasma Flow and Grid Erosion", accepted for publication in *J. Propulsion & Power, 2003*

5. Wang, J., et al., "Deep Space 1 Investigation of Ion Propulsion Plasma Environment", *J. Spacecraft & Rockets, 37(5), pp545*