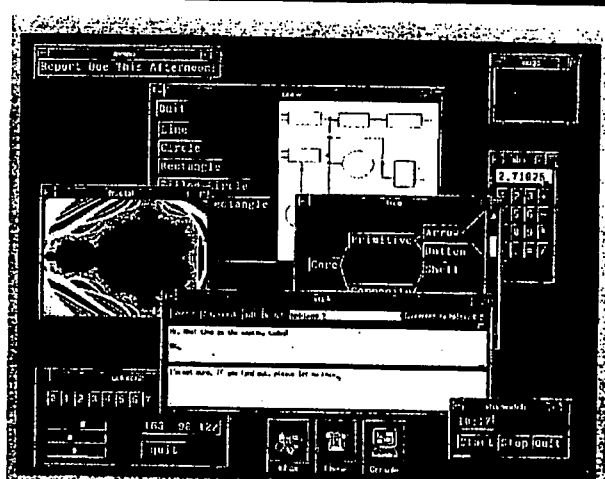
Exhibit K

THE

WINDOW SYSTEM®
PROGRAMMING AND APPLICATIONS WITH Xt
OSF/MOTIF EDITION



DOUGLAS A. YOUNG

The X Window System®

Programming and Applications with Xt OSF/Motif® Edition

Douglas A. Young
Hewlett-Packard Laboratories
Palo Alto, California

Prentice Hall, Englewood Cliffs, New Jersey 07632

Library of Congress Cataloging-in-Publication Data
Young, David A.
The X window system : programming and applications with Xt /
Douglas A. Young. -- OSF/Motif ed.
p. cm.
Includes bibliographical references.
ISBN 0-13-497074-8
1. X Window System (Computer system) I. Title.
QA76.76.W56Y67 1990
005.4'3--dc20 89-19224
CIP

Cover design: *Photo Plus Art*
Manufacturing buyer: *Ray Sintel*

Back cover line art:
courtesy of John Humphrys

Printed in the United States of America

10 9 8 7 6 5 4 3 2

ISBN 0-13-497074-8

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

1

AN INTRODUCTION TO THE X WINDOW SYSTEM

The X Window System is an industry-standard software system that allows programmers to develop portable graphical user interfaces. One of the most important features of X is its unique device-independent architecture. X allows programs to display windows containing text and graphics on any hardware that supports the X protocol without modifying, recompiling, or relinking the application. This device independence, along with X's position as an industry standard, allows X-based applications to function in a heterogeneous environment consisting of mainframes, workstations, and personal computers.

X was developed at Massachusetts Institute of Technology (MIT), with support from the Digital Equipment Corporation (DEC). The name, X, as well as some initial design ideas were derived from an earlier window system named W, developed at Stanford University. X was designed at MIT's Laboratory for Computer Science for Project Athena to fulfill that project's need for a distributed, hardware-independent user interface platform. Early versions of X were used primarily within MIT and DEC, but with the release of version 10, many manufacturers expressed interest in X as a commercial product. The early versions of X were designed and implemented primarily by Robert Scheifler and Ron Newman from MIT and Jim Gettys from DEC, although many additional people contributed to X, Version 11. Version 11 of the X Window System is supported by a consortium of hardware and software vendors who have made a commitment to X as a standard base for user interfaces across each of their product lines. The X Consortium supports and controls the standard specification of the X Window System. X is available on most UNIX systems, Digital's VAX/VMS operating

Exhibit L

Microsoft Product Support Services Application Note (Text File)
GC0165: RICH-TEXT FORMAT (RTF) SPECIFICATION

Revision Date: 6/92
No Disk Included

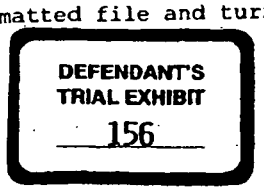
The following information applies to using RTF version 1.0 with Microsoft MS-DOS(R), Windows(TM), OS/2(R), and Apple(R) Macintosh(R) Applications.

OVERVIEW

The rich-text format (RTF) standard is a method of encoding formatted text and graphics for easy transfer between MS-DOS, Windows, OS/2, and Apple Macintosh applications.

The RTF standard provides a format for text and graphics interchange that can be used with different output devices, operating environments, and operating systems. RTF uses the ANSI, PC-8, Macintosh, or IBM PC character set to control the representation and formatting of a document, both on the screen and in print. With the RTF standard, you can transfer documents created under different operating systems and with different software applications among those operating systems and applications.

Software that takes a formatted file and turns it into an RTF file is



called a "writer." An RTF writer separates the application's control information from the actual text and writes a new file containing the text and the RTF groups associated with that text. Software that translates an RTF file into a formatted file is called a "reader."

RTF SYNTAX

An RTF file consists of unformatted text, control words, control symbols, and groups. For ease of transport, a standard RTF file can consist of only 7-bit ASCII characters. However, Word for Windows uses 8-bit characters in the RTF stream given to converter DLLs.

A "control word" is a specially formatted command that RTF uses to mark printer control codes and information that applications use to manage documents. A control word takes the following form:

```
\<LetterSequence>[<NumericParameter>]<delimiter>
```

For example:

```
A   B   C
|   |   |
| ----- |
\ r t f 1
```

A A backslash begins each control word
B Letter sequence
C Numeric parameter

The "Letter Sequence" is made up of uppercase or lowercase alphabetic characters between A and Z inclusive.

The "Numeric Parameter" begins with a digit or a minus sign (-). The range of the values for the numeric parameter is -32,767 through 32,767. However, Microsoft Word for Windows, Word for OS/2, and Word for the Macintosh restrict the range to -31,680 through 31,680. If a numeric parameter immediately follows the control word, this parameter becomes part of the control word and the delimiter follows the parameter.

The "delimiter" marks the end of an RTF control word or symbol. A delimiter can be one of the following:

- A space. If a space delimits a control word, the space does not appear in the document. Any characters following the delimiter, including spaces, do appear in the document. For this reason, you should use spaces only where necessary; do not use spaces merely to break up RTF code.

- A backslash (\), opening brace ({), or closing brace (}). These characters are used to mark the beginning of a new control word or symbol, the beginning of a group, and end of a group, respectively. More information about control symbols and groups is provided later in this document.

- Any character other than a letter or a digit. In this case, the character terminates the control word but is not actually part of the control word. The character is considered part of the document text.

Some control words govern properties that have only two states (for example, bold, which is either turned on or turned off). When such a control word has no parameter or has a nonzero parameter, it is assumed that the control word turns on the property. When such a control word has a parameter of 0 (zero), it is assumed that the control word turns off the property. For example, \b turns on bold, whereas \b0 turns off bold.

A "control symbol" consists of a backslash followed by a single, nonalphabetic character. For example, \~ represents a nonbreaking space. Control symbols take no delimiters.

A "group" consists of text and control words or control symbols enclosed in braces ({ }). The opening brace indicates the start of the group and the closing brace indicates the end of the group. Each group specifies the text affected by the group and the different attributes of that text. The RTF file can also include groups for fonts, styles, screen color, pictures, footnotes, annotations, headers and footers, summary information, fields, and bookmarks, as well as document-, section-, paragraph-, and character-formatting properties. If the font, style, screen color, and summary information groups and document formatting properties are included, they must precede the first plain-text character in the document. These groups form the RTF file header. If the group for fonts is included, it should precede the group for styles. If any group is not used, it can be omitted. The groups are discussed in the following sections.

Certain groups, referred to as "destinations," mark the beginning of a collection of related text that could appear at another position, or destination, within the document. Destinations can also be text that is used but should not appear within the document at all. Destinations are sometimes distinguished from other groups by a * control symbol immediately following the opening brace of the group. The * is followed by a control word that defines the type of the destination. This control symbol identifies destinations whose related text should be ignored if the RTF reader does not recognize the destination. (RTF writers should follow the convention of using this control symbol when adding new control words.) Destinations whose related text should be inserted into the document even if the RTF reader does not recognize the destination should not use *. An example of a destination is the \footnote group, where the footnote text follows the control word.

Formatting specified within a group affects only the text within that group. Generally, text within a group inherits the formatting of the text in the preceding group. However, Microsoft implementations of RTF assume that the footnote, header and footer, and annotation groups (described later in this document) do not inherit the formatting of the preceding text. Therefore, to ensure that these groups are always formatted correctly, you should set the formatting within these groups to the default with the \sectd, \pard, and \plain control words, and then add any desired formatting.

The control words, control symbols, and braces constitute control information. All other characters in the file are plain text. The following is an example of plain text that does not exist within a group:

```
...
\linex0endnhere \pard\plain \fs20 This is plain text.
...
```

The phrase, "This is plain text" is not part of a group and is treated as document text.

As previously mentioned, the backslash (\) and braces ({ }) have special meaning in RTF. To use these characters as text, precede them with a backslash, as follows:

```
\\
\{
\}
```

CONVENTIONS OF AN RTF READER

The reader of an RTF stream is concerned with the following:

- Separating control information from plain text
- Acting on control information
- Collecting and properly inserting text into the document, as directed by the state of the current group

The process of acting on control information is designed to be relatively simple. Some control information only contributes special characters to the plain text stream. Other information changes the "program state," which includes properties of the document as a whole, or changes a collection of "group states," which applies to parts of the document.

As previously mentioned, a group state can specify the following:

- The "destination," or part of the document that the plain text is constructing
- Character formatting properties, such as bold or italic
- Paragraph formatting properties, such as justified or centered
- Section formatting properties, such as the number of columns
- Table formatting properties, which define the number of cells and dimensions of a table row

An RTF reader performs the following procedure:

1. The reader reads the next character.

2. If the next character is:

    - An opening brace, the reader stores the current state of the document on the stack.

    - A closing brace, the reader retrieves the current state of the document from the stack.

    - A backslash, the reader marks the beginning of an RTF control. The reader collects the control word or control symbol and its parameter, if any, and carries out the action prescribed for that control. The meaning of each of the controls is discussed in the section "Contents of an RTF File" in this document. The read pointer is left on the next nonspace delimiter following the control.

    - Anything other than an opening brace, closing brace, or

backslash, the reader assumes that the character is plain text and writes the character to the current destination using the current formatting properties.

3. The reader then reads the next character.

If the RTF reader does not recognize a particular control word or control symbol, the reader ignores it. When the reader encounters a control word or control symbol preceded by an opening brace, the reader recognizes the control as part of a group. The reader saves the current state on the stack but does not change the state. When the reader encounters a closing brace, the reader retrieves the current state from the stack, thereby resetting the current state. If the * control symbol precedes the unknown control word, * defines a destination group and is preceded by an opening brace. The RTF reader will discard all text and subgroups up to and including the brace that closes this group.

For control words or control symbols recognized by the RTF reader, the possible actions are as follows:

Action	Description
Change destination	The RTF reader changes the destination to the destination described in the table entry. Destination changes are legal only immediately after an opening brace. (Other restrictions may also apply; for example, footnotes cannot be nested.) Many destination changes imply that the current property settings will be reset to their default settings. Examples of control words that change the destination are \footnote, \header, \footer, \pict, \info, \fonttbl, \stylesheet, and \colortbl.
Change formatting property	The RTF reader changes the property as described in the table entry. The entry specifies whether a parameter is required.
Insert special character	The reader inserts into the document the character code or codes described in the table entry.
Insert special character and perform action	The reader inserts into the document the character code or codes described in the table entry and performs whatever other action the entry specifies. For example, when Microsoft Word interprets \par, Word inserts a paragraph mark in the document, along with the paragraph properties belonging to that paragraph mark.

CONTENTS OF AN RTF FILE

An RTF file can contain combinations of the following elements:

THE RTF VERSION

An entire RTF file is considered a group and must be enclosed in braces. The control word \rtf<N> must follow the opening brace. The numeric parameter <N> identifies the version of the RTF standard used. The RTF standard described in this document is version 1.0.

THE CHARACTER SET

After specifying the RTF version, you must declare the character set. The control word for the character set must precede any plain text or any table control words. The RTF specification currently supports the following character sets:

Control Word	Character Set
\ansi	ANSI (the default)
\mac	Apple Macintosh
\pc	IBM(R) PC Code Page 437
\pca	IBM PC Code Page 850, used by IBM Personal System/2(R) (not implemented in version 1.0 of Microsoft Word for OS/2)

The Font Table

This group begins with the control word \fonttbl and contains descriptions of fonts. All fonts available to the RTF writer can be included in the font table, even if the document doesn't use all the fonts.

A font is defined by a font number, a font family, and a font name as shown in the following example:

```
      A     B     C      D
      |     |     |      |
   -------- | ------- ---------
  {\fonttbl\f0\froman Tms Rmn;}...
```

A Control word
B Font number
C Font family
D Font name

Semicolons separate one font from the next. The font numbers represent the full font definitions in the group and vary with each document. The font families are listed below:

Control Word	Font Family
\fnil	Unknown or default fonts (the default)
\fRoman	Roman, proportionally spaced serif fonts (for

	example, Times Roman(R) and Palatino(R))
\fswiss	Swiss(TM), proportionally spaced sans serif fonts (for example, Swiss Helvetica(R))
\fmodern	Fixed-pitch serif and sans serif fonts (for example, Courier and Pica)
\fscript	Script fonts (for example, Cursive)
\fdecor	Decorative fonts (for example, Old English and ITC Zapf Chancery(R))
\ftech	Technical, symbol, and mathematical fonts (for example, Symbol)

If an RTF file uses a default font, the default font number is specified with the \deff<N> control word, which must precede the font-table group. The RTF writer supplies the default font number used in the creation of the document as the numeric argument <N>. The RTF reader then translates this number through the font table into the most similar font available on the reader's system.

THE STYLE SHEET

The style sheet group begins with the control word \stylesheet. This group contains definitions and descriptions of the various styles used in the document. The style sheet is declared only once, before any document text. All styles in the document's style sheet can be included, even if not all the styles are used.

Control Word	Meaning
\sbasedon<N>	Defines the number of the style on which the current style is based. Word for Windows defaults to the Normal style--style number 222--if \sbasedon is omitted.
\snext<N>	For paragraph styles, \snext defines the style automatically assigned to a paragraph created following the paragraph with the current style. If \snext omitted, the next paragraph is given the same style as the current paragraph.
\keycode	Specified within the description of a style in the style sheet in the RTF header. The syntax for this group is {*\keycode <Keys>} where <Keys> are the characters used in the key code. For example, a Normal style may be defined {\s0 {*\keycode \shift\ctrl n}Normal;} within the RTF style sheet. See the "Special Characters" control words for the characters outside of the alphanumeric range that can be used.

An example of an RTF style sheet and styles follows:

...

```
     |   {\stylesheet{\fs20 \sbasedon222\snext10{keycode \shift...}
  A--|   {\s1 \ar \fs20 \sbasedon0\snext1 FLUSHRIGHT}{\s2\fi...}
     |   \sbasedon0snext2 IND:}}
      --
        ...
      --
     |  \widowctrl\ftnbj\ftnrestart \sectd \linex0\endnhere
     |  \pard\plain \fs20 This is Normal style.
  B--|  \par \pard\plain \s1
     |  This is right justified. I call this style FLUSHRIGHT.
     |  \par \pard\plain \s2
     |  This is an indented paragraph. I call this style IND...
      --
        \par}
        ...

  A  Style sheet
  B  Styles applied to text

  This is Normal Style.
                This is right justified. I call this style FLUSHRIGHT.
  This is an indented paragraph. I call this style IND. It produces
        a hanging indent.
```

In this example, PostScript(R) is declared but not used. Some of the control words in this example are discussed in the following sections.

THE COLOR TABLE

Screen colors, character colors, and other color information are contained in the color table group. The control word \colortbl begins this group. Additional control words for red, green, and blue and the foreground and background colors then use parameter values (0-255) corresponding to the color indexes used by Microsoft Windows to define the amount of red, green, and blue that makes up a color. For more information on color setup, see your Windows documentation.

The following are valid control words for this group:

Control Word	Meaning
\red<N>	Red index
\green<N>	Green index
\blue<N>	Blue index
\cf<N>	Foreground color (the default is 0)
\cb<N>	Background color (the default is 0)

Each definition must be delimited by a semicolon, even if the definition is omitted. If a color definition is omitted, the RTF reader uses its default color. In the example that follows, three colors are defined. The first color is omitted, as shown by the semicolon following the \colortbl control word.

```
...
{\colortbl;\red0\green0\blue0;\red0\green0\blue255;}
...
```

The following example defines a block of text in color (where supported). Note that the cf/cb index is the index of an entry in the color table, which represents a red/green/blue color combination.

```
...
{\f1\cb1\cf2 This is colored text. The background is color
1 and the foreground is color 2.
...
```

If the file is translated for software that does not display color, the reader ignores the color table group.

PICTURES

An RTF file can include picture files created with other applications. These files are in hexadecimal (the default) or binary format. The control word \pict begins this group. Control words that define and describe the picture parameters follow the \pict control word. These control words are listed in the following table (some measurements in this table are in twips; a twip is one-twentieth of a point):

General

Control Word	Meaning
\macpict	The source file of the picture is QuickDraw(TM).
\pmmetafile<N>	The source file of the picture is an OS/2 metafile; the <N> argument identifies the metafile type.
\wmetafile<N>	The source file of the picture is a Windows metafile; the <N> argument identifies the metafile type (the default is 1, meaning the metafile type is MM_TEXT).
\dibitmap<N>	The source file of the picture is a device independent bitmap; the <N> argument identifies the bitmap type (the default is 0).
\wbitmap<N>	The source file of the picture is a bitmap; the <N> argument identifies the bitmap type (the default is 0, meaning that the bitmap is a logical bitmap).

Bitmap Information

Control Word	Meaning
\wbmbitspixel<N>	The number of bitmap bits per pixel (the default is 1).
\wbmplanes<N>	The number of bitmap planes (the default is 1).
\wbmwidthbytes<N>	The bitmap width in bytes.

Picture Size, Scaling, and Cropping

Control Word	Meaning
\picw<N>	The <xExt> field if the picture is a metafile; the picture width in pixels if the picture is a bitmap or is from QuickDraw.
\pich<N>	The <yExt> field if the picture is a metafile; the picture height in pixels if the picture is a bitmap or from QuickDraw.
\picwgoal<N>	The desired width of the picture in twips.
\pichgoal<N>	The desired height of the picture in twips.
\picscalex<N>	The horizontal scaling value; the <N> argument is a value representing a percentage (the default is 100).
\picscaley<N>	The vertical scaling value; the <N> argument is a value representing a percentage (the default is 100).
\picscaled	Scales the picture to fit within the specified frame; used only with \macpict pictures.
\piccropt<N>	The top cropping value in twips; a positive value crops toward the center of the picture; a negative value crops away from the center, adding a space border around the picture (the default is 0).
\piccropb<N>	The bottom cropping value in twips; a positive value crops toward the center of the picture; a negative value crops away from the center, adding a space border around the picture (the default is 0).
\piccropl<N>	The left cropping value in twips; a positive value crops toward the center of the picture; a negative value crops away from the center, adding a space border around the picture (the default is 0).
\piccropr<N>	The right cropping value in twips; a positive value crops toward the center of the picture; a negative value crops away from the center, adding a space border around the picture (the default is 0).

Picture Border

Control Word	Meaning
\brdrs	A single picture border.

\brdrdb	A double picture border.
\brdrth	A thick picture border.
\brdrsh	A shadow picture border.
\brdrdot	A dotted picture border.
\brdrhair	A hairline picture border.
\brdrw<N>	<N> is the width in twips of the picture border line. This control should follow the picture border controls \brdrt, \brdrr, \brdrb, and \brdrl.
\brdrcf<N>	<N> is the color of the picture border from the color table in the RTF header. This control should follow the picture border controls \brdrt, \brdrr, \brdrb, and \brdrl.

Picture Background Shading

Control Word	Meaning
\shading<N>	<N> is the shading of the picture in hundredths of a percent.
\bghoriz	Specifies a horizontal background pattern for the picture.
\bgvert	Specifies a vertical background pattern for the picture.
\bgfdiag	Specifies a forward diagonal background pattern for the picture (\\\\).
\bgbdiag	Specifies a backward diagonal background pattern for the picture (////).
\bgcross	Specifies a cross-hatched background pattern for the picture.
\bgdcross	Specifies a diagonal cross-hatched background pattern for the picture.
\bgdkhoriz	Specifies a dark horizontal background pattern for the picture.
\bgdkvert	Specifies a dark vertical background pattern for the picture.
\bgdkfdiag	Specifies a dark forward diagonal background pattern for the picture (\\\\).
\bgdkbdiag	Specifies a dark backward diagonal background pattern for the picture (////).
\bgdkcross	Specifies a dark cross-hatched background pattern for the picture.
\bgdkdcross	Specifies a dark diagonal cross-hatched background pattern for the picture.

\cfpat<N>	<N> is the line color of the background pattern.
\cbpat<N>	<N> is the background color of the background pattern.

Picture Data

Control Word	Meaning
\bin<N>	The picture is in binary format; the numeric parameter <N> is the number of bytes that follow.

The \wbitmap control word is optional; if neither \wmetafile nor \macpict is specified, the picture is assumed to be a Windows bitmap. If \wmetafile is specified, the <N> argument can be one of the following types:

Type	<N> Argument
MM_TEXT	1
MM_LOMETRIC	2
MM_HIMETRIC	3
MM_LOENGLISH	4
MM_HIENGLISH	5
MM_TWIPS	6
MM_ISOTROPIC	7
MM_ANISOTROPIC	8

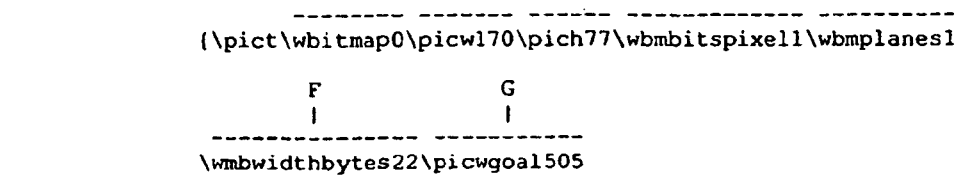
If \pmmetafile is specified, the <N> argument can be one of the following types:

Type	<N> Argument
PU_ARBITRARY	4
PU_PELS	8
PU_LOMETRIC	12
PU_HIMETRIC	16
PU_LOENGLISH	20
PU_HIENGLISH	24
PU_TWIPS	28

Be careful with spaces following control words when dealing with pictures in binary format. When reading files, RTF considers the first space after a control word the delimiter and subsequent spaces part of the document text. Therefore, any extra spaces are attached to the picture, with unpredictable results.

RTF writers should not use the carriage-return/linefeed (CRLF) combination to break up pictures in binary format. If they do, the CRLF is treated as literal text and considered part of the picture data.

The picture in hexadecimal or binary format follows the picture-group control words. The following example illustrates the group format:




```
H--- \pichgoal221
I--- \picscalex172
J--- \picscaley172
   - 49f2000000000273023d1101a030
   | 3901000a000000000273023d98
   | 0048000200000275
   | 0204000020001027 5023e000000000
K--| 273023d000002b90002b90002
   | b90002b90002b9
   | 0002b90002b90002b90002b90002b90002
   | b92222b90002b90002b90
   | 002b90002b9
   - 0002b90002b90002b90002b9000
     ...
A  Source
B  Width
C  Height
D  Bits per pixel
E  Bitmap planes
F  Width of picture in bytes
G  Desired picture width
H  Desired picture height
I  Horizontal scaling value
J  Vertical scaling value
K  Hexadecimal data
```

NOTE: The controls \pichgoal, \picscalex, and \picscaley in this example are on separate lines for the purpose of illustration only. In actual RTF code, they can be included on the same line.

FOOTNOTES

The group containing footnote text begins with the control word *\footnote. Footnotes are anchored to the character that immediately precedes the footnote group. If automatic footnoting is defined, the group can be preceded by a footnote reference character, identified by the control word \chftn.

The following is an example of a group containing footnotes:

```
       ...
       \ftnbj\ftnrestart \sectd \\linemod0\linex0\endnhere
       \pard\plain \ril170 \fs20 {pu6 Mead's landmark study has been
       amply. annotated.\chftn
A--|  {\*\footnote \pard\plain \s246 \fs20 {\up6\chftn }See Sahlins,
   |  Bateson, and Geertz for a complete bibliography.}
       It was her work in America during the Second World War,
       however, that forms the basis for the paper. As others have
       noted, \chftn
A--|  {\*\footnote \pard\plain \s246 \fs20 {\up6\chftn}
```

```
| A complete bibliography will be found at the end of the
| chapter.}
  this period was a turning point for Margaret Mead.}
  \par
  ...
```

Mead's landmark study has been amply annotated.1 It was her work in America during the Second World War, however, that forms the basis for this paper. As others have noted,2 this period was a turning point for Margaret Mead.

1 See Sahlins, Bateson, and Geertz for a complete bibliography.

2 A complete bibliography will be found at the end of the chapter.

A: Footnotes

For other control words relating to footnotes, see the sections titled "Document Formatting Properties," "Section Formatting Properties," and "Special Characters" later in this document.

ANNOTATIONS

==========

The group containing annotation text begins with the control word *\annotation. Annotations are anchored to the character that immediately precedes the annotation group. The group must be preceded by an annotation reference character identified by the control word \chatn. The annotation reference character must in turn be preceded by a group that begins with the control word *\atnid and that contains the identification text for the author of the annotation. An example of annotation text follows:

```
...
An example of a paradigm might be Newtonian physics or
Darwinian biology.{\v\fs16 {\atnid bz}\chatn{\*\annotation
\pard\plain \s224 \fs20 {field{fldinst page \\*"'Page:
'#\line'"}{\fldrslt}}{\fs 16 \chatn }
How about some examples that deal with social science?
That's what this paper is about.}}
...
```

HEADERS AND FOOTERS

==========

Headers and footers are treated as separate groups in RTF. Different headers and footers can be defined for different sections in the document. If none is defined for a given section, the headers and footers from the previous section (if any) are used. These groups must precede the first plain-text character in the document section to which they apply.

The control words \header and \footer begin these groups. These control words can be replaced by the following control words, as appropriate:

Control Word	Meaning
\headerl	The header is on left pages only.
\headerr	The header is on right pages only.
\headerf	The header is on the first page only.
\footerl	The footer is on left pages only.
\footerr	The footer is on right pages only.
\footerf	The footer is on the first page only.

The \headerl and \headerr and \footerl and \footerr control words are used in conjunction with the \facingp control word, and the \headerf and \footerf control words are used in conjunction with the \titlepg control word. For more information, see "Document Formatting Properties" and "Section-Formatting Properties" in this document.

SUMMARY INFORMATION

The RTF file can also contain a summary information group, which is translated but not displayed with the text. This information can include the title, author, keywords, comments, and other information specific to the file. This information is for use by a document-management utility, if available.

This group begins with the control word \info. Some applications, such as Word, ask the user to type this information when saving the document in its native format. If the document is then saved as an RTF file or translated into RTF, the RTF writer specifies this information using the following control words. These control words are destinations, and both the control words and the text should be enclosed within braces ({ }).

Control Word	Meaning
\title	The title of the document
\subject	The subject of the document
\author	The author of the document
\operator	The person who last made changes to the document
\keywords	Selected keywords for the document
\comment	Comments; text is ignored
\version<N>	The version number of the document
\doccomm	Comments displayed in Word's Summary Info dialog box

The RTF writer can automatically enter other control words, including the following:

Control Word	Meaning
\vern<N>	The internal version number
\creatim	The creation time
\revtim	The revision time
\printim	The last print time
\buptim	The backup time
\edmins<N>	The total editing time (in minutes)

\yr<N>	The year
\mo<N>	The month
\dy<N>	The day
\hr<N>	The hour
\min<N>	The minute
\nofpages<N>	The number of pages
\nofwords<N>	The number of words
\nofchars<N>	The number of characters
\id<N>	The internal ID number

Entries without the <N> parameter have the \yr \mo \dy \hr \min format. An example of a summary information group follows:

```
...
{\info{\title The Panda's Thumb}{\author Stephen J.
Gould}{\keywords science natural history }}
```

FIELDS

The field group contains the text of Word fields. For more detailed information on fields, choose Help in Microsoft Word for Windows.

The field group begins with the control word \field. The following control words can follow the \field control word:

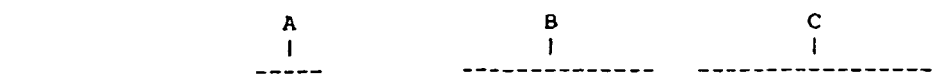
Control Word	Meaning
\flddirty	A formatting change has been made to the field result since the field was last updated.
\fldedit	The text has been added to, or removed from, the field result since the field was last updated.
\fldlock	A field is locked and cannot be updated.
\fldpriv	A result is not in a form suitable for display (for example, binary data used by fields whose result is a picture).

Two subgroups are available within the \field group. They must be enclosed within braces ({ }) and begin with the following control words:

Control Word	Meaning
*\fldinst	Field instructions
\fldrslt	The most recently calculated result of the field

The \fldrslt control word should be included even if no result has been calculated because even readers that do not recognize fields can generally include the value of the \fldrslt group in the document.

An example of field text follows:



```
{\field\fldedit{\fldinst author}{\fldrslt Joe Smith}}\par\pard
{\field{\fldinst time \\@ "h:mm AM/PM"}{\fldrslt 8:12 AM}}
 -----  ----------------------------  -----------------
   |                 |                          |
   A                 B                          C
```

A Begins field group
B Field instructions
C Field result

INDEX ENTRIES

The index entry group begins with the control word \xe. Following this control word is the text of the index entry and other optional control words that further define the index entry.

If the text of the index entry is not formatted as hidden text with the \v control word, the text is put into the document as well as into the index. For more information on the \v control word, see "Character Formatting Properties" in this document. Similarly, the text of the \txe subgroup, described later in this section, becomes part of the document if it is not formatted as hidden text.

The following control words can also be used:

Control Word	Meaning
\bxe	Formats the page number or cross-reference in bold
\ixe	Formats the page number or cross-reference in italic

The following control words are destinations within the \xe group and are followed by text arguments. These control words and their arguments must be enclosed within braces ({ })

Control Word	Meaning
\txe <Text>	The text argument is to be used instead of a page number.
\rxe <BookmarkName>	The text argument is a bookmark for the range of page numbers.

An example of an index entry follows:

```
  A        B              C                D         E    F
  |        |              |                |         |    |
  --  -------------  ---------------  -------------  ---  ---
{\xe{\v Index Entry}{\rxe Index Range}{\txe See Index}\bxe \ixe }
```

A Begins index-entry group
B Entry text
C Bookmark defining range of pages
D Replacement for page number
E Bold entry in index table
F Italic entry in index table

TABLE OF CONTENTS ENTRIES

The table of contents entry group begins with the control word \tc. It is followed by the text of the table of contents entry and other optional control words that further define the group.

As with index entries, text that is not formatted as hidden with the \v character-formatting control word is put into the document. The following control words can also be used in this group:

Control Word	Meaning
\tcf<N>	The type of table being compiled; <N> is mapped by existing Microsoft software to a letter between A and Z (the default is 67, which maps to C and is used for tables of contents).
\tcl<N>	The level number (the default is 1).

OBJECTS

Objects are structures in a document that contain a data portion and a result portion. The data portion generally appears hidden to the application that produced the document. A separate application uses the data and supplies the result or appearance of the data. This appearance is the result portion of the object. Some examples of objects include object linking and embedding (OLE) objects and Edition Manager Subscriber objects on the Macintosh platform.

In RTF, the results of objects are represented so that RTF readers that don't understand objects or don't use a particular type of object are able to use the current result in place of the object. This allows the appearance of the object to be maintained through the conversion even though the object functionality is lost. For this reason, it is important for RTF writers to supply the object result. The format of the result should be standard RTF.

The data portion of an OLE object is the structure produced from the OLE SaveToStream function. Some OLE clients rely on the OLE system to render the object, and a copy of the result is not available to the RTF writer for that application. For these cases, the object result can be obtained from the structure produced from the OLE SaveToStream function.

An object group is defined by the control word \object. These objects can be either embedded objects, OLE links, or subscriber objects on the Macintosh platform. If the RTF reader's application does not use the type of object represented, then all the object information and data should be ignored and the object result should be inserted into the document in place of the object.

Each object comes with optional information about the object, a required data group that contains the object data, and an optional result that contains the last display of the object. This result contains standard RTF. It is the responsibility of the RTF writer to provide the result so that existing RTF readers will be able to display the object without having full object support.

The syntax for this group is as follows:

{\object [\objemb|\objlink|\objautlink|\objsub|\objpub|\objicemb]
[\linkself] [\objlock] [{*\objclass <Name>}] [{*\objname <Name>}]
[\rsltmerge] [\rsltrtf|\rslttxt|\rsltpict|\rsltbmp] [\objsetsize]
[\objalign<N>] [\objtransy<N>] [\objh<N> \objw<N>]
[\objcropt<N>][\objcropb<N>][\objcropl<N>][\objcropr<N>]
[\objscalex<N>][\objscaley<N>] {*\objdata[{*\objalias[\bin<N>]
<Data>} {*\objsect[\bin<N>] <Data>}] [\bin<N>] <Data>}<Result>}

See the \result control word in the following table for a description of <Result>:

Object Type

Control Word	Meaning
\objemb	An object type of OLE embedded object. If no type is given for the object, the object is assumed to be of type \objemb.
\objlink	An object type of OLE link.
\objautlink	An object type of OLE autolink.
\objsub	An object type of Macintosh Edition Manager subscriber.
\objicemb	An object type of MS Word for Macintosh Installable Command (IC) Embedder.

Object Information

Control Word	Meaning
\linkself	The object is a link to another part of the same document.
\objlock	Locks the object from any updates.
\objclass	The syntax for the group is {*\objclass <Name>} where <Name> is the name of the object class. This is optional in the \object group.
\objname	The syntax for group is {*\objname <Name>} where <Name> is the name of the specific object instance. This is optional in the \object group.

Object Size, Position, Cropping, and Scaling

Control Word	Meaning
\objh<N>	<N> is the original object height in twips.

\objw<N>	<N> is the original object width in twips.
\objsetsize	Forces the object server to set the object's dimensions to those specified by the client.
\objtransy<N>	<N> is the distance in twips an object should be moved vertically with respect to the baseline. This control word is needed to place MathType(TM) equation objects correctly in line. This is an optional control of the \object group.
\objcropt<N>	<N> is the top cropping distance in twips.
\objcropb<N>	<N> is the bottom cropping distance in twips.
\objcropl<N>	<N> is the left cropping distance in twips.
\objcropr<N>	<N> is the right cropping distance in twips.
\objscalex<N>	<N> is the horizontal scaling percentage.
\objscaley<N>	<N> is the vertical scaling percentage.

Object Data

Control Word	Meaning
\objdata	The \objdata subgroup is required to be in the \object group. The syntax for this group is {*\objdata[{*\objalias[\bin<N>] <Data>}{*\objsect[\bin<N>] <Data>}][\bin<N>] <ObjectData>} where <ObjectData> represents the complete data of the object. If the \bin<N> option is used, then <N> represents the number of bytes of binary data. Otherwise, the <ObjectData >is in hexadecimal.
\objalias	This group contains the Alias Record for the publisher object for the Macintosh Edition Manager. If the \bin control is used, the data is in binary numerical format. Otherwise, it is represented in the RTF stream in hexadecimal.
\objsect	This group contains the Section Record for the publisher object for the Macintosh Edition Manager. If the \bin control is used, the data is in binary numerical format. Otherwise, it is represented in the RTF stream in hexadecimal.

Object Result

Control Word	Meaning
\rsltrtf	Forces the result to be RTF, if possible.
\rsltpict	Forces the result to be a Windows metafile or MacPict image, if possible.

\rsltbmp	Forces the result to be a bitmap, if possible.
\rslttxt	Forces the result to be plain text, if possible.
\rsltmerge	Uses the formatting of the current result whenever a new result is obtained.
\result	The result group is optional in the \object group. It contains the last update of the result of the object. The data of the result group should be standard RTF so that RTF readers that don't understand objects or the type of object represented can use the current result in the object's place to maintain the object's appearance. The syntax for this group is {\result <ResultData>} where <ResultData> is standard RTF syntax.

BOOKMARKS

This group contains two control words: *\bkmkstart, which indicates the start of the specified bookmark, and *\bkmkend, which indicates the end of the specified bookmark. A bookmark is shown in the following example:

```
...
\pard\plain \fs20 Kuhn believes that science, rather than
discovering in experience certain structured
relationships, actually creates (or already participates in)
a presupposed structure to which it fits the data.
{\bkmkstart paradigm} Kuhn calls such a presupposed
structure a paradigm.{\bkmkend paradigm}
...
```

If a bookmark covers a partial selection of columns of a table, \bkmkcolf<N> is used to denote the first column of a table covered by a bookmark and \bkmkcoll<N> is used to denote the last column. If \bkmkcolf is not included, the first column of the table is used as the first column for the bookmark. If \bkmkcoll is not included, the last column of the table is used for the bookmark. These controls are used within the *\bkmkstart group following the \bkmkstart control. For example, {*\bkmkstart\bkmkcolf2\bkmkcoll5 Table1} places the bookmark "Table1" on columns 2 through 5 of a table.

MACINTOSH EDITION MANAGER PUBLISHER OBJECTS

These controls define the RTF controls used to define publisher objects for the Macintosh Edition Manager. The range of publisher objects are marked as bookmarks; therefore, these controls are all used within the \bkmkstart group. The RTF syntax for a publisher object is:

```
{\*\bkmkstart\bkmkpub[\pubauto]{\*\objalias[\bin]
<Data>}{\*\objsect[\bin] <Data>} <BookmarkName>}
```

Control Word	Meaning

\bkmkpub	The bookmark marks a Macintosh Edition Manager publisher object.
\pubauto	The publisher object will automatically update all Macintosh Edition Manager subscribers of this object whenever it is edited.
\objalias	This group contains the Alias Record for the publisher object for the Macintosh Edition Manager. If the \bin control is used, the data is in binary format. Otherwise, it is in hexadecimal.
\objsect	This group contains the Section Record for the publisher object for the Macintosh Edition Manager. If the \bin control is used, the data is in binary format. Otherwise, it is in hexadecimal.

DOCUMENT FORMATTING PROPERTIES

This section specifies the attributes of the document, such as margins and footnote placement. These attributes must precede the first plain-text character in the document.

The control words that specify document formatting are listed in the following table (the measurements are in twips).

The default values are used for omitted control words.

Control Word	Meaning
\deftab<N>	The default tab width (the default is 720).
\hyphhotz	The hyphenation hot zone (the amount of space at the right margin in which words are hyphenated--the default for Word is 360 twips).
\linestart<N>	The beginning line number (the default is 1).
\fracwidth	Uses fractional character widths when printing (QuickDraw only).
*\nextfile	The destination; the argument is the name of the file to print or index next; must be enclosed with braces.
*\template	The destination; the argument is the name of a related template file; must be enclosed with braces.
\makebackup	The backup copy is made automatically when the document is saved.
\defformat	Tells the RTF reader that the document should be saved in RTF format.
\psover	Prints PostScript over the text.

\deflang<N>	Defines the default language used in the document when character formatting is reset with the \plain control word. See "Character Formatting Properties" in this document for a list of possible values for <N>.

Footnotes

Control Word	Meaning
\ftnsep	The text argument separates footnotes from the document.
\ftnsepc	The text argument separates continued footnotes from the document.
\ftncn	The text argument is a notice for continued footnotes.
\endnotes	Print the footnotes at the end of the section (the default).
\enddoc	Print the footnotes at the end of the document.
\ftntj	Print the footnotes beneath text.
\ftnbj	Print the footnotes at the bottom of the page.
\ftnstart<N>	The beginning footnote number (the default is 1).
\ftnrestart	The footnote numbers restart at each section.

Page Information

Control Word	Meaning
\paperw<N>	The paper width (the default is 12,240).
\paperh<N>	The paper height (the default is 15,840).
\margl<N>	The left margin (the default is 1,800).
\margr<N>	The right margin (the default is 1,800).
\margt<N>	The top margin (the default is 1,440).
\margb<N>	The bottom margin (the default is 1,440).
\facingp	Facing pages (activates odd/even headers and gutters).
\gutter<N>	The gutter width (the default is 0).
\margmirror	Switches margin definitions on left and right pages.
\landscape	Landscape format.

\pgnstart<N>	The beginning page number (the default is 1).
\widowctrl	Widow control.

Revision Marks

Control Word	Meaning
\revisions	Turns on revision marking.
\revprop<N>	Argument indicates how revised text will be displayed: 0 for no properties shown; 1 for bold; 2 for italic; 3 for underline (the default); 4 for double underline.
\revbar<N>	Vertical lines mark altered text, based on the argument: 0 for no marking; 1 for left margin; 2 for right margin; 3 for outside (left on left pages, right on right pages; the default).

SECTION FORMATTING PROPERTIES

This group specifies section formatting properties, which apply to the text FOLLOWING the control word, with the exception of the section break control words (those beginning with \sbk). Section break control words describe the break PRECEDING the text.

Section Formatting Control Words

Control Word	Meaning
\sectd	Reset to the default section properties.
\endnhere	Endnotes included in the section.
\binfsxn<N>	<N> is the printer bin used for the first page of the section. If this control is not defined, the first page uses the same printer bin as defined by the \binsxn<N> control.
\binsxn<N>	<N> is the printer bin used for the pages of the section.

Section Break

Control Word	Meaning
\sbknone	No section break.
\sbkcol	The section break starts a new column.
\sbkpage	The section break starts a new page (the default).

\sbkeven	The section break starts at an even page.
\sbkodd	The section break starts at an odd page.

Columns

Control Word	Meaning
\cols<N>	The number of columns for "snaking" (the default is 1).
\colsx<N>	The space between columns in twips (the default is 720).
\linebetcol	Insert a line between columns.

Line Numbering

Control Word	Meaning
\linemod<N>	The line number modulus amount to increase each line number (the default is 1).
\linex<N>	The default value for <N> is 360 if \linex is omitted. A value of 0 for <N> means the automatic distance for the application reading the RTF file.
\linestarts<N>	The beginning line number (the default is 1).
\linerestart	The line numbers restart at the \linestarts value.
\lineppage	The line numbers restart on each page.
\linecont	The line numbers continue from the preceding section.

Page Information

Control Word	Meaning
\pgwsxn<N>	<N> is the page width in twips. A \sectd control resets the value to that specified by \paperw<N> in the document properties.
\pghsxn<N>	<N> is the page height in twips. A \sectd control resets the value to that specified by \paperh<N> in the document properties.
\marglsxn<N>	<N> is the left margin of the page in twips. A \sectd control resets the value to that specified by \margl<N> in the document properties.
\margrsxn<N>	<N> is the right margin of the page in twips. A \sectd control resets the value to that specified by \margr<N> in the document properties.

\margtsxn<N>	<N> is the right margin of the page in twips. A \sectd control resets the value to that specified by \margr<N> in the document properties.
\margbsxn<N>	<N> is the top margin of the page in twips. A \sectd control resets the value to that specified by \margt<N> in the document properties.
\guttersxn<N>	<N> is the width of the gutter margin for the section in twips. A \sectd control resets the value to that specified by \gutter<N> in the document properties. If facing pages is turned off, the gutter will be added to the left margin of all pages. If facing pages is turned on, the gutter will be added to the left side of odd-numbered pages and the right side of even-numbered pages.
\lndscpsxn	Page orientation is in landscape format. To mix portrait and landscape sections within a document, the \landscape control should not be used so that the default for a section is portrait, which may be overridden by the \lndscpsxn control.
\titlepg	The first page has a special format.
\headery<N>	The header is <N> twips from the top of the page (the default is 720).
\footery<N>	The footer is <N> twips from the bottom of the page (the default is 720).

Page Numbers

Control Word	Meaning
\pgnstarts<N>	The beginning page number (the default is 1).
\pgncont	The continuous page numbering (the default).
\pgnrestart	The page numbers restart at the \pgnstarts value.
\pgnx<N>	The page number is <N> twips from the right margin (the default is 720).
\pgny<N>	The page number is <N> twips from the top margin (the default is 720).
\pgndec	The page-number format is decimal.
\pgnucrm	The page-number format is uppercase Roman numerals.
\pgnlcrm	The page-number format is lowercase Roman numerals.
\pgnucltr	The page-number format is uppercase letters.
\pgnlcltr	The page-number format is lowercase letters.

Vertical Alignment

Control Word	Meaning
\vertalt	The text is top aligned on the page (the default).
\vertal	The text is bottom aligned on the page.
\vertalc	The text is centered vertically on the page.
\vertalj	The text is justified vertically on the page.

PARAGRAPH FORMATTING PROPERTIES

This group specifies paragraph formatting properties. To ensure compatibility with previous versions of RTF, if border properties are specified, the border segment control word (\brdrt,\brdrb,\brdrl, \brdrr, or \box) must precede the control word(s) that specify the pattern for the border.

Paragraph Formatting Control Words

Control Word	Meaning
\pard	Resets to the default paragraph properties.
\s<N>	Designates the reference number of the style in the RTF document; if a style is referenced within the document, style properties must be included along with the style reference in the document text.
\intbl	The paragraph is part of a table.
\keep	Keep the paragraph intact.
\keepn	Keep the paragraph with the next paragraph.
\noline	No line numbering.
\pagebb	Break the page before the paragraph.
\sbys	Side-by-side paragraphs.

Alignment

Control Word	Meaning
\ql	Left aligned (the default)
\qr	Right aligned
\qj	Justified
\qc	Centered

Indentation

Control Word	Meaning

\fi<N>	First-line indent (the default is 0).
\li<N>	Left indent (the default is 0).
\ri<N>	Right indent (the default is 0).

Spacing

Control Word	Meaning
\sb<N>	Space before (the default is 0).
\sa<N>	Space after (the default is 0).
\sl<N>	Space between lines. If this control word is missing or if \sl1000 is used, the line spacing is automatically determined by the tallest character in the line; if <N> is a positive value, this size is used only if it is taller than the tallest character (otherwise, the tallest character determines the size); if <N> is a negative value, the absolute value of <N> determines the size, even if it is shorter than the tallest character.

Tabs

Control Word	Meaning
\tx<N>	The tab position in twips from the left margin.
\tqr	A flush-right tab.
\tqc	A centered tab.
\tqdec	A decimal tab.
\tb<N>	A bar tab position in twips from the left margin.
\tldot	Leader dots.
\tlhyph	Leader hyphens.
\tlul	A leader underline.
\tlth	A leader thick line.
\tleq	A leader equal sign.

Paragraph Borders

Control Word	Meaning
\brdrt	Border top.
\brdrb	Border bottom.
\brdrl	Border left.
\brdrr	Border right.
\brdrbtw	Consecutive paragraphs with identical border formatting are considered part of a single group with the border information applying to the entire group. To have borders around individual paragraphs within the group, the \brdrbtw control must be

	specified for that paragraph.
\brdrbar	Box border outside (right side of odd-numbered pages, left side of even-numbered pages).
\box	Border around the paragraph.
\brdrs	Single-thickness border.
\brdrth	Thick border.
\brdrsh	Shadowed border.
\brdrdb	Double border.
\brdrdot	Dotted border.
\brdrhair	Hairline border.
\brdrw<N>	<N> is the width in twips of the paragraph border line. This control should follow the paragraph border controls \brdrt, \brdrr, \brdrb, and \brdrl.
\brdrcf<N>	<N> is the color of the paragraph border from the color table in the RTF header. This control should follow the paragraph border controls \brdrt, \brdrr, \brdrb, and \brdrl.
\brsp<N>	Space in twips between borders and the paragraph.

Paragraph Shading and Background Pattern

Control Word	Meaning
\shading<N>	<N> is the shading of the paragraph in hundredths of a percent.
\bghoriz	Specifies a horizontal background pattern for the paragraph.
\bgvert	Specifies a vertical background pattern for the paragraph.
\bgfdiag	Specifies a forward diagonal background pattern for the paragraph (\\\\).
\bgbdiag	Specifies a backward diagonal background pattern for the paragraph (////).
\bgcross	Specifies a cross-hatched background pattern for the paragraph.
\bgdcross	Specifies a diagonal cross-hatched background pattern for the paragraph.
\bgdkhoriz	Specifies a dark horizontal background pattern for the paragraph.
\bgdkvert	Specifies a dark vertical background pattern for the

paragraph.

\bgdkfdiag	Specifies a dark forward diagonal background pattern for the paragraph (\\\\).
\bgdkbdiag	Specifies a dark backward diagonal background pattern for the paragraph (////).
\bgdkcross	Specifies a dark cross-hatched background pattern for the paragraph.
\bgdkdcross	Specifies a dark diagonal cross-hatched background pattern for the paragraph.
\cfpat<N>	<N> is the line color of the background pattern.
\cbpat<N>	<N> is the background color of the background pattern.

POSITIONED OBJECTS AND FRAMES

The following paragraph-formatting control words specify the location of a paragraph on the page. Consecutive paragraphs with the same frame formatting are considered to be part of the same frame. For two framed paragraphs to appear at the same position on a page, they must be separated by a paragraph with different or no frame information.

Control Word	Meaning
\absw<N>	<N> is the width of the frame in twips.
\absh<N>	<N> is the height of the frame in twips. A positive number indicates the minimum height of the frame, and a negative number indicates the exact height of the frame. A value of 0 indicates that the height of the frame adjusts to the contents of the frame. Zero is the default for frames where no height is given.

Horizontal Position

Control Word	Meaning
\phmrg	Use the margin as the horizontal reference frame.
\phpg	Use the page as the horizontal reference frame.
\phcol	Use the column as the horizontal reference frame. This is the default if no horizontal reference frame is given.
\posx<N>	Positions the frame <N> twips from the left edge of the reference frame.
\posxc	Centers the frame horizontally within the reference frame.

\posxi	Positions the paragraph horizontally inside the reference frame.
\posxo	Positions the paragraph horizontally outside the reference frame.
\posxr	Positions the paragraph to the right within the reference frame.
\posxl	Positions the paragraph to the left within the reference frame. This is the default if no horizontal positioning information is given.

Vertical Position

Control Word	Meaning
\pvmrg	Use the margin as the vertical reference frame.
\pvpg	Use the page as the vertical reference frame.
\pvpara	Positions the reference frame vertically relative to the top of the top-left corner of the next unframed paragraph in the RTF stream. This is the default if no vertical frame positioning information is given.
\posy<N>	Positions the paragraph <N> twips from the top edge of the reference frame.
\posyil	Positions the paragraph vertically to be in line.
\posyt	Positions the paragraph at the top of the reference frame.
\posyc	Centers the paragraph vertically within the reference frame.
\posyb	Positions the paragraph at the bottom of the reference frame.

Text Wrapping

Control Word	Meaning
\dxfrtext<N>	Distance in twips of a positioned paragraph from the main text flow in all directions. In Word for Windows 2.0, this control affects only the horizontal distance from the text on each side of the frame.
\dyfrtext<N>	<N> is the vertical distance in twips from the text above and below the frame.

The following is an example of absolute-positioned text in a document:

A
|

```
...                                                          -------------
\par \pard \pvpg\phpg\posxc\posyt\absw5040\dxfrtest173 abs pos para1
\par \pard \phmrg\posxo\posyc\dxfrtext1152 abs pos para2
                                                 -------------
                                                       |
                                                       A
```

A: Text to be positioned

TABLES

There is no RTF table group. A table is a collection of paragraphs, and a table row is a continuous sequence of paragraphs partitioned into cells. The \intbl paragraph formatting control word identifies the paragraph as part of a table. This control is inherited between paragraphs that do not have paragraph properties reset with a \pard. The last paragraph of a cell is terminated by a cell mark (the \cell control word), and the row is terminated by a row mark (the \row control word). The following control words further define the table:

Control Word	Meaning
\trowd	Sets the table row defaults.
\trgaph<N>	Half the space between the cells of a table row in twips.
\cellx<N>	Moves the right boundary of a table cell, including its half of the space between cells.
\clmgf	The first cell in a range of table cells to be merged.
\clmrg	The contents of the table cell are merged with those of the preceding cell.

Cell Borders and Shading

Control Word	Meaning
\clbrdrb	The bottom table cell border.
\clbrdrt	The top table cell border.
\clbrdrl	The left table cell border.
\clbrdrr	The right table cell border.

Row Formatting

Control Word	Meaning
\trql	Left-justifies a table row with respect to its containing column.
\trqr	Right-justifies a table row with respect to its containing column.

\trqc	Centers a table row with respect to its containing column.
\trleft<N>	The position of the leftmost edge of the table with respect to the left edge of its column.
\trrh<N>	The height of a table row in twips; when 0, the height is sufficient for all the text in the line; when positive, the height is guaranteed to be at least the specified height; when negative, the absolute value of the height is used, regardless of the height of the text in the line.

Cell Shading and Background Pattern

Control Word	Meaning
\clshdng<N>	<N> is the shading of a table cell in hundredths of a percent. This control should be included in RTF along with cell border information.
\clbghoriz	Specifies a horizontal background pattern for the cell.
\clbgvert	Specifies a vertical background pattern for the cell.
\clbgfdiag	Specifies a forward diagonal background pattern for the cell (\\\\).
\clbgbdiag	Specifies a backward diagonal background pattern for the cell (////).
\clbgcross	Specifies a cross-hatched background pattern for the cell.
\clbgdcross	Specifies a diagonal cross-hatched background pattern for the cell.
\clbgdkhor	Specifies a dark horizontal background pattern for the cell.
\clbgdkvert	Specifies a dark vertical background pattern for the cell.
\clbgdkfdiag	Specifies a dark forward diagonal background pattern for the cell (\\\\).
\clbgdkbdiag	Specifies a dark backward diagonal background pattern for the cell (////).
\clbgdkcross	Specifies a dark cross-hatched background pattern for the cell.
\clbgdkdcross	Specifies a dark diagonal cross-hatched background pattern for the cell.
\clcfpat<N>	<N> is the line color of the background pattern.

\clcbpat<N>	<N> is the background color of the background pattern.

The following example shows some table text:

```
...
\par \trowd \trqc\trgaph108\trrh280\trleft36
\clbrdrt\brdrth \clbrdrl\brdrth \clbrdrb\brdrdb
\clbrdrr\brdrdb \cellx3636\clbrdrt\brdrth
...
```

CHARACTER FORMATTING PROPERTIES

The last group controls character formatting properties. A control word preceding plain text turns on the specified attribute. Some control words (indicated in the following table by an asterisk following the description) can be turned off by the control word followed by 0 (zero). For example, \b turns on bold, whereas \b0 turns off bold.

The character formatting control words are listed in the following table:

Control Word	Meaning
\plain	Resets application's default character formatting properties.
\b	Turns on bold.*
\caps	Turns on all uppercase.*
\deleted	Marks the text as deletion revision marked.*
\dn<N>	Sets the subscript position in half-points (the default is 6).
\expnd<N>	Sets the expansion or compression of the space between characters in quarter-points; a negative value compresses the space (the default is 0).
\f<N>	Identifies the font number.
\fs<N>	Sets the font size in half-points (the default is 24).
\i	Turns on italic.*
\outl	Turns on outline.*
\revised	Indicates that text has been added since revision marking was turned on.
\scaps	Turns on small capitals.*
\shad	Turns on shadow.*
\strike	Turns on strikethrough.

\ul	Turns on continuous underline. \ul0 turns off all underlining.
\uld	Turns on dotted underline.
\uldb	Turns on double underline.
\ulnone	Turns off all underlining.
\ulw	Turns on word underline.
\up<N>	Sets the superscript position in half-points (the default is 6).
\v	Turns on hidden text.*
\lang<N>	Applies a language to a character. <N> is a number corresponding to a language. The following table defines the standard languages used by Microsoft. This table was generated by the Unicode group for use with TrueType(R) and Unicode. A \plain control resets the language property to the language defined by \deflang<N> in the document properties.

Language Name	Language ID
No Language	0x0400
Albanian	0x041c
Arabic	0x0401
Bahasa	0x0421
Belgian Dutch	0x0813
Belgian French	0x080c
Brazilian Portuguese	0x0416
Bulgarian	0x0402
Catalan	0x0403
Croato-Serbian (Latin)	0x041a
Czech	0x0405
Danish	0x0406
Dutch	0x0413
English (Aus.)	0x0c09
English (U.K.)	0x0809
English (U.S.)	0x0409
Finnish	0x040b
French	0x040c
French (Canadian)	0x0c0c
German	0x0407
Greek	0x0408
Hebrew	0x040d
Hungarian	0x040e
Icelandic	0x040f
Italian	0x0410
Japanese	0x0411
Korean	0x0412
Norwegian (Bokmal)	0x0414
Norwegian (Nynorsk)	0x0814
Polish	0x0415
Portuguese	0x0816
Rhaeto-Romanic	0x0417
Romanian	0x0418

Russian	0x0419
Serbo-Croatian (Cyrillic)	0x081a
Simplified Chinese	0x0804
Slovak	0x041b
Spanish (Castilian)	0x040a
Spanish (Mexican)	0x080a
Swedish	0x041d
Swiss French	0x100c
Swiss German	0x0807
Swiss Italian	0x0810
Thai	0x041e
Traditional Chinese	0x0404
Turkish	0x041f
Urdu	0x0420

To read negative \expnd values from Word for the Macintosh, an RTF reader should use only the low-order 6 bits of the value read. Word for the Macintosh does not emit negative values for \expnd. Instead, it treats values from 57 through 63 as -7 through -1, respectively (the low-order 6 bits of 57 through 63 are the same as -7 through 1).

SPECIAL CHARACTERS

The RTF standard includes control words for special characters. If a special character control word is not recognized by the RTF reader, the control word is ignored, and the text following it is considered plain text. The RTF specification is flexible enough to allow new special characters to be added for interchange with other software.

The special RTF characters are listed in the following table:

Control Word	Meaning
\chdate	The current date (as in headers).
\chdpl	The current date in long format; for example, Tuesday, June 28, 1992.
\chdpa	The current date in abbreviated format; for example, Tue, Jun 28, 1992.
\chtime	The current time (as in headers).
\chpgn	The current page number (as in headers).
\chftn	An automatic footnote reference (the footnotes follow in a group).
\chatn	An annotation reference (the annotation text follows in a group).
\chftnsep	An anchoring character for the footnote separator.
\chftnsepc	An anchoring character for the footnote continuation.
\cell	The end of a table cell.

\row	The end of a table row.
\par	The end of a paragraph.
\sect	The end of a section and a paragraph.
\page	A required page break.
\column	A required column break.
\line	A required line break (no paragraph break).
\tab	A tab character; the same as ASCII 9.
\emdash	An em (--) dash. This is character 151 in the ANSI character set and character 208 in the Macintosh character set.
\endash	An en (-) dash. This is character 150 in the ANSI character set and character 209 in the Macintosh character set.
\bullet	A bullet character. This is character 149 in the ANSI character set and character 165 in the Macintosh character set.
\lquote	An opening single quotation mark. This is character 145 in the ANSI character set and character 212 in the Macintosh character set.
\rquote	A closing single quotation mark. This is character 146 in the ANSI character set and character 213 in the Macintosh character set.
\ldblquote	An opening double quotation mark. This is character 147 in the ANSI character set and character 210 in the Macintosh character set.
\rdblquote	A closing double quotation mark. This is character 148 in the ANSI character set and character 211 in the Macintosh character set.
\\|	A formula character.
\~	A nonbreaking space.
\-	An optional hyphen.
_	A nonbreaking hyphen.
\:	Specifies a subentry in an index entry.
*	Marks a destination whose text should be ignored if not understood by the RTF reader.
\'hh	A hexadecimal value, based on the specified character set (may be used to identify 8-bit values).
\alt	The ALT modifier key. Used to describe shortcut-key codes for styles.

\shift	The SHIFT modifier key. Used to describe shortcut-key codes for styles.
\ctrl	The CTRL modifier key. Used to describe shortcut-key codes for styles.
	Specifies a function key where <N> is the function key number. Used to describe shortcut-key codes for styles.

NOTE: An ASCII 9 is accepted as a tab character. A carriage return (character value 13) or linefeed character (character value 10) will be treated as a \par control if the character is preceded by a backslash. You must include the backslash, or RTF ignores the control word. (You may also want to insert a carriage-return/linefeed pair without backslashes at least every 255 characters for better text transmission over communication lines.)

Exhibit M

of the association for computing machinery

newsletter

special interest group on biomedical computing

TABLE OF CONTENTS

February 1993 Volume 13 Number 1

Processing of Cross-Sectional Image Data for Reconstruction of Human Developmental Anatomy from Museum Specimens

Michael D. Doyle, Cheong Ang, Rakesh Raju, *Gary Klein, **Betsey S. Williams, Thomas DeFanti, Ardeshir Goshtasby, ***Robert Grzesczuk and ****Adrianne Noe

The University of Illinois at Chicago, *Amoco Oil Co., **Harvard Medical School, ***the University of Chicago, and ****the National Museum of Health and Medicine, AFIP

A project is described (The Visible Embryo Project) to develop software strategies for the creation of large-scale databases of 3-dimensional image data on human developmental anatomy. The issues discussed relate to the processing of serial-section image data for the purpose of reconstructing volumetric models of individual embryos from museum specimens. Attempts were made to fully automate the processes of image registration and artifact correction, in order to allow for the eventual unattended reconstruction of thousands of such embryos from such resources as the Carnegie Collection of Human Embryology. Tools were also developed to allow the real-time interactive visualization of the massive databases that this type of reconstruction project creates. The implementation of a 3-D model of a human embryo in a viewer-centric virtual reality environment is also described.

Introduction

Congenital malformations have challenged the humanistic and intellectual resources of mankind for ages. Generations of scientists have labored to understand their causes and effect their treatments. Despite this legacy, there are a variety of problems currently plaguing the discipline of human embryology. Although many remarkable collections of human embryological materials have been built up over the last century, the nature of these materials dictates that they be housed in museums under tight lock and key. The wealth of knowledge represented by such archives is largely accessible only through the publications of earlier researchers who studied various aspects of the collections. New methods for computer-based 3-dimensional morphometric analysis are enabling developmental biologists to study the dynamics of human development in ways that were not possible until only a few years ago. The publications of early researchers in the field are of little value to these new quantitative biologists. Yet these researchers have been severely limited by a lack of free access to large populations of embryological specimens. New technologies in high performance computing and communications are beginning to enable museums and research centers to open up their collections, in electronic form, to researchers, educators and students throughout the nation without endangering the fragile specimens themselves. The presently-described research is part of an effort to develop the technologies necessary for such a "museum without walls" to take form.

The Center for Human Developmental Anatomy

For over a century, scholars have perceived a need for an internationally accessible center for research in human developmental anatomy. Consistent with its missions to support consultation, education, and research in medicine and related disciplines, the staff at the United States Armed Forces Institute of Pathology began in 1990 to develop such a center at its National Museum of Health and Medicine. Its long-time experience in both conducting and supporting collections-based scientific research have allowed the staff and associates to identify prospectively the collections that will serve critical medical and scientific needs. Beginning with the acquisition of the world-renowned Carnegie Human Embryology Collection (O'Rahilly, 1987), the Institute continues to identify

and collect well-documented materials and carefully houses them for perpetual availability to scholars. The Institute's mission is to preserve these and other materials relating to developmental anatomy in a careful, organized manner, which will eventually include extensive computer-based image archiving and the development of large-scale, remotely accessible, database on the Center's collections. The Center relies on supported research agendas to activate these materials and make them useful around the globe.

The Visible Embryo Project

A multi-institutional, multidisciplinary research project (the Visible Embryo Project) is being led by the Developmental Anatomy Center and the Biomedical Visualization Laboratory at the University of Illinois at Chicago to develop software strategies for the development of distributed biostructural databases using current technologies for high-performance computing and communications (HPCC), and to implement these tools in the creation of a large-scale digital archive of multidimensional data on normal and abnormal human development (Doyle, et al., 1992).. This project relates to BVL's long-term activity in the areas of health informatics, educational multimedia, and biomedical imaging science (Doyle, et al., 1990-92, Carlbom, et al., 1992).

Serial-section reconstruction has long been a tool used by morphologists, and especially embryologists, to study the three-dimensional structure of microscopic anatomy. The early workers typically hand traced the contours of microscopic sections projected through a camera lucida, cut out the contours from pieces of cardboard of a thickness corresponding to the scale of the reconstruction, and carefully glued them together, in register, to form a model of the 3-D surface of the structure of interest (O'Rahilly,1987).

Several workers have attempted to extend this approach into computer environments by tracing the contours of consecutive sections on a digitizing tablet, and then using computer graphics to render the 3D surface described by the stacked contours (Prothero,1986, Wilkinson,1990). Two embryos from the Blechschmidt collection have been reconstructed in this manner and visualized using a polygonal surface-based reconstruction approach (personal communication Cornelius Rosse). In the mouse, Lawson of the Hubrecht Laboratory, has attempted reconstruction from plastic histologic sections. These two approaches have in common the limitation that only the surface of the structure can be studied from a given model. If a reconstruction of the exterior of the heart were performed, for example, entirely separate reconstructions would have to be performed in order to visualize the chambers, or the myocardial vascular patterns.

Most of the early work in the Visible Embryo project has involved using serial section reconstruction from microscope slides, however we extracted volumetric data from these specimens, rather than just surface data. Sets of serial microscopic cross-sections through human embryos (prepared between the 1890s and 1970s) are being used as sample image data around which to design and implement various components of the system. These images are digitized and processed to create both 3D voxel datasets and polygonal models representing embryonic anatomy. Standard techniques for 3D volume visualization could then be applied to these data (Carlbom 1992, Doyle 1989-92, Karssemeijer 1988, Leichtman 1990, Levoy 1990, Lorenson 1987, McClean 1991, Prothero 1986, Terzopoulos 1987, Udupa 1990, Wilkinson 1990, Yuille 1986). Processing of these artifacts was required to correct for certain artifacts that were found in the original microscope sections from routine histological techniques of tissue preparation.

This processing includes, as will be discussed below, registration of adjacent serial sections, specialized methods of histogram equalization to correct for staining irregularities, and various levels of conformal warping to correct for dimensional instability (stretching, etc.) in the embedding material. Predominantly-automatic software tools have been developed by the UIC BVL team to accomplish these tasks, thereby enabling the future processing of huge numbers of images to create large-scale datasets describing human development through the first trimester. Remote-access visualization and database tools are under development to allow real-time interaction with these enormous datasets by distributing certain computational tasks to super computers.

Materials and Methods

A human embryo specimen was acquired, on loan, from the collection of the Human Developmental Anatomy Center. This specimens comprised a set of serial transverse sections through a 7-week-old embryo. The sectioned specimen was in the form of paraffin embedded 10μm-thick sections stained with haemotoxylin and eosin and mounted on glass slides with coverslips. Figure 1 shows a stereo pair image of a human embryo of similar age.

The microscopic sections through the embryo specimen were transilluminated with a voltage-stabi

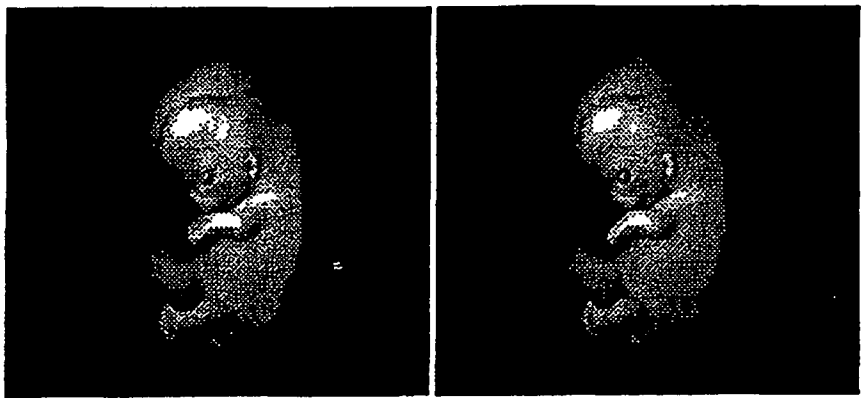


Figure 1: Stereo-pair image of a 7 to 8 week old human embryo from the Carnegie Collection of Human Embryology.

lized dichroic light source. Video images were created with a Sony XC-77 CCD monochrome video camera through a Micro Nikkor 105mm macro lens with extension tubes, and then digitized with a Jovian Logic frame grabber at a spatial resolution of 640x480 pixels with 256 gray levels.

Images were stored in the TIF format with LZW (lossless) compression on 3.5" magneto-optical disks. A total of 1297 images were thereby obtained of the embryo, with each image corresponding to an individual physical section through the embryo. The magnification of the imaging system was such that the digitized images were made up of square pixels with a side length of 9mm.

Image Processing

The serial cross-section images were first processed in order to isolate the embryonic tissue from the surrounding area. This was accomplished through the use of the automatic edge-tracing feature of Image Pro Plus from Media Cybernetics Corp. Image Pro Plus also allowed the x,y coordinates of the external contour of each embryo section to be saved to an ASCII file. This proved to be very useful in the later stage of the project devoted to creating a surface-based polygonal model of the embryo exterior.

Registration of Images

Since the embryo specimen had been embedded and sectioned in the 1930's, fiducial markers were not included in the embedment to assist in the computer reconstruction of the embryo. It was therefore necessary to process the image data in order to restore topological relationships between tissue elements, and to restore dimensional accuracy in the direction perpendicular to the plane of section. We were therefore forced to "reverse engineer" proper image registration based solely upon intrinsic image features. This was accomplished through the creation of an algorithm by one of us (Klein) that used an error minimization technique he calls "iterative chasing." This technique attempts to find the "best" difference result between two adjacent images in a series, such that the following value is minimized:

$$H(x,y) = SS\ |r(x,y) - R(x,y)|^{xy}$$

where r(x,y) represents the first image (the reference image) and R(x,y) represents the second image (the mobile image) in a sequential pair. The program iteratively performs transformations in terms of x and y translations and rotations of the mobile image, while holding the position of the reference image stationary. The assumption is made that the closer one gets to the optimal transformation, the lower will be the value H. The 3-dimensional matrix of h (x translation), k (y translation), and θ (rotation), one dimension at a time. First, the best h transformation is found while k and q are kept at zero. Then the best k transformation is sought while h = *h-best* and θ=0. A third set of transformations are then done to find q while h = *h-best* and k = *k-best*. This sequence is then repeated continuously until H(x,y) does not decrease any more. We then use *h-best*, *k-best* and θ-*best* as the coefficients of transformation for the mobile image. The program then saves the transformed mobile image to disk, replaces the image in the reference buffer with the mobile image, and then loads the next sequential serial-section image into the mobile image buffer. The registration proce-

dure then begins again. This process is repeated automatically until all of the images of sequential cross-sections are processed and saved to disk.

Once proper registration of adjacent images was restored to the embryo data, these data were loaded into a 3 dimensional array representing voxels where each voxel had the dimensions 9mm x 9mm (the pixel size) x 10 μm (the standard section thickness). Standard techniques for volume visualization could then be used to examine the data.

Figure 2 shows a para-saggital section through the 3-D embryo dataset in the region of the thorax, at a plane perpendicular to the original plane of sectioning. Two important categories of image artifacts can be seen in this image, both a result of the original histological techniques used when the embryo was sectioned. First, although the heart (arrow) looks reasonably-well reconstructed, a closer examination shows a horizontal banding resulting from both staining irregularities and dimensional instability (shrinkage variations) among the set of serial cross sections. This can be seen throughout the sagittal image, but is especially evident in the area of the liver, which is immediately below the heart. There is a strong periodicity to this effect which makes the image appear as if it was made up of a set of "slabs" which were found to correspond to roughly 12 cross-sections each in the original data. The source of the periodicity of this effect may be clearer when one considers the routine techniques that were probably used by the 1930's era histology technician that prepared the original specimen.

It can be assumed that the embryo was first dehydrated with a graded alcohol series and then embedded in paraffin before sectioning began. The sectioning occurred on a guillotine-style microtome that made precise 10 mm-thick sections perpendicular to the long axis of the embryo. As the sections were cut, they formed ribbons that floated out onto a "boat" filled with water. They were then lifted out of the water directly onto a glass microscope slide and placed on a slide-warmer to dry. Once a number of slides were prepared in this way, they were probably then placed, end up, into a typical "staining rack" before being immersed in a series of solutions for staining of the important structures. They were then probably left to dry in the staining rack and then removed from the rack to be examined or stored.

When we examined the positions of the sections on the microscope slides, we found that they were placed on the glass in a rectangular grid with 4 to 5 rows of 12 section ribbons. If these slides had been dried in a vertical position after staining, then the sections at one end of a row would have dried faster than those at the other end of the row. The sections that dried faster may have shrunk at a different rate that those that spent more time being wet. In addition some rows of sections may have been exposed to stain for a longer time than others, and this would result in those rows staining darker than the others. Therefore, the handling of the specimens during histological processing may account for the banding artifacts in the original dataset. This is supported by the fact that the 12-section thickness of the bands correlates with the 12

Figure 2: Parasagittal section through the thorax region of the auto-registered data. The arrow points toward the heart area. The liver can be seen below the heart.

sections per row found on the glass slides.

Staining irregularities were corrected for by histogram equalization. Optimal histograms were calculated for every 26th image among the 1300 images from the embryo; 50 such histograms in all. Histogram equalization curves were saved to disk for each of these optimal histograms. Each of the equalization curves were then used for the correction of the following 26 images. Each of those images were processed so that their histogram matched the optimal histogram for that 26 image segment. This produced an image dataset with minimized banding and enhanced contrast, yet in a way that was sensitive to the topographical variations in image character throughout the embryo. Dimensional instability was corrected for through a the use of an algorithm (Goshtasby, 1986) that conformally-warped the images based upon a pixel-to-pixel comparison between adjacent images, and that was "aware" of the periodicity of the shrinkage artifact measured in the original data.

It was found that additional artifacts were introduced into the data by the registration program itself. Due to the nature of the registration algorithm, it tended to "straighten out" any global curvatures or twists that existed in the original embryo. These "global artifacts" were partially corrected by interactively defining a "correction spline" in a direction parallel to the cranial-caudal axis of the body. The correction spline was then used as a guide for a smooth progression of x and y translations that restored a natural-appearing curve to the embryo.

3-D Visualization Tools

Software tools were developed to allow the interactive three-dimensional visualization of the embryo reconstruction in real time. Figure 3 shows the display of the application as it appeared at the SIGGRAPH '92 conference in Chicago (Doyle, et al., 1992). The left of the screen shows a surface-based model of the embryo's exterior. This model was built from data which was derived, through three-dimensional interpolation, from the original embryo dataset . Two-hundred volume slices of the embryo (stored as texture maps) can be interactively displayed at this lower resolution while the model is rotated freely in three dimensions. A cutting-plane can be seen to intersect the surface-based model. This cutting plane can be interactively controlled to intersect with the embryo model at any arbitrary angle and position. To the right of the screen, one can see a window that displays a high-resolution image of the oblique section through the embryo as indicated by the interactive cutting plane. In order to maintain the quick response needed for effective real-time interaction, the computational load of this application was distributed so that the interface panel, seen at the bottom of the screen, and the 3-D surface model were running on the CPU of the Silicon Graphics workstation. Computation of the high-resolution oblique section image displayed in the right window took place on the Convex supercomputer. Both of these operations occurred simultaneously, communicating through a high-speed fiber optic network.

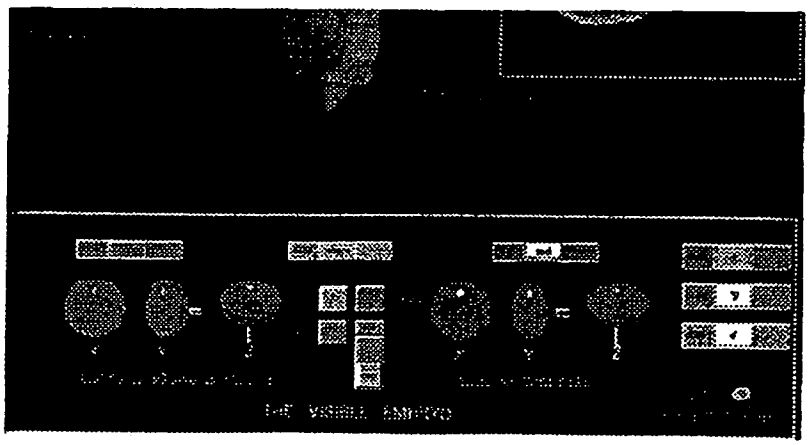


Figure 3: Screen image of the 3-D visualization program demonstrated at SIGGRAPH '92, in Chicago. This application was distributed between a Silicon Graphics Crimson VGXT workstation and a Convex supercomputer.

Current efforts are being directed towards the development of a very portable tool for viewing arbitrary oblique slices through such data. This program allows interactive display of orthogonal and oblique slices through volumetric data without using any machine-specific functions. The application is written in pure ANSI-standard C and uses the X Window (Motif) toolkit for its interface. It has already been successfully tested on workstations from Silicon Graphics, Sun, and IBM.

Virtual Reality

The surface-based embryo-model described above was also implemented within a virtual reality environment, called "The Cave," at the 1992 SIGGRAPH conference. The Cave was a 10' x 10' room made up of back-projection screens upon which were projected stereo views of three dimensional data that a viewer, wearing LCD stereo glasses and a 3-D tracking device, could move around in as the system tracked his or her motion. One could walk around the data and receive the sensation that one was actually "in" the computer graphic environment. This system was also shown at the Supercomputing '92 conference and the 1992 meeting of the Radiological Society of North America.

Summary

The work described here represents the very earliest stages of the Visible Embryo project. Its long term goals are simply to create a comprehensive multimedia database that documents and describes both normal and abnormal human morphological development in multiple dimensions. This is in all respects a long term endeavor. Yet even the earliest stages of progress towards this goal will be of importance to a variety of investigators, educators and students of embryology and many ancillary fields. Among the obvious areas that will benefit from this work are medical education, cell and developmental biology, biomedical informatics, biomedical imaging science and high performance computing. Perhaps the most important benefit will be that it will form a model for similar projects in other fields ranging from the life sciences to the humanities.

Bibliography

1) Atlas, S.W., Braffman, B.H., LoBrutto, R., Elder, D.E. and Herlyn, D. Human malignant melanomas with varying degrees of melanin content in nude mice: MR imaging, histopathology, and electron paramagnetic resonance; *J. Comp. Asst. Tomog.* 14/4: 547-554 (1990).

2) Carlbom, I., Hsu, W.M., Klinker, G., Szelski, R., Waters, K., Doyle, M.D., Gettys, J., Harris, K.M., Levergood, T.M., Palmer, R., Palmer, L., Picart, M., Terzopoulos, D., Tonnessen, D., Vannier, M., and Wallace, G. Modeling and Analysis of Empiracle Data in Collaborative Environments, *Communications of the ACM*, 33/6: 75-84, (1992).

3) Cersen, S., Lotscher, H.R., Kummecke, B. and Seelig, J. Monoclonal antibody-coated magnetite particles as contrast agents in magnetic resonance imaging of tumors *Magnetic Resonance in Medicine* 12:151-163 (1989).

4) Doyle, M.D., Raju,R., Ang,C., Goshtasby,A., and DeFanti, T. The Virtual Embryo: real-time viewer-centric 4-D visualization of human embryonic anatomy, presented at the High Performance Computing and Communication Showcase, SIGGRAPH '92, the annual meeting of the Association for Computing Machinery's Special Interest Group for Graphics, Chicago. (August 1992).

5) Doyle, M.D., Raju, R., Ang, C., Klein,G., Goshtasby, A., DeFanti, T., and Noe, A.: The Visible Embryo distributed (workstation/supercomputer) interactive visualization of high-density 3D image data of embryonic anatomy, presented at the High Performance Computing and Communication Showcase, SIGGRAPH '92, the annual meeting of the Association for Computing Machinery's Special Interest Group for Graphics, Chicago. (August, 1992).

6) Doyle, M.D. and Sadler, L.L. Health Informatics: MetaMap indexing and retrieval of image-based data for an institutional PACS system, presented at the 1991 Forum in Cellular and Organ Biology, American Association of Anatomists, Chicago, (April 1991).

7) Doyle, M.D. Palette Segmentation Indexing: The MetaMap Process, *SIGBIO Newsletter* (The Journal of the ACM Special Interest Group for Biological Computing), (February, 1992).

8) Doyle, M.D. The MetaMap Process: A New Approach to the Creation of Object-oriented Image Databases for Medical Education, *Proceedings of the 13th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, IEEE Press (1991).

9) Doyle, M.D., Raju, R., Ang, C., Klein, G., Gosh-

tasby, A.,, and DeFanti, T. Multidimensional reconstruction of human embryonic anatomy from museum-archived serial section collections, to be submitted to *Medical Imaging* (1993).

10) Doyle, M.D., Williams,B.S., Noe, A. The Visible Embryo Project: high performance computing, vizualization, and .communications for developmental anatomy research, to be submitted to *Anatomical Record* (1993).

11) Goshtasby, A.: A region based approach to digital image registration with subpixel accuracy, *IEEE Transactions on Geoscience and Remote Sensing*, GE-24, V03, May (1986)

12) Karssemeijer, N. Three-dimensional stochastic organ-models for segmentation in CT-scans *SPIE* 1030: 177-184 (1988).

13) Leichtman, G.S. and Anderson, D.J. Linking a relational database of biological features to computer-aided reconstruction of tissue. *Proc. 1st Conf. Visualization in Biomed. Computing*. IEEE Computer Society Press 288-294 (1990).

14) Levoy, M. A hybrid ray tracer for rendering polygon and volume data *IEEE Comp. Graphics and Applic.* 33-40 (1990).

15) Lorensen, W.E. Marching cubes: A high resolution 3-d surface construction algorithm. *ACM SIGGRAPH* 21/4: 38-44 (1987).

16) McLean, M. and Prothero, J.W. Three-dimensional reconstruction from serial sections. V. Calibration of dimensional changes incurred during tissue preparation and data processing. Anal. and Quant. Cytol. and Hist. 13: 269-278 (1991).

17) O"Rahilly, R. and Muller, F. Developmental stages in human embryos, Carnegie Institution of Washington, Publication no. 637. Carnegie Institute, Washington, DC. (1987).

18) Prothero, J.S. and Prothero, J.W. Three-dimensional reconstruction from serial sections. IV: The reassembly problem. Computers and Biomed. Res., 19: 361-373 (1986).

19) Terzopoulos, D. On matching deformable models to images. Topical Meeting on Machine Vision, Technical Digest Series (Optical Society of America, Washington, DC.) 12: 160-163 (1987).

20) Udupa, J.K. Course notes: Visualization of biomedical data: principles and algorithms. 1st Conf. on Visualization in Biomed. Computing. IEEE Computer Society Press, p14-16 (1990).

21) Wilkinson, D.G. and Green, J. In situ hybridization and the three-dimensional reconstruction of serial sections. IN: Copp, A.J. and Cockroft, D.L. eds. Postimplantation Mammalian Embryos: A Practical Approach. IRL Press, Oxford University. pp155-171(1990).

22) Yuille, A.L., Cohen, D.S., and Hallinan, P.W. Feature extraction from faces using deformable templates. Harvard Robotics Lab. Tech. Rep. 88/2: 104-109 (1988).

Calendar of Events

- AMIA 1993 Spring Congress: The Electronic Medical Record:
  (St. Louis, MO, May 9-12, 1993)

- First Utah Workshop on the Applications of Intelligent and Adaptive Systems:
  (Salt Lake City, Utah, May 10, 1993)
  [Appl-IAS-Utah]

- The Third International Workshop on Human & Machine Cognition
  (Expertise in Context: Human & Machine):
  (Seaside, Florida, May 13-15, 1993)
  [Expertise-93]

- A Workshop On Computational Neurosciences
  (Austin, TX, May 14-15, 1993)
  [Comp-Neurosci]

- 7th International Workshop on Qualitative Reasoning about Physical Systems:
  (Orcas Island, WA, May 16-19, 1993)
  [QR-93]

Exhibit N

OBJECT Linking & Embedding

OLE 2.01 Design Specification

Microsoft OLE2 Design Team
27 September, 1993

OBJECT Linking & Embedding

OLE 2.01 Design Specification

Microsoft OLE2 Design Team
27 September, 1993

MS-ET 0000066

Writers: Bob Atkinson, Tony Williams, Randy Kerr, Barry Potter, and the OLE2 Documentation Team
Editor: Bob Atkinson
Contributors: Bob Atkinson, Kraig Brockschmidt, Douglas Hodges, Randy Kerr, Srini Koppolu, Barry Potter, Tony Williams, Craig Wittenberg, the OLE2 Development Team and the OLE2 Documentation Team

2.5. States and Visualizations of Objects

With editing in place, OLE 2 enhances the conceptual structure of a compound document by exposing in a single window the entire hierarchy of containment. In OLE 1 there were two kinds of objects to be considered: the container and the embedded objects it contained. That those objects themselves might also contain embedded objects was of no consequence, since this fact was not relevant until they were opened into their own windows. The container and the open object were always the same, and it was not necessary to think of embedded objects as containers themselves. In OLE 2 this situation changes: the open object becomes the root of a hierarchy of embedded objects, *each* of which may be a container in its own right:

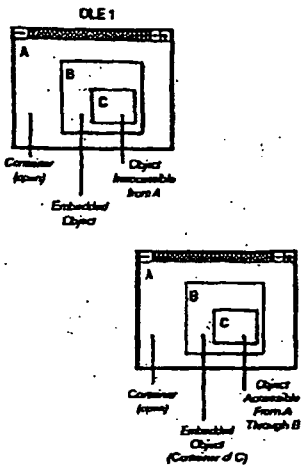


Figure 20. OLE 1 and OLE 2 object hierarchies.

As the user navigates through this hierarchy, OLE objects assume different states and appearances. An OLE object may be *inactive*, *selected*, *active*, or *open*.

2.5.1. Inactive

An object is said to be inactive when it is neither active nor part of a selection. It is displayed in its *presentation* form which is (usually) conveyed through its cached meta-file description. It may be desirable to know whether an object is embedded or linked at a glance without having to interact with it. Container applications should provide a "show objects" option that places a single pixel wide black *solid* border around the extent of an embedded object and a *dotted* border (Figure 22) around linked objects. If the container application cannot guarantee that a linked object is up to date with its source (because an automatic update was unsuccessful or the link is manual), the dotted border should appear in the Windows "disabled text" color (typically gray), suggesting the link is likely out of date. Only the container document's first level objects should be bordered. For example, if in Figure 7 "show objects" were chosen from the outer-level container (Microsoft Word), then only the worksheet object would be bordered, not the graph object nested inside of it. These borders should be clearly distinct from the visualizations for the other states, and are useful to distinguish embedded from linked objects.

An inactive object may be selected by single clicking anywhere within its extent, or it may be double-clicked which will perform its primary verb. The state diagram in Figure 27 shows that double-clicking will activate or open the object for editing, the usual primary verbs.

U.S. Compact Disc vs. LP Sales ($)

	1989	1990	1991
CD's	6,349K	18,652K	32,657K
LP's	31,539K	26,571K	17,429K
Total	37,883K	45,223K	50,086K

Figure 21. Unmodified presentation

2.5.2. Selected

The embedded object is selected when it is either single clicked or included in a multiple selection. It is selected (and deselected) and rendered according to the normal highlight rules of its container (see Figure 3), and responds to appropriate commands as any selected object of the container would. When the object is the singly selected, object-specific operations may be performed on it; the container retrieves these verbs from the system registry. When the object is selected the container may supply handles (for resizing, etc.) which affect the object as a unit with respect to the container. It is recommended that resizing an OLE object while it is selected results in a scaling operation, since there is no mechanism by which the container can communicate a cropping area that would be honored by the object when it is active.

MS-ET 0000095

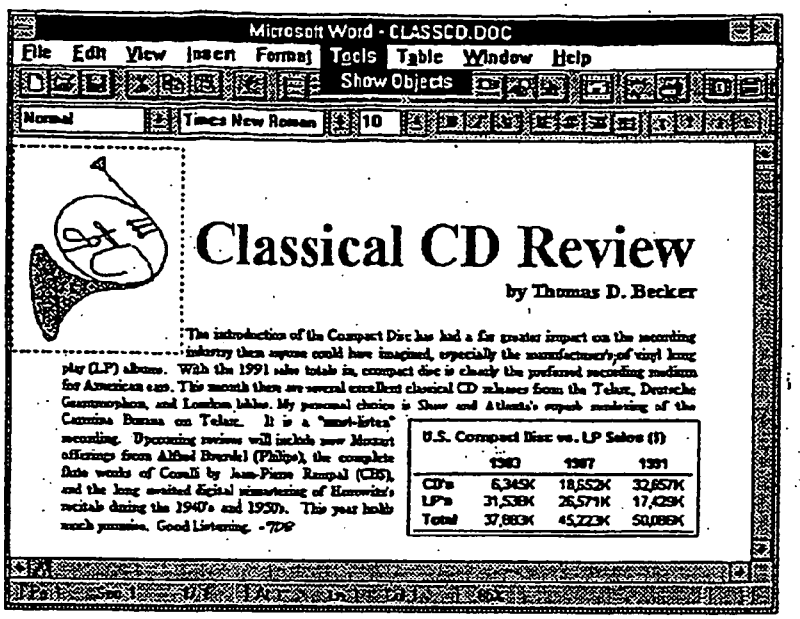


Figure 22. The Show Objects option.

When an object is singly selected, any of its registered verbs may be applied. "Edit" and "Open" will activate and open the object respectively, but other verbs (e.g. "Play") might perform and then leave the object selected. Any number of single clicks will simply reselect the object, while clicking outside will deselect the object. Verbs whose appropriateness depends on the state of the object should ideally enable and disable appropriately. For example, a media object which has Play and Rewind as verbs should disable Rewind when the object is already at the beginning. Similarly if an open object has two verbs Edit (for in-place editing) and Open (for opened editing), Edit should be disabled since the object cannot directly achieve the in-place active state without first closing.

2.5.3. Active

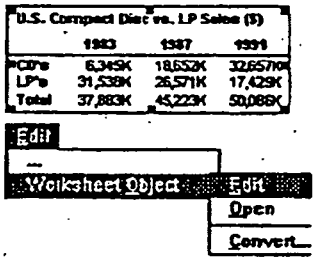


Figure 23. Selected OLE object.

OLE 2 objects may enter an *active* state in which the user may interact with the object's contents in-place, reusing its container document's window for its application's menus and interface controls. The user can make an object active either by performing its appropriate verb ("Edit"), double-clicking it (since for many objects the primary verb will be "Edit"), or selecting the object and pressing Enter. If Enter already has reserved meaning within the container, then Alt + Enter is recommended. When an object becomes active its application's menus and interface controls are grafted into the document's window and apply over the extent of the active object. Frame adornments appear outside the extent of the object, and thus may cover neighboring material in the document temporarily. Row/column headers (as pictured below), handles, or scrollbars are examples of frame adornments an object may wish to present. Scrollbars would allow the scrolling of a large spreadsheet within the object's viewport for example. The object and its frame adornments are surrounded by a black diagonal hatch border as an indication of the active state and to suggest the area of focus. The hatch is always black; it does not change color as focus changes between windows. There is only one object activated at a time; there is no attempt to activate all objects that use the same application as a set.[3] The hatch pattern is comprised of right-ascending diagonal lines as

[3] If for instance all Paintbrush Bitmaps were activated together, commands such as "Select All" or "Clear All" are ambiguous. The user would not know which object(s) would be affected by such document-scoped commands.

illustrated below. The object takes on the appearance which is best suited for its own editing: frame adornments may appear, table gridlines, handles, and other editing aids. The hatch border is considered to be part of the object's territory so it is the object's cursor that appears when the mouse hovers above the border. Clicking in the hatch pattern (and not on resize handles) should be re-interpreted by the object as clicking just inside the edge of the border. The hatch area is effectively a click slop zone that prevents inadvertent deactivations and makes it easier to select the contents of the object which lies right along its edge. The examples below show the border around an active worksheet (notice the border surrounds frame adornments).

Should the container may set at a view-scale (zoom ratio) which the object cannot match in order to perform in-place activation, the object should instead open into a separate window; if the object does not support an open mode, then it should not respond to the verb but issue an appropriate error message (in a dialog) indicating why.

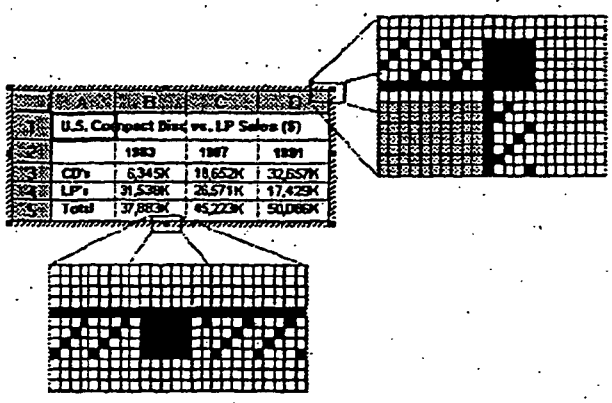


Figure 24. Hatch border around in-place activated objects.

Note that at any given instant there is at most one object that is active in-place per container. A single click in the container area, or a double click on a new OLE object (which may be nested in the currently active object) deactivates the current object and gives the focus to the new object.

An active object may be deactivated by clicking outside its extent in the container document or by pressing the Escape key. If an object uses the Escape key at all times, it is recommended that Shift+Escape is used to deactivate, after which it becomes the selected element of its container.

Edits made to an active object immediately and automatically become apart of the container document, just like edits to native data. Consequently, there is no "update changes?" prompt when an in-place active object deactivates. Of course, changes to the entire document, embedded or otherwise, can be abandoned by declining to save the file to disk. As we shall later see, in-place active objects participate in the Undo stack of the window in which they are activated.

Those objects which support resizing while in-place active should include square resize handles within the active hatch pattern. The solid black handles should be of the same width as the hatch pattern and have a single white pixel separation from the diagonal lines. It is recommended that in-place resizing exposes more or less of the object's content (adding or removing rows/columns in the case of this worksheet). In-place resizing should be seen as adjusting the viewport rather than scaling the object's appearance. Certain objects however may default to in-place scaling if cropping is not meaningful.

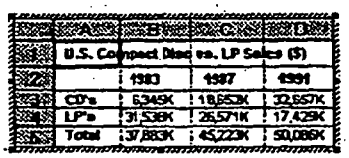


Figure 25. In-place resizing.

MS-ET 0000097

3.2.4. States of Embedded Objects

Figure 67 illustrates the main states of objects as they relate to the execution of servers when the objects are instantiated by containers, etc. In addition to the object states, there will be relevant states of server applications (e.g., dormant: invisible, not object, but server task is running so that getting into the Editing state is quick).

There is no requirement that all objects in a container be loaded at once; the container may choose to optimize the loading process. Similarly, containers may defer making the object running, in order to conserve memory, or may pre-activate server tasks for fast startup.

Passive: Object is on disk.

Loaded: Object structure is in client memory. The object may use the cached picture for rendering. The native object data will not normally be in memory.

Running: The server .EXE (or handler) for the object is running, with the object data available to it. The object can do full negotiation and rendering, etc. Clients who have links to this object may bind to it or receive notification of availability.

From the server's perspective, it is sometimes (though less often) useful to distinguish variations in the running state:

Ready: The server or handler has created UI resources, but does not have the focus. The server's tools and menu may be visible but inactive if the focus is in a different window, or invisible if the focus is elsewhere in the container's window hierarchy.

Active: The server or handler has the menus and is getting input.

Open: The server or handler has opened the object in a pop-up window, which may look like its normal frame window.

Executing: The server is asynchronously executing some command invoked by the client.

Note that there is no direct correspondence between the UI states described earlier and the states illustrated above. A UI Inactive object, for example, could be Passive, Loaded, Running, Ready, or Executing. The states where the two classifications coincide are indicated by identical names: objects that are *Active* or *Open* in the UI sense, are also Active or Open in the sense defined here.

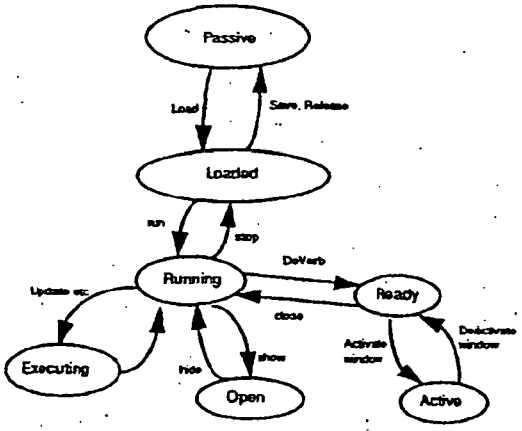


Figure 67. Sketch of object states transitions

MS-ET 0000135

Exhibit O

Medicine Meets Virtual Reality II: Interactive Technology & Healthcare

The Virtual Embryo: VR Applications in Human Developmental Anatomy

Authors:

Michael D. Doyle, Ph.D.
Director, Center for Knowledge Management
University of California, San Francisco
530 Parnassus Ave, Box 0840
San Francisco, CA 94143-0840
(415)476-9742
doyle@ckm.ucsf.edu

Adrianne Noe, Ph.D.
Curator, Human Developmental Anatomy Center
National Museum of Health and Medicine
Armed Forces Institute of Pathology
Walter Reed Army Medical Center, Bldg. 54
6825 Alaska Ave. N.W.
Washington, D.C. 20306-6000
(202)576-0401
NOE@padstars.afip.osd.mil

Ingrid Carlbom, Ph.D.
Director, Visualization Group
Cambridge Research Laboratory
Digital Equipment Corp.
1 Kendal Square, Bldg. 700
Cambridge, MA 02139
(617)621-6629
carlbom@crl.dec.com

Cheong Ang, M.S.
Programmer, Innovative Software Systems Division
Center for Knowledge Management
University of California, San Francisco
530 Parnassus Ave, Box 0840
San Francisco, CA 94143-0840
(415)476-8293
ang@ckm.ucsf.edu

David Martin, M.S.
Head, Innovative Software Systems Division
Center for Knowledge Management
University of California, San Francisco
530 Parnassus Ave, Box 0840
San Francisco, CA 94143-0840
(415)476-8293
martin@ckm.ucsf.edu

The Virtual Embryo: VR Applications in Human Developmental Anatomy

If virtual reality applications in biology and medicine are going to fufill their promise of recreating the subtleties and complexities of biological environments, they must draw from a rich assortment of heterogeneous and highly dense databases containing information about both the structures and functions of the systems to be simulated. This presents a problem for workers in virtual reality, for there are few readily available resources for accurate and detailed data on human biological structure and physiology. Several national-level research initiatives, such as the Visible Human Project and the Human Brain Project, are taking great strides towards creating information resources that can fill this gap. These projects are creating, for the first time, canonical datasets on adult human anatomic topography and topology which will serve as reference datasets for many areas of research into the applications of computer technology to biomedicine. The Visible Embryo project is an attempt to create such information resources for the fields of developmental and molecular biology. It also represents an integrated approach to developing applications of this technology in the areas of image databases, software tools, educational applications, and both basic and clinical sciences. A critical component of this research involves the development of virtual reality applications for research and education in the basic science of human developmental anatomy, as well as surgical simulation tools for practitioners of pediatric medicine.

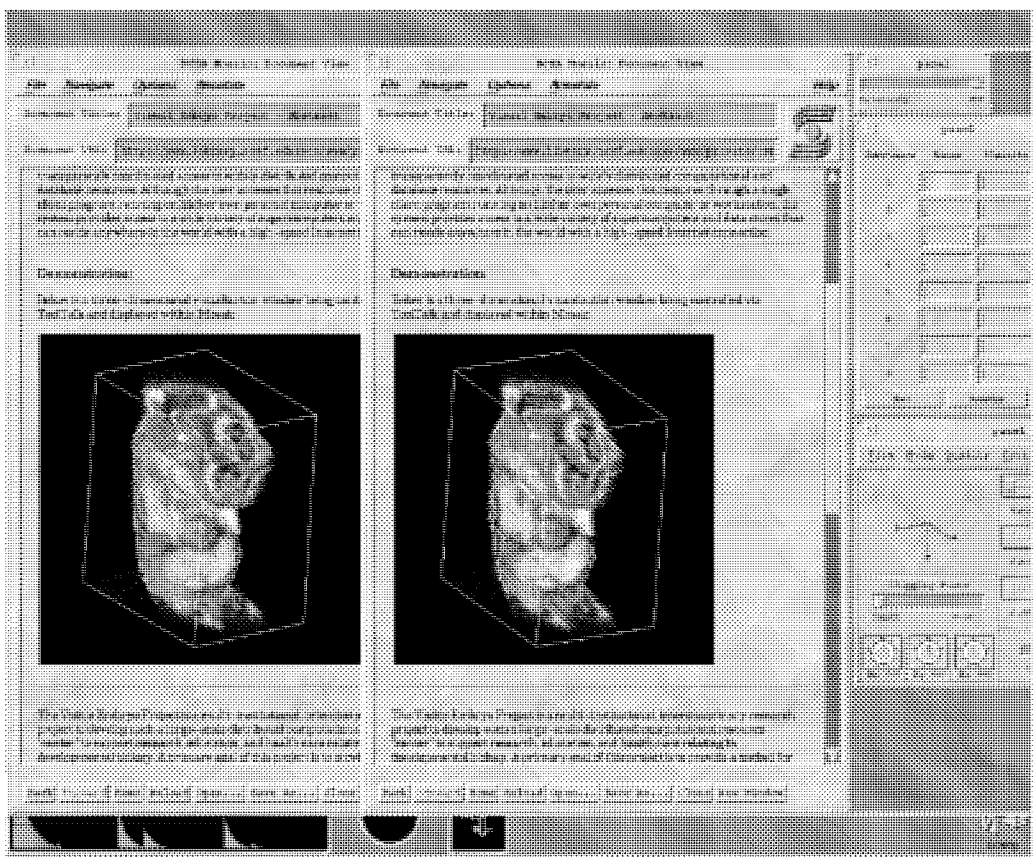
The term "metacenter" was coined several years ago by Larry Smarr, the Director of the National Center for Supercomputing Applications, to propose network-based computing and information "centers that actually represent transparently coordinated access to widely distributed computational and database resources. Although the user accesses this resource through a single client program running on his/her own personal computer or workstation, the system provides access to a wide variety of supercomputers and data stores that can reside anywhere in the world with a high-speed Internet connection.

The Visible Embryo Project is a multi-institutional, interdisciplinary research project to develop such a large-scale distributed computational resource "center" to support research, education, and health care relating to developmental biology. A primary goal of this project is to provide a testbed for the development of new technologies, and the refinement of existing ones, for the application of high-speed, high-performance computing and communications to current problems in biomedical science.

Sets of serial microscopic cross-sections through human embryos, within the collection of the National Museum of Health and Medicine, will be digitized and processed to create volumetric reconstructions of normal human embryonic anatomy. During the five years of this initial project, the entire contents of the Museum's Carnegie Collection of Human Embryology, over 600 embryos, will be digitized, reconstructed and archived, together with case histories, scientific articles, research notes, didactic descriptions, and other data contained within the collection. This massive database will be housed at the Museum at Washington D.C., while teams of researchers at more than 20 universities and companies around the United States will access widely distributed super computing resources to

develop visualization, analysis and telecollaboration software tools, educational materials, virtual reality simulations, basic science investigations, and clinical research projects based upon the data contained within the collection.

This project will serve the dual purpose of providing a testbed for new technology development in high performance computing and communications, as well as creating powerful new tools for the developmental biology research community. New advances in visualization technology are beginning to allow investigators to break through previous technical limitations and discover universally-applicable rules for pattern formation and shape development in organisms. By applying these new technologies to the existing archives of cross-sectional image information that exist in the literature and in collections around the world, we can tap into an enormous amount of new information that can be extracted from these databases.



This image illustrates UCSF's integration of World Wide Web documents with remote access to real-time interactive volume visualization tools. The two embryo images are slightly rotated to form a stereo-pair image which can be viewed in 3D by allowing your eyes to diverge slightly so that three images apear, and then focusing in on the middle image (it takes practice). This stereo pair was created by "cloning" the NCSA Mosaic window and then rotating the embryo 6º to the right. The interactive visualization controls are seen to the right of the Mosaic windows.

The task of integrating access to such massive information and computational resources is nontrivial. Just one embryo from the 650 serially-sectioned specimens in the collection can yield as much as a terabyte of anatomical volume data (a 20mm specimen, sectioned at 5 microns and digitized at a resolution of 8000x8000 pixels/section at 36bits RGB produces 1.073 TB of voxel data). It is clear that no single workstation, or even supercomputer, can manipulate, process and analyze such a large quantity of data as a single unit, much less perform computational operations on a database of hundreds of such datasets. For this reason, the Visible Embryo team at UCSF has developed tools to allow integrated Internet access (through NCSA Mosaic) to remote volume visualization engines which can distribute computation across a large number of graphics supercomputers connected by high-speed networking. This allows the integration, through NCSA Mosaic and the World Wide Web, of text-based, image, audio, and video data with real-time interactive control of high-performance visualizations embedded within Mosaic documents. We are also exploring the potential of using this technology for delivering interactive access to virtual reality applications through the Internet. The success of these efforts will allow widespread access to highly accurate and complex virtual reality simulations of human developmental anatomy, using inexpensive workstations and personal computers.

Reprint from SIGBIO Newsletter (The Journal of the ACM Special Interest Group for Biological Computing), 13/1 (1993) *page 1*

Processing cross-sectional image data for reconstruction of human developmental anatomy from museum specimens.

by Michael D. Doyle, Cheong Ang, Rakesh Raju, *Gary Klein, **Betsey S. Williams, Thomas DeFanti, Ardeshir Goshtasby, ***Robert Grzeszczuk and ****Adrianne Noe

The University of Illinois at Chicago, *Amoco Oil Co., **Harvard Medical School, ***the University of Chicago, and ****the National Museum of Health and Medicine, AFIP

Abstract

A project is described (The Visible Embryo Project) to develop software strategies for the creation of large-scale databases of 3-dimensional image data on human developmental anatomy. The issues discussed relate to the processing of serial-section image data for the purpose of reconstructing volumetric models of individual embryos from museum specimens. Attempts were made to fully automate the processes of image registration and artifact correction, in order to allow for the eventual unattended reconstruction of thousands of such embryos from such resources as the Carnegie Collection of Human Embryology. Tools were also developed to allow the real-time interactive visualization of the massive databases that this type of reconstruction project creates. The implementation of a 3-D model of a human embryo in a viewer-centric virtual reality environment is also described.

Introduction

Congenital malformations have challenged the humanistic and intellectual resources of mankind for ages. Generations of scientists have labored to understand their causes and effect their treatments. Despite this legacy, there are a variety of problems currently plaguing the discipline of human embryology. Although many remarkable collections of human embryological materials have been built up over the last century, the nature of these materials dictates that they be housed in museums under tight lock and key. The wealth of knowledge represented by such archives is largely accessible only through the publications of earlier researchers who studied various aspects of the collections. New methods for computer-based 3-dimensional morphometric analysis are enabling developmental biologists to study the dynamics of human development in ways that were not possible until only a few years ago. The publications of early researchers in the field are of little value to these new quantitative biologists. Yet these researchers have been severely limited by a lack of free access to large populations of embryological specimens. New technologies in high performance computing and communications are beginning to enable museums and research centers to open up their collections, in electronic form, to researchers, educators and students throughout the nation without endangering the fragile specimens themselves. The presently-described research is part of an effort to develop the technologies necessary for such a "museum without walls" to take form.

The Center for Human Developmental Anatomy

For over a century, scholars have perceived a need for an internationally accessible center for research in human developmental anatomy. Consistent with its missions to support consultation, education, and research in medicine and related disciplines, the staff at the United States Armed Forces Institute of Pathology began in 1990 to develop such a center at its National Museum of Health and Medicine. Its long-time experience in both conducting and supporting collections-based scientific research have allowed the staff and associates to identify prospectively the collections that will serve critical medical and scientific needs. Beginning with the acquisition of the world-renowned Carnegie Human Embryology Collection (O'Rahilly, 1987), the Institute continues to identify and collect well-documented materials and carefully houses them for perpetual availability to scholars. The Institute's mission is to preserve these and other materials relating to developmental anatomy in a careful, organized manner, which will eventually include extensive computer-based image archiving and the development of large-scale, remotely accessible, database on the Center's collections. The Center relies on supported research agendas to activate these materials and make them useful around the globe.

The Visible Embryo Project

A multi-institutional, multidisciplinary research project (the Visible Embryo Project) is being led by the Developmental Anatomy Center and the Biomedical Visualization Laboratory at the University of Illinois at Chicago to develop software strategies for the development of distributed biostructural databases using current technologies for high-performance computing and communications (HPCC), and to implement these tools in the creation of a large-scale digital archive of multidimensional data on normal and abnormal human development (Doyle, et al., 1992).. This project relates to BVL's long-term activity in the areas of health informatics, educational multimedia, and biomedical imaging science (Doyle, et al., 1990-92, Carlbom, et al., 1992).

Serial-section reconstruction has long been a tool used by morphologists, and especially embryologists, to study the three-dimensional structure of microscopic anatomy. The early workers typically hand traced the contours of microscopic sections projected through a camera lucida, cut out the contours from pieces of cardboard of a thickness corresponding to the scale of the reconstruction, and carefully glued them together, in register, to form a model of the 3-D surface of the structure of interest (O'Rahilly,1987).

Several workers have attempted to extend this approach into computer environments by tracing the contours of consecutive sections on a digitizing tablet, and then using computer graphics to render the 3D surface described by the stacked contours (Prothero,1986, Wilkinson,1990). Two embryos from the Blechschmidt collection have been reconstructed in this manner and visualized using a polygonal surface-based reconstruction approach (personal communication Cornelius Rosse). In the mouse, Lawson of the Hubrecht Laboratory, has attempted reconstruction from plastic histologic sections. These two approaches have in common the limitation that only the surface of the structure can be studied from a given model. If a reconstruction of the exterior of the heart were performed, for example, entirely separate reconstructions would have to be performed in order to visualize the chambers, or the myocardial vascular patterns.

Most of the early work in the Visible Embryo project has involved using serial section reconstruction from microscope slides, however we extracted volumetric data from these specimens, rather than just surface data. Sets of serial microscopic cross-sections through human embryos (prepared between the 1890s and 1970s) are being used as sample image data around which to design and implement various components of the system. These images are digitized and processed to create both 3D voxel datasets and polygonal models representing embryonic anatomy. Standard techniques for 3D volume visualization could then be applied to these data (Carlbom 1992, Doyle 1989-92, Karssemeijer 1988, Leichtman 1990, Levoy 1990, Lorenson 1987, McClean 1991, Prothero 1986, Terzopoulos 1987, Udupa 1990, Wilkinson 1990, Yuille 1986). Processing of these artifacts was required to correct for certain artifacts that were found in the original microscope sections from routine histological techniques of tissue preparation.

This processing includes, as will be discussed below, registration of adjacent serial sections, specialized methods of histogram equalization to correct for staining irregularities, and various levels of conformal warping to correct for dimensional instability (stretching, etc.) in the embedding material. Predominantly-automatic software tools have been developed by the UIC BVL team to accomplish these tasks, thereby enabling the future processing of huge numbers of images to create large-scale datasets describing human development through the first trimester. Remote-access visualization and database tools are under development to allow real-time interaction with these enormous datasets by distributing certain computational tasks to super computers.

Materials and Methods

A human embryo specimen was acquired, on loan, from the collection of the Human Developmental Anatomy Center. This specimens comprised a set of serial transverse sections through a 7-week-old embryo. The sectioned specimen was in the form of paraffin embedded 10μm-thick sections stained with haemotoxylin and eosin and mounted on glass slides with coverslips. Figure 1 shows a stereo pair image of a human embryo of similar age.

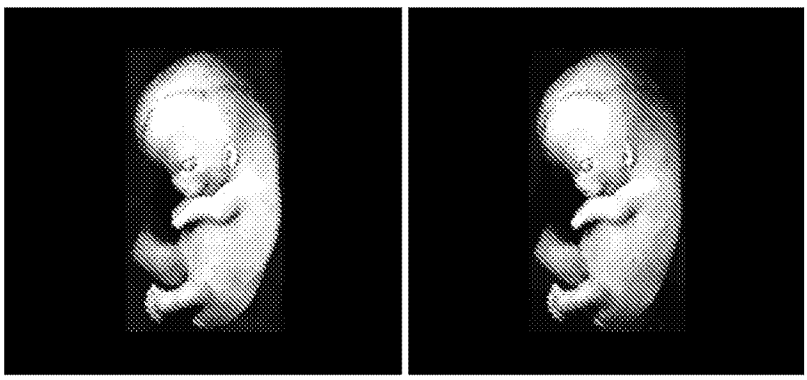


Figure 1: Stereo-pair image of a 7 to 8 week old human embryo from the Carnegie Collection of Human Embryology.

The microscopic sections through the embryo specimen were transilluminated with a voltage-stabilized dichroic light source. Video images were created with a Sony XC-77 CCD monochrome video camera through a Micro Nikkor 105mm macro lens with extension tubes, and then digitized with a Jovian Logic frame grabber at a spatial resolution of 640x480 pixels with 256 gray levels.

Images were stored in the TIF format with LZW (lossless) compression on 3.5" magneto-optical disks. A total of 1297 images were thereby obtained of the embryo, with each image corresponding to an individual physical section through the embryo. The magnification of the imaging system was such that the digitized images were made up of square pixels with a side length of 9μm.

Image Processing

The serial cross-section images were first processed in order to isolate the embryonic tissue from the surrounding area. This was accomplished through the use of the automatic edge-tracing feature of Image Pro Plus from Media Cybernetics Corp. Image Pro Plus also allowed the x,y coordinates of the external contour of each embryo section to be saved to an ASCII file. This proved to be very useful in the later stage of the project devoted to creating a surface-based polygonal model of the embryo exterior.

Registration of Images

Since the embryo specimen had been embedded and sectioned in the 1930's, fiducial markers were not included in the embedment to assist in the computer reconstruction of the embryo. It was therefore necessary to process the image data in order to restore topological relationships between tissue elements, and to restore dimensional accuracy in the direction perpendicular to the plane of section. We were therefore forced to "reverse engineer" proper image registration based solely upon intrinsic image features. This was accomplished through the creation of an algorithm by one of us (Klein) that used an error minimization technique he calls "iterative chasing." This technique attempts to find the "best" difference result between two adjacent images in a series, such that the following value is minimized:

$$H(x,y) = \sum_{xy}\sum |r(x,y) - R(x,y)|$$

where r(x,y) represents the first image (the reference image) and R(x,y) represents the second image (the mobile image) in a sequential pair. The program iteratively performs transformations in terms of x and y translations and rotations of the mobile image, while holding the position of the reference image stationary. The assumption is made that the closer one gets to the optimal transformation, the lower will be the value H. The 3-dimensional matrix of h (x translation), k (y translation), and θ (rotation), one dimension at a time. First, the best h transformation is found while k and θ are kept at zero. Then the best k transformation is sought while h = *h-best* and θ=0. A third set of transformations are then done to find

θ while h = *h-best* and k = *k-best*. This sequence is then repeated continuously until H(x,y) does not decrease any more. We then use *h-best*, *k-best* and θ-*best* as the coefficients of transformation for the mobile image. The program then saves the transformed mobile image to disk, replaces the image in the reference buffer with the mobile image, and then loads the next sequential serial-section image into the mobile image buffer. The registration procedure then begins again. This process is repeated automatically until all of the images of sequential cross-sections are processed and saved to disk.

Once proper registration of adjacent images was restored to the embryo data, these data were loaded into a 3 dimensional array representing voxels where each voxel had the dimensions 9μm x 9μm (the pixel size) x 10 μm (the standard section thickness). Standard techniques for volume visualization could then be used to examine the data.

Figure 3 shows a para-saggital section through the 3-D embryo dataset in the region of the thorax, at a plane perpendicular to the original plane of sectioning. Two important categories of image artifacts can be seen in this image, both a result of the original histological techniques used when the embryo was sectioned. First, although the heart (arrow) looks reasonably-well reconstructed, a closer examination shows a horizontal banding resulting from both staining irregularities and dimensional instability (shrinkage variations) among the set of serial cross sections. This can be seen throughout the sagittal image, but is especially evident in the area of the liver, which is immediately below the heart. There is a strong periodicity to this effect which makes the image appear as if it was made up of a set of "slabs" which were found to correspond to roughly 12 cross-sections each in the original data. The source of the periodicity of this effect may be clearer when one considers the routine techniques that were probably used by the 1930's era histology technician that prepared the original specimen.

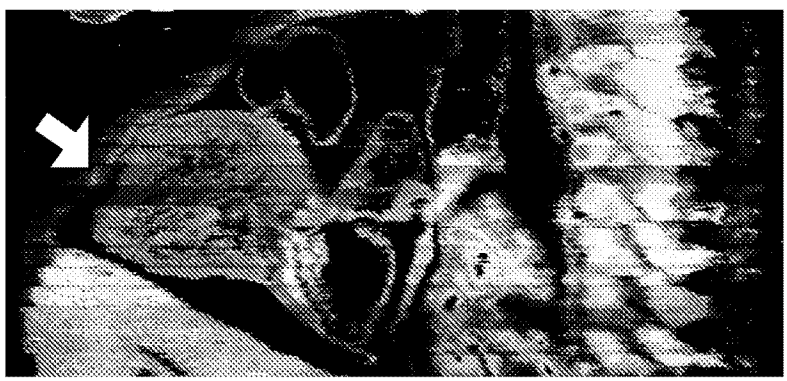


Figure 2: Parasagittal section through the thorax region of the auto-registered data. The arrow points toward the heart area. The liver can be seen below the heart.

It can be assumed that the embryo was first dehydrated with a graded alcohol series and then embedded in paraffin before sectioning began. The sectioning occurred on a guillotine-style microtome that made precise 10 μm-thick sections perpendicular to the long axis of the embryo. As the sections were cut, they formed ribbons that floated out onto a "boat" filled with water. They were then lifted out of the water directly onto a glass microscope slide and placed on a slide-warmer to dry. Once a number of slides were prepared in this way, they were probably then placed, end up, into a typical "staining rack" before being immersed in a series of solutions for staining of the important structures. They were then probably left to dry in the staining rack and then removed from the rack to be examined or stored.

When we examined the positions of the sections on the microscope slides, we found that they were placed on the glass in a rectangular grid with 4 to 5 rows of 12 section ribbons. If these slides had been dried in a vertical position after staining, then the sections at one end of a row would have dried faster than those at the other end of the row. The sections that dried faster may have shrunk at a different rate that those

that spent more time being wet. In addition some rows of sections may have been exposed to stain for a longer time than others, and this would result in those rows staining darker than the others. Therefore, the handling of the specimens during histological processing may account for the banding artifacts in the original dataset. This is supported by the fact that the 12-section thickness of the bands correlates with the 12 sections per row found on the glass slides.

Staining irregularities were corrected for by histogram equalization. Optimal histograms were calculated for every 26th image among the 1300 images from the embryo; 50 such histograms in all. Histogram equalization curves were saved to disk for each of these optimal histograms. Each of the equalization curves were then used for the correction of the following 26 images. Each of those images were processed so that their histogram matched the optimal histogram for that 26 image segment. This produced an image dataset with minimized banding and enhanced contrast, yet in a way that was sensitive to the topographical variations in image character throughout the embryo. Dimensional instability was corrected for through a the use of an algorithm (Goshtasby,1986) that conformally-warped the images based upon a pixel-to-pixel comparison between adjacent images, and that was "aware" of the periodicity of the shrinkage artifact measured in the original data.

It was found that additional artifacts were introduced into the data by the registration program itself. Due to the nature of the registration algorithm, it tended to "straighten out" any global curvatures or twists that existed in the original embryo. These "global artifacts" were partially corrected by interactively defining a "correction spline" in a direction parallel to the cranial-caudal axis of the body. The correction spline was then used as a guide for a smooth progression of x and y translations that restored a natural-appearing curve to the embryo.

3-D Visualization Tools

Software tools were developed to allow the interactive three-dimensional visualization of the embryo reconstruction in real time. Figure 3 shows the display of the application as it appeared at the

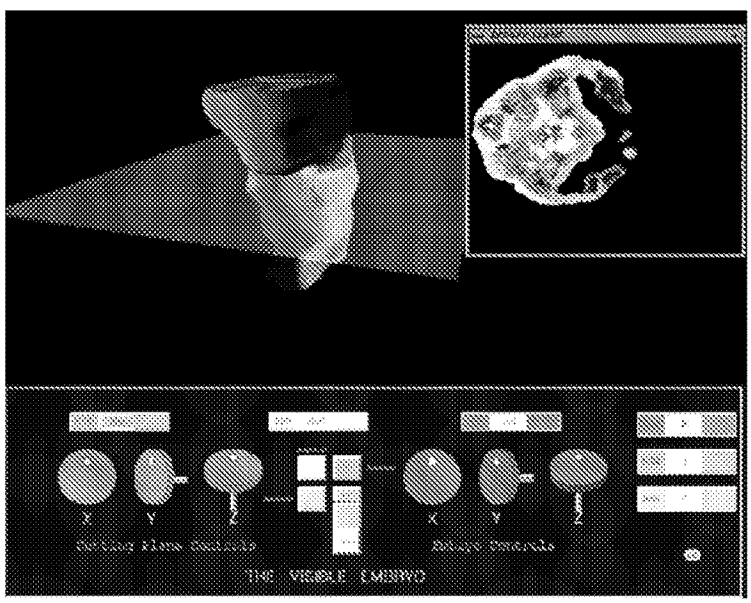


Figure 3: Screen image of the 3-D visualization program demonstrated at SIGGRAPH '92, in Chicago. This application was distributed between a Silicon Graphics Crimson VGXT workstation and a Convex supercomputer.

SIGGRAPH '92 conference in Chicago (Doyle, et al., 1992). The left of the screen shows a surface-based model of the embryo's exterior. This model was built from data which was derived, through three-

dimensional interpolation, from the original embryo dataset . Two-hundred volume slices of the embryo (stored as texture maps) can be interactively displayed at this lower resolution while the model is rotated freely in three dimensions. A cutting-plane can be seen to intersect the surface-based model. This cutting plane can be interactively controlled to intersect with the embryo model at any arbitrary angle and position. To the right of the screen, one can see a window that displays a high-resolution image of the oblique section through the embryo as indicated by the interactive cutting plane. In order to maintain the quick response needed for effective real-time interaction, the computational load of this application was distributed so that the interface panel, seen at the bottom of the screen, and the 3-D surface model were running on the CPU of the Silicon Graphics workstation. Computation of the high-resolution oblique section image displayed in the right window took place on the Convex supercomputer. Both of these operations occurred simultaneously, communicating through a high-speed fiber optic network.

Current efforts are being directed towards the development of a very portable tool for viewing arbitrary oblique slices through such data. This program allows interactive display of orthogonal and oblique slices through volumetric data without using any machine-specific functions. The application is written in pure ANSI-standard C and uses the X Window (Motif) toolkit for its interface. It has already been successfully tested on workstations from Silicon Graphics, Sun, and IBM.

Virtual Reality

The surface-based embryo-model described above was also implemented within a virtual reality environment, called "The Cave," at the 1992 SIGGRAPH conference. The Cave was a 10' x 10' room made up of back-projection screens upon which were projected stereo views of three dimensional data that a viewer, wearing LCD stereo glasses and a 3-D tracking device, could move around in as the system tracked his or her motion. One could walk around the data and receive the sensation that one was actually "in" the computer graphic environment. This system was also shown at the Supercomputing '92 conference and the 1992 meeting of the Radiological Society of North America.

Summary

The work described here represents the very earliest stages of the Visible Embryo project. Its long term goals are simply to create a comprehensive multimedia database that documents and describes both normal and abnormal human morphological development in multiple dimensions. This is in all respects a long term endeavor. Yet even the earliest stages of progress towards this goal will be of importance to a variety of investigators, educators and students of embryology and many ancillary fields. Among the obvious areas that will benefit from this work are medical education, cell and developmental biology, biomedical informatics, biomedical imaging science and high performance computing. Perhaps the most important benefit will be that it will form a model for similar projects in other fields ranging from the life sciences to the humanities.

Bibliography

1) Atlas, S.W., Braffman, B.H., LoBrutto, R., Elder, D.E. and Herlyn, D. Human malignant melanomas with varying degrees of melanin content in nude mice: MR imaging, histopathology, and electron paramagnetic resonance; J. Comp. Asst. Tomog. 14/4: 547-554 (1990)

2) Carlbom, I., Hsu, W.M., Klinker, G., Szelski, R., Waters, K., Doyle, M.D., Gettys, J., Harris, K.M., Levergood, T.M., Palmer, R., Palmer, L., Picart, M., Terzopoulos, D., Tonnessen, D., Vannier, M., and Wallace, G. Modeling and Analysis of Empiracle Data in Collaborative Environments, Communications of the ACM, 33/6: 75-84, (1992).

3) Cersen, S., Lotscher, H.R., Kummecke, B. and Seelig, J. Monoclonal antibody-coated magnetite particles as contrast agents in magnetic resonance imaging of tumors Magnetic Resonance in Medicine 12:151-163 (1989).

4) Doyle, M.D., Raju,R., Ang,C., Goshtasby,A., and DeFanti, T. The Virtual Embryo: real-time viewer-centric 4-D visualization of human embryonic anatomy, presented at the High Performance Computing and Communication Showcase, SIGGRAPH '92, the annual meeting of the Association for Computing Machinery's Special Interest Group for Graphics, Chicago. (August 1992).

5) Doyle, M.D., Raju, R., Ang, C., Klein,G., Goshtasby, A., DeFanti, T., and Noe, A.: The Visible Embryo distributed (workstation/supercomputer) interactive visualization of high-density 3D image data of embryonic anatomy, presented at the High Performance Computing and Communication Showcase, SIGGRAPH '92, the annual meeting of the Association for Computing Machinery's Special Interest Group for Graphics, Chicago. (August,1992).

6) Doyle, M.D. and Sadler, L.L. Health Informatics: MetaMap indexing and retrieval of image-based data for an institutional PACS system, presented at the 1991 Forum in Cellular and Organ Biology, American Association of Anatomists, Chicago, (April 1991).

7) Doyle, M.D. Palette Segmentation Indexing: The MetaMap Process, SIGBIO Newsletter (The Journal of the ACM Special Interest Group for Biological Computing), In Press (1992).

8) Doyle, M.D. The MetaMap Process: A New Approach to the Creation of Object-oriented Image Databases for Medical Education, Proceedings of the 13th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, IEEE Press (1991).

9) Doyle, M.D., Raju, R., Ang, C., Klein, G., Goshtasby, A.,, and DeFanti, T. Multidimensional reconstruction of human embryonic anatomy from museum-archived serial section collections, to be submitted to Medical Imaging (1993).

10) Doyle, M.D., Williams,B.S., Noe, A. The Visible Embryo Project: high performance computing, vizualization, and .communications for developmental anatomy research, to be submitted to Anatomical Record (1992)

11) Goshtasby, A.: A region based approach to digital image registration with subpixel accuracy, IEEE Transactions on Geoscience and Remote Sensing, GE-24, V03, May (1986)

12) Karssemeijer, N. Three-dimensional stochastic organ-models for segmentation in CT-scans SPIE 1030: 177-184 (1988).

13) Leichtman, G.S. and Anderson, D.J. Linking a relational database of biological features to computer-aided reconstruction of tissue. Proc. 1st Conf. Visualization in Biomed. Computing. IEEE Computer Society Press 288-294 (1990).

14) Levoy, M. A hybrid ray tracer for rendering polygon and volume data IEEE Comp. Graphics and Applic. 33-40 (1990).

15) Lorensen, W.E. Marching cubes: A high resolution 3-d surface construction algorithm. ACM SIGGRAPH 21/4: 38-44 (1987).

16) McLean, M. and Prothero, J.W. Three-dimensional reconstruction from serial sections. V. Calibration of dimensional changes incurred during tissue preparation and data processing. Anal. and Quant. Cytol. and Hist. 13: 269-278 (1991).

17) O"Rahilly, R. and Muller, F. Developmental stages in human embryos, Carnegie Institution of Washington, Publication no. 637. Carnegie Institute, Washington, DC. (1987).

18) Prothero, J.S. and Prothero, J.W. Three-dimensional reconstruction from serial sections. IV: The reassembly problem. Computers and Biomed. Res., 19: 361-373 (1986).

19) Terzopoulos, D. On matching deformable models to images. Topical Meeting on Machine Vision, Technical Digest Series (Optical Society of America, Washington, DC.) 12: 160-163 (1987).

20) Udupa, J.K. Course notes: Visualization of biomedical data: principles and algorithms. 1st Conf. on Visualization in Biomed. Computing. IEEE Computer Society Press, p14-16 (1990).

21) Wilkinson, D.G. and Green, J. In situ hybridization and the three-dimensional reconstruction of serial sections. IN: Copp, A.J. and Cockroft, D.L. eds. Postimplantation Mammalian Embryos: A Practical Approach. IRL Press, Oxford University. pp155-171(1990).

22) Yuille, A.L., Cohen, D.S., and Hallinan, P.W. Feature extraction from faces using deformable templates. Harvard Robotics Lab. Tech. Rep. 88/2: 104-109 (1988).

Integrated Control of Distributed Volume Visualization Through the World-Wide-Web

Cheong S. Ang, M.S.
David C. Martin, M.S.
Michael D. Doyle, Ph.D.

University of California, San Francisco
Library and Center for Knowledge Management
San Francisco, California 94143-0840

The World-Wide-Web (WWW) has created a new paradigm for online information retrieval by providing immediate and ubiquitous access to digital information of any type from data repositories located throughout the world. The web's development enables not only effective access for the generic user, but also more efficient and timely information exchange among scientists and researchers. We have extended the capabilities of the web to include access to three-dimensional volume data sets with integrated control of a distributed client-server volume visualization system. This paper provides a brief background on the World-Wide-Web, an overview of the extensions necessary to support these new data types and a description of an implementation of this approach in a WWW-compliant distributed visualization system.

1. Introduction

Advanced scanning devices, such as magnetic resonance imaging (MRI) and computer tomography (CT), have been widely used in the fields of medicine, quality assurance and meteorology [Pommert, Zandt, Hibbard]. The need to visualize resulting data has given rise to a wide variety of volume visualization techniques and computer graphics research groups have implemented a number of systems to provide volume visualization (e.g. AVS, ApE, Sunvision Voxel and 3D Viewnix)[Gerleg, Mercurio, VandeWettering]. Previously these systems have depended upon specialized graphics hardware for rendering and significant local secondary storage for the data. The expense of these requirements has limited the ability of researchers to exchange findings. To overcome the barrier of cost, and to provide additional means for researchers to exchange and examine three-dimensional volume data, we have implemented a distributed volume visualization tool for general purpose hardware, we have further integrated that visualization service with the distributed hypermedia [Flanders, Broering, Kiong, Robison, Story] system provided by the World-Wide-Web [Nickerson].

Our distributed volume visualization tool, VIS, utilizes a pool of general purpose workstations to generate three dimensional representations of volume data. The VIS tool provides integrated load-balancing across any number of heterogeneous UNIX™ workstations (e.g. SGI, Sun, DEC, etc...) [Giertsen] taking advantage of the unused cycles that are generally available in academic and research environments. In addition, VIS supports specialized graphics hardware (e.g. the RealityEngine from Silicon Graphics), when available, for real-time visualization.

Distributing information that includes volume data requires the integration of visualization with a document delivery mechanism. We have integrated VIS and volume data into the WWW, taking advantage of the client-server architecture of WWW and its ability to access hypertext documents stored anywhere on the Internet [Obtaszka, Nickersen]. We have enhanced the capabilities of the most popular WWW client, Mosaic [Andreessen] from the National Center for Supercomputer Applications (NCSA), to support volume data and have defined an inter-client protocol for communication between VIS and Mosaic for volume visualization. It should be noted that other types of interactive applications could be "embedded" within HTML documents as well. Our approach can be generalized to allow the implementation of object linking and embedding over the Internet, similar to the features the OLE 2.0 provides users of Microsoft Windows on an individual machine.

1.1 The World-Wide-Web

The World-Wide-Web is a combination of a transfer protocol for hyper-text documents (HTTP) and a hyper-text mark-up language (HTML) [Nickersen]. The basic functionality of HTTP allows a client application to request a wide variety of data objects from a server. Objects are identified by a universal resource locator (URL)[Obraczka] that contains information sufficient to both locate and query a remote server. HTML documents are defined by a document type definition

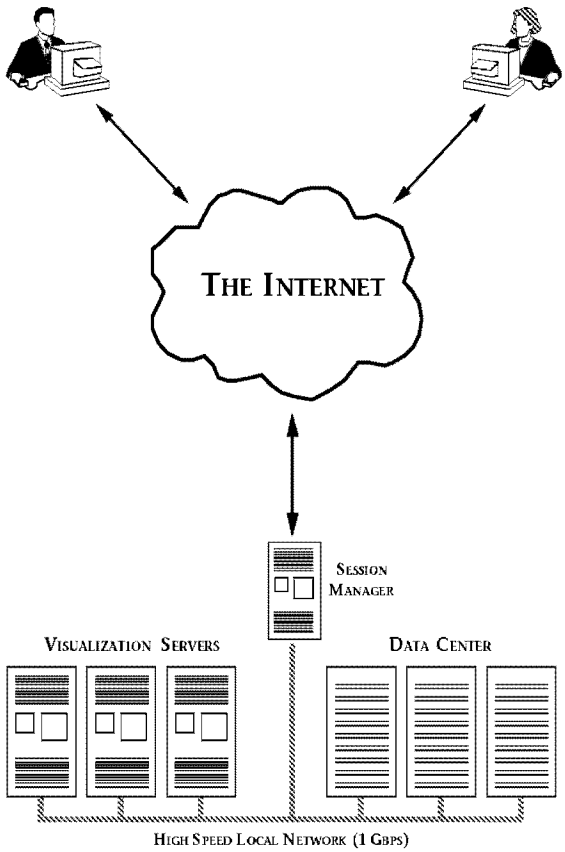


Figure 1: VIS client/server model.

(DTD) of the Standard Generalized Mark-up Language (SGML). These documents are returned to WWW clients and are presented to the user. Users are able to interact with the document presentation, following hyper-links that lead to other HTML documents or data objects. The client application may also directly support other Internet services, such as FTP, Gopher, and WAIS, [Andreessen] or may utilize gateways that convert HTTP protocol requests and return HTML documents. In all interactions, however, the user is presented with a common resulting data format (HTML) and all links are accessible via URL's.

1.2 Mosaic

The National Center for Supercomputer Applications (NCSA) has developed one of the most functional and popular World-Wide-Web clients: Mosaic. This client is available via public FTP for the most popular computer interfaces (Motif, Windows and Macintosh). Mosaic interprets a majority of the HTML DTD elements and presents the encoded information with page formatting, type-face specification, image display, fill-in forms, and graphical widgets. In addition, Mosaic provides inherent access to FTP, Gopher, WAIS and other network services [Andreessen].

1.3 VIS

VIS is a simple but complete volume visualizer. VIS provides arbitrary three-dimensional transformation (e.g. rotation and scaling), specification of six axial clipping planes (n.b. a cuboid), one arbitrary clipping plane, and control of opacity and intensity. VIS interactively transforms the cuboid, and texture-maps the volume data onto the transformed geometry. It supports distributed volume rendering [Argrio, Drebin, Kaufman] with run-time selection of computation servers, and isosurface generation (marching cubes) [Lorenson, Levoy] with software Gouraud shading for surface-based model extraction and rendering. It reads NCSA Hierarchical Data Format (HDF) volume data files, and has a graphical interface utility to import volume data stored in other formats.

2. VIS: A Distributed Volume Visualization Tool

VIS is a highly modular distributed visualization tool, following the principles of client/server architecture (figure 1), and consisting of three cooperating processes: VIS, Panel, and VRServer(s). The VIS module handles the tasks of transformation, texture-mapping, isosurface extraction, Gouraud shading, and manages load distribution in volume rendering. VIS produces images that are drawn either to its own top-level window (when running stand-alone) or to a shared window system buffer (when running as a cooperative process). The Panel module provides a graphical user-interface for all VIS functionality and communicates state changes to VIS. The VRServer processes execute on a heterogenous pool of general purpose workstations and perform volume rendering at the request of the VIS process . The three modules are integrated as shown in figure 3 when cooperating with another process. A simple output window is displayed when no cooperating process is specified.

2.1 Distributed Volume Rendering

Volume rendering algorithms require a significant amount of computational resources. However, these algorithms are excellent candidates for parallelization. VIS distributes the volume rendering among workstations with a "greedy" algorithm that allocates larger portions of the work to faster machines [Bloomer]. VIS segments the task of volume rendering based on scan-lines, with segments sized to balance computational effort versus network transmission time. Each of the

user-selected computation servers fetches a segment for rendering via remote procedure calls (RPC), returns results and fetch another segment. The servers effectively compete for segments, with faster servers processing more segments per unit time, ensuring relatively equal load balancing across the pool. Analysis of this distribution algorithm [Giertsen, 93] shows that the performance improvement is a function of both the number of segments and the number of computational servers, with the optimal number of sections increasing directly with the number of available servers. Test results indictate that performance improvement flattens out between 10 to 20 segments distrihuted across an available pool of four servers. Although this algorithm may not be perfect, it achieves acceptable results.

2.2 Cooperative Visualizaton

The VIS client, together with its volume rendering servers, may be lauched by another application collectively as a visualization server. The two requirements of cooperation are a shared window system buffer for the rendered image and support for a limited number of inter-process messages. VIS and the initiating application communicate via the ToolTalk service, passing messages specifying the data object to visualize as well as options for visualization, and maintaining state regarding image display. The VIS Panel application appears as a new top-level window and allows the user control of the visualization tool.

3. Visualization with Mosaic

We have enhanced the Mosaic WWW browser to support both a three-dimensional data object and communication with VIS as a coopezrating application (figure 2). HTTP servers respond to requests from clients, e.g. Mosaic, by transferring hypertext documents to the client. Those documents may contain text and images as intrinsic elements and may also contain external links to any arbitrary data object (e.g. audio, video, etc...). Mosaic may also communicate with other Internet servers, e.g FTP, either directly – translating request results into HTML on demand – or via a gateway that provides translation services. As a WWW client, Mosaic communicates with the server(s) of interest in response to user actions (e.g. selecting a hyperlink), initiating a connection and requesting the document specified by the URL. The server delivers the file specified in the URL, which may be a HTML document or a variety of multimedia data files (for example, images, audio files, and MPEG movies) and Mosaic uses the predefined SGML DTD for HTML to parse and present the information. Data types not directly supported by Mosaic are displayed via user-specifiable external applications and we have extended that paradigm to both include three-dimensional volume data, as well as to integrate the external applications more completely with Mosaic.

3.1 Mosaic 3D image support

We have extended the HTML DTD to support three-dimensional data via the introduction of a new SGML element: EMBED. This element provides information to the presentation system (i.e. Mosaic) about the content that is referenced in the document. The EMBED element is defined in the HTML DTD as shown in Example 1, which is translated as "SGML document instance element tag EMBED containing no content; four required attributes: TYPE, the type of the external application, in the MIME-type format; HREF, the location/URL of the datafile; WIDTH, the window width and, HEIGHT, the window height. The TYPE attribute give this specification the flexibility to accomodate different types of external applications. In a HTML document, a 3D image element would be represented as shown in Example 2, which may be interpreted as "create a drawing-area window of width 400 pixels, height 400 pixels, and use the application associated to hdf/volume MIME content-type to visualize the data Embryo.hdf located at the HTTP server site www.library.ucsf.edu".

3.2 Interface with Mosaic

The VIS/Mosaic software system consists of three elements: VIS, Mosaic, and Panel. Currently, the VIS application communicates with Mosaic via ToolTalk™, but the system will work with any interclient communication protocol. When Mosaic interprets the HTML tag EMBED, it creates a drawing area widget in the document page presentation and requests a shared buffer or pixmap from the windowing system to receive visualization results. In addition, Mosaic launches the Panel process, specifying the location of the data object to render and identifying the shared image buffer. The Panel process begins execution by first verifying its operating parameters, then launching the VIS process. The Panel process also presents the user with the control elements for data manipulation and manages the communication between the whole VIS application and Mosaic.

The VIS process, on the other hand, serves as a rendering engine. It executes the visualization commands from the Panel process, integrates the image data segments from various VRServers, and presents the complete array of image data to the Panel.

Thus the scenario following a user's action on the Panel will be (1) Panel issues visualization commands to the VIS rendering engine, (2) VIS sends rendering requests to

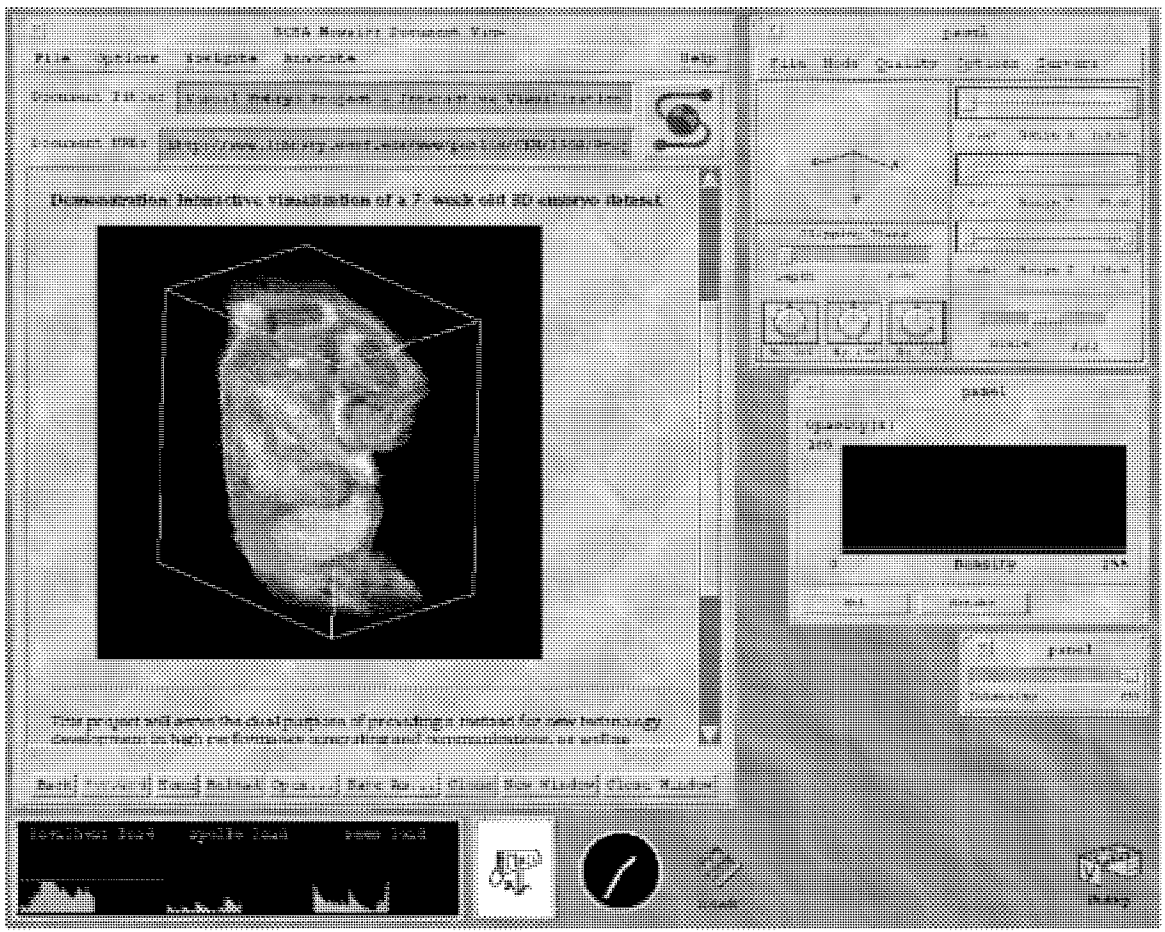


Figure 2: VIS embeded within Mosaic for interactive visualization in a HTML document.

VRServer(s), then gathers the resulting image segments, (3) Panel fetches the returned image data, then writes it to the pixmap, (4) Panel notifies Mosaic upon completion, and (5) Mosaic bit-blots the pixmap contents into its corresponding DrawingArea widget. The interprocess communication issue will be addressed in more details under section 3.3. The configuration of this software system is depicted in figure 3.

3.3 Interclient communication

We recognized the minimum set of communication protocols between Mosaic and a particular Panel process:

(a) Messages from Mosaic to a Panel process include the following:

(i) ExitNotify - requesting the Panel to terminate itself.when Mosaic exits.

(ii) MapNotify - requesting the Panel to map itself to the screen when the HTML document containing the DrawingArea corresponding to the above panel is visible.

(iii) UnmapNotify - requesting the Panel to unmap/iconify itself when the HTML page containing the DrawingArea corresponding to the above Panel is cached.

(b) Messages from a Panel process to Mosaic may be one of the following:

(i) RefreshNotify - informing Mosaic of an update in the shared pixmap, and requesting Mosaic to update the correspinding DrawingArea.

(ii) PanelStartNotify - informing Mosaic the Panel is started successfully, and ready to receive messages.

(iii) PanelExitNotify - informing Mosaic the Panel is exiting, and Mosaic should not send any more messages to the Panel.

We have packaged the above protocols and all the required messaging functions into a library. Modification of an existing external application merely involves registration of the external application's messaging window (the window to receive Mosaic's messages), installation of callback functions corresponding to

the messages from Mosaic, and addition of message-sending routine invocations. The protocol is summarized in Table 1.

Messages	Descriptions
ExitNotify	Mosaic exiting
MapNotify	DrawingArea visible
UnmapNotify	DrawingArea cache
RefreshNotify	DrawingArea update
PanelStartNotify	Panel starting
PanelExitNotify	Panel exiting

Table 1: Mosaic/VIS IPC communication.

4. Results

The results of the above implementation are very encouraging. The Mosaic/VIS sucessfully allows users to visualize HDF volume datasets from various HTTP server sites. Fig 2 shows a snapshot of the WWW visualizer. Distributing the volume rendering loads results in a remarkable speedup in image computations. Our performance ananlysis with a homogeneous pool of Sun SPARCstation 2's on a relatively calm network produced reasonable results (Figures 4a, 4b, and 4c. Three trials per plot). The time-versus-number-of-workstations curve decreases as more servers participate, and plateaus when the number of SPARCstations is 11 in the case of 256x256 image (9 for 192x192 image, and 7 for 128x128 image). The speed increases at the plateaus are very significant: about 10 times for the 256x256 image, 8 times for the 192x192 image, and 5 times for the 128x128 image. The outcomes suggest that performance improvement is a function of the number of volume rendering servers. Furthermore, the optimal number of workstations and the speed increase are larger when the image size is bigger. This is in complete agreement with Giertsen's analysis. We have also successfully tested the software system in an environment consisting of heterogenous workstations: a SGI Indigo2 R4400/150MHz, two SGI Indy R4000PC/100MHz, a DEC Alpha 3000/500 with a 133MHz Alpha processor, two Sun SparcStations 10, and two Sun SparcStations 2, which were located arbitrarily on an Ethernet network. To our knowledge this is the first demonstration of the embedding of interactive control of a client/server visualization application within a multimedia document in a distributed hypermedia environment, such as the World Wide Web.

5. Ongoing/Future work

We have begun working on several extensions and improvements on the above software system:

```
<!ELEMENT    EMBED   EMPTY>
<!ATTLIST    EMBED
             HREF    %URL     #REQUIRED
             TYPE    CDATA    #REQUIRED
             WIDTH   NUMBER   #REQUIRED
             HEIGHT  NUMBER   #REQUIRED>
```

Example 1: SGML definition for EMBED element.

```
<EMBED
   HREF="http://<host>/…/Embryo.hdf">
   TYPE="hdf/volume"
   WIDTH=400
   HEIGHT=400>
```

Example 2: EMBED element usage.

5.1 MPEG Data Compression

The data transferred between the visualization servers and the clients consists of the exact byte streams computed by the servers, packaged in the XDR machine independent format. One way to reduce network transferring time would be to compress the data before delivery. We propose to use the MPEG compression technique, which will not only perform redundancy reduction, but also a quality-adjustable entropy reduction. Furthermore, the MPEG algorithm performs

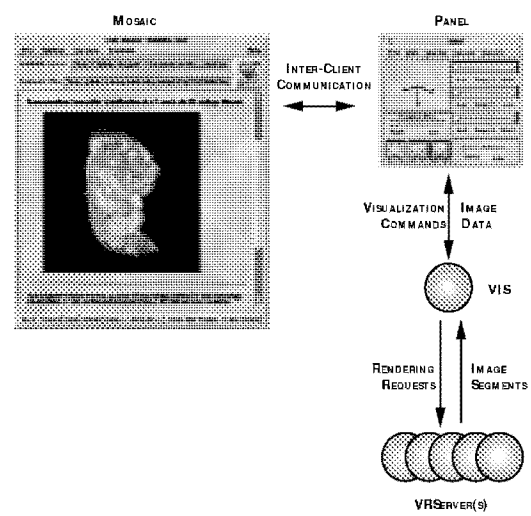


Figure 3: Communication among Mosaic, VIS and distributed rendering servers.

interframe, beside intraframe, compression. Consequently, only the compressed difference between the current and the last frames is shipped to the client.

5.2 Generalized External-Application-to-Mosaic-Document-Page Display Interface

The protocols specified in Table 1 are simple, and general enough to allow most image-producing programs be modified to display in the Mosaic document page. We have successfully incorporated an in-house CAD model rendering program into Mosaic. Our next undertakings will be to extend the protein database (PDB) displaying program, and the xv 2D image processing program, to create a Mosaic PDB visualization server, and a Mosaic 2D image processing server.

5.3 Multiple Users

With multiple users, the VIS/Mosaic distributed visualization system will need to manage the server resources, since multiple users utilizing the same computational servers will slow the servers down significantly. The proposed solution is depicted in Fig 5. The server resource manager will allocate servers per VIS client request only if those servers are not

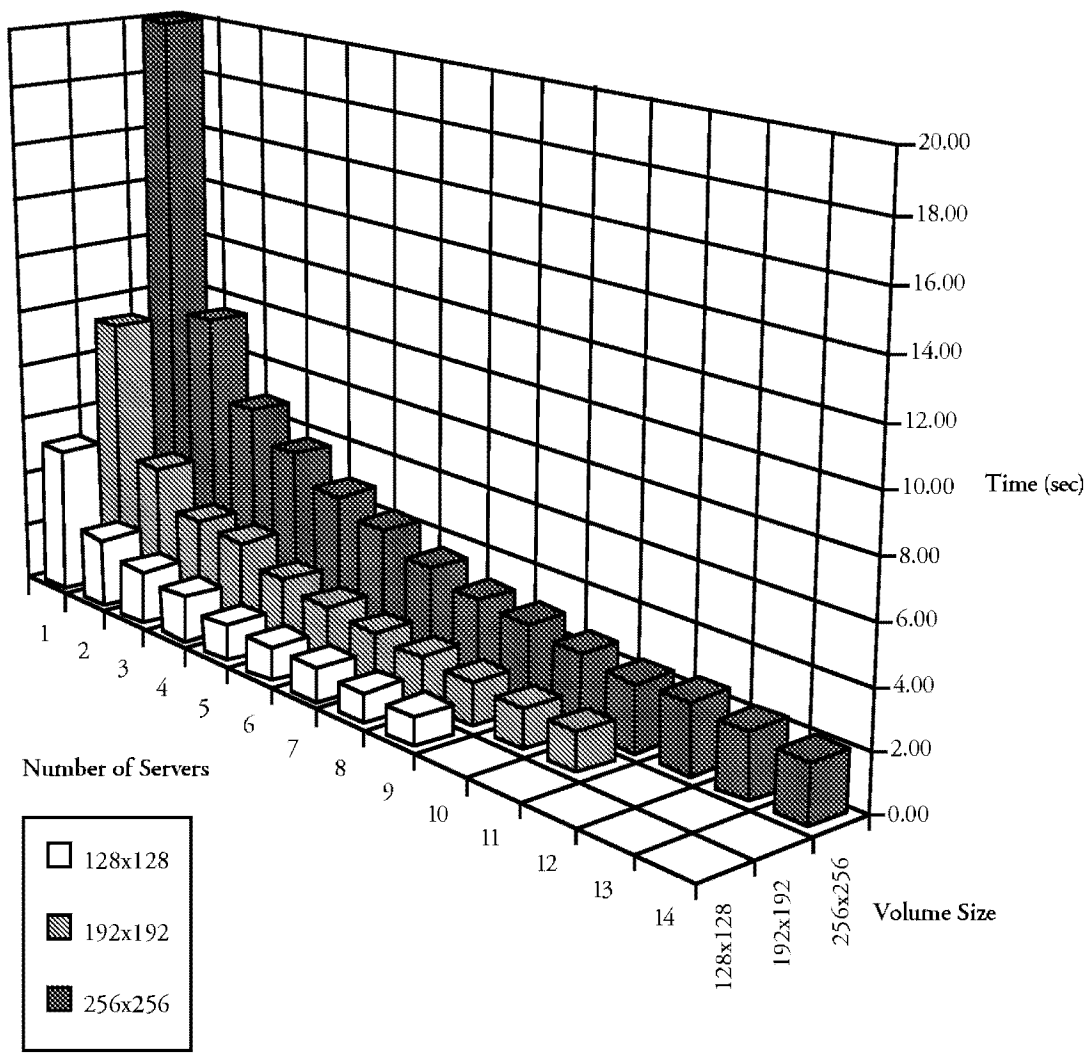


Figure 4: Volume rendering performance for 128^2, 192^2, and 256^2 data sets .

overloaded. Otherwise, negotiation between the resource manager and the VIS client will be necessary, and, perhaps the resource manager will allocate less busy alternatives to the client.

5.4 Load Distributing Algorithm

Since the load distributing algorithm in the current VIS implementation is not the most optimal load distribution solution, we expect to see some improvement in the future implementation, which will be using sender-initiated algorithms, described in [Shivaratri].

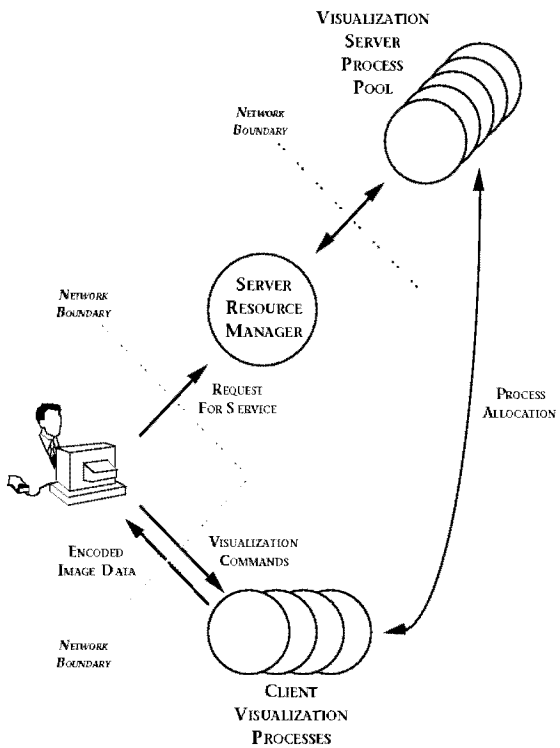


Figure 5: Server Resource Management

6. Conclusions

Our system takes the technology of networked multimedia system (especially the World Wide Web) a step further by proving the possibility of adding new interactive data types to both the WWW servers and clients. The addition of the 3D volume data object in the form of an HDF file to the WWW has been welcomed by many medical researchers, for it is now possible for them to view volume datasets without a high-cost workstation. Furthermore, these visualizations can be accessed via the WWW, through hypertext and hypergraphics links within an HTML page. Future implementations of this approach using other types of embedded applications will allow the creation of a new paradigm for the online distribution of multimedia information via the Internet.

7. References

Argiro, V. "Seeing in Volume", Pixel, July/August 1990, 35-39.

Avila, R., Sobierajski, L. and Kaufman A., "Towards a Comprehensive Volume Visualization System", Visualization '92 Proceedings, IEEE Computer Society Press, October 1992, 13-20.

Andreessen, M., "NCSA Mosaic Technical Summary", from FTP site ftp.ncsa.uiuc.edu, 8 May 1993.

Bloomer, J., "Power Programming with RPC", O'Reilly & Associate, September 1992, 401-451.

Brinkley, J.F., Eno, K., Sundsten, J.W., "Knowledge-based client-server approach to stuctural information retrieval: the Digital Anatomist Browser", Computer methods and Programs in Biomedicine, Vol. 40, No. 2, June 1993, 131-145.

Broering, N. C., "Georgetown University, The Virtual Medical Library," Computers in Libraries, Vol. 13, No. 2, February 1993, 13.

Drebin, R. A., Carpenter, L. and Hanrahan, P., "Volume Rendering", Computer Graphics, Vol. 22, No. 4, August 1988, 64-75.

Flanders, B., "Hypertext Multimedia Software: Bell Atlantic DocuSource", Computers in Libraries, Vol 13, No. 1, January 1993, 35-39.

Gelerg, L., "Volume Rendering in AVS5", AVS Network news, Vol. 1, Issue 4, 11-14.

Giertsen, C. and Petersen, J., "Parallel Volume Rendering on a Network of Workstations", IEEE Computer Graphics and Applications, November 1993, 16-23.

Jäger, M., Osterfeld, U., Ackermann, H, and Hornung, C., "Building a Multimedia ISDN PC", IEEE Computer Graphics and Applications, September 1993, 24-33.

Kaufman, A., Cohen, D., and Yagel, R., "Volume Graphics", Computer, July 1993, 51-64.

Kiong B., and Tan, T., "A hypertext-like approach to navigating through the GCG sequence analysis package", Computer Applications in the Biosciences, Vol. 9, No. 2, 1993, 211-214.

Levoy, M., "Display of Surfaces from Volume Data", IEEE Computer Graphics and Applications, Vol. 8, No. 5, May 1988, 29-37.

Lorensen, W., Cline, H.E., "Marching Cubes: A High Resolution 3D Surface Construction Algorithm", Computer Graphics, Vol. 21, No. 4, July 1987, 163-169.

Mercurio, F., "Khoros", Pixel, March/April 1992, 28-33.

Narayan, S., Sensharrma D., Santori, E.M., Lee, A.A., Sabherwal, A., Toga, A.W., "Animared visualization of a high resolution color three dimensional digital computer model of the whole human head", International Journal of Bio-Medical Computing, Vol 32, No. 1, January 1993, 7-17.

Nickerson, G., "WorldWideWeb Hypertext from CERN", Computers in Libraries, Vol. 12, No. 11, December 1992, 75-77.

Obraczka, K., Danzig, P, and Li, S., "Internet Resource Discovery Services", Computer, Vol. 26, No. 9, September 1993, 8-22.

Pommert, A., Riemer, M., Schiemann, T., Schubert. R., Tiede, U., Hoehne, K-H, "Methods and Applications of Medical 3D-Imaging", SIGGRAPH 93 course notes for volume visulization, 68-97.

Robison, D., "The Changing States of Current Cites: The Evolution of an Electronic Journal", Computers in Libraries, Vol. 13, No. 6, June 1993, 21-26.

Shivaratri, N.G., Krueger, P., and Singhal, M., "Load Distributing for Locally Distributed Systems", Computer, December 1992, 33-44.

Singh. J, Hennessy, J. and Gupta A., "Scaling Parallel Programs for Multiprocessors: Methodology and Examples", Computer, July 1993, 42-49.

Story, G., O'Gorman, L., Fox. D, Schaper, L. and Jagadish, H.V., "The RightPages Image-Based Electronic Library for Alerting and Browsing", Computer, September 1992, 17-26.

VandeWettering, M., "apE 2.0", Pixel, November/December 1990, 30-35.

Woodward, P., "Interactive Scientific Visualization of Fluid Flow", Computer, Vol. 26, No. 10, June 1993, 13-25.

Zandt, W.V., "A New 'Inlook' On Life", UNIX Review, Vol 7, No. 3, March 1989, 52-57.

Exhibit P

IN THE UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF ILLINOIS
EASTERN DIVISION

EOLAS TECHNOLOGIES, INC., a Delaware corporation, and THE REGENTS OF THE UNIVERSITY OF CALIFORNIA, a California educational corporation,	)	No. 99 C 626
Plaintiffs,	)	
vs.	)	Chicago, Illinois
MICROSOFT CORPORATION, a Washington corporation,	)	
Defendant.	)	August 7, 2003 7:00 a.m.

VOLUME 20
TRANSCRIPT OF PROCEEDINGS
BEFORE THE HONORABLE JAMES B. ZAGEL, and a Jury

APPEARANCES:

FOR THE PLAINTIFFS: ROBINS, KAPLAN, MILLER & CIRESI
2800 LaSalle Plaza, 800 LaSalle Avenue,
Minneapolis, MN 55402-2015
BY: MR. MARTIN R. LUECK
MR. RICHARD M. MARTINEZ
MS. JAN M. CONLIN
MR. MUNIR MEGHJEE
MS. KEIKO SUGISAKA
MS. EMILY ROME

FOR THE DEFENDANT: LEYDIG, VOIT & MAYER, LTD.
Two Prudential Plaza, Suite 4900,
Chicago, IL 60601
BY: MR. H. MICHAEL HARTMANN
and
SIDLEY, AUSTIN, BROWN & WOOD
Bank One Plaza, 10 South Dearborn Street,
Chicago, IL 60603
BY: MR. DAVID T. PRITIKIN
MR. WILLIAM H. BAUMGARTNER
MR. RUSSELL CASS

Jury Instructions

One: Executable application. A key part of the invention is the ability of the browser to automatically invoke some external program vis-a-vis the hypermedia document to process the data object. Generic examples of such programs are image viewers, word processors and spread sheets, programs that display and allow the user to interact with data. Such programs are referred to in the claims as an executable application.

I have defined executable application as any computer program code that is not the operating system or a utility that is launched to enable an end user to directly interact with data.

The word external means external to a web page. Executable application may be a stand-alone program, a component such as a dynamic link library, or multiple components working together. A component may be used to perform more than one function and may be used by more than one program.

Type information. The invention parses a hypermedia document, also known as a web page, and learns the location of at least a portion of some object. This object has type information associated with it, and this type information is utilized by the browser to identify and locate the executable application. In other words, the browser connects the type information to the executable application.

Type information may include the name of an application associated with the object, which may convey information to the browser for it to use in identifying and locating the executable application.

Third. The phrase utilized by said browser to identify and locate. Utilized by said browser to identify and locate means that the enumerated functions are performed by the browser. In other words, the browser connects the type information to identify and locate the executable application. Executing the application once it has been identified and located is not part of this linking.

The inventors contemplated the browser's use of some