Visualization Products On-Demand Through the Web

Suzana Djurcilov and Alex Pang
University of California, Santa Cruz

Abstract

We describe our efforts in providing REINAS (Realtime Environmental Information Network and Analysis System) weather visualization products on demand through the web. As the REINAS research enterprise transition into operational mode, there is increasing demand for wide and effective distribution of results. To meet this demand, we provide weather products on the web as both images and VRML worlds. The viewer selects geographic extents and time range, as well as specific weather products. Data is then pulled out of the database based on the user's specifications and fed to the visualization tools to produce the image or VRML file. This paper describes the architecture of this end-to-end system for delivering sensor and forecast data through the database and to the public as VRML-based visualization products.

Key words and phrases: VRML, HDF, database, environmental visualization, weather, data fusion, data assimilation.

1 INTRODUCTION

As the Internet becomes commonplace, there is a greater demand for access to scientific information, in a readily accepted multimedia format such as VRML (Virtual Reality Modeling Language). Scientists and programmers trying to make their results and tools available on the web often face a tremendous difficulty in bridging the gap between large amounts a varied scientific data and the visualization packages requiring rigid input formats. Environmental data, for example, is diverse in terms of origin and formating and is produced at a high rate. A lay user would require extensive training before he or she is able to formulate a proper request for a specific section of a dataset, and then visualize it. Therefore the task of sifting out the required data is put in the hands of the web-page creator. The final outcome is usually a pre-packaged web product, where the data is collected beforehand and an image is produced at regular intervals. The end user is thus stripped of the privilege of viewing the latest available data, or interacting with the dataset; instead they face the choice of several predetermined outputs.

We propose an integrated visualization system, where the basic selection of the data is done by the end-user. We accomplish this by integrating our visualization system with a database containing environmental data collected in the Monterey Bay for the last four years.

The highlights of the visualization system that we describe in this paper are: its integration with a database front end that enables data to be queried and retrieved using a user-driven, web interface; the ability to specify both the geographical and time parameters of the data requested; the ability to query and display realtime data as soon as they become available in the database; and the generation of visualization product on demand through the web.

In the next sections, we discuss the background work and our motivation for creating a new visualization system and placing it on the World-Wide Web, followed by an overview of the REINAS project in order to provide a context for the design considerations that were made; next we have the architecture and major components of the our system; followed by a description of the individual tools and elements involved in creating the VRML output; a short discussion on the desired features in VRML and a status and summary section.

2 BACKGROUND AND MOTIVATION

The traditional means of distributing hand drawn weather products have been painstakingly slow. As computing power and communication bandwidth increased, the turnaround time for generating weather products have been shortened. For example, the National Weather Service (NWS) provides hourly updates of wind readings around the San Francisco peninsula [8].

With the growth of Internet we are starting to see more applications interactively available over the Web, such as WXP [10] - "The Weather Processor". Nowadays we can "monitor" the environment with instantaneous readings from a pre-specified view and a number of sensors through the web [4]. Other tools, like Vis-a-Web [7], even provide visualization service on the net. Furthermore, [12] describes a scenario where the web publisher mounts the raw data on the Web, and the viewer accesses this data through a modular visualization environment such as the IRIS Explorer. Our goal is to combine the power of the database, the web, and VRML to provide sophisticated 3D weather products on-demand that the users can interact with.

To accomplish that one needs visualization software that can map the data returned from the database into an interactive image or a VRML file. Through the years, universities and research labs have created a number of tools to address the visualization needs of different scientific communities. For example, Vis5D [3] and GEMPAK [1] are geared towards 3D meteorological data and provide almost all standard imaging options, while VTK [6] is an all-around visualization toolkit. In addition, powerful general-purpose commercial visualization packages such as Data Explorer (DX) and

AVS can certainly serve the purpose of making visualizations out of scientific data.

While these systems focus on or can be tailored to the environmental visualization arena, they do not address all of our project requirements, which are:

- Integrate data from geographically dispersed sensors showing both observed values and interpolated fields. These include sparsely distributed in-situ sensors, remote sensors, as well as numerical model simulations.
- Support sensors that collect different types of data, stored in different formats, and of different temporal density and spatial dimensionality.
- Interactively select the region, time range and type of data of interest.
- Access to data in realtime as they are collected, as well as access through files for backward compatibility.

In addition, we had commitments to our users to provide a visualization environment that can monitor current conditions, but also provide a retrospective analysis of past data; tools that can display forecast products and experimental results; and last, but not least, we wanted to service the general public by providing web products on demand. To accomplish this we had to resort to a new paradigm: a data-independent system which would allow file-loading, but primarily rely on interactive requests for specific data from a vast database.

Our work in creating this model has been preceded by several important examples of linking data inquiry and retrieval with VRML-based visualizations. As an example, Elvins and Jain [2] present a system for choosing a volumetric dataset, and creating isosurfaces "on-the-fly". Our project follows this line of work, but concentrates on extracting data with very precise requirements and produces a variety of visualizations.

3 REINAS OVERVIEW

The REINAS system is now an operational regional laboratory for research in meso-scale meteorology and oceanography. It is developed through a multi-year effort by the Baskin Center for Computer Engineering and Computer Science of the University of California, Santa Cruz (UCSC), in cooperation with environmental scientists from the Naval Postgraduate School (NPS), and Monterey Bay Aquarium Research Institute (MBARI). REINAS provides access to a variety of measurements, in real-time and retrospectively, for the Monterey Bay region of the California Coast. Data from REINAS supports research in modeling and forecasting, as well as nowcasting that is accessed by an large number of users over the Internet / World Wide Web interface. Advance visualization tools provide the research users with support of examining data in multi-dimensions, and for combining the results of measurements and models.

Online information about REINAS, including technical reports, are available from http://www.cse.ucsc.edu/research/reinas.

The goal of this project is to bring modern technologies to bear upon the problem of realtime environmental (oceanographic and meteorological) data acquisition, management, and visualization/analysis. As a focus area, we are looking at the phenomena within the regional scale of Monterey Bay, California. Since physical changes happen at much smaller time scales within our region of interest (compared to global or climate studies), the main challenge of REINAS is to provide realtime environmental information of the physical parameters.

Figure 1 shows a high-level view of the system and its three main components: data acquisition, data management, and data visualization.

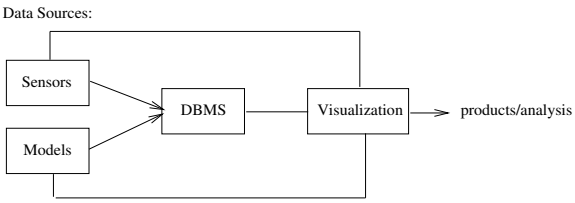

Figure 1: Major REINAS components.

Data is acquired from a variety of sensors, some of which are on buoys, CODARS (radar based ocean current measurements), wind profilers (measuring wind at different elevations), radiosondes, but most commonly from meteorological stations. These data are tagged and stored in our database, one datatable per sensor. Each table contains fields specifying the time and location of the measurement, along with the sensor-specific data output. This design is matched by the datastructures used in the database APIs.

3.1 Database API

We have two primary application programming interfaces (APIs) for allowing REINAS applications a uniform access to our database. These are the RSObject, which provides an interface to the full featured schema at a higher level; and the REINAS low level API, which provides a mechanism to pass SQL queries to the database engine directly, or call up predefined statements which encapsulate the most popular SQL queries. For all but the simplest queries we rely upon RSObject to provide the type checking, security and data organization necessary for effective gathering of specific parameters from the database.

The RSObject API allows the user to specify the parameter type and the temporal and geographic range of the desired data as part of an RSObject structure. If known, a specific instrument or sensor name can also be requested. A simple *RSGet* call then populates the structure with the requested measurements.

4 VISUALIZATION ARCHITECTURE

We now describe how the data returned through the RSObject API are rendered by the REINAS visualization system called PET SLUG [5]. PET is an acronym for Products, Elements, and Tools which reflects the modular architecture of the visualization system. It was originally developed using OpenGL, C++, and xforms [13] for improved portability. PET SLUG has since been extended to produce VRML output files as well.

It uses a tool-based approach where users activate one or more visualization tools to visualize their data. Figure 2 shows the graphical interface of PET SLUG.

PET SLUG is designed to be readily extended by users with varying requirements and levels of expertise. As such, it is organized into a three level hierarchy: products, tools, and elements. Products are collections of tools with specific preset parameters and data (see Figure 3). In this way, a first time or casual user can simply open a product file and see a meaningful visualization. An example might be a temperature isosurface that is pseudo-colored with pressure values over the Monterey Bay. The more sophisticated user, on the other hand, is more likely to use the tools directly to customize their visualization. Typically, a tool is used to implement a common visualization method such as isosurfaces, contour lines, streamlines, etc. Tools can be composed from other tools and even

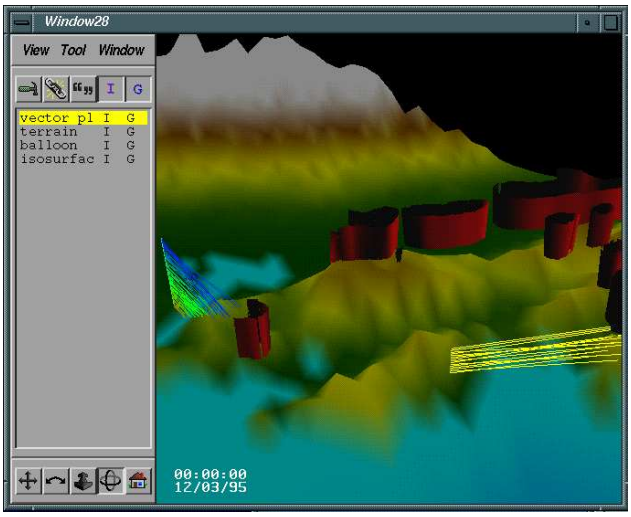

Figure 2: Graphical interface of PET SLUG showing visualizations a variety of realtime environmental data.

smaller building blocks called elements (see Figure 4). Elements are small C++ programs with a common API that are linked into PET SLUG at run time.

Examples of elements include different types of spatial and temporal interpolators, glyph makers, data transformation modules, etc.

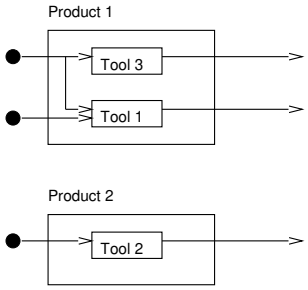

Figure 3: Visualization products are composed from one or more tools with inputs bound to specific data sets.

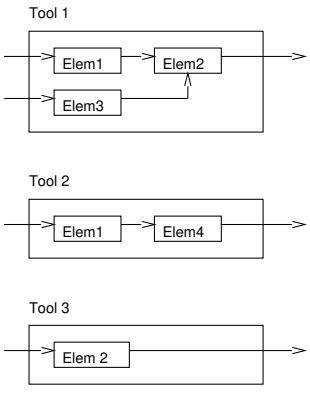

Figure 4: Visualization tools are made up of one or more elements.

PET SLUG currently provides a number of visualization tools – including both single-element and multiple-element tools. Each tool can be instantiated more than once if multiple copies are needed. Different data sets can be bound to different tool instances. Each tool may have an optional graphical interface specific to that tool. For example, an isosurface tool will have a slider for specifying a threshold value. Among the tools available in our current release are:

1. Balloon tool for displaying one or more radiosonde data from weather balloons. A variation of this tool can be used to display data from drifting buoys. Another variation is used to display data collected from an aircraft.
2. Contour tool for displaying iso-bars and iso-therms.
3. Coastline tool for displaying higher resolution coastline data over selected region.
4. Interpolation element for spatial/temporal interpolation of environmental data.
5. Isosurface tool used to visually delimit the portion of gridded data that is below the given threshold value.
6. Mini-cubes/mini-spheres tool for displaying properties of large volumes at each data point.
7. Pseudo-colored grid tool for displaying 2D layers of data. Several options for mapping data values to color are provided and can further be edited by the user.
8. Quake tool for displaying fault lines and earthquake events.
9. Satellite tool for displaying GOES7, GOES9 (1km and 4km) images at various display resolutions.
10. Station tool for displaying information from one or more stationary meteorological stations and buoy sites.
11. Terrain tool for displaying topography/bathymetry of selected region.
12. Vector plot tool for displaying vector information such as those from CODAR (ocean surface current data) and vertical wind profilers. This tool provides several types of glyphs for representing vector information: arrows, barbs, and uncertainty glyphs [11].

The output from these visualization tools are sent to one or more graphics windows. Each window may have its own region, camera, and associated set of tools. Thus, a graphics window and its contents corresponds directly to a visualization product. One of features supported by PET SLUG is the ability to customize, save, and edit these windows. Information about tools, datasets, including window placement, etc. are included in the saved information. This is useful if a user usually examines a set of queries over a particular region or if the user need to continue from an earlier session. This is also the mechanism in which a visualization product request is passed from the web interface to our server, as described in more detail in the following section.

5 WEB INTERFACE AND IMPLEMENTATION

5.1 Adding VRML support to PET SLUG

When working on converting our PET SLUG to a program accessible on the Web, we immediately took to switching OpenGL objects to VRML shapes. The initial experience was that VRML is extremely flexible and simple in terms of object definition and positioning and our OpenGL graphics primitives were quickly transformed into respective VRML shapes.

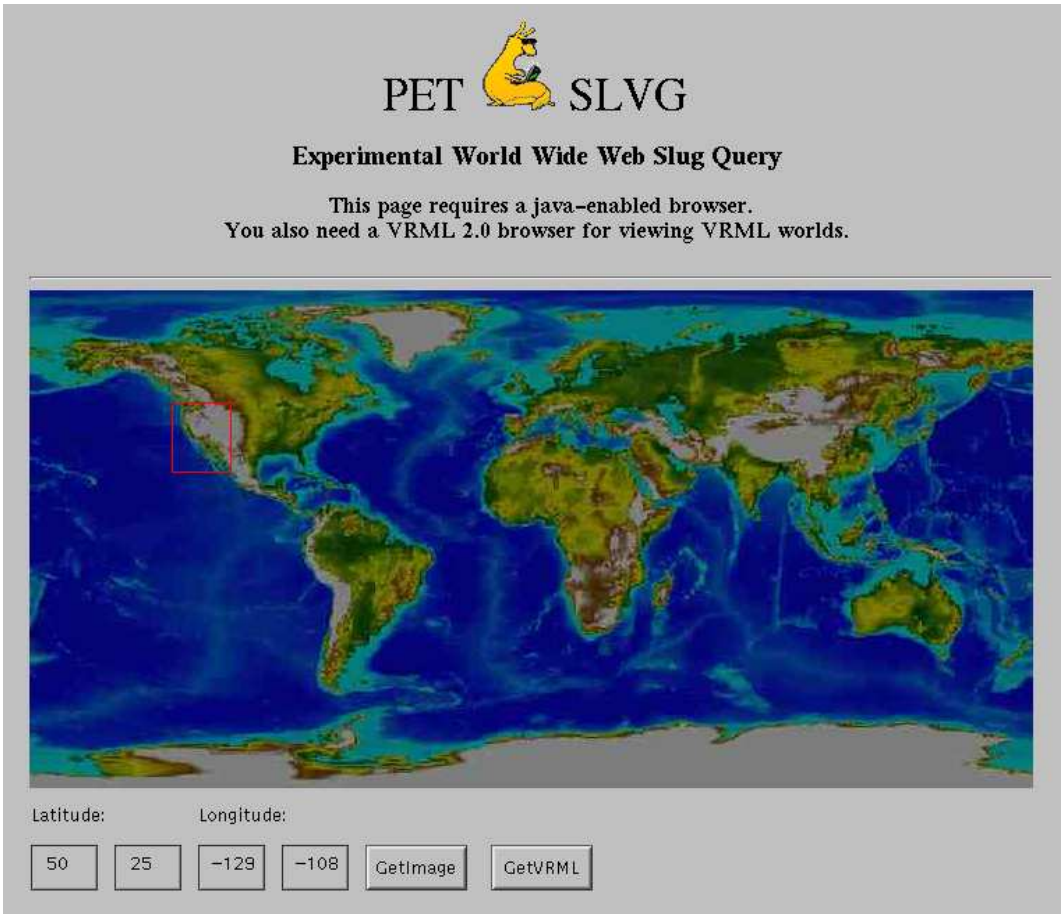

Figure 5: The region selection interface http://www.cse.ucsc.edu/research/slvg/slug.html

However, in the course of joining the pieces together and creating the web interface we came upon a few stumbling blocks specific to creating VRML worlds, namely:

- Combining the graphical output of several tools. Since each tool is a separate entity creating only nodes and viewpoints, there had to be an additional step to inline and provide a header for all the individual VRML outputs. This task was left to the cgi script at the end of the session. The fact that VRML allows multiple independent object and viewpoint definitions made our job much easier.

- Choosing the initial viewpoint. We decided to have the main viewpoint be the one created by the tool producing the terrain, and each component adding a viewpoint that zooms into the graphical element created by it. This way the user is first given an overview of the entire scene, and then allowed to view the details at closer range.

- Placing the individual graphical elements in world space. While some tools produce geometry placed at the origin (such as the terrain tool), most follow the geographic location of the data. These differences were reconciled by subtracting from all tools the minimum latitude and longitude of the terrain boundary, thus effectively registering all objects with the terrain.

5.2 User Interface

For the web interface we used a Java program with an embedded image of the world's topography. Using the mouse users can rubberband a rectangular region which is converted to latitude and longitude coordinates, as pictured in Figure 5. Also provided is the choice of tools and the time range for the data selection.

5.3 Server Setup

On the http server side, the script parses the parameters received from the Java applet and creates a session file to be used as input to PET SLUG. The time range specified in the file is ten minutes before, up to the time chosen by the user.

To avoid files being overwritten due to multiple hits, each input file is tagged with the process ID of the running CGI script. The cgi script then calls the PET SLUG program with the input filename as argument. The input file contains the description of the requested times, datasets and tools, as well as the coordinates of the spatial boundary. Upon processing the input, PET SLUG opens a connection to the database and obtains the requested data. At the same time tools listed by the session file are started and matched with the appropriate datasets as they become available.

Each tool creates a single VRML file containing the shapes and viewpoints. As the last step, the individual outputs are combined into a single file which is then returned by the cgi script back to the user in the shape of a dynamic link to the VRML world.

6 RESULTS

In this section, we present some of the visualization tools that have VRML interfaces for generating visualization products on the web.

6.1 Terrain

Using the given coordinates, the terrain tool extracts the necessary subset out of the world elevation database and creates an topography map. We have used the Digital Elevation Map file etopo5, available from USGS [9] The elevation values are exaggerated to give the terrain a visible relief, and height is also used to produce a texture map coded to the appropriate geographical color-scheme. The output is an elevation grid, the VRML shape well suited for the task of creating terrains. Dependent on the scale of the selected region, the resolution of the terrain grid varies, in order to keep the size of the output file relatively small. We have decided not to compress the .wrl files, as some browsers still don't recognize the file extensions.

There is one aspect of the elevation grid construction that caused some confusion - the fact that the grid is layed out in the XZ plane. Our system was created with the assumption that the terrain and other GIS objects exist in the XY plane with the viewer position along the negative Z axis. Due to this restriction, all other tools had to be adjusted for the new layout.

6.2 Station

This tool creates an object representation for weather stations and their corresponding readings – temperature, wind, etc. The stations are chosen based on the terrain boundary, that is, the database is searched for all met-stations in the region and the corresponding visualizations created. A glyph is used to represent each station and consists of a cylinder with an arrow stemming from its center - a graphical setting we have in both versions of PET SLUG. The color of the cylinder is mapped to temperature, the direction of the arrow to wind, and length of it to wind speed.

Figures 6 and 7 show the images of the station and terrain tools before and after VRML conversion of PET SLUG.

The graphical elements are used as a quick overview, but for the exact measurements the user is invited to click on an individual station, which causes a listing of all parameter names and values to pop up This option is a feature we built into the VRML version of PET SLUG since it was relatively simple to add without having to handle picking. Each station node has a hidden text field, displayed only when the object is clicked on. We accomplish this by creating transparent text, and routing the **isActive** event of cylinder node to the **set_transparency** event of the text node. Here is the piece of code facilitating the message passing:

```
DEF Toggle Script {
  url "vrmlscript:
    function set_boolean (bool) {
      if (bool==true) {tran = 0;}
      else {tran = 1;}
  }"
  eventIn SFBool set_boolean
  eventOut SFFloat tran
}

ROUTE CylSens.isActive TO Toggle.set_boolean
ROUTE Toggle.tran TO TxtMat.set_transparency
```

The text is kept facing the viewer with the use of a billboard node as a parent to the text node.

6.3 Wind Profiler

Wind profiler tool is similar to the station in that it displays a glyph at the location of each wind profiler. A windprofiler glyph consists of arrows of different vertical levels showing the direction and magnitude of wind vectors at that height.

6.4 Interpolated Winds

With singular wind profilers, it is difficult to observe the wind patterns in an area. For this reason we provide an alternative, where we gather wind information from all available sensors (on met-stations or wind-profilers) and use inverse multiquadric interpolation to calculate the intermediate values. The area over which we interpolate is a gridded region bounded by the extreme latitude and longitude values found among the sensors. Wind intensity is mapped to color, in order to avoid clutter among narrowly spaced glyphs. The VRML shape of choice for both of the wind tools is IndexedLineSet.

An example of interpolated winds is seen in Figure 8.

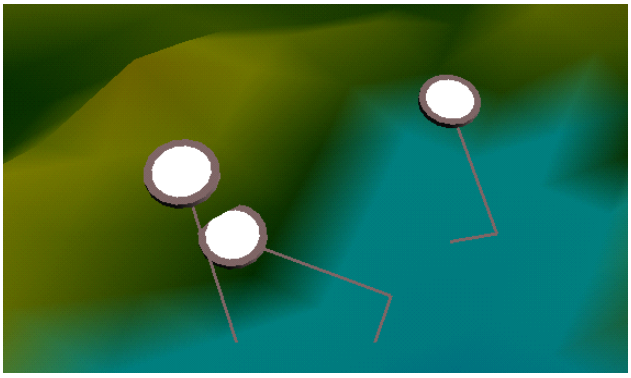

Figure 6: Three station glyphs with terrain

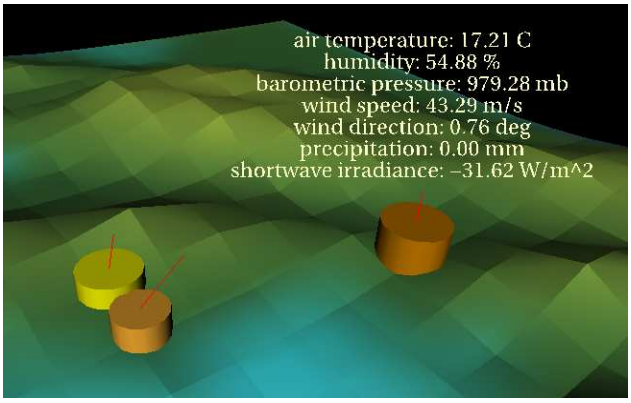

Figure 7: Three station glyphs with terrain - the VRML version. The text is appearing above the activated station

The glyphs for all tools are scaled according to the terrain boundaries, a step intended to keep a match among the object sizes.

7 DISCUSSION

In general, our experience in creating VRML-based visualizations has been a positive one. We do however regret not being able to create the interface tool in Java and shortcut the long-winded path from Java to the cgi script, to PET SLUG, to VRML, and back as a

URL. The Java security restrictions have forced us to use cgi and c programming as a way to access the database and local files.

In terms of VRML, we see several desirable extensions:

- Glyph prototypes. While it is easy to create simple glyphs, we believe it would aid novice VRML users to have a few common graphical constructs handy. For example, a garden-variety of arrow prototypes could become part of standard VRML.
- Irregular grids. In order to adequately represent Earth's curvature it will be necessary to support grids other than standard. We need to have a method for describing grids where the distances between cells may vary - say an array for row and column cell distances.
- Automatic LOD for ElevationGrid. Varying levels of detail are easily achieved with grids by subsampling. Why not allow users to create a new LOD by simply specifying the skip in both directions ?

8 SUMMARY

We described an end-to-end system that brings data from remote sensors and computer simulations, through a database, then a visualization system, and to users through a web interface that provides VRML output. One unique aspect and challenge of this work is the generation of the visualization product on-demand using the latest available information, sifted out from gigabytes of data. Our goal is to provide as much of the PET SLUG functionality to web users. We have achieved the first step in providing users with a set of visualization products to choose from. The logical next step is to provide interaction capability at the tool level where users can set tool parameters and customize their own visualization product. Among our planned future enhancements are adding more tools with web interactivity including a new source of data, NEXRAD, which provides volumetric information of the atmosphere.

ACKNOWLEDGEMENTS

We would like to thank other students and former students involved in the REINAS visualization project including Zoe Wood, Chris Oates, Michael O'Neil, Elijah Saxon, Jeremy Story, and Steven McDonald. We would also like to thank Eric Rosen for help with database issues, Dan Fernandez and Wendell Nuss for helping drive the development of PET SLUG from the users' point of view, and Craig Wittenbrink for collaborations on earlier REINAS visualization efforts. The Santa Cruz Laboratory for Visualization and Graphics provided a wonderful research environment and we would like to acknowledge that as well. Finally, the authors would also like to acknowledge support from ONR grant N00014-92-J-1807, NSF grant IRI-9423881, NASA Cooperative Agreement NCC2-5176, and DARPA grant N66001-97-8900.

References

[1] M. L. desJardins, K. F. Brill, and S. S. Schotz. Use of GEMPAK on Unix workstations. In *Seventh Conference on Interactive Information Processing Systems for Meteorology, Hydrology, and Oceanography,New Orleans, LA*, pages 449–451, 1991.

[2] T.T. Elvins and R. Jain. Web-based volumetric data retrieval. In *Proceedings of the VRML Symposium*, pages 11–15, 1995.

[3] William Hibbard and David Santek. The Vis-5D system for easy interactive visualization. In *Visualization '90*, pages 28–35, 1990.

[4] Eric Rosen. Slug video weather. URL: http://sapphire.cse.ucsc.edu/SlugVideo/di-weather.html.

[5] Elijah Saxon, Zoe Wood, Michael O'Neil, Chris Oates, Jeremy Story, Suzana Djurcilov, and Alex Pang. Integrated visualization of realtime environmental data. In *Proceedings of the Spring Conference on Computer Graphics*, pages 135–143. Comenius University, Bratislava, 1997.

[6] Will Schroeder, Ken Martin, and Bill Lorensen. *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*. Prentice Hall, New Jersey, 1996.

[7] J.C. Trapp and H.-G. Pagendarm. A prototype for a www-based visualization service. Online web site - http://www.sm.go.dlr.de/sm-sk_info/library/documents/EGSciVis97/.

[8] USGS. URL: http://sfbay7.wr.usgs.gov/wind.

[9] USGS. Etopo-5 data set. URL: http://grid2.cr.usgs.gov/data/etopo5.elev.html.

[10] Dan Vietor. WXP-the weather processor. URL: http://wxp.eas.purdue.edu.

[11] Craig M. Wittenbrink, Alex T. Pang, and Suresh K. Lodha. Glyphs for visualizing uncertainty in vector fields. *IEEE Transactions on Visualization and Computer Graphics*, 2(3):266–279, September 1996. Short version in SPIE Proceeding on Visual Data Exploration and Analysis, pages 87-100, 1995.

[12] J. Wood, K. Brodlie, and H. Wright. Visualization over the world wide web and its application to environmental data. In R. Yagel and G.M. Nielson, editors, *Proceedings of Visualization 96*, pages 81–86, 470. ACM, 1996.

[13] T.C. Zhao. XFORMS home page. URL: http://bragg.phys.uwm.edu/xforms.

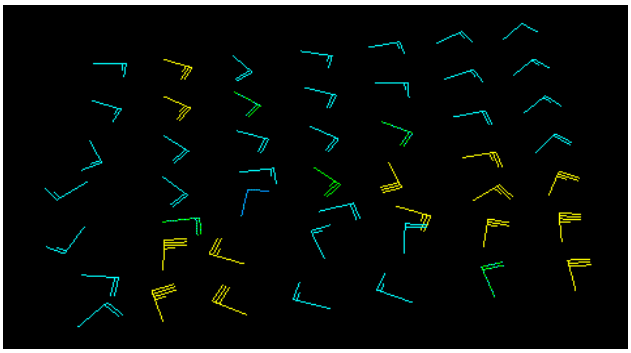

Figure 8: Interpolated wind vectors