A Distributed Interactive Ocean Visualization System

by

Steve S. Lin

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degrees of

Bachelor of Science in Electrical Engineering and Computer Science

and Master of Engineering in Electrical Engineering and Computer Science

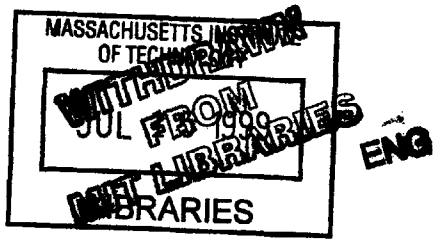
at the Massachusetts Institute of Technology

May 21, 1999
[June 1999]

Author_____
Department of Electrical Engineering and Computer Science
May 21, 1999

Certified by_____
Leonard Mcmillan
Thesis Supervisor

Accepted by_____
Arthur C. Smith
Chairman, Department Committee on Graduate Theses



A Distributed Interactive Ocean Visualization System
by
Steve S. Lin

Submitted to the
Department of Electrical Engineering and Computer Science

May 21, 1999

In Partial Fulfillment of the Requirements for the Degree of
Bachelor of Science in Electrical Engineering and Computer Science
and Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

A distributed interactive ocean visualization system was developed to provide high quality output for the World Wide Web. Our system also provides a wide range of tools for visualization and analysis. The system provides a simple yet flexible user interface for producing visualizations from multiple data sets that can be used to create animations as well as interactive worlds using a Virtual Reality Modeling Language. Through the collaborative effort of an interdisciplinary team we attempt to address some of the issues inhibiting scientists from utilizing and integrating visualization in their research. I also introduce a generalized particle animation tool as well as a glyph annotation technique in combining multiple data sets simultaneously.

This research was supervised by Gloria Brown-Simmons, Research Fellow, Center for Advanced Visual Studies, MIT, and supported in part by the Earth Science Enterprise, National Aeronautics and Space Administration (NASA), under contract NAG5-6600.

Thesis Supervisor: Leonard Mcmillan
Title: Assistant Professor, Department of Electrical Engineering and Computer Science

Table of Contents

List of Figures

1 Introduction

"Visualization is a vital component of the tools required for meeting Earth Observing System (EOS) scientific objectives. Neglecting visualization could result in failure to meet these objectives" (Botts, 1993). The National Aeronautics and Space Administration (NASA) dedicated much of its initial efforts to data retrieval, data management, and scientific objectives. The EOS Data and Information System (EOSDIS), NASA's Global Change Master Directory (GCMD), and the Pathfinder Program created jointly by NASA and the National Oceanic and Atmospheric Administration (NOAA) are all products of this work. Visualization, however, has been left mostly to commercial applications and, in some cases, systems developed within NASA.

Although technological advances and reduced costs have increased the amount of computing power directly available to scientists, they have been slow to accept the visualization tools that are currently available. Some of the reasons include lack of extensibility to incorporate specific functionality, complexity of supporting multiple computing environments, difficulty of importing scientific data sets, cost, inadequate support for data analysis, difficulty of communicating results to others, and difficulty of learning or using the system (Botts, 1993). Sometimes scientists are also unaware of the tools that are available.

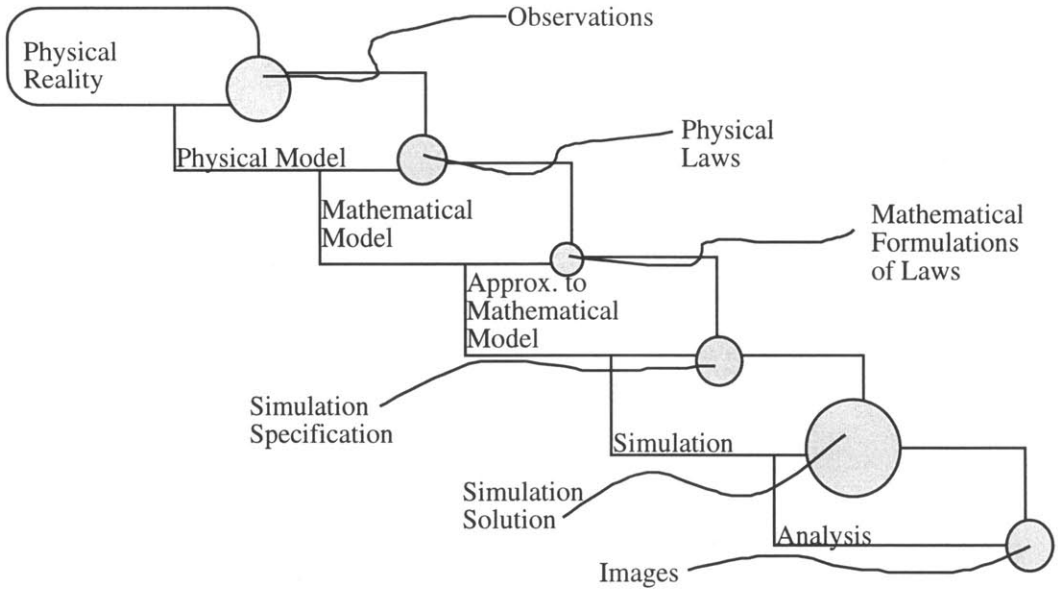


Figure 1. Simulating natural phenomena: the boxes represent processes, the circles' sizes indicate the relative volume of information passing between each pair of processes. (Earnshaw and Wiseman, 1992)

To address some of these issues, I worked jointly with a group of scientists from the Program in Atmospheres, Oceans, and Climate (POAC) in the Department of Earth, Atmospheric, and Planetary Sciences (EAPS) at MIT in a collaborative effort to build an interactive system for visualizing oceanographic data. This visualization system, developed at the Center for Advanced Visual Studies (CAVS), is built upon IBM's Visualization Data Explorer (DX), a visualization application that utilizes a data-flow model and specializes in scientific visualization and analysis.

Previous interactive scientific visualization efforts have attempted to address issues such as complexity, costs, communication of results, and ease of use through the implementation of Web interfaces. However, these systems fall short with regard to the sophistication of the visualization. One notable system is the Ingrid server at the International Research Institute / Lamont-Doherty Earth Observatory Climate Data Library at Columbia University (Blumenthal, 1998). This system provides an extensive

selection of data sets and allows the user to control simple parameters such as zooming. (See Figure 2.) The response time is reasonable for each data set, but the quality of the image returned is sufficient only for data browsing.

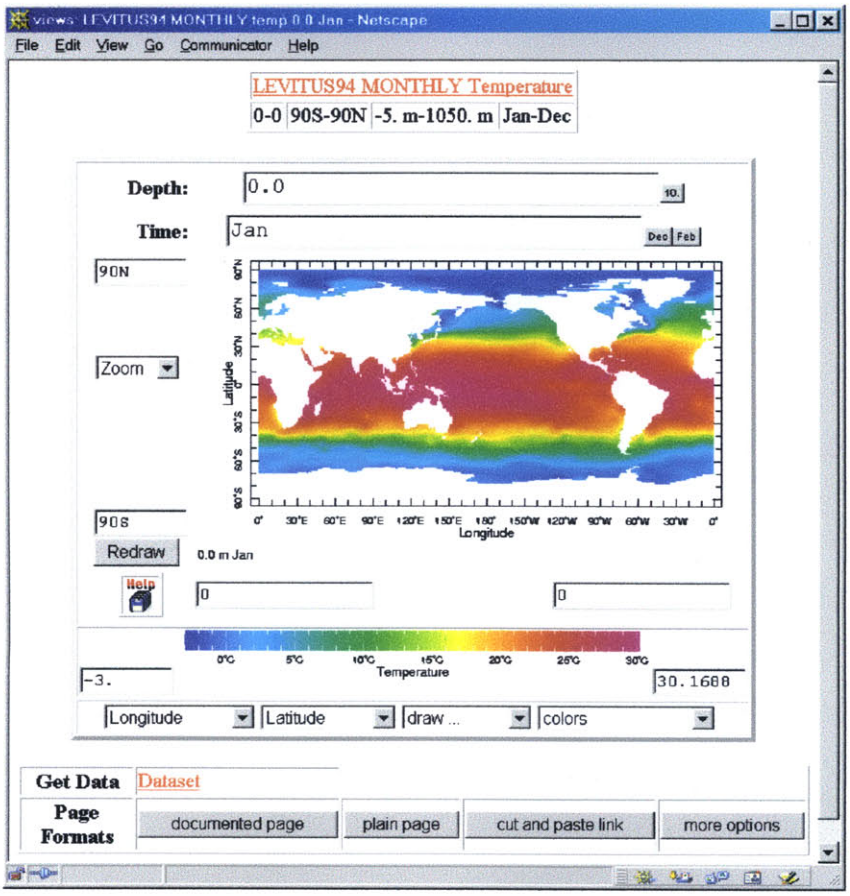


Figure 2. The Ingrid server at the International Research Institute / Lamont-Doherty Earth Observatory Climate Data Library at Columbia University.

Our system differs from other visualization applications in its emphasis on high quality graphical models that are suitable for scientific analysis. Support is available for viewing multiple data sets simultaneously, a feature most interactive systems lack. A wide range of visualization tools is also present and controlled by a flexible user

interface. Output in the form of images, models constructed using a Virtual Reality Modeling Language (VRML), and movies are accessible through the World Wide Web.

Although the current prototype lacks the Web-based front end of other systems, I designed the user interface with largely the same functionality as a higher level interface. The user can control and customize many aspects of the visualization through a set of control panels implemented in DX. The parameters controlled by the control panels can easily be modified to handle values passed to DX from an external interface.

A major feature of the system is the support for a wide range of visualization and analysis tools. These tools include contour surfaces, glyph annotations, wire-frame meshes, data probes and various data filters. These features are mostly built-in or derived from DX modules. DX also enables users to write customized modules if a desired tool is not present.

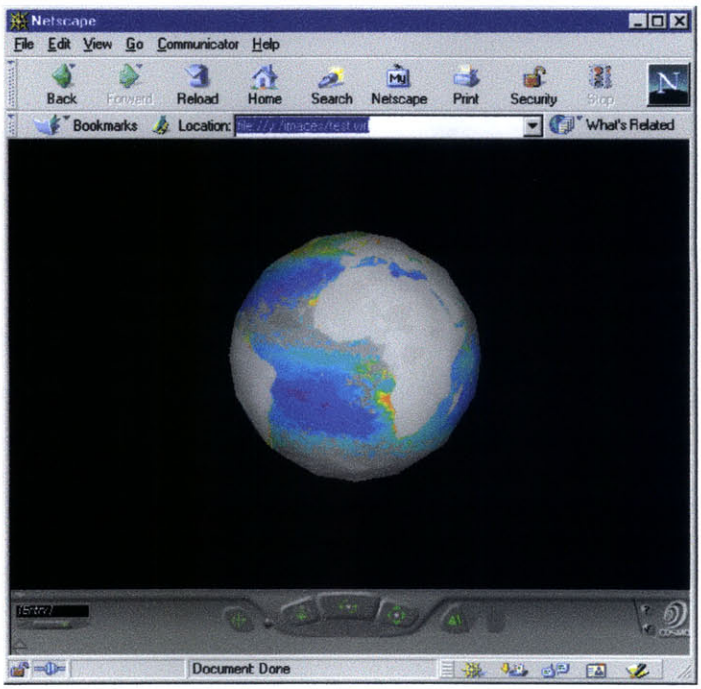


Figure 3. SeaWiFS ocean-color data viewed on a VRML model.

In addition to standard image output, the user can also produce VRML worlds as well as MPEG animations. The VRML worlds are graphical models that allow users to interactively select different viewpoints through a customized viewer or Web-browser plug-in. These output types facilitate the communication of results quickly to other users.

The ability to interactively incorporate multiple data sets into a model is useful in depicting a potential correlation between the data sets. When a multidimensional vector data set is involved, the user can use a particle animation to trace a set of points through time. I developed a generalized tool for tracing particle flows for Earth data given an appropriately scaled vector field. When the user views multiple scalar data sets simultaneously, glyphs, or objects such as spheres or arrows used for annotation, can be used to enhance the graphical model. The team developed a technique utilizing overlaid glyphs to effectively incorporate an additional data set into the model without overwhelming or misleading the user.

With the multitude of data formats in existence, a major issue is the incompatibility of scientific data sets with the various applications used for analysis, visualization, and animation. Moreover, although many visualization programs include support for the various data types, the lack of a standard presentation or orientation makes each data set annoyingly unique. For example, some formats select rows to represent the major axis while others use columns. Also, data sets run from –180 degrees longitude to +180 degrees longitude (our default), whereas others choose to vary from 0 degrees to 360 degrees. I chose to abstract many of the data related issues in the form of assumptions for purposes of this prototype as these issues comprise a major task on their own.

Our interactive ocean visualization system concentrates on the presentation of high quality output over the World Wide Web as well as the presence of a wide range of tools for visualization and analysis that the user can control through a flexible user interface. Through the collaborative effort of an interdisciplinary team we attempt to address some of the issues inhibiting scientists from utilizing and integrating visualization in their research.

2 Methods

The introduction of new technologies continually brings media, art, and science closer together. From groups of interdisciplinary teams working together through the ability to collaborate and share one's work on the World Wide Web and the Internet, people contribute their varying expertise effectively using technology as a channel of communication and distribution. One such area that benefits from this is scientific visualization. Scientists, artists, and computer scientists collaborate to produce graphical models that are artistically appealing yet scientifically accurate. People thrive from seeing others' research and obtain valuable feedback instantly. These tools facilitate communication, experimentation, and analysis, and stimulate further scientific discovery.

2.1 Interdisciplinary Teams

We deliberately chose to foster these sorts of interaction by forming an interdisciplinary team. A team of one artist, specializing in scientific visualization and systems, and one graduate student, studying computer science, researched various visualization techniques in addition to system design and integration. To assure scientific accuracy of all images and models, the team also collaborated with scientists whose expertise with the data sets supported the other skill sets present. These scientists provided feedback for information such as color, regions of interest, correlation between data sets, and the correctness of all results. They also added valuable ideas regarding features that might be of use to them in a visualization system.

In addition to the collaborating scientists at MIT, we obtained additional feedback from other scientists at NASA and visualization specialists who have worked with Earth data in the past. These people, whose contributions helped shape the application, also formed the first tier of end-users for the visualization system.

Scientists for the most part lack the time and experience required to work with more powerful visualization tools available today. Constantly improving technology and standards also make it difficult to rely on a new application or specification knowing it will soon become obsolete. Thus, scientists rely on time-tested methods for their research, using mathematical and data analysis software for their work.

Since humans rely heavily upon visual cues to interpret their surroundings, advanced visualization should play an integral role in scientific research. Visualization of scientific data has occasionally led to interesting discoveries. However, this method of analysis has thus far been underutilized, perhaps due to the lack of responsiveness in high quality visualization systems. We attempt to address these issues by designing a system better suited to interactive analysis.

During the course of the project, I interfaced mainly with two groups of scientists in the department of Earth, Atmospheric, and Planetary Sciences at MIT. The first group focused on sea-surface-height data as measured by Topex/Poseidon and its variants. This is a measurement of the absolute displacement from sea level of the ocean surface in centimeters. More recently, I have been working with a group of scientists studying SeaWiFS ocean-color data. This is a measurement of the chlorophyll concentration in the oceans in milligrams per cubic meter.

2.2 *System Resources*

Although the price of powerful machines continues to drop, available hardware and software resources still limits our efforts. Since both system and visualization research depended on common resources, I describe them in this section.

The primary development machine was a 300 MHz Intel Pentium II processor with 384 Megabytes of SDRAM running Windows NT. Later, a Silicon Graphics Onyx2 InfiniteReality workstation with 256 Megabytes of RAM and dual MIPS R10000 processors was used as well.

The primary visualization package utilized was IBM Visualization Data Explorer (DX) Version 3.1.4 for both the Windows NT operating system as well as for Silicon Graphics workstations. We chose DX over other visualization packages such as Research Systems' Interactive Data Language (IDL) and Advanced Visual Systems' Application Visualization System (AVS) due to cost, ease of use, scripting capabilities, and support for both Windows NT and Unix platforms.

Often the effort to create an effective scientific visualization is dominated by importing the data and determining its orientation, scale, and other attributes. Data Explorer's built-in data import features are a good starting point. They make it relatively easy to import various data sets of different formats. Native support is present for importing HDF and netCDF formats, and DX provides a general array importer that is fairly straightforward to use.

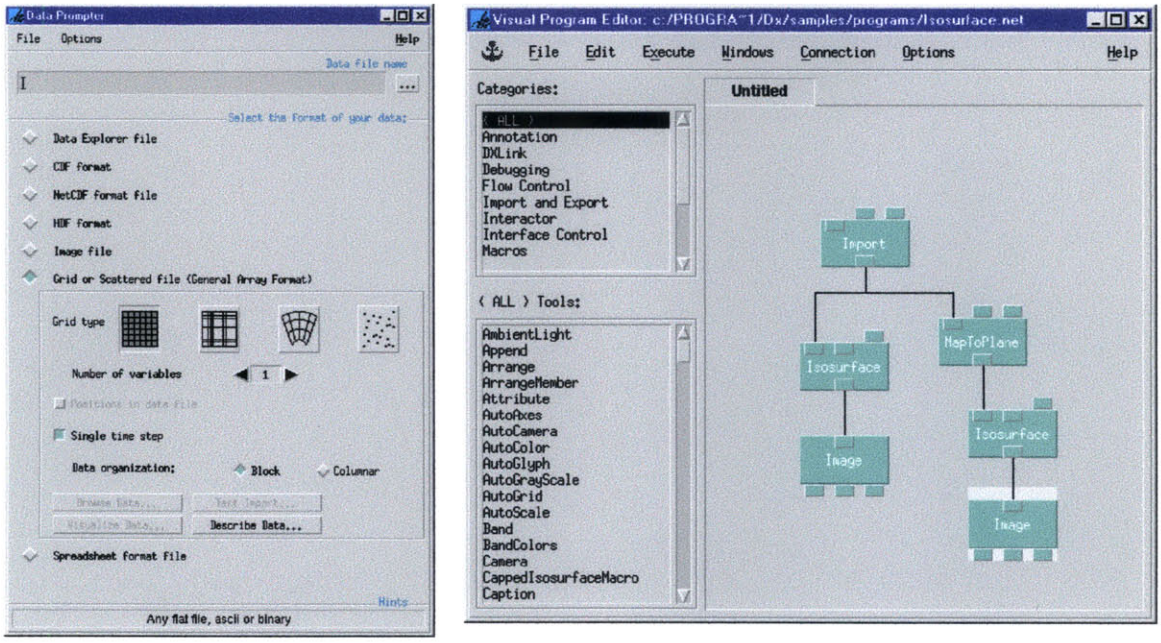


Figure 4. (Left) DX Data Importer. (Right) Sample visualization network.

DX relies on a visual programming metaphor using networks of functional blocks to describe and manipulate the data flow from data import through the generation of a graphical representation of the data. DX provides a number of standard modules to import, process, and visualize the data. The user can also write a C program to build customized modules to perform necessary functions. Macros can be saved by combining modules and other macros to perform broader tasks.

A key feature is the support for scripting in DX. Each network file is stored in Data Explorer's scripting language, which makes it easy to dynamically generate networks based on specific parameters. Furthermore, there are different levels of interface that one can use to link DX code to external applications. These different levels provide support for calling DX modules as well as controlling the DX Executive from an external program. The ability to control DX modules through its libDXcallm library is more powerful and requires user management of objects and memory. Controlling the

Executive via libDXL gives you the option of designing your own user interface for a network. However, the libDXcallm library is not available under Windows. Another major drawback of DX is its requirement of an X server, which translates to extra load on the machine and inefficient utilization of the graphics capabilities under Windows NT.

I used Cosmo Worlds to create template worlds in VRML. This program allows the user to define objects, interaction, and environment settings in a virtual world. I explored models using objects constructed from the actual data as well as using texture-mapped spheres. These worlds can then be viewed with a VRML viewer or a Web-browser plug-in.

Adobe Premier is a leading movie production and movie editing program for the Windows environment. It supports the creation of movies from a sequence of images with input and output in many common formats. With the addition of a DV Spark board, it supports high-resolution output in AVI format for storage and playback from DV-based media. Lower quality animations can also easily be developed for Web-sharing purposes.

The Berkeley MPEG encoder is a command line program that can be invoked to create MPEG clips that are reasonable in quality. Although this program is not as powerful as commercial video editing packages like Adobe Premiere, it provides a simple way to create animations given a parameter file. The program is available for a number of UNIX platforms in addition to Windows NT.

2.3 Interactivity and the World Wide Web

Two important goals in my system design are user interactivity and making the results publishable on the World Wide Web. Interactivity is a key element in a flexible

and dynamic design, allowing the user to customize and explore various possibilities that cater to his or her unique needs. Providing the user with a means to interface with the data also promotes creativity and diversity, which is far more useful in research and analysis than a presentation of static parameters.

The Internet and the World Wide Web have become an essential and integral part of academic research and communication. Communication through digital media allows results to be quickly distributed over wide distances. In addition to collaborating more effectively, results can be presented and shared using common formats, resulting in instantaneous testing and feedback from a large number of users in any location.

However the greatest potential for our scientific visualization system is that it can combine data gathered from scientists throughout the world into a single model. These same scientists can download and view the results locally in the form of images, movies, and graphical models in their respective labs.

3 Results

Working with the scientists, I developed a system with a flexible user interface that allows the user to control various aspects of the visualization. The user can generate output in the form of MPEG animations or VRML worlds in addition to standard images. The system interface controls a wide range of data visualization techniques. I have also developed a new particle animation tool and a glyph annotation technique for viewing multiple data sets simultaneously.

I explored different combinations of visualization methods including incorporating isosurfaces, glyphs, wire meshes, and particle trajectories. Isosurfaces are surfaces of constant value. In the case of two-dimensional data sets, a set of contour lines is created. Glyphs are objects such as spheres or arrows used for annotation. Glyphs can represent data values through variation in size and color. Wire meshes are grids stretched over the height data to help emphasize the changing data values. I also varied other variables such as color maps and scaling to optimize artistic appeal and scientific technicality. This research enhanced the toolkit of methods currently used in scientific visualization, and focused on images and animations stemming from graphical models using sea-surface-height and ocean-color data.

The construction of the visualization system revolved around these techniques. The individual components were generalized to allow for different values to specify each unique data set. These parameters defined the user interface in the first stage of system development, which provided direct support for outputting images of varying resolutions. These images could then be applied to produce animations or VRML models.

3.1 Visualization Techniques

Scientists have traditionally used powerful mathematical programs such as Matlab to produce simple image plots of their data. The graphical capabilities of these programs tend to be limited, and do not provide the ease of use to apply different graphical techniques in experimental research. They can generate histograms, contour lines, and scatter plots, but not much else. Moreover, reading in the scientific data occupies much of the time, and is not inherently easy within such mathematical analysis programs.

By introducing a program that specializes in scientific visualization such as DX, these tasks become significantly easier. Reading in the ocean color data for the first time and producing a simple image plot took only a few days compared to several weeks for similar tasks using mathematical tools. The interface provides an easy way of tracking the data flow and each computation performed along the path from data input to the composition of an image. These programs also facilitate experimentation with various visualization techniques, and provide a set of standard tools available for the user. Such tools include glyphs and other forms of annotation, histograms, isosurfaces, streamlines, as well as modules to apply color to and filter the data.

In addition to working with traditional visualization techniques, we developed new ones to highlight certain features in particular data sets. The two main data sets we worked with were the Topex/Poseidon sea-surface-height data and SeaWiFS ocean-color data. In particular, I developed a particle animation module portraying the sea-surface-height data and associated flow fields as well as a glyph technique combining ocean-color and sea-surface-temperature data.

3.1.1 Particle Animation

The Topex/Poseidon sea-surface-height, sea-surface-height-anomaly, geostrophic velocity, and relative geostrophic velocity data formed the foundation for our research in particle animation. The sea-surface-height-anomaly data is derived from the original sea-surface-height data, and is calculated by taking the difference between the absolute data and a temporal average for each coordinate. The geostrophic velocity data indicates ocean currents with respect to the absolute sea-surface-height, while the relative velocity data is used with the anomaly data (Wunsch and Stammer, 1998).

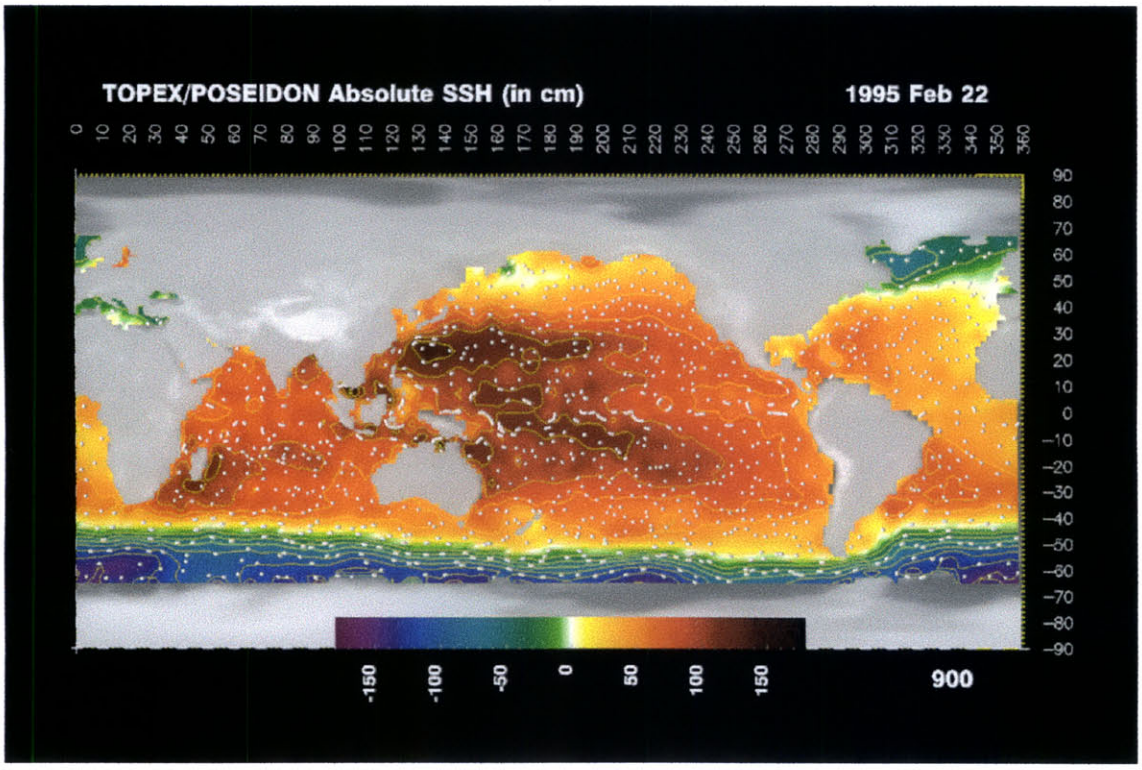


Figure 5. Sea-surface-height data overlaid with trajectories of Lagragian particles advected by the associated geostrophic flow field. Processed by the Department of Earth, Atmospheric and Planetary Sciences, visualization by the Center for Advanced Visual Studies, MIT, 1998.

For these data sets, we concentrated on accenting contrasts between various boundaries of different value regions. At first we tried to use variations in color to bring out these contrasts. We found experimenting with both smooth gradients as well as distinct color bars was able to bring out contrast in specific regions of interest. In the case of the absolute sea-surface-height data, contour lines served well to enhance these features. We also introduced particles to further emphasize variations in these models.

The primary goal of our visualization emphasized accuracy in relative scaling and time representation. I constructed the particle animation tool with the aid of the DX streamline module, which takes a vector field and computes a series of lines through the flow field at a particular instant in time. I compiled a custom module that takes the output from the streamline module and interpolates the position of each particle for a single time step. This new position is then fed back into the streamline module for the new time step and the loop continues.

Initially, a regular grid of points is submitted to the streamline module. I chose this method as an unbiased way of introducing a set of particles that the user can trace. It is also possible to start with other configurations such as a random sampling of particles. A line through the static vector field at the starting time interval is computed for each point in the grid. For each of these lines, my module determines the appropriate position of the particle after a single time step. To control the number of particles present at any given point in time, it then checks if any other particles are present within a fixed radius. Particles are removed if any other particles are found within that radius or if they reach the end of the flow field. (See Figure 6.) Otherwise, the particle is added to the output list. Finally, each position in the original grid is checked and particles are introduced into

the scene if no other particles are present within a certain radius. This new set of particles is then used as the initial set of points submitted to the streamline module for the next iteration. This code is included in Appendix A.

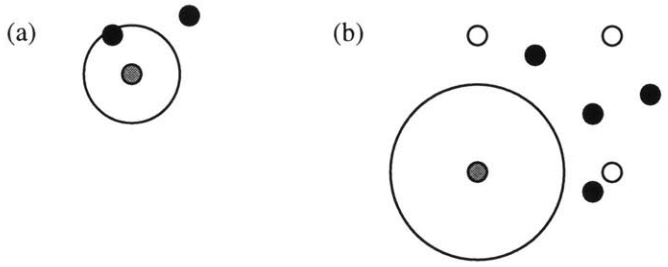
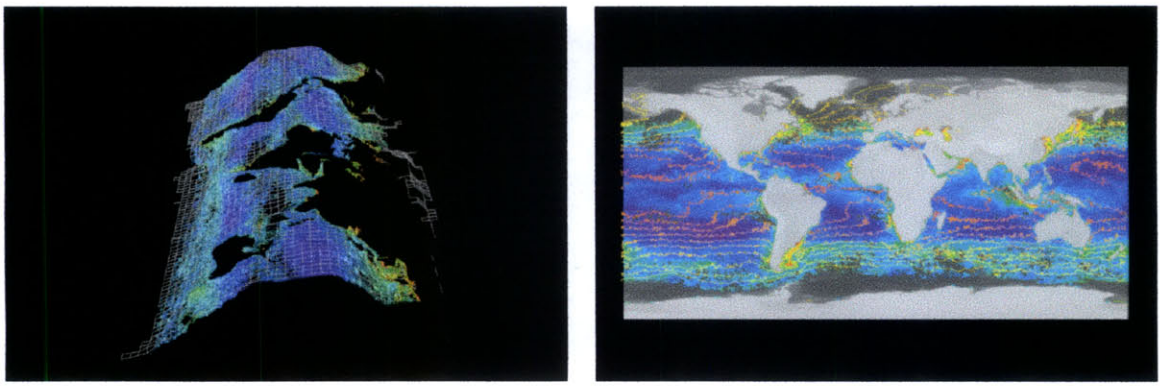


Figure 6. (a) Particles are ignored if they converge within a fixed radius of each other. (b) Particles are introduced into the system if no particles are present within a certain radius of any original grid point.

Another important factor is the appropriate scaling of the data. For example, the original geostrophic and relative geostrophic velocity data is given in centimeters per second. These units must first be converted to degrees per day for the grid projection (See *Projection*, Section 3.2.1.) given the daily time interval of our data. Scaling varies with latitude, as one degree at the Equator is greater in distance than one degree at the North and South Poles.

3.1.2 Overlaid Glyphs

The SeaWiFS ocean-color data was used in the next part of the project. In addition, we also explored integrating the ocean-color data with sea-surface-temperature and sea-surface-height data to help visualize a potential correlation among these data sets. The main focus here was to develop effective techniques to visualize these data sets together without overwhelming the user with too much information.



(a) (b)

Figure 7. (a) SeaWiFS ocean-color data draped over NOAA AVHRR MCSST surface-temperature data enhanced with a wire-frame mesh. (b) SeaWiFS ocean-color data overlaid with NOAA AVHRR MCSST surface-temperature isosurfaces. Program in Atmospheres, Oceans and Climate, and Center for Advanced Visual Studies, MIT, 1998.

While working with the combination of sea-surface-temperature data and ocean-color data, we introduced glyphs as a means of overlaying one data set on top of the other while not obscuring or blurring the data. We attempted different configurations, including populating glyphs along contour lines and sampling glyphs randomly within subdivided regions.

We spread the glyphs evenly within each region to prevent the misleading impression that the distribution of the glyphs was dependent on some data value. The spacing between glyphs decreases in regions that have higher data values. All the glyphs are the same size regardless of value.

The glyphs represent the ocean-color data, while the underlying surface is composed from sea-surface-temperature values. Isosurfaces can also contribute to the model for either the ocean-color or sea-surface-temperature data. Either the sea-surface-

temperature data or sea-surface-height data can define the height displacement for the model.

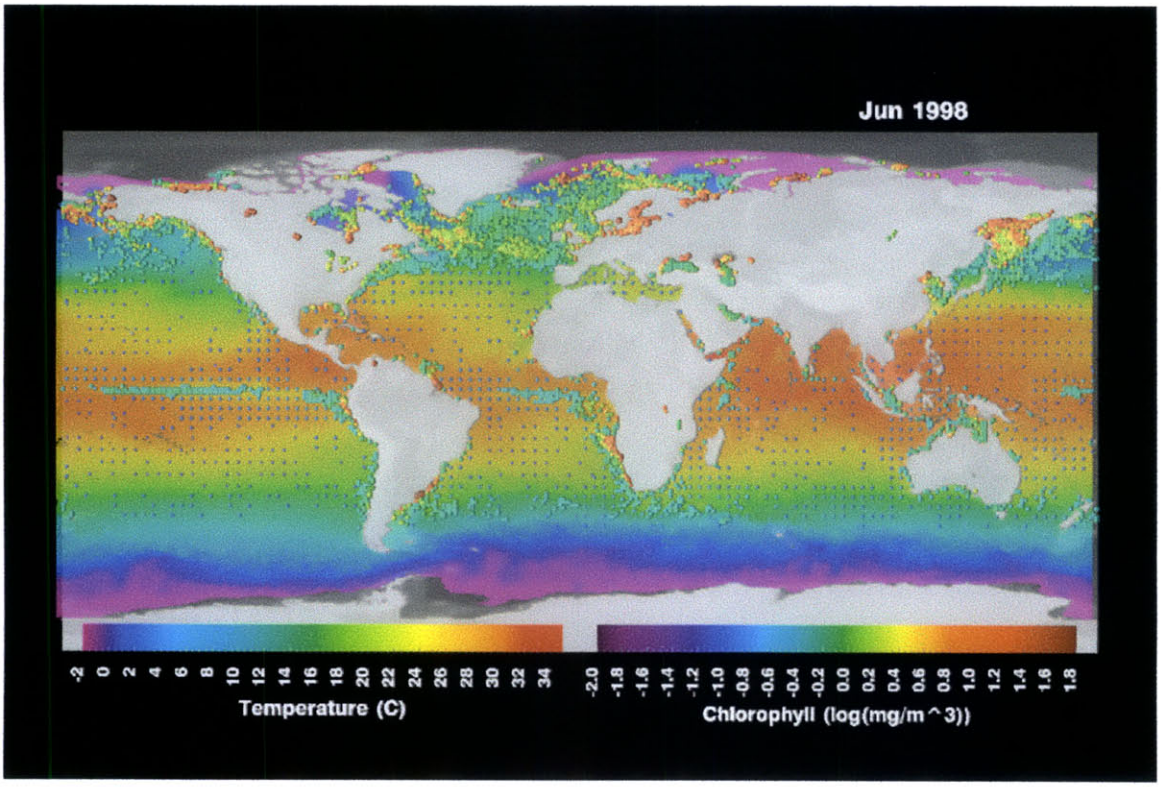


Figure 8. SeaWiFS chlorophyll glyphs draped over NOAA AVHRR MCSST surface-temperature data. Program in Atmospheres, Oceans and Climate, and Center for Advanced Visual Studies, MIT, 1998.

3.2 System Architecture

I then shifted my efforts towards generalizing the previous research and introducing more interactivity into the visualization system. A major goal of the design was to allow users to customize the visualization using their own parameters and produce an end product such as an image, movie, or VRML model that they can access and view on the Web.

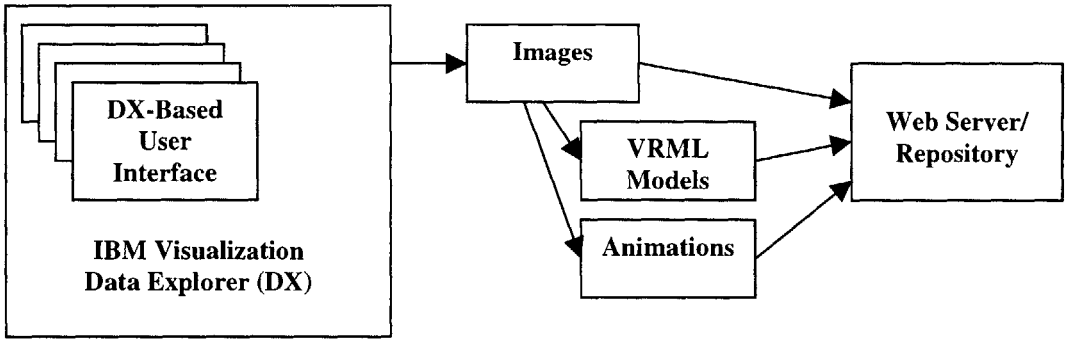


Figure 9. System Diagram. The user controls a DX-based user interface. The output images can be accessed from a Web repository or incorporated into VRML models and animations.

The initial prototype features a user interface constructed using DX. The interface, described in more detail in the following section, gives the user the ability to control aspects about each particular data set, camera angles, as well as methods of combining the data. All the processing from data import to the output of an image takes place within the DX environment. Some additional steps are required to incorporate the images into movies and VRML models. The goal of Web distribution is satisfied with each output type available.

3.2.1 User Interface

The user interface defines the range of interactions that a user can have over the system. The preliminary user interface for the ocean visualization system depends on DX and must be accessed from within the program. The control parameters provided by the interface can easily be considered or even implemented as arguments passed from an external program when a Java-based interface is created. The variables and output types remain mostly unchanged and I discuss these below.

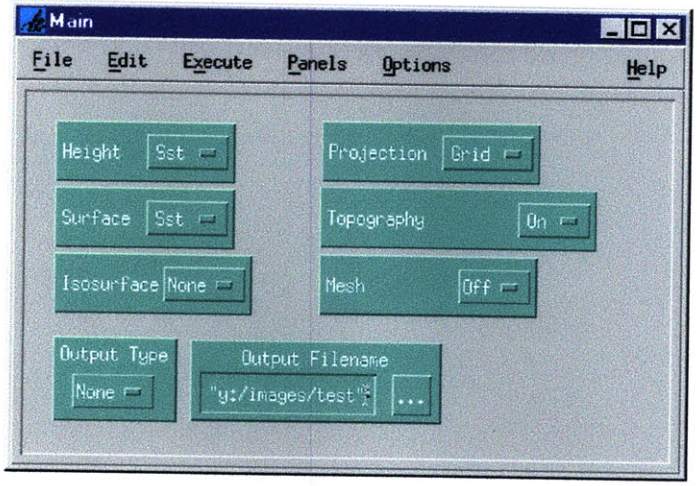


Figure 10. Main control panel.

Output Types – [GIF | RGB | TIFF | YUV | WRL | None]

This selection represents the type of output generated. If the user selects "None," no output will be saved, but the resulting image will appear on the screen. Four different image formats can be saved in DX. The formats supported are GIF, RGB, TIFF, and YUV. The YUV format is supported for MPEG encoding purposes. If the user selects "WRL," the system produces a 1024x512 resolution GIF image in grid projection without borders and in orthographic mode. This image can then be called as an argument for the texture map applied to a sphere in the VRML world. To view these models, a VRML plug-in is needed for the particular web browser, such as Silicon Graphics' Cosmo Player. The current defaults for other image types create 720x480 resolution images in perspective mode.

Movie files can also be created indirectly through DX. As an intermediate step in the movie generation process, images must be output in the YUV format for MPEG movies using the Berkeley MPEG Encoder. For more information on how to generate animations, see section 3.2.3, Generating Animations.

Output Filename

This entry lets the user specify the name for saved files. A file selector allows users to browse the directories available.

Surface – [Ssh | Sst | Chlo]

The surface attribute specifies whether sea-surface-height (Ssh), sea-surface-temperature (Sst), or ocean-color (Chlo) data illustrates the ocean surface. The colors displayed correspond to the color map applied to that data set. The data value at each location is looked up in the color map table and values between data locations are linearly interpolated. I configured the system so that the surface data is independent from its height component. See *Height,* below.

Height – [Ssh | Sst | Chlo]

This field determines which data set applies the displacement in the z-axis for the surface data. To enhance viewing of a particular data set, one can set the height to the same value as the surface. This setting takes the least time to process. Sometimes, it is preferable to use a different data set for the height than for the surface. This process requires mapping the surface-data-set grid to the height-data-set grid, which can take quite a long time. To significantly reduce the time involved, one should consider mapping the data sets in a preprocessing stage. For more information, please read section 4.2, Data Assumptions. The user can exaggerate the height data by scaling the data set selected for the height parameter.

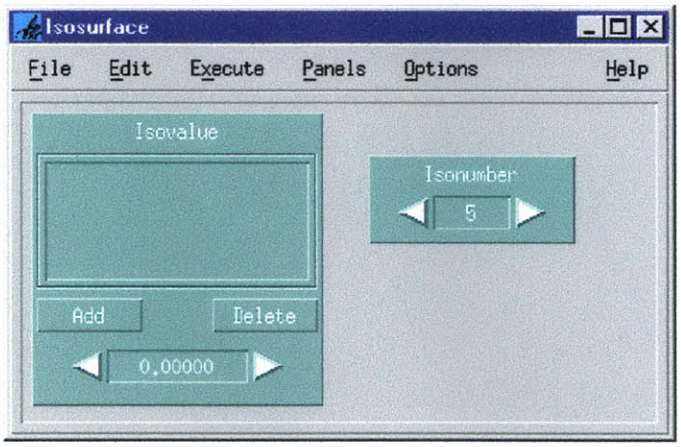


Figure 11. Isosurface control panel.

Isosurface – [Ssh | Sst | Chlo | None]

The isosurface variable specifies the data set used for drawing isovalue curves or contour lines overlaid on the surface. If the user selects "None," this field is ignored. If a three-dimensional data set is used, the result will be an isosurface, where all values on the surface represent a constant value. If the data set is two-dimensional, a set of contour lines will be drawn instead.

The user can configure the number of isosurfaces in two ways. One option is to enter specific values in the "Isovalue" parameter. This will draw an isosurface at each of the constants listed. Alternatively, one can input a value for the "Isonumber" parameter. This automatically generates the specified number of isosurfaces approximately equally spaced apart in value. If any numbers are specified in "Isovalue", the system disregards the "Isonumber" parameter.

Mesh – [On | Off]

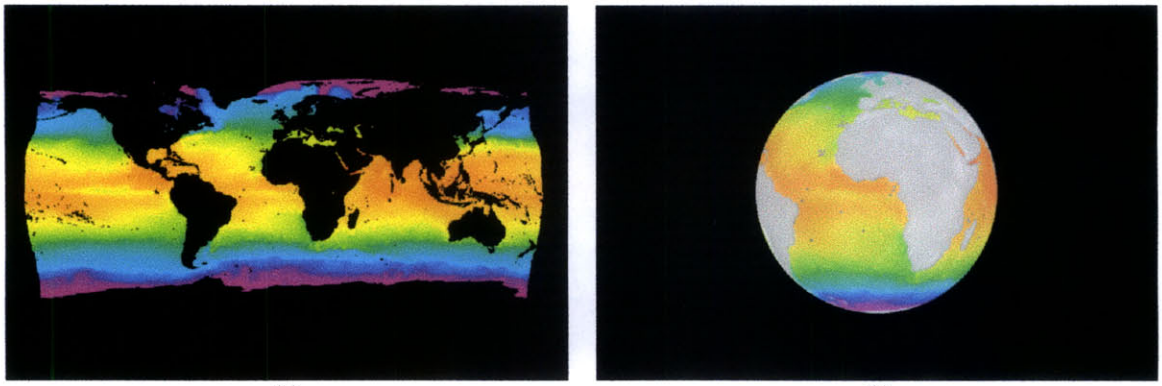
This lets the user display a wire mesh over the surface to enhance viewing of the ocean data terrain. The spacing between gridlines is approximately four degrees. This option is only available when using the grid projection. See Figure 7(a).

Topography – [On | Off]

This option toggles between viewing or hiding the land and ocean-floor topography. This half-degree-resolution data set covers the entire surface of the Earth. This data set is by default in kilometers measured from sea level, and can be scaled by the user. The topography values do not effect the displacement of the ocean data.

Projection – [Grid | Globe]

The grid projection allows the user to see the entire map of the Earth at once. All latitudinal and longitudinal values are equally spaced. The map is centered at zero degrees longitude and zero degrees latitude. Longitude values range from -180 degrees to +180 degrees from left to right along the x-axis and –90 degrees to +90 degrees from bottom to top along the y-axis. Note that in this projection, latitude values farther away from the equator are stretched more than near the equator. Data values are represented by displacement along the z-axis.



(a) (b)

Figure 12. NOAA AVHRR MCSST sea-surface-temperature data in (a) grid projection and (b) globe projection.

The globe projection views the Earth in its natural form as a sphere. The polar radius and equatorial radius differ as with the Earth's actual dimensions and are accurate to four significant digits. The globe is aligned such that the North Pole is along the positive z-axis, zero degrees longitude lies along the negative y-axis, and ninety degrees longitude lies along the positive x-axis. Data values are represented by displacement outward along the radius of the globe.

The user can control the scale of the displacement in both projections. This scale factor must be applied to the data set used for the *Height* parameter. See *Height*, above.

Particles

For any vector data sets present, the option to generate a particle trajectory is available. Due to the memory required for this process, particle trajectories must be computed independently before integration with the overall visualization. Each vector file should be numbered in sequential order. The data assumptions listed in section 4.2 also apply with the vector data sets.

The user must supply four parameters for the particle export visualization network. The input and output filename formats specify the format of the filenames to be imported and exported, respectively. The user must also include the minimum and maximum values for the sequence numbers. For example, if the vector data files are named vel.100, vel.101, ... vel.150, the user should specify the input format as "vel.%d" while setting the sequence minimum to 100 and maximum to 150. The output filename should be specified similarly. The user can also enter the approximate number of particles to follow. Since the particle network maintains state at each frame, the entire sequence of data files must be processed continuously without interruption.

Glyphs – [Ssh | Sst | Chlo | None]

This allows the user to select which data set is applied to the glyph visualization. If "None" is selected, this field is ignored. This feature also allows users to specify regions of interest to segment the glyphs. The numbers should be listed starting with the minimum value and ending with the maximum value. For example, if the values listed are 0.0, 1.2, 3.7, and 4.5, the system will define three distinct glyph ranges from 0.0 to 1.2, 1.2 to 3.7, and 3.7 to 4.5.

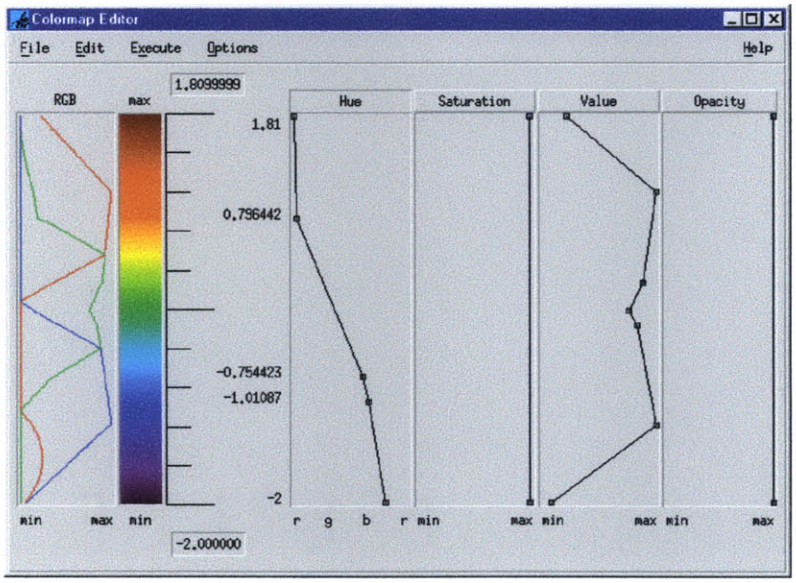


Figure 13. DX Color map editor.

Color Maps

For each data set, a color map editor can be invoked that allows the user to load a preset color map or modify an existing one. This lets the user customize his or her particular preferences for bringing out certain ranges of interest.

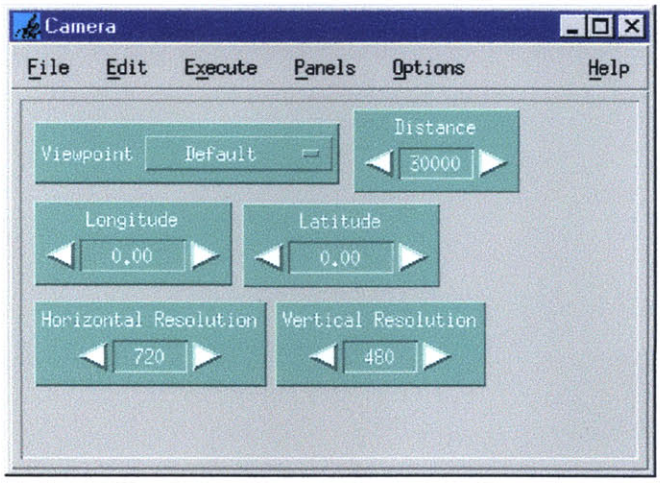


Figure 14. Camera control panel.

Viewpoint – [Manual | Default | North Atlantic]

This option lets the user choose data viewing parameters from a set of predetermined locations of interest. The current preset values are *Default* (0, 0, 500, 30000), and *North Atlantic* (-45, 40, 120, 9000). The coordinate values represent longitude, latitude, distance above sea level for the grid projection, and distance above sea level for the globe projection, respectively. Choosing one of these locations overrides the manual camera and distance inputs.

By setting the viewpoint to manual, one can alternatively select the coordinates of the point seen at the center of the image. Choosing the distance will determine how far away from the surface the camera is located. This value is in kilometers for the globe projection and for the grid projection is in the same units as the height data set.

3.2.2 Data Pathway

By executing DX with a set of parameters, the process of generating the visualization begins. Assuming the data has been preprocessed as described in Section 4.2 Data Assumptions, all imported data follows a prescribed pathway depending on the visualization parameters set. If the user desires a globe projection, the system wraps the data at the ends to fill the gap between the rightmost data points and the leftmost ones. Furthermore, the system recalculates each position to its respective location in the spherical projection.

Each appropriate data set is then routed to each visualization technique as requested. For the grid projection, the data sets are draped over the data set specified by

the *height* variable. This enables glyphs, isosurfaces, and other data sets to follow the height component of a common data set.

The applicable techniques are then collected and the camera angle determined from the viewpoint selected or the coordinates and distances entered. The system displays the final antialiased model on the screen and saves it if an appropriate format has been selected.

3.2.3 Generating Animations

The Berkeley MPEG encoder has been integrated into our ocean visualization system to support the generation of animations. The encoder provides a relatively simple method of constructing animations with sufficient quality. Since most of the controlling parameters are supplied via a script file and command-line arguments, it was easy to interface with the DX system. An external wrapper program can integrate this into the system, thereby minimizing extra work on the part of the user. One invokes this program by calling *mpeg_encode* with the name of the parameter file supplied as an argument. A sample template parameter file is supplied in Appendix B. The following options need to be modified within the file to reflect the particular animation.

OUTPUT

This is used to specify the path and name of the output file generated.

INPUT_DIR

This parameter specifies the directory where all the input images can be found.

INPUT ... END_INPUT

The user should include each file used in generating the animation between these lines in the appropriate order. Sequences of images can be specified with a '*', where the numbers are specified within brackets. For example, img*.yuv [1-3] indicates the files img1.yuv, img2.yuv, and img3.yuv, and img*.yuv [01-07+3] specifies the files img01.yuv, img04.yuv, and img07.yuv.

In this section I described a flexible user interface for the ocean visualization system that allows for customized visualizations. The user can control techniques such as contour lines, glyphs, data mapping, and wire-frame meshes and can also select a predefined viewpoint or define one by specifying the coordinates and distance. I provide support for outputting MPEG animations and VRML models in addition to standard image formats. Moreover, the visualization parameters allow the system to be easily modified for Web-based control. The user can also add new techniques and features to the system, which allows for the necessary extensibility in a scientific visualization system.

4 Discussion

This ocean data visualization system forms the foundation for an automated system in which data can be retrieved from remote locations, processed, and returned to the user in a variety of formats. The current prototype presents a DX-based user interface and supports various image output options. These can also be incorporated into animations and VRML models that can be accessed on the World Wide Web. The visualization techniques developed open the door for new ways of combining and displaying scientific and other data. These build upon a toolkit of methods that can be applied in other situations when visualizing multiple data sets simultaneously.

4.1 System Prototype

The prototype system developed represents a sample of the types of visualizations that can be created and viewed in a distributed environment. The system deviates from traditional distributed, interactive, scientific visualization systems in output quality, types of output produced, and interactivity, and is customized for a specific group of data sets. We focused on producing higher resolution images and output in a VRML model in addition to animations and standard images. We provide the user with more flexibility to apply new visualization techniques to data sets and more control over other areas such as projection and viewpoint.

Rather than produce fly-by animations and other such effects, we emphasized producing visualizations that enhance the user's comprehension of the data and analyze

what he or she is seeing. The sea-surface-height particle animation allows particles to be followed as they move through the Earth's oceans. The importance placed on scaling and accuracy through time make it one of the most accurate visualizations to date. This has been verified by external sources. The ocean-color glyph technique represents a new way of combining multiple data sets while minimizing misleading information in the placement of the particles.

The tradeoff between output quality and responsiveness is noticeable. The longer processing time is due in part to the modular, more flexible data-flow system employed by DX. I chose to ignore output latency for purposes of this prototype. However, various techniques including data preprocessing and enhancing hardware can be used to make interaction more real-time. Although the current system is not optimized for performance, the resources available to us were only moderately adequate for visualizations of this scale. Having more processors for parallel computing as well as having faster central processing units would significantly decrease the time to manipulate the data and produce more timely output. For more information on how data preprocessing can enhance system performance, see the following section, Data Assumptions.

The DX-based user interface is a limiting factor in making the system truly distributed. By maintaining this interface, the features of the system are directly limited to what can be produced within DX. This reduces the output to producing only images. VRML models can also be exported from within DX, but models of any reasonable quality are exceedingly large for transfer over networks and loading into VRML viewers. Currently producing and viewing animation sequences and VRML models require extra

steps for the user outside of the system. Although Web viewing of the final output is achievable, one must still interface with a particular machine to perform the necessary processing.

4.2 Data Assumptions

Working with the data is, in itself, a project on its own. I chose to make some assumptions about the data and suggest some methods for generalizing data import and incorporating it within the visualization system. For the prototype, I explicitly import four data sets that show the user how to register data in different situations.

The system requires that the data be properly formatted before it is input. Until a general data importer is developed to account for the numerous data formats and other variables, all of the data must be preformatted to meet a set of specifications for use in the visualization network. Data must be oriented in a regularly spaced grid centered at zero degrees longitude and zero degrees latitude and placed such that the rows represent latitude values and the columns represent longitude values, ascending from bottom to top and left to right, respectively.

Some data sets require a considerable computation to convert them to the prescribed format. This calculation should be performed in advance, although it can be applied when scaling the data manually within the system. Furthermore, minimum and maximum data ranges should be noted and invalid data points excluded before any computations are performed. Although custom scaling is allowed for each data set, any data whose units are measurements of distance should by default be converted to kilometers for proper scaling with the topological data set and the sphere in the globe

projection. Other data sets should be scaled such that the overall data range stays within a reasonable magnitude. For vector data sets, the value at each location should also take the appropriate time factor into consideration.

Separate visualization networks can be constructed for each logical file group modeled after the prototype. An external interface can dynamically generate a network for each set of files using the scripting features of DX, or by developing a general data importer, to automate this process. Currently, each network must be constructed manually, and is better suited to a system where data is retrieved and processed automatically and a predetermined set of visualizations are updated and placed on a Web server.

Preprocessing the data can significantly speed up the visualization process. When draping one data set over another, a mapping must first take place synchronizing the two data sets on the same grid. With the current set of visualization techniques, this step takes by far the most amount of time – approximately fifty minutes to process a single image. Therefore, one can reduce the bulk of the processing time by mapping each data set onto a common grid before using it in the visualization system. With further advanced registration of the data, filtering and scaling the data appropriately in advance, more valuable time can be saved which will make the system more responsive.

4.3 Future work

One of the first additions to the system should be a Java interface extension that extrapolates the current user interface and brings system control to users via the Internet. This can reside in the form of a Java applet for Web browsers, or a Java application

running on a user's local machine that connects directly to the visualization server. In addition to accessing the user interface remotely, this modification allows VRML models and animations to be directly integrated into the visualization system.

By changing the current user interface controls to parameters that can be supplied externally using the DXLink feature, the user can repeatedly call a visualization network via an external C program with different values for each of the parameters. One way of linking this program and the user interface uses the Common Object Request Broker Architecture (CORBA), which is designed to connect distributed systems written in different languages easily with a set of interface parameters.

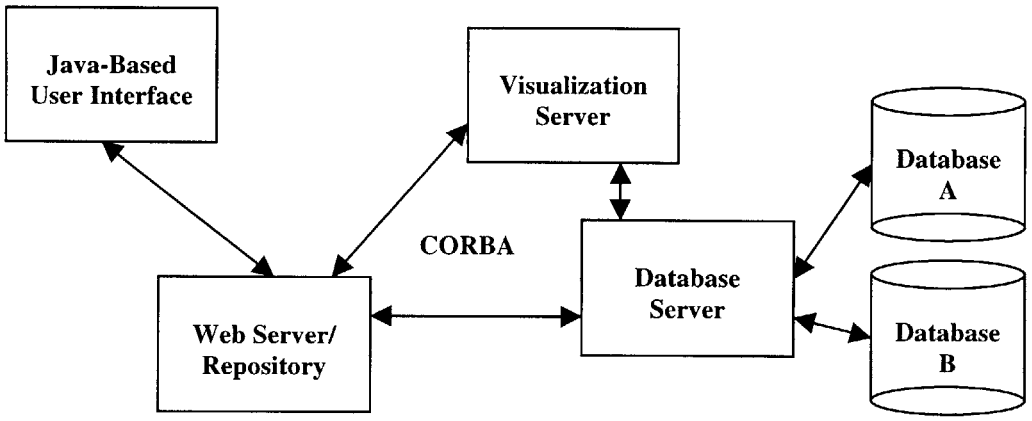


Figure 15. Forecasted system diagram. The user interacts with a local Java-based user interface that interacts with a Web server or repository. Depending on the parameters selected, the appropriate data is obtained through the database server for processing in the visualization server. Communication between the server components is aided by the Common Object Request Broker Architecture (CORBA).

The use of a centralized visualization server for a distributed system poses some disadvantages. Due to the server dedication necessary to process each request, it can process no more than one request at any given time. Adding machines to the visualization server can help disperse this problem to some, but this quickly adds up in both hardware and software costs. If responsiveness is not required, the calling C

program can maintain a queue to process server requests. Caching may also reduce some of the server load.

The addition of a database helps generalize data importation. The numerous data formats present and lack of a standard orientation for data make it difficult to develop a "universal data importer". The database can keep track of a set of values specific to each data set. Some of the parameters that can be used to describe different data sets include longitudinal and latitudinal resolution, whether the data set is centered at zero, orientation, format type, valid data range, appropriate scaling factor, and color map to use. In addition, a database can organize various combinations of data sets that are appropriate for visualization, such as ocean-color, sea-surface-temperature, and sea-surface-height.

Although the system describes a general method of importing various data sets and visualizing them, the visualization techniques presented can in no way encompass the multitude of different viewing possibilities for all data sets. They should, rather, be considered a starting set of features in a visualization toolkit. Visualization is an effective tool because different people can look at data in very different ways depending on their needs. There is no right or wrong method of studying data. One can use these techniques as a guide to stimulate creativity and enhance the analytical process.

Acknowledgments

This research was supervised by Gloria Brown-Simmons, Research Fellow, Center for Advanced Visual Studies, MIT, who developed the overall research agenda, designed the Creative Visualization Program, formed the collaborative team that I was a part of, and secured funding for this project.

Special thanks to Professor Leonard Mcmillan, MIT Laboratory for Computer Science, for offering many interesting and useful technical suggestions for the visualization system and serving as my thesis supervisor. Professor Stephen A. Benton, Director, Center for Advanced Visual Studies, MIT, deserves equal recognition for the support he has given me and for providing me with the structure to work on this project at the Center for Advanced Visual Studies. The effort taken to provide me with essential resources is truly appreciated. Kent Larson, Ben Denckla, and Jim Cain have been valuable resources as well in minimizing my system administration tasks and providing me with support to keep my work more focused and productive.

Professor John Marshall, Stephanie Dutkiewicz, Mick Follows, and Chris Hill, Program in Atmospheres, Oceans, and Climate, MIT, have been a wonderful group of people to collaborate with. Without the ocean-color data expertise and unparalleled enthusiasm during group meetings, my efforts would have been much more difficult. I would also like to recognize Professor Carl Wunsch, Detlef Stammer, and Charmaine King, Department of Earth, Atmospheric, and Planetary Sciences, MIT, for providing valuable experience with the sea-surface-height data processing.

Appendix A: InterpolateStreamlines.c

```
#include <dx/dx.h>


static int   nLines;
static int   nPts;
static int   nDim;
static float *ends;

static int Interpolate(Field, float);
void Regenerate(Array, int);

Error m_InterpolateStreamlines(Object *in, Object *out)
{
   float t;
   int   n, nd, i;
   Array a_samples;
   Object line;

   out[0] = NULL;
   nPts = 0;
   nDim = 0;
   ends = NULL;

   if (! in[0])
      if (DXGetObjectClass(in[0]) != CLASS_FIELD)
         if (DXGetObjectClass(in[0]) != CLASS_GROUP ||
               DXGetGroupClass((Group)in[0]) != CLASS_GROUP)
         {
               DXSetError(ERROR_BAD_PARAMETER, "line group not valid");
               return ERROR;
         }

   if (! DXExtractFloat(in[2], &t))
   {
         DXSetError(ERROR_BAD_PARAMETER, "time parameter not valid");
         return ERROR;
   }

   if (DXGetObjectClass(in[0]) == CLASS_FIELD)
   {
      out[0] = (Object)DXNewArray(TYPE_FLOAT, CATEGORY_REAL, 1);
      return OK;
   }

   DXGetMemberCount((Group)in[0], &nLines);

   if (! in[1])
      if (DXGetObjectClass(in[1]) != CLASS_FIELD)
      {
```

```
            DXSetError(ERROR_BAD_PARAMETER, "samples not valid");
            return ERROR;
    }

a_samples = (Array)DXGetComponentValue((Field)in[1], "positions");
if (! a_samples)
        return ERROR;

DXGetArrayInfo(a_samples, &n, NULL, NULL, NULL, &nd);

if (! n || ! nd)
        return ERROR;

nDim = nd;
ends = (float *)DXAllocate(nDim * (nLines+n) * sizeof(float));

i = 0;
while ((line = DXGetEnumeratedMember((Group)in[0], i++, NULL)) != NULL)
{
        if (DXGetObjectClass(line) == CLASS_FIELD)
    {
            Interpolate((Field)line, t);
    }
        else if (DXGetObjectClass(line) == CLASS_GROUP &&
                DXGetGroupClass((Group)line) == CLASS_SERIES)
        {
            Field segment;
            int s = 0;
            while ((segment = (Field)DXGetEnumeratedMember((Group)line, s++, NULL)) != NULL)
                if (Interpolate(segment, t))
                    break;
        }
        else
        {
            DXSetError(ERROR_BAD_PARAMETER, "line must be field or series");
            if (ends)
            {
                DXFree((Pointer)ends);
                ends = NULL;
            }
            return ERROR;
        }
}

Regenerate(a_samples, n);

if (nPts)
{
        out[0] = (Object)DXNewArray(TYPE_FLOAT, CATEGORY_REAL, 1, nDim);
        DXAddArrayData((Array)out[0], 0, nPts, (Pointer)ends);
        DXFree((Pointer)ends);
}
else
        out[0] = (Object)DXEndField(DXNewField());

ends = NULL;
```

```
    return OK;
}

static int
Interpolate(Field f, float t)
{
    Array a_time = (Array)DXGetComponentValue(f, "time");
    Array a_pts = (Array)DXGetComponentValue(f, "positions");
    int n, nd, i, j;
    float a, b;
    float *pts, *time, *p, *s, *e;
    float d;

    if (! a_pts || ! a_time)
        return 0;

    DXGetArrayInfo(a_pts, &n, NULL, NULL, NULL, &nd);

    if (! n || ! nd)
        return 0;

    pts = (float *)DXGetArrayData(a_pts);
    time = (float *)DXGetArrayData(a_time);

    if (time[0] > t || time[n-1] < t)
        return 0;

    for (i = 1; i < n; i++)
        if (time[i] >= t)
            break;

    if (i == n)
        return 0;

    d = (t - time[i-1]) / (time[i] - time[i-1]);

    p = ends + nPts*nd;
    s = pts + (i-1)*nd;
    e = s + nd;

    // Check if point too close to other points
    for (j = 0; j < nPts; j++)
    {
        a = *(ends + j*nd) - (*s + d*(*e - *s));
        b = *(ends + j*nd + 1) - (*(s + 1) + d*(*(e+1) - *(s+1)));
      if ((a*a + b*b) < 2)
            return 0;
    }

    // Add point to output
    for (i = 0; i < nd; i++, p++, s++, e++)
        *p = *s + d*(*e - *s);

    // Increment point counter
    nPts++;
```

```
    return 1;
}


// Function to determine when to regenerate new points
void Regenerate(Array a_samp, int n)
{
    int i, j, k, max, flag;
    float a, b;
    float *samp, *p, *s;

    max = nPts;
    samp = (float *)DXGetArrayData(a_samp);

    for (i=0; i < n; i++)
    {
        j = 0;
        flag = 1;
        while (j < max)
        {
            a = *(ends + j*nDim) - *(samp + i*nDim);
                b = *(ends + j*nDim + 1) - *(samp + i*nDim + 1);
                if ((a*a + b*b) < 35)
            {
                flag = 0;
                        break;
            }
                j++;
        }

        if (flag)
        {
                p = ends + nPts*nDim;
                s = samp + i*nDim;

                for (k = 0; k < nDim; k++, p++, s++)
                        *p = *s;

            nPts++;
        }
    }
}
```

Appendix B: Sample.param

```
PATTERN             I
OUTPUT              y:\movies\sample.mpg
BASE_FILE_FORMAT    YUV
YUV_FORMAT ABEKAS
YUV_SIZE        720x480
INPUT_CONVERT *
GOP_SIZE        60
SLICES_PER_FRAME    1
INPUT_DIR       y:\images\ssh
INPUT
title.yuv
ssh.*.yuv           [100-671+10]
credits.yuv
END_INPUT
PIXEL           HALF
RANGE           10
PSEARCH_ALG SUBSAMPLE
BSEARCH_ALGCROSS2
IQSCALE             1
PQSCALE             10
BQSCALE             25
REFERENCE_FRAME     ORIGINAL
```

References

Ames, Andrea L., Nadeau, David R., and Moreland, John L. VRML 2.0 Sourcebook. 2nd Edition. New York: John Wiley & Sons, Inc., 1997.

Blumenthal, Benno. International Research Institute / Lamont-Doherty Earth Observatory Climate Data Library, 1998.

Botts, M.E. The State of Scientific Visualization with Regard to the NASA EOS Mission to Planet Earth, *A Report to NASA Headquarters*. Earth System Science Laboratory, The University of Alabama in Huntsville, Huntsville, Alabama, 1993.

Brown-Simmons, Gloria. Creative Visualization Project. Center for Advanced Visual Studies, MIT, 1998.

Distributed Active Archive Center, NASA Goddard Space Flight Center. http://daac.gsfc.nasa.gov/.

Dutkiewicz, Stephanie, et al. *Modelling Interannual Variability of Phytoplankton Abundance in the North Atlantic*. EOS, Transactions, AGU, Volume 79, Number 45, Nov. 1998.

Earnshaw, Rae, Vince, John, and Jones, Huw, ed. Visualization and Modeling. San Diego: Academic Press, 1997.

The Global Change Master Directory. http://gcmd.gsfc.nasa.gov/

Gong, K.L., and Rowe, L.A. *Parallel MPEG-1 Video Encoding*. Picture Coding Symposium, Sacramento, CA, 1994.

Hooker, S.B., W.E. Esaias, G.C. Feldman, W.W. Gregg, and C.R. McClain. An Overview of SeaWiFS and Ocean Color. NASA Tech. Memo. 104566, Vol. 1, S.B. Hooker and E.R. Firestone, Eds., NASA Goddard Space Flight Center, Greenbelt, Maryland, 1992.

Hoque, Reaz. CORBA 3. Foster City, CA: IDG Books Worldwide, Inc., 1998.

IBM Visualization Data Explorer Programmer's Reference. 7th Edition, 1997.

IBM Visualization Data Explorer User's Guide. 7th Edition, 1997.

IBM Visualization Data Explorer User's Reference. 4th Edition, 1997.

MacEachren, Alan M., and Taylor, Fraser, ed. Visualization in Modern Cartography. New York: Elsevier Science Ltd, 1994

Meyer, P.J. *Survey of Visualization and Analysis Tools* (NASA TM-108436). Space Sciences Laboratory, George C. Marshall Space Flight Center, National Aeronautics and Space Administration, 1994.

NetCDF User's Guide for C, *An Access Interface for Self-Describing, Portable Data.* Version 3, 1997.

Szuszczewicz, E. P., and Bredekamp, J. H., ed. Visualization Techniques in Space and Atmospheric Sciences. Washington, D.C.: National Aeronautics and Space Administration.

Wolff, Robert S., and Yaeger, Larry. Visualization of Natural Phenomenon. New York: Springer-Verlag New York, Inc., 1993.

Wunsch, Carl, and Stammer, Detlef. *Satellite Altimetry, the Marine Geoid, and the Oceanic General Circulation*. Annual Review of Earth and Planetary Sciences, 1998.