The Information Mural: A Technique for Displaying and Navigating Large Information Spaces

Dean F. Jerding and John T. Stasko
Graphics, Visualization, and Usability Center
College of Computing
Georgia Institute of Technology, Atlanta, GA 30332-0280
{dfj,stasko}@cc.gatech.edu

Abstract

Visualizations which depict entire information spaces provide context for navigation and browsing tasks; however, the limited size of the display screen makes creating effective global views difficult. We have developed a technique for displaying and navigating large information spaces. The key concept is the use of an Information Mural, *a two-dimensional reduced representation of an entire information space that fits completely within a display window or screen. Information murals use grayscale shading and color along with anti-aliasing techniques to create a miniature version of the entire data set. By incorporating navigational capabilities, information murals become a tool that can be used as a global view along with more detailed informational displays. Information murals are utilized in our software visualization research to help depict the execution of object-oriented programs, and can also be used in more general information visualization applications.*

1 The Problem

Emerging networking and database technologies are making vast amounts of information available on-line. However, the information itself is useless without effective display, query, and browsing mechanisms. Information visualization research must be concerned not only with how visual displays of this information are constructed, but also with how the information displays can be navigated. We believe that navigational mechanisms in visualization systems are as important as the visual displays themselves.

Developers of effective information visualizations are confronted with a fundamental problem: Design a visualization of a particular information space that 1) displays what the user wants to see, and 2) allows the user to focus quickly on areas of interest. The type of information being examined often dictates the answer to part one, while more generic techniques can be applied in part two.

A particular area of our research involves the use of visualization techniques to help programmers understand and debug object-oriented programs. For example, consider an object-oriented program which has been annotated to produce a trace of interesting events upon execution. These events include function invocations/returns and objects being created/destroyed, among other possibilities. From the function invocation trace, the sequence of messages sent during the program execution can be determined.

A textual display of such voluminous information is difficult to read and understand. A graphical view of the message trace could better help a software developer understand what occurs during a program's execution. One potential visualization of this information might show classes on the vertical axis and the progression of time on the horizontal axis. A message from one class to another would be depicted as a line between the two classes, at the appropriate time coordinate. The visual appearance would thus be a sequence of vertical lines representing messages sent from class to class during the execution of a program.

Such a view should allow a user to browse the entire stream of messages, which in practice could easily number in the hundreds of thousands. The problem then is how to concisely present the messages and allow users to easily navigate through the message display. Before designing the navigation mechanisms for such a view, we first looked at alternatives based on related work in information visualization. As a result, we have developed a technique that can be applied to more general types of information visualizations.

1.1 Browser Alternatives

Plaisant, Carr, and Shneiderman recently proposed a taxonomy of *image-browsers*, or applications which browse a graphical image in one or more dimensions[10]. The remainder of this section discusses possible alternatives for our message view, along the categories of their taxonomy.

Detail-only. One potential method for displaying the example message trace information is in a single detail-only view. This presentation would allow users to see and interact with messages drawn at a reasonable scale.[1] Since the messages may number in the hundreds of thousands or more, some mechanism for scrolling or panning through the entire message trace must be provided. The simplest solution is to provide a scrollbar for such navigation.

Scrollbars provide two important cues to the user: relative size of the currently displayed information (the current focus) compared to the size of the entire information space, and the relative position in the overall information space of the current focus. However, scrollbars do not provide

[1]A reasonable range would be from one to eight pixels wide.

semantic information about the *contents* of the focus area relative to the *contents* of the entire information space.

We believe that a message view should permit the user to observe a representation of patterns in the entire message trace, and selectively zoom in on areas of interest. Adding zooming to a detail-only view allows the user to shift between a very detailed display and a more global overview. While the global perspective provides context for the details of the display, with a single view zooming to reveal details eliminates the context (and vice-versa).

Fish-eye. *Fish-eye* or *focus+context* views address this problem. These views display information of particular interest in detail, and show the rest of the contextual information in a smaller representation relative to the areas of focus. The main advantage is that all of the information fits in a single view, and the user is relieved from having to shift back-and-forth from a detailed view to a separate global view. Animation is often used to allow interactive, real-time update when the area(s) of focus is changed.

An early example of a fish-eye view was developed by Spence and Apperly[16]. Their Bifocal Display uses a central focus area to display detailed information and two "de-magnified" regions on each side to display the rest of the information in a more compressed form.

Furnas formalized the fish-eye display concept, where the size and position of information in a display is calculated using a degree-of-interest (DOI) function[6]. The DOI function is typically defined using distance from the current focus and *a priori* importance (API).

A number of other fish-eye methods have been proposed. A brief description is included in the following list.

- The Perspective Wall of the Information Visualizer shows a detailed section of an information space in 2D, while also showing the remaining information in a 3D perspective[8]. It works well for linear information spaces that are much larger in one dimension than the other. Using the 3D perspective view allows more information to be displayed than would be possible with just two dimensions.
- Sarkar and Brown developed fish-eye views of planar graphs[14]. This work was extended to more general layouts using rubber sheet stretching metaphors[15]. By placing handles on objects in a view, areas of detail can be defined and sized using a stretching metaphor.
- The Document Lens applies the Perspective Wall technique to general 2D information spaces[13]. The strategy shows a 2D focus area (or lens) at a detailed scale, and distorts the rest of the information in perspective using a truncated pyramid.
- Magic Lens filters change the graphical display of information by applying a viewing operation to the underlying data[18]. These movable lens can show different details within arbitrary regions of interest, while still preserving context outside of those regions.
- The Table Lens applies fish-eye viewing techniques to table-oriented data[12]. By combining symbolic and graphical representations, various rows, columns, or cells can be shown at different levels of focus.

Although combining the focus and context into a single view can be worthwhile for certain information visualizations, focus+context views suffer from some problems. As the slope of the DOI function increases, the contextual information in the display becomes more distorted, making it difficult to perceive contextual relationships. Additionally, the shear volume of information relative to the size of the focus area may force the DOI function to be almost a step function in order for all the data to be displayed. The size of the distant information may then become so small that it cannot even be perceived. Worse, the information space may contain more elements than there are pixels in the display.

Another serious problem with focus+context views is performance[14]. With every change in focus, the size and position of every element in the view may change. In order to smoothly animate these changes, the entire view has to be redrawn many times per second. This may be difficult for large amounts of information. The voluminous nature of our object-oriented program message trace presents such problems.

Multiple-views. A third alternative is to separate the detailed view from the global view. Beard and Walker use a navigational or *map* window to show a miniature version of the entire information space along with some sort of "you-are-here" indicator[2]. In their system, the global map window supports roaming and zooming over the entire information space, which contains a balanced binary tree of words. Experiments confirmed that user performance during navigational tasks improved using the map window instead of horizontal and vertical scrollbars.

Many other applications utilize multiple-view visualization displays, including medical imaging, geographic information systems, data visualization, and CAD/CAM. For our program visualization application, using multiple views would allow the user to interact with the message display in a detailed portion of the view, and provide a separate area to display the entire message trace.

1.2 Our Solution

We have developed a method for displaying and navigating large 2D information spaces using the multiple-view technique. This work is derived from our implementation of the message trace visualization. The next section describes the design of the message view and its navigation mechanisms. We call our views of large information spaces *Information Murals*, and describe them in Section 3. In Section 4 we discuss several application areas where the information murals are useful, and compare our methods with related work in those areas.

2 The Execution Mural

As mentioned in the first section, one area of our *software visualization*[11] research involves visualizing the execution of object-oriented programs. As a component of an integrated set of views, we are designing a display of the messages exchanged between objects during the execution of a `C++` program. This section describes the *Execution Mural*, focusing specifically on the visual mechanisms used to provide navigation capabilities. While the current state of the design does not contain all the functionality envisioned for this view, it does provide an effective demonstration of our methods. The techniques discussed in this section will be generalized to other information spaces in Section 3.

2.1 Program Information Generation

In order to create an Execution Mural for a given program, static and dynamic information about that program are required. Static information includes details regarding classes and functions that exist in the program. Dynamic information includes a trace of interesting events that occur during the program's execution, such as instance creation/deletion and function invocation/return. The Sage++[19] toolkit for designing `C++` profilers is currently utilized to collect this information. We have written a `C++` instrumentor which performs annotation of source code to generate the necessary trace information, based partly on work done at the University of Oregon[9]. When the annotated source is compiled and executed, a trace of the interesting events is created. A separate static analyzer reads Sage++ descriptions of source files and generates information on classes and functions in the program. These tracing techniques can be activated automatically during compilation; the programmer is not required to do any manual annotation of the code.

2.2 Presentation of the View

Given the static and dynamic information for a particular `C++` program, a program simulator is used to recreate what happened during the program's execution. While the simulator executes, information pertinent to the Execution Mural's display is stored. This information is then used to drive the visual display.

The majority of the Execution Mural view is occupied by the focus area, which presents the basic message representations. Classes in the program are assigned rows along the vertical axis. The classes are shown in the same order as they are declared in the header files of the program. A single "global" class is used to represent `C++` global functions (non-member functions). The horizontal axis represents time, or progression of the program's execution. Each message in the program is depicted as a gray vertical line originating at the source class and ending at the destination class. A red rectangle at the originating end of the line marks the source class. Messages are drawn corresponding to either function invocations or returns, with the latter shown as a dashed line. The width of the line representing a message can be changed by the user.

Only a small portion of the entire set of messages can be displayed in the focus area at a particular time. Additionally, there may be more classes in the program than can fit on the visible vertical axis. In order to provide the user with a global view of the entire message trace, the bottom section of the view shows a reduced representation of the entire set of messages. This portion of the view is an example of what we call an *Information Mural*. The details of information murals are discussed further in Section 3. Essentially, anti-aliasing techniques along with grayscale shading are used to reproduce a miniature drawing of the entire message trace. The representation can effectively show patterns in the program execution, even when the ratio of messages to pixels is greater than 100:1.

The global portion of the view includes a navigation rectangle which indicates where the focus area fits within the entire execution. This view also serves as a navigational widget for the focus area, providing panning and zooming capabilities. Execution Murals of a bubble-sort algorithm animation program are shown in Figures 1 and 2.

2.3 Interacting with the View

The Execution Mural helps programmers understand the execution of an object-oriented program. By examining the message trace, users can discover phases in the execution, relationships between classes, and generally how the objects accomplish the functional purpose of the program. Users will often only be interested in particular classes or in particular parts of the execution. This view must then make it possible to

- uncover/investigate global patterns in the execution
- focus on the details of particular execution phases
- examine the role of different messages
- focus on relationships between particular classes

The interaction and navigation mechanisms which support these tasks are of first class importance if the Execution Mural is to be effective. Because each individual interaction with the view may be for a different program understanding task, we have adopted the strategy of presenting all information to the user and letting him or her highlight and ellide information as appropriate.

2.4 Usage Scenario

In order to describe the interaction techniques implemented in the view and give the reader a feel for how the Execution Mural can assist with program understanding tasks, an example usage scenario is presented. Obviously, the static textual medium does not really convey the visual effectiveness of this view, but it gives the reader an idea of what can be done. The example deals with an algorithm animation created using the Polka animation toolkit[17].

The Polka animation toolkit is an object-oriented library which supports the development of software visualizations and algorithm animations. In this animation, two views display an array of four values being sorted using the bubble-sort algorithm.

When the Execution Mural first opens, the display looks like that of Figure 1. There are 38 classes and over 40,000 messages in this program. The way in which the Execution Mural is utilized to help the user understand a program is very much task dependent. The mural of the entire message trace provides a good starting point, especially for more global, high-level understanding tasks. In this case it is clear that there are about four different "phases" in the execution, of which the middle two appear to contain a very repetitive pattern.

The mouse can be used to pan the detailed view and zoom in on subsets of the message mural, allowing further investigation of the message trace at a more detailed level. In the focus view, the left mouse button can be used to highlight the message under the mouse pointer. A text field above the display shows the name of the message, and the name of the source and destination classes. Pressing the right mouse button while a message is highlighted opens a text window showing the program source code, positioned at the function which was called in response to that message.

The message options dialog box allows the user to alter the width of messages in the focus area, to make viewing individual messages easier. Additionally, the color of individual messages can be changed, permitting highlighting of particular messages in the program. For example, we can set the color of all constructors in the program to blue. It is then quite apparent where new objects are created in the execution–mostly in the early initialization phase of the program.

Another possible task might be to explore the relationship between specific classes in the program. The class options dialog box provides control over which classes are displayed. By default, messages sent and received by all classes are shown. However, if the user is not interested in a particular class, (s)he may choose to hide that class's messages. The messages to and from hidden classes disappear and the labels for the hidden classes are grayed out.

Dragging with the right mouse button in the global view can be used to select a part of the message trace to replace the existing global view. In Figure 2, the first third of the execution is displayed in the global information mural. Some of the classes have been hidden to focus on particular class relationships, messages which correspond to constructors are blue, and the message width has been set to three pixels.

3 Information Murals

Within the confines of typical computer displays there are only a limited number of pixels. As the amount of information in a visualization increases, the informational entities quickly outnumber the pixels. A form of abstraction is often used to superimpose a hierarchy onto the information. However, information is not always hierarchical in nature, and a useful display which can show all of the information is usually preferable to one showing only a subset.

The visualization and navigation techniques implemented in the Execution Mural described in the previous section can be extended to create visualizations of other large information spaces. We use the term *Information Mural* to describe a reduced representation of an information space that fits entirely within a display window or screen. Many examples exist of information visualizations that cannot be displayed completely within a window or screen. If a user of such a view were presented with an information mural of the data, navigating the information display would become easier.

There are several different types of information spaces which could be represented using information murals:

- A text file or document usually does not fit entirely on the screen, because its vertical dimension far exceeds its horizontal dimension. Typically, a text editor displays only a portion of the file being edited.
- Graphs of data often require some compression technique to fit on the screen. Scaling and rounding of data values is often necessary to draw the entire graph. Other alternatives are to display an average of the data values, or only a subset of the data.
- Program visualizations often span many computer screens if laid out completely. This is especially true for those views where one dimension corresponds to time. The Execution Mural discussed in Section 2 is an example of such a view.
- Even images might be represented using information murals. Although an image usually fits on a screen, it is often desirable to change the size of the image. As an image is shrunk, information in the image is inevitably lost.

3.1 Creating an Information Mural

The idea of an information mural is to take the basic visual representation of information and scale it to fit into the available space on the display. Instead of rounding the position of information elements to the nearest pixel or using a form of abstraction to compress the representation, anti-aliasing techniques and grayscale shading are used to create a miniature version of the entire dataset. These strategies are well-known in computer graphics and adapt well to the information visualization problems that we address.

An anti-aliasing technique is used render informational entities that when scaled into the available screen space fall in-between actual pixels. Since data points may lie almost on top of one-another, grayscale shading is further used to represent the density of the data at each particular pixel. The more information compressed into that pixel, the darker the pixel's color.

The basic algorithm for creating an information mural is listed below. The algorithm takes an image of `M x N` elements and scales it into a mural of `I x J` pixels. In addition to the data structures which store the information, the algorithm requires an `I x J` array of floats:

```
1. for each $i,j$ set $mural\_array[i][j]$ to zero
2. for each element $m,n$ of information
     a. compute $x = m/M * I,\ y = n/N * J$
     b. determine the proportion of this
        point that lies in each of the four
        surrounding $mural\_array$ entries
        (totals to 1.0):
        $mural\_array[floor(x)][floor(y)]$
        $mural\_array[floor(x)][ceil(y)]$
        $mural\_array[ceil(x)][floor(y)]$
        $mural\_array[ceil(x)][ceil(y)]$
     c. add each of the proportions
        determined in the previous step to
        the existing values of each
        corresponding $mural\_array$ entry
4. set $max\_mural\_array\_value$ to the maximum
   $mural\_array[i][j]$, of all $i,j$
5. for each $i,j$ in the $mural\_array$
     a. compute the grayscale value
        corresponding to
        $mural\_array[i][j]/max\_mural\_array\_value$,
        with a value of 0.0 being white and
        1.0 being black
     b. color the pixel at $i,j$ of the mural
        with the grayscale value computed
        in the previous step
```

In actual implementations of the algorithm, one can take advantage of the information structure to make the algorithm more efficient.

A useful information mural will convey the same visual patterns to the viewer as the original image, although some of the detail may be lost. We do not propose information murals as a generic technique that can be used to browse any form of information. The distribution of information in the original image may be such that a useful information mural cannot be created.

3.2 Information Murals as Navigational Tools

Information murals can be used as global views of information spaces. In this manner, they can serve as navigational views to support panning and zooming of more detailed focus views. Plaisant, Carr, and Shneiderman point out the importance of global views in a multiple-view browser to allow quick access to detailed information[10].

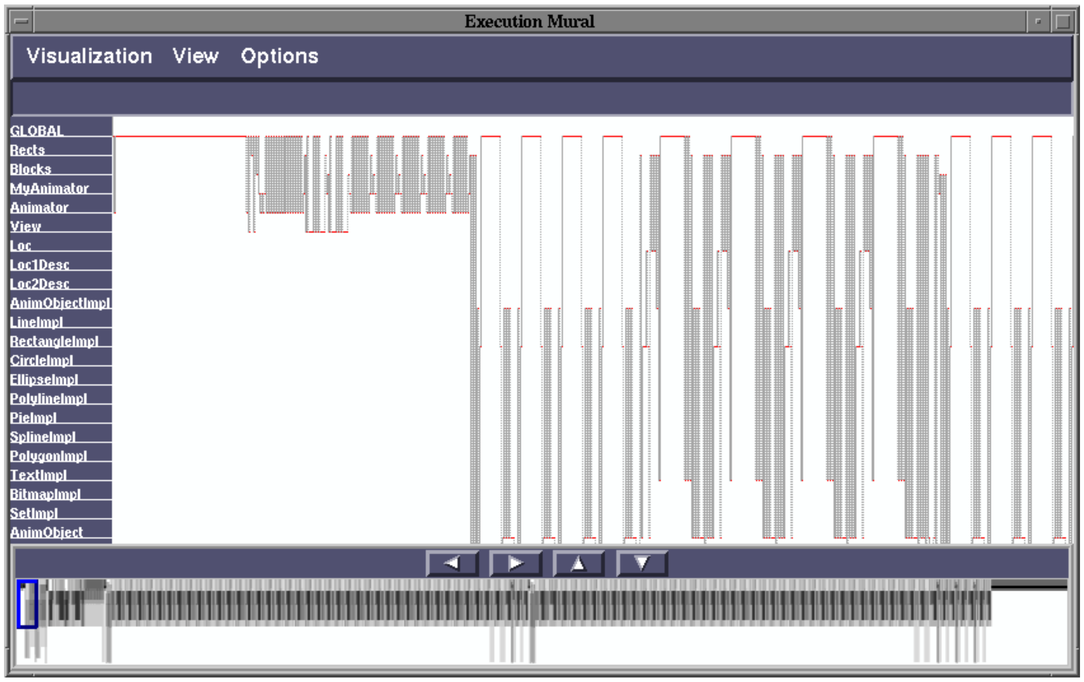


Figure 1: Initial Execution Mural of a Polka bubble sort animation. The focus area displays only a portion of the 38 classes and 40,000 messages in this animation. The global/navigation view at the bottom shows an information mural of the entire message trace.

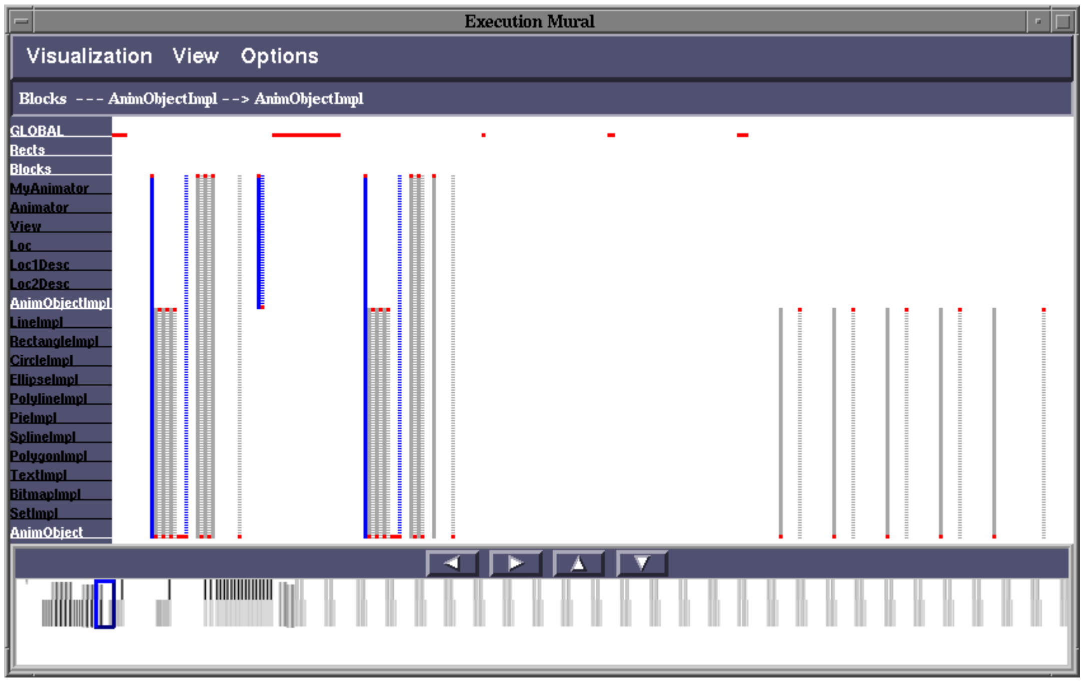
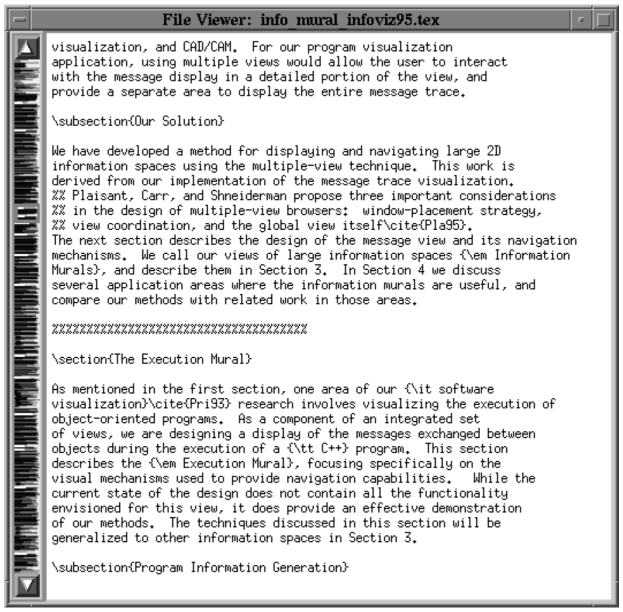


Figure 2: Execution Mural of bubble sort after hiding classes and zooming in on first third of the execution. Messages which correspond to constructors are colored blue. Note that the global view in this figure represents the first third of the information mural shown in the previous figure.

While it is clear that global views are important, the real problem that information murals address is *how effective representations of large information spaces are created*. Without a good visual representation, the global view cannot serve as an effective navigation tool. The usefulness of a visualization tool often depends on the effectiveness of its navigation capabilities: Can the user navigate quickly to locate an area of particular interest?

The idea of using a separate view to support navigation is not new[2, 10]. However, the techniques we use to create the information mural allow the display of very large information spaces in a small area. Most current global navigational views either work with smaller information spaces whose dimensions are not much more than several times the screen size, or show large information spaces using abstraction. In contrast, information murals have been created which compress informational entities into pixels with ratios of more than 100:1.

The information mural can be used as the background for a 1D or 2D navigational viewspace. Such a tool is called a *Mural Navigator*, which we have implemented as an abstract "widget" in X/Motif. It can be used within an information visualization application to act as a global navigational view, or by itself as a standalone browser of an entire information space. Several panning and zooming capabilities are provided which allow the user to control the presentation of the global and/or detailed information in the application.



```
visualization, and CAD/CAM.  For our program visualization
application, using multiple views would allow the user to interact
with the message display in a detailed portion of the view, and
provide a separate area to display the entire message trace.

\subsection{Our Solution}

We have developed a method for displaying and navigating large 2D
information spaces using the multiple-view technique.  This work is
derived from our implementation of the message trace visualization.
%% Plaisant, Carr, and Shneiderman propose three important considerations
%% in the design of multiple-view browsers:  window-placement strategy,
%% view coordination, and the global view itself\cite{Pla95}.
The next section describes the design of the message view and its navigation
mechanisms.  We call our views of large information spaces {\em Information
Murals}, and describe them in Section 3.  In Section 4 we discuss
several application areas where the information murals are useful, and
compare our methods with related work in those areas.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

\section{The Execution Mural}

As mentioned in the first section, one area of our {\it software
visualization}\cite{Pri93} research involves visualizing the execution of
object-oriented programs.  As a component of an integrated set
of views, we are designing a display of the messages exchanged between
objects during the execution of a {\tt C++} program.  This section
describes the {\em Execution Mural}, focusing specifically on the
visual mechanisms used to provide navigation capabilities.   While the
current state of the design does not contain all the functionality
envisioned for this view, it does provide an effective demonstration
of our methods.  The techniques discussed in this section will be
generalized to other information spaces in Section 3.

\subsection{Program Information Generation}
```

Figure 3: Editor showing LaTeX file of this paper. Red is used to highlight section headers, and pink is used to highlight sub-sections.

4 Sample Applications

Given that an information mural supports visualizing and navigating large information spaces, this section presents examples of possible application areas. We have implemented simple visualizations to demonstrate the Mural Navigator's usefulness.

4.1 Document Editing

Standard text editors provide horizontal and vertical scrollbars for scrolling through a document. While scrollbars provide size and position information along with their navigational features, they provide no informational context for the subset of the file displayed in the editor.

Silicon Graphics, Inc. multi-media presentation tool IRIS Showcase^{TM} provides a view displaying a miniature version of each page which can be used to make changes to the pages and navigate through their document. Because the view is intended to be more than a navigational tool, it is separate from the editor and can take up a large amount of screen space.

The SeeSoft tool developed at Bell Laboratories displays files graphically, using one row of pixels to represent one line of the file[5]. By appropriately indenting each line, a miniature image of the entire file is created. SeeSoft conveys information about the contents and structure of the file by appropriately coloring the individual lines. The original implementation of SeeSoft requires one row of pixels for every line in the file, and the display wraps around for files which do not fit in a single column. More recent SeeSoft implementations have tried to reduce the required screen space by wrapping multiple lines of the file into a single row of pixels. A number of other applications have adopted this technique to display information about program files[1, 7].

The information mural technique can be used to create a reduced representation of a text file even when the number of lines is greater than the number of available pixel rows. The compression preserves the overall structure of the document because it uses shading to overlap adjacent lines rather than wrapping them. Color can be added to the mural to display structural details such as comments, function declarations, page breaks, or highlighted regions.

Figure 3 shows a simple text editor with the LaTeX version of this paper. Color is used to mark section headers and highlighted regions. A Mural Navigator could easily be added to a word processing application, providing a miniature image of each page in place of the scrollbar. This contextual information about the document could greatly reduce the navigational time between the user deciding (s)he needs to move to a certain position and the moment that position is actually reached.

4.2 Data Visualization

The Mural Navigator can be incorporated into a data visualization view to support one- or two-dimensional navigation through a large data space. For example, a detailed plot can be drawn at a scale which fits on the screen and the Mural Navigator used as a global view to show the entire dataset. Displaying large amounts of data using an information mural allows the density or overlap of data values to be indicated even when there are more than 100 data values mapped onto a single pixel.

Recently, Eick has proposed the concept of data visualization sliders[4]. These sliders are used mainly for filtering the visual display of information. The background of the slider typically shows the distribution of the data within the range of values, either marking individual values or showing a continuous distribution. The slider itself is used to set visual attributes (such as color) for regions of the data shown in a more detailed view. The Mural Navigator is not intended as a filtering device, but rather as a navigational tool which displays all of the actual data values, providing contextual relationships between a detailed subset of the data and the entire dataset.

Chimera's Value Bars are used to analyze the global distribution of data attributes for multi-attribute listings which typically have a hierarchical structure (such as directories of files)[3]. Each item in the entire listing is given an area in the value bar proportional to the value of some particular attribute (such as file size). A value bar whose attribute is the information itself might be considered an information mural, although it is not clear that value bars can scale to display attributes of large information spaces.

Examining a visual plot of data to uncover basic patterns and characteristics in the data set would be a useful first step before proceeding with more detailed analyses. Simply plotting large data spaces can be difficult with current tools. Typical spreadsheets which run on personal computers have difficulty processing large data files. Dedicated data analysis and visualization packages such as PV-WAVE^{TM} or **S** can plot large datasets, but they typically use over-plotting methods which obscure the true density of data values.

For example, assume that the data is in the form of a square wave. If the wavelength is such that an entire cycle is compressed into one pixel, the plot will be a line from one peak value to the other peak value. If the data is a simple pulse waveform with one narrow pulse of that same amplitude, the resulting plot will look the same. An information mural of the square wave data using grayscale shading would be black at the peak values and the line connecting them would appear gray, indicating to the viewer that there is a concentration of data points at each of the peak values. For the pulse waveform, the low value would be dark, with the upper peak and the connecting line gray. In practice we may encounter large data sets with local behavior that will not be depicted using standard plotting methods. Using the information mural, some indication of the behavior will be apparent.

Sun Spot Data. The first example is a visualization of sunspot data collected daily from 1850-1993, a total of over 52,000 data points. Figure 4 shows a plot of the data, with a Mural Navigator used as a one-dimensional slider to control the portion of the dataset shown in the plot. In the Mural Navigator, the width of the navigation rectangle is almost nil because 52,000+ values are being shown in the mural. Darker portions of the mural indicate that there were many days with no spots recorded.

Dow Jones Industrial Average. The second example plots the Dow Jones Industrial Average from 1915-1990, a dataset with over 19,000 daily values. Because the data varies widely in the x and y dimensions, a two-dimensional Mural Navigator is used along with the plot. The plot shows the high, low, and closing values for each day. Figure 5 shows a snapshot of the visualization.

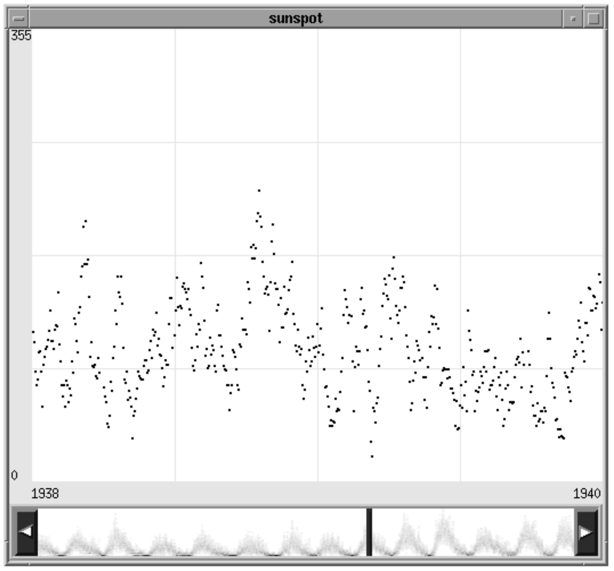


Figure 4: Plot of sunspot numbers recorded daily from 1850-1993, a total of over 52,000 values. The Mural Navigator at the bottom shows an information mural of the whole dataset.

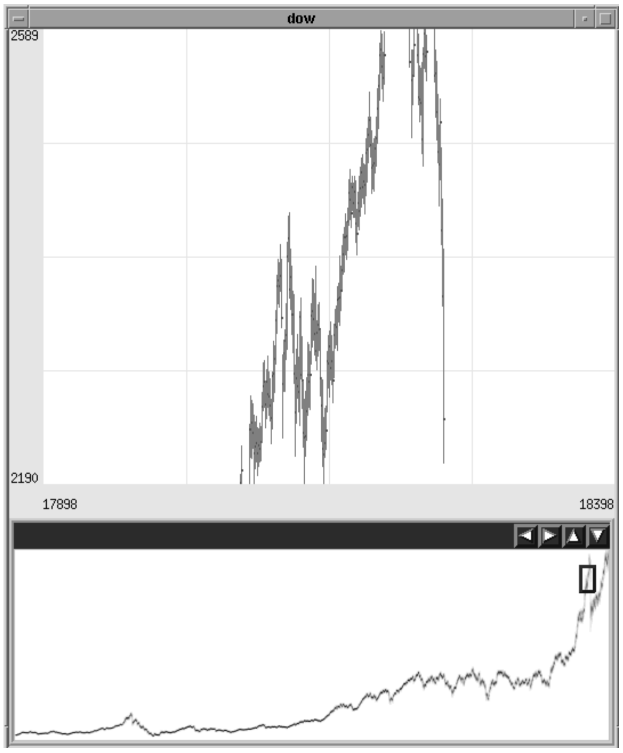


Figure 5: Plot of daily Dow Jones Industrial Average high/low/close from 1915-1990, with the information mural showing over 19,000 daily readings.

5 Conclusions and Future Work

An Information Mural is a graphical representation of a large information space which fits entirely within a display window or screen. The miniature representation is drawn using anti-aliasing techniques and grayscale shading, and is useful for visualizing trends and patterns in the overall distribution of information. By adding panning and zooming capabilities to an information mural, a Mural Navigator can be used as a global view along with more detailed informational displays.

The techniques used to create information murals can be integrated into various information visualization applications to help display large information spaces. We have demonstrated applications in program visualization, data visualization, and information visualization. Although our techniques work in two dimensions, they could be applied in 3D visualizations such as the Perspective Wall[8] or the Document Lens[13] to better present the information compressed in the perspective parts of these views.

The current information mural implementation uses a grayscale (white to black) intensity variation on a white background. We are experimenting with different encodings such as grayscale from black to white, and an equalized intensity color scale. We are also exploring the use of color on top of the mural to depict attributes of the underlying data.

References

[1] T. Ball and S. G. Eick. Visualizing program slices. In *Proceedings of the 1994 IEEE Symposium on Visual Languages*, pages 288–295, October 1994.

[2] David V. Beard and John Q. Walker II. Navigational techniques to improve the display of large two-dimensional spaces. *Behaviour and Information Technology*, 9(6):451–466, 1990.

[3] R. Chimera. Value Bars: An information visualization and navigation tool for multiattribute listings (demo summary). In *Proceedings of the ACM SIGCHI '92 Conference on Human Factors in Computing Systems*, pages 293–294, 1992.

[4] Stephen G. Eick. Data visualization sliders. In *Proceedings of the 1994 ACM Symposium on User Interface Software and Technology*, pages 119–120, November 1994.

[5] Stephen G. Eick, Joseph L. Steffen, and Eric E. Sumner Jr. SeeSoft—A tool for visualizing line oriented software statistics. *IEEE Transactions on Software Engineering*, 18(11):957–968, November 1992.

[6] George W. Furnas. Generalized fisheye views. In *Proceedings of the ACM SIGCHI '86 Conference on Human Factors in Computing Systems*, pages 16–23, Boston, MA, April 1986.

[7] Doug Kimelman and Bryan Rosenburg. Strata-Various: Multi-layer visualization of dynamics in software system behavior. In *Proceedings of the IEEE Visualization 1994 Conference*, October 1994.

[8] Jock Mackinlay, George G. Robertson, and Stuart K. Card. The Perspective Wall: Detail and context smoothely integrated. In *Proceedings of the ACM SIGCHI '91 Conference on Human Factors in Computing Systems*, pages 173–180, New Orleans, LA, May 1991.

[9] Bernd Mohr. A portable dynamic profiler for c++ based languages. Technical report, University of Oregon, 1993.

[10] Catherine Plaisant, David Carr, and Ben Shneiderman. Image-browser taxonomy and guidelines for designers. *IEEE Software*, 12(2):21–32, March 1995.

[11] Blaine A. Price, Ronald M. Baecker, and Ian S. Small. A principled taxonomy of software visualization. *Journal of Visual Languages and Computing*, 4(3):211–266, September 1993.

[12] Ramana Rao and Stuart K. Card. The Table Lens: Merging graphical and symbolic representations in an interactive focus+context visualization for tabluar information. In *Proceedings of the ACM SIGCHI '94 Conference on Human Factors in Computing Systems*, pages 318–322, Boston, MA, April 1992.

[13] George G. Robertson and Jock D. Mackinlay. The Document Lens. In *Proceedings of the 1993 ACM Symposium on User Interface Software and Technology*, pages 101–108, Atlanta, GA, November 1993.

[14] Manojit Sarkar and Marc H. Brown. Graphical fisheye views of graphs. In *Proceedings of ACM SIGCHI '92 Conference on Human Factors in Computing Systems*, pages 83–91, May 1992.

[15] Manojit Sarkar, Scott S. Snibbe, Oren J. Tversky, and Steven P. Reiss. Stretching the rubber sheet: A metaphor for viewing large layouts on small screens. In *Proceedings of the 1993 ACM Symposium on User Interface Software and Technology*, pages 81–91, November 1993.

[16] Robert Spence and Mark Apperley. Data base navigation: an office environment for the professional. *Behaviour and Information Technology*, 1(1):43–54, 1982.

[17] John T. Stasko and Eileen Kraemer. A methodology for building application-specific visualizations of parallel programs. *Journal of Parallel and Distributed Computing*, 18(2):258–264, June 1993.

[18] Maureen C. Stone, Ken Fishkin, and Eric A. Bier. The movable filter as a user interface tool. In *Proceedings of the ACM SIGCHI '94 Conference on Human Factors in Computing Systems*, pages 306–312, Boston, MA, April 1992.

[19] University of Indiana. *Sage++: A Class Library for Building Fortran 90 and C++ Restructuring Tools*, November 1993.