IICE: BRINGING INTERACTIVITY TO IMAGE-BASED WWW PRODUCTS

David Wojtowicz[1,2], Robert Wilhelmson[1,2], Mohan Ramamurthy[1]
University of Illinois at Urbana-Champaign
[1]Department of Atmospheric Sciences
[2]National Center for Supercomputing Applications (NCSA)

1. INTRODUCTION

Over 250 World Wide Web sites now offer a variety of up-to-date observational and forecast weather maps, radar and satellite images, and the like. This has introduced net-savvy weather data consumers to an unprecedented availability of meteorological information. An ordinary desktop computer and a high speed modem now grants access to data previously only available to the professional meteorologist through expensive services and equipment.

High school students studying basic earth sciences can learn by relating the current conditions in the school yard with their causes as detailed on weather maps they access from their classroom PC. Disaster relief agencies and insurance companies alike can access hourly satellite images of a hurricane preparing to batter the east coast in order to determine where volunteers and claim adjusters will be needed most.

As impressive and valuable an achievement of the information revolution this is, there is often something vaguely unsatisfying about the basic process of selecting a map or image from a list, viewing it for a few moments, dismissing it and then repeating this process for several other images. Not that the process of use could be made much simpler or that we desire to make it more complicated, but perhaps because it remains largely a passive activity.

Sure, browsing the Web is an interactive process, but sometimes not all that much more so than flipping through television channels can be called interactive....especially when all you're doing is picking from lists of images. We expect more from our computers. We know that we can learn faster and understand more if we are able to look at the same data in several different ways, easily compare related variables (like wind and temperature change) and can try out "what if?" ideas. We find displays that focus in on the area in which we live more compelling than generalized national views. In general, how can we achieve greater interactivity when it comes to image based Web products?

Interactive Image Client Environment (or IICE) is a project that seeks to address these needs by designing prototype environments that extend the interactive capabilities of traditional Web environments. The University of Illinois at Urbana-Champaign Department of Atmospheric Sciences is carrying out this work as part of a larger NASA-funded project called HORIZON, which seeks to enable greater public access to earth and space science data through enhancements of and innovations in World Wide Web technology.

From the user's point of view, what specific capabilities should this new more interactive Web environment have? From the developer's point of view, how should these capabilities be implemented? The IICE project has considered both of these questions.

2. INTERACTIVE CAPABILITIES

The concepts presented here are by no means all that one might want to accomplish in this area, however these are some that we consider initially important to work on, especially in the context of providing interactive meteorological products for the Web.

2.1 More Versatility and Control

In general, more versatility and customizability would be very useful. This includes the versatility to select from many possible combinations of meteorological variables and visualization methods for each. It includes the versatility to change analysis parameters such

* *Corresponding Author Address:* David Wojtowicz, University of Illinois at Urbana-Champaign, Dept. of Atmospheric Sciences, 105 S. Gregory Ave., Urbana, IL 61801, davidw@uiuc.edu

[1] See Dan Vietor's "Interactive Weather Maps" at Purdue University: http://thunder.atms.purdue.edu/interact.html or Charles Henrich's "The Interactive Weather Browser" at Michigan State University: http://rs560.cl.msu.edu/weather/interactive.html

as contour intervals. It also includes the customizability to show regional views with more detail than can be presented in national views.

Instead of just choosing from a limited set of pre-defined weather maps, you could design your own map by picking and choosing its components and presentation. For example, you might choose to look at just surface pressure contours at a small contour interval over the northeastern U.S. to study them in detail. Then, you may wish to overlay wind vectors to study the relationship between pressure and wind. You might also want to overlay temperature contours and wind vectors over a satellite image to show frontal positions.

Controls for such customization could take the form of the kinds of option menus, toggle buttons, fill-in blanks, etc. that HTML "forms" already offers. However, the problem with HTML forms is that they are indeed forms. A form is something you fill in, and when entirely complete, you send off and wait for a response. A more interactive mode of operation would allow for the controls to be modified individually with a response occurring after each modification.

2.2 Image Hyperlinking

While one can hyperlink an entire inlined image such as an icon to a new URL, the ability to hyperlink portions of images to additional data would also be useful. We're used to seeing highlighted phrases in HTML documents and know that we will likely be able to obtain additional information on the highlighted topic simply by clicking. However, this capability is not available in image based products (other than with the limited capabilities provided by the HTML "imagemap" feature). For example, clicking on an unfamiliar weather symbol yields no additional information.

Actually, there are several types of image hyperlinking to consider. The most basic is being able to hyperlink a portion of an image to an additional WWW resource such as a descriptive document. The weather symbol example above would be an example of this. The next level would be to associate the actual location on map of a user's click point (in real world coordinates) with data. This might be from a fixed set of points....clicking in the southeast corner of Arizona might yield additional information about the conditions in Tucson or it might be for any point....clicking on a pixel of an enhanced infrared satellite image might return the cloud top temperature at that point.

Of course, this is already possible to a limited extent. Several real-world applications make use of HTML's existing "imagemap" capability which simply transmits the pixel coordinates of a user's clickpoint back to the server where these coordinates are used to select specific data that is transmitted back to the user's browser in the form of a new document. The typical application here is to click on a city shown on a national map and receive back the local conditions and forecast for that city.[1]

However, the actions of image hyperlinking should not be limited to simply causing the browser to fetch a new document or other WWW resource. Often, that approach is too heavy-handed...one has to wait for a new page to load, which replaces the current page and then must be returned from. This is especially the case where the resulting hyperlinked data is small and the number of items to be requested is potentially large. In this case it would be more practical to simply temporarily display the additional information adjacent to the image.

If the information could be derived instantly from data already available to the browser at little computational cost, such information could be displayed instantaneously without clicking as the mouse pointer passes over items of interest. This would be similar to the way many applications programs display brief explanatory messages about certain parts of their user interfaces whenever the pointer is near them such as with the "Balloon Help" feature on Macintosh computers.[2]

2.3 Animation

The animation of weather images over time has always been useful as weather occurs as a sequence of time-ordered events...fronts move across the land, storms grow to full size and later dissipate and hurricanes move along a path, swirling as they do. These actions are best displayed by looping through a set of hourly images. While "movies" of satellite images and such do exist on the Web in MPEG, Quicktime and other similar formats, image quality and the

---

[2] This capability is well demonstrated in relation to weather maps by the University of Michigan's *Blue-Skies* package. It can display temperatures at the top of its window as the user passes their mouse pointer over various cities on a weather map and provides detailed information when a user clicks on a city. This gopher-based application, which works on Macs and PC/Windows, is available at http://blueskies.sprl.umich.edu/WUnderground/Blue-Skies_MacV1.1.html

user's control over the looping are generally limited.

The image quality suffers because the compression schemes used by these formats is generally a "lossy" compression, which additionally is compromised by the fact that most of the movie compression algorithms were designed for full-motion video rather than for computer graphics.[3]

Most MPEG and Quicktime players also offer limited control...generally just the ability to play, restart or stop. A few also offer single step controls. Backwards animation, though sometimes possible in single-step mode, is generally not possible at normal speed. This is because the frames can be decoded in forward sequence only due to the nature of the encoding algorithm. Speed control is also usually absent.

On the other hand, stand-alone meteorological analysis packages often offer more powerful capabilities to step or play forward or backwards at any speed using simple keyboard or mouse controls. This allows one to, for example, "bounce" between two frames allowing one to note small changes. This is aided by the fact that the images have not been degraded by lossy compression.

3. IMPLEMENTATION

Of course, little of this is possible with the Web technology in use at the time of this writing. Certainly, some of this functionality can partially be implemented with existing technology, such as by using the HTML forms to request customized weather maps or using the "imagemap" feature to implement basic image hyperlinking. Obviously though, additional technology is needed.

3.1 Client / Server Division

One must be aware that WWW-based systems have a client component (a user's Web browser running on their local machine), a network component and a server component. These new interactive capabilities will require additional capabilities from and impose additional load on each of these components. Past experience has proven that it is essential to carefully consider how the work is to be divided among these components.

This is well illustrated by recent experience on a related project. The *CoVis Weather Visualizer version 2* is a Web-based tool we recently prototyped here in collaboration with Northwestern University in support of the *Collaborative Learning Through Visualization Project*.[4]

This tool allows one to custom design a current weather map with various backgrounds and overlays. The user makes these selections via HTML forms widgets. All of the processing involved in generating the custom map is handled on the server. This works very nicely...until a few dozen users try to access it simultaneously. At that point, the server is simply overloaded due to the large amount of processing needed per request.

The problem is that server capacity (when the amount of processing per request is large) simply can not scale to keep up with demand. However, client computing capacity scales well with increasing users. The ratio between users and client computers always remains close to one. Therefore, it makes sense to move most of the interactive part of the processing to the client end. Not only does ensure sufficient computational capability for each user, but it simply makes sense to do the interactivity computation as close to the point of use as possible. This is why IICE concentrates its efforts on the development of client technology.

3.2 Turning To Java

Until recently, the problem with implementing capabilities on the client end was the one needed to write, maintain and distribute different software for each type of platform to be used. The effort required to support such software on multiple platforms is well beyond the capacity of most projects and typically, only one or two platforms are supported. Additionally, users had to obtain and install updated versions of this software on their machines in order to use its capabilities.

The introduction of the Java environment, hailed by many experts as THE answer to many of the Web's current limitations, from Sun Microsystems changed everything. This environment consists the object-oriented C++-like Java language, a Java interpreter and a

[3] Contrary to what one might expect, in a good number of cases, the MPEG loops on our WWW server actually take up MORE bytes than the total of the same set of individual images in GIF format....taking longer to transfer and providing less quality!

[4] Please see http://www.covis.nwu.edu/ for more information on this project.

WWW browser called HotJava that has the interpreter imbedded in it.[5]

Programs written in the Java language, after being compiled into a compact and efficient form, can be run anywhere there exists a Java interpreter, regardless of the platform type. The HotJava browser, in addition to being able to display inlined images, can run inlined Java programs...with the interface/output appearing directly inside a HTML document.

For example, many of the Java demo documents include what appears to be an inlined image of the Java cartoon mascot, Duke. However, it is not an inlined image, but the display of a small inlined Java program called an applet. This enables Duke to continually wave at you. The code for these applets is loaded from a standard WWW server at the time of use just as inlined images are. This code is executed in a secure, contained environment such that potentially ill-intentioned code simply does not have sufficient access to the rest of one's system to do anything nefarious such as erase files, plant a virus or search through confidential data.

The platform independent nature of Java ends the problems with having to port a client application to many different platforms. Its load on demand capability allows one to centrally maintain and update a copy that is automatically accessed by anyone who needs to reference it.

At first thought, one way to go about implementing our interactive capabilities would be to build entire analysis and display tools in Java for the user to run on their end and simply ship them the raw data. The problem here is that it is not possible to incorporate the capabilities of the large number of existing software packages and libraries that already do very good analysis. One would have to reproduce these capabilities in the Java language from scratch.

Instead, the idea is to develop a basic set of Java classes (a class is roughly a unit of software capable of performing a set of operations on certain kinds of data) which could be customized as needed. These classes would perform basic interactive operations and the server would use existing and new analysis packages to generate semi-preprocessed data to be used by the client.

This later approach does put some of the work back on the server, but only to generate one set of each of the possible analysis components which can be reused until they expire. The client is responsible for combining them in any of the thousands of possible user chosen combinations and dealing with other user interaction.

[5]See Sun's Java Home Page at http://java.sun.com/ for full details.

4. CLIENT CAPABILITIES

The following outlines some of the basic capabilities that IICE will implement as Java classes:

4.1 User Interface Controls

These classes will implement controls similar to those found in HTML forms such as option menus, checkboxes, fill-in blanks, pushbuttons, etc. Just like HTML forms widgets, these will be specifiable by HTML-like tags and can be flexibly inlined in documents. The difference is that they are not restricted to the simple behavior of HTML forms which simply transfers the values of all the widgets in the form to the server and awaits a new document in response. Individual actions can be attached to each making it possible to implement immediate responses to user interaction with them, such as immediately verifying the validity of a user's input or causing a new display to be generated.

4.2 Image Construction

These classes will be used to construct the images from the pre-processed analysis components provided by the server. A typical instance would be to overlay contour plots. The data (in the form of pre-generated contours) for the each data variable would be fetched from the server as needed. These classes would then be used to composite them into a single image. If the user should turn off or on one of the components, the layer of the composite containing that component would simply be removed or replaced.

Mentioning pre-generated contours above brings up the question as to what form this data is in. The simplest way to do this would be to have the contours plotted on otherwise blank images and then composite the image layers together. However, one concept that IICE is likely to explore is the use of a CGM-like system (CGM stands for Computer Graphics Metacode). In such a system, drawing commands such as "draw line segments through these points" are transmitted rather than an image.

The use of a CGM-like system has some particular advantages, the first of which is that for simple vector graphics such as contour plots, the drawing commands take up fewer bytes than would the resulting image of a reasonable size.

This would, of course, reduce network load and improve transfer time. Another advantage is that this gives the client the ability to draw high quality graphics at any user defined size. Components of a raster nature such as satellite images would still be transmitted in raster form as a special case.

4.3 Image Hyperlinking / Information Extraction

These classes will retrieve additional information from the server that specify pre-defined areas in the image that either display data when the mouse pointer passes over them (the data to be displayed is also sent) or hyperlink to additional resources when these areas are clicked.

Additionally, control can be transferred to a Java routine in either of these cases. Several such pre-defined routines will be provided. One will have the ability to translate colormap entries to data values (using a table retrieved from the server). This can be used to implement an interactive image that shows cloud top temperatures as the user moves the mouse pointer over the image.

4.4 Animation

This class will provide for simple animation of several images, both ones that are constructed locally and ones that are retrieved from the server. As discussed earlier, this has advantages over commonly used “movie” formats. Controls will be provided to play and step both forward and backwards at various speeds.

5. ADDITIONAL INFORMATION

Additional information concerning the IICE project can be found on the IICE Home Page: http://www.atmos.uiuc.edu/horizon/subprojects/iice/.