```
09:35:21            OCA PAD AMENDMENT - PROJECT HEADER INFORMATION            05/19/92
                                                                                Active
Project #: C-43-613            Cost share #:                          Rev #: 5
Center # : 10/24-6-R6887-0A1   Center shr #:                          OCA file #:
                                                                      Work type : RES
Contract#: STD. AGREEMENT                     Mod #: LTR DTD 5/13/92  Document  : AGR
Prime    #:                                                           Contract entity: GTRC

Subprojects ? : N                                                     CFDA: N/A
Main project #:                                                       PE #: N/A


Project unit:                SRC           Unit code: 02.010.307
Project director(s):
    PUTNAM W O III           COMPUTING     (404)894-5551




Sponsor/division names: BELLSOUTH ENTERPRISES            /
Sponsor/division codes: 251                              / 004


Award period:     900102    to    920630  (performance)    920630  (reports)

Sponsor amount            New this change               Total to date
      Contract value                0.00                    74,160.00
      Funded                        0.00                    74,160.00
Cost sharing amount                                              0.00

Does subcontracting plan apply ?: N

Title: MULTI-MEDIA ELECTRONIC MAIL



                                    PROJECT ADMINISTRATION DATA

OCA contact: Don S. Hasty                 894-4820

 Sponsor technical contact                 Sponsor issuing office

 NEIL WALKER                               NEIL WALKER
 (404)249-4518                             (404)249-4518

 BELLSOUTH ENTERPRISES, INC.               BELLSOUTH ENTERPRISES, INC.
 1100 PEACHTREE ST., NE, ROOM 11C02        1100 PEACHTREE ST., NE, ROOM 11C02
 ATLANTA, GA 30309-4599                    ATLANTA, GA 30309-4599


Security class (U,C,S,TS) : U              ONR resident rep. is ACO (Y/N): N
Defense priority rating   :  N/A           N/A supplemental sheet
Equipment title vests with:    Sponsor X        GIT

Administrative comments -
  SPONSOR LTR DTD 5/13/92 AUTHORIZES A 3-MONTHS NO-COST EXTENSION, AS REQUESTED
  IN OUR LTR DTD 4/30/92.
```

GEORGIA INSTITUTE OF TECHNOLOGY
OFFICE OF CONTRACT ADMINISTRATION

NOTICE OF PROJECT CLOSEOUT

SL-580-1632-1

Closeout Notice Date 07/07/92

Project No. C-43-613 Center No. 10/24-6-R6887-0A1

Project Director PUTNAM W O III School/Lab SRC

Sponsor BELLSOUTH ENTERPRISES/

Contract/Grant No. STD. AGREEMENT Contract Entity GTRC

Prime Contract No.

Title MULTI-MEDIA ELECTRONIC MAIL

Effective Completion Date 920630 (Performance) 920630 (Reports)

Closeout Actions Required:	Y/N	Date Submitted
Final Invoice or Copy of Final Invoice	Y	
Final Report of Inventions and/or Subcontracts	N	
Government Property Inventory & Related Certificate	Y	
Classified Material Certificate	N	
Release and Assignment	N	
Other	N	

Comments

Subproject Under Main Project No.

Continues Project No.

Distribution Required:

Project Director	Y
Administrative Network Representative	Y
GTRI Accounting/Grants and Contracts	Y
Procurement/Supply Services	Y
Research Property Managment	Y
Research Security Services	N
Reports Coordinator (OCA)	Y
GTRC	Y
Project File	Y
Other	N
	N

Mutimedia Electronic Mail

Final Report

William O. Putnam
Keith Edwards
Tom Rodriguez

Georgia Institute of Technology
College of Computing
Software Research Center
Atlanta GA 30332-0280

Georgia Institute of Technology Project C–43–613

Abstract

Electronic mail systems capable of transmitting compositions consisting of various unconventional media (such as voice, video, and images) have attracted a substantial amount of interest in recent years. The Montage multimedia electronic mail system, along with its model for multimedia documents, was developed to explore the issues involved in multimedia electronic communication The system is built on top of the MIT X Window System and standard mail transport protocols for flexibility and portability. It supports electronic mail message composed of an arbitrary mixture of text, still graphics, digital sound, voicemail, and motion video, and is extensible at runtime to allow user to add support for new media types. This document presents a summary of the development of Montage and a description of the prototype system.

This work was sponsored by BellSouth Enterprises, Inc.
and by a grant from the Advanced Technology Development Center of the State of Georgia.

Mutimedia Electronic Mail

Final Report

William O. Putnam
Keith Edwards
Tom Rodriguez

Georgia Institute of Technology
College of Computing
Software Research Center
Atlanta GA 30332–0280

This work was sponsored by BellSouth Enterprises, Inc.
and by a grant from the Advanced Technology Development Center of the State of Georgia.

Abstract

Electronic mail systems capable of transmitting compositions consisting of various unconventional media (such as voice, video, and images) have attracted a substantial amount of interest in recent years. The Montage multimedia electronic mail system, along with its model for multimedia documents, was developed to explore the issues involved in multimedia electronic communication The system is built on top of the MIT X Window System and standard mail transport protocols for flexibility and portability. It supports electronic mail message composed of an arbitrary mixture of text, still graphics, digital sound, voicemail, and motion video, and is extensible at runtime to allow user to add support for new media types. This document presents a summary of the development of Montage and a description of the prototype system.

Introduction

The Software Research Center (SRC) at the Georgia Institute of Technology is conducting ongoing research into the integration of multi–media applications and tools into workstation environments. This project involved the design and construction of a multi–media electronic mail handling system. Whereas current electronic mail systems can generally only handle simple text messages, we have built a system called *Montage* which is capable of sending and receiving "documents" of complex media types, such as audio and video.

Our broad goal has been to produce a prototype Multi–Media Workstation. This workstation will be representative of the small systems of the nineties: harnessing large amounts of processing power, able to deal with media other than traditional text, and possessing vast storage capabilities. We believe that the ability to process what are by today's standards unusual media types will be integral to future machines.

This Multi–Media Workstation must be able to integrate and synthesize multi–media inputs from many sources in as smooth and seamless a fashion as conventional machines manipulate text today. Toward this end, we have made use of accepted standards wherever possible. All of our applications and tools function under the X Window System. X has widespread acceptance and is able to function with the high level of network connectivity that is required by demanding applications. X also provides the advantages of portability (to different hardware platforms and operating systems) and scalability (to new and more powerful technologies as they become available). The use of X insures that our research investment will not be wasted as new machines appear on the market.

Electronic mail systems have redefined the ways in which many groups communicate. From small work groups to large corporations, electronic mail is fast becoming an indispensable method of interpersonal communication. Early electronic mail systems were limited to the transmission of simple textual data only. More recently, electronic mail systems capable of transmitting compositions consisting of various "unconventional" media (such as voice, video, and images) have attracted a substantial amount of interest.

Sending text via electronic mail systems is one of the most important uses of traditional networked workstations. Systems of the nineties, however, should be more flexible in the types of data they can transmit. Thus, our first application for the Multimedia Workstation was a complex electronic mail system, capable of transmitting, receiving, and viewing multimedia documents.

Project Summary

During the term of the project we have developed a multimedia electronic mail system which allows the creation and distribution of documents containing text, graphics, and digitally recorded sound. The system is called *Montage*, and it runs on Unix workstations under the industry standard X Window System.

The development of Montage took place in two stages. From January, 1990 through June 1991 we designed the message architecture and built the initial prototype using X and the HP widget set. A paper on the Montage message architecture was presented at the IEEE Tricomm 1991 conference, and is included here as Appendix 2.

In the Winter of 1991 we applied for a Research Commercialization Grant from the Advanced Technology Development Center to support further work on Montage. We received a $25,000 grant from ATDC and matching funds from BellSouth Enterprises in July 1991.

From July 1991 through June 1992 we reworked the prototype system, porting it to the Motif graphical user interface and widget set and adding many usability features, including support for many more media types, customization and configuration panels, mail folders, and functions such as Reply and Forward which had been omitted in the original prototype.

The advanced prototype is described by a paper and a draft user guide which are included here as Appendices 3 and 4.

In April 1992 the advanced prototype was completed. It was demonstrated at the ICA Conference as part of the BellSouth SMDS Showcase.

Montage Description

Montage is a prototype multimedia electronic mail system for Unix workstations. It is similar to the NeXT mail system but is built on top of the X window system, so it is more portable. Montage supports text, graphics in a variety of formats, digital audio and voice, and data from commercial applications such as spreadsheets and word processors. Users can configure Montage to support whatever data types and applications they have available.

Montage messages have a primary component, which can be plain text or a mixture of text and graphics, and any number of attachments or annotations to the primary component. The attachments, which may be of any media type, are represented by icons located along the right margin of the message body and are tied to a particular location in the message. While the prototype system does not yet implement use of dynamic media such as motion video or audio as the primary message body, they can be used as attachments, and the Montage architecture does support their use in the message body in the future.

Montage uses standard Unix mail transport mechanisms (sendmail and SMTP) and can ship messages transparently across heterogeneous networks. The Montage software is required only at the sending and receiving ends, not by intervening gateway systems.

The prototype system runs on Sun SPARCstations and makes use of the embedded A/D converter for voice input and audio playback. That is the only part of the system that is Sun-specific.

Most current multimedia electronic mail systems impose several substantial limitations on their users. Perhaps chief among the problems of current systems is their lack of extensibility. Usually these systems have compiled into them a predefined set of media types which the mailer can compose, transmit, and view. The mailer itself has, in essence, its own built-in spreadsheet, audio recorder, text processor, and drawing package (plus whatever other media types the mail system may support). These systems suffer from two major problems: users cannot use the tools which are familiar to them to compose and view messages, and the mail system cannot easily support new data formats and applications.

Montage takes a different approach to electronic mail. In Montage, there is no built-in support for various media. Instead, all media handling is externalized from the mail system. Montage messages consist of multiple components, each of which has a "tag" associated with it. A tag is simply an ASCII string which users have agreed to associate with a certain type of medium. Montage itself associates no meanings with tags; instead tags are used to index into a user-defined database which maps tags to external handlers. These handlers are complete, stand-alone programs which "know" how to edit and view the given medium.

In this way, Montage overcomes many of the problems associated with other multimedia electronic mail systems: users may use the tools which they are most familiar with to compose and view messages, and the system may be easily extended by users at runtime to support new media types. No recompilation is necessary.

In addition to its powerful extension features, Montage also makes use of a two-level message presentation format. Montage messages consist of a main body (which may itself consist of components of various media types) and various annotations to the main body (which themselves may be of any media type). Annotations are analogous to margin notes on a written document. They allow users to exchange compound documents with various attachments which can provide extra information about some main body component. Annotations are represented as icons in the margin of the main message body. The data in the annotation is not presented until it is requested by the reader of the message.

In summary, Montage provides a powerful, extensible system for exchanging compound documents. These documents themselves are composed into a two-level presentation format. Montage runs under the Motif look-and-feel on any X Window-capable workstation. Currently supported media include standard text, several image formats (including PostScript), Group-3 FAX, and audio.

Montage Features

Montage offers the following features:

Plain or formatted text messages.

Still images in several formats, including scanned images, computer generated graphics, line art, clip art, bitmaps, postscript, and still frames from video.

Digital video attachments from pre-recorded clips at speeds up to 30 frames per second (monochrome).

Sound attachments or annotations embedded in messages (recorded in real-time or from a stored digital recording) along with text and graphics.

In addition to all types of data files, executable programs can be mailed as attachments.

Easy extensibility to support full-motion video, high quality audio, commercial document preparation system formats, and other media types not currently supported.

Configuration panels for customization and media configuration.

Motif graphical user interface on top of the industry standard X Window System.

Modular design and adherence to open systems technologies and standards for maximum portability.

Use of X resources to link with commercial word processing, graphics, database, spreadsheet, and desktop publishing packages.

Does not require any extra hardware (audio or video boards) beyond the basic workstation.

Uses standard Unix mail protocol (SMTP): compatible with all existing Unix Email transmission systems.

The current version of Montage runs on Sun SPARCstation workstations under X Version 11 Release 4 or Release 5.

Montage Documentation

Montage is described and documented in two research reports and a draft user manual. These reports are provided as Appendices 2, 3, and 4. Copies of the reports are included on the source code tape in the Papers directory in FrameMaker, PostScript, and ASCII formats.

An example Montage message is included in Appendix 1. It is a one page description of Montage demonstrating usage of text and graphics in several formats. The message was composed using FrameMaker and several image editing tools. It is included in the Montage demo message set on the distribution tape, along with several other sample messages and a variety of sample media clips and files.

To view the demo messages, start Montage as directed by the README file in the "ica" directory and look in the "new demos" folder. A copy of the Montage poster message is in the file "ica/MMMail/poster".

Montage Distribution Tape

The Montage source code, documentation, executable programs, and data files are contained on a 1/4 inch cartridge tape included with this report. The tape was made on a Sun SPARCstation with the command:

```
cd montage; tar cvf /dev/rst0 .
```

which created a tape archive of the montage project directory and its contents. The tape can be loaded with the command:

```
mkdir montage; cd montage; tar xvf /dev/rst0
```

which will create a directory called "montage" and copy into it the archive from the tape.

After loading the tape, the montage directory will contain the following directories and files:

./README	A file describing the Montage distribution directory.
./bak	Snapshots of various older versions of Montage.
./bin	Latest binary versions of Montage and associated programs.
./ica	Montage distribution used at the ICA conference demo.
./ideas	Directory for TODO lists and other ideas.
./images	Some images which may be used by Montage messages (bball.ras is the image in the startup window).
./papers	Some papers and documentation for the system.
./src	Montage source code.

Other README files describe the contents of the "src" (source) and "papers" (documentation) directories.

The directory "ica" contains a demo installation of Montage used at the BellSouth SMDS Showcase at the May 1992 ICA Conference. All programs and data files needed to run Montage are present in that directory. See the README file for installation and startup instructions.

All programs have been compiled to execute on a Sun SPARCstation under SunOS 4.1 and X11R4 using the OSF Motif 1.1 distribution. You will need the X11R4 or X11R5 distribution and the OSF Motif distribution to compile Montage.

Licensing

In return for sponsorship of the Montage development, BellSouth Enterprises has a non–exclusive, non–transferrable, royalty–free right–to–use license for the Montage system. Further, BellSouth Enterprises will receive 50% of any non–educational royalties generated by licensing of Montage by the Georgia Tech Research Corporation (GTRC).

GTRC is currently in negotiations with SecureWare, Inc., who wish to license Montage for further development, productization, and distribution.

Areas for Further Work

At this stage Montage is considered to be an advanced prototype. Before it can be used as a commercial quality email system the following should be considered:

Further testing and debugging will be needed to make it more robust and less prone to crash.

Some buttons and menu functions are un–implemented and should be completed or removed.

Support for multiple browser windows should be added.

Several Motif bugs in release 1.1 should be fixed by rebuilding with the latest Motif release.

Motion Video Support could be extended to support live recording and to allow video in the primary message component.

Fax support could easily be added using the media configuration panel if a commercial fax-modem package were obtained.

Appendix 1

Example Message: Montage Multimedia Electronic Mail Poster

This poster is an example of a Montage multimedia message, combining text and graphics in several formats.

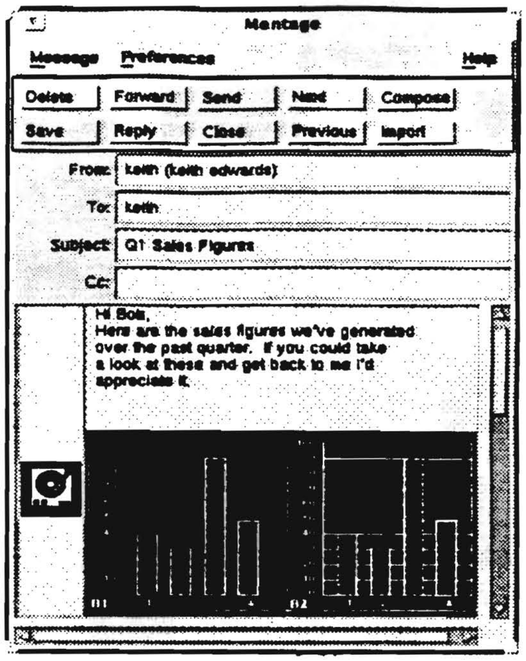
Montage
Multimedia Electronic Mail

Georgia Tech Multimedia Computing Group

Montage is an extensible multimedia electronic mail system which supports the composition, transmission, and viewing of documents consisting of arbitrary media, including even executable programs and commercial file data.

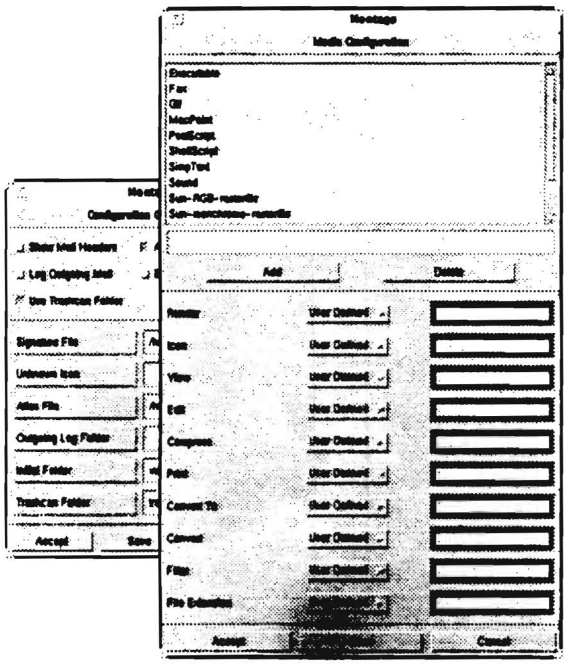
Montage messages also support easy interactive message annotation in *any* medium including audio, video, and text.

Montage runs under OSF/Motif on Unix workstations and is compatible with SMTP mail agents.



From: keith (keith edwards)
To: keith
Subject: Q1 Sales Figures
Cc:

Hi Bob,
Here are the sales figures we've generated
over the past quarter. If you could take
a look at these and get back to me I'd
appreciate it.



Montage Features:

- Easy runtime extensibility by end-users to any media, including commercial file formats.
- Supports "mark-up" or annotation of messages in any medium.
- Will work with any SMTP mail transport agents; independent of transport-level protocols.
- Built-in support for GIF, XBM, XPM, Macpaint, Group-3 FAX, JPEG, and other image formats.
- Uses the OSF/Motif interface on Unix workstations; fully X11 compatible.

For more information, contact the Georgia Tech Multimedia Computing Group.
Email: montage-info@multimedia.cc.gatech.edu

Bill Putnam
College of Computing
Georgia Tech
Atlanta, GA 30332-0280
(404) 894-5551

Keith Edwards
College of Computing
Georgia Tech
Atlanta, GA 30332-0280
(404) 894-6266

Appendix 2

The Design and Implementation of the MONTAGE Multimedia Mail System

This paper was presented at IEEE Tri-Comm 1991, and describes the Montage message architecture and the first generation prototype.

The Design and Implementation of the MONTAGE Multimedia Mail System

W. Keith Edwards

Georgia Institute of Technology
Software Engineering Research Center
Multimedia Computing Group
Atlanta, GA 30332-0280
keith@cc.gatech.edu

ABSTRACT

Electronic mail systems capable of transmitting compositions consisting of various unconventional media (such as voice, video, and images) have attracted a substantial amount of interest in recent years. It is important that mailers capable of delivering such documents present them in an organized fashion. We present the Montage multimedia electronic mail system, along with its model for multimedia documents. Montage makes use of a simpler format than more generalized hypermedia systems. It is our belief that the Montage model is more effective than general hypertext for the task of creating user-to-user messages. Furthermore, Montage is designed to be runtime extensible to new media types by its users. Thus the system does not have to know ahead of time all the possible media users may want to send. The system is built on top of existing mail transport protocols for flexibility and portability. We discuss the design of the mailer along with experiences gained from its implementation. The user interface to the system is also presented.

KEYWORDS

Multimedia mail, electronic mail, compound documents.

INTRODUCTION

mon•tage män-'täzh, *n.*

1 *the production of a rapid succession of images in a motion picture to illustrate an association of ideas*

2a *a literary, musical, or artistic composite of juxtaposed more or less heterogeneous elements*

2b *a composite picture made by combining several separate pictures*

3 *a heterogeneous mixture*

Webster's Ninth New Collegiate Dictionary
Copyright © 1989

In the past decade, the proliferation of fast, inexpensive, networked computer workstations has produced an explosion in the use of electronic mail. Electronic mail systems have traditionally been limited to the transmission of pure textual information only.

As computer workstations increase in power, and as the use of windowing interfaces becomes more widespread and standardized, it becomes apparent that capabilities now exist for the transmission and reception of complex multimedia documents consisting of voice, images, video, and other media types in addition to plain text.

The chief advantage of such a system is the increased communication bandwidth between the users of the system. A multimedia mail system should allow people to communicate as freely and without restriction as conventional "paper" mail systems do. In current paper mail systems, users can seal most anything within an envelope and expect prompt delivery. Electronic multimedia mail systems should allow similar flexibility.

Furthermore, to be useful for the widest range of people, attention should be paid to current standards for electronic mail transport. It should be possible to build multimedia mail systems on top of existing lower-level mail transport protocols and thereby use existing mail routing software for transport. Furthermore, the system should allow users to send plain "flat text" mail messages to users who do not have multimedia capabilities.

We present an experimental multimedia electronic mail system, called *Montage*, which has been developed at the Georgia Institute of Technology's Software Engineering Research Center. We believe that this system provides a flexible and convenient means to send and receive complex multimedia documents. This system is built strictly on top of existing lower-level mail routing protocols and thus should work on most systems which support the Unix operating system, X Windows, and Simple Mail Transport Protocol (SMTP) mail transfer systems.

WHY MULTIMEDIA MAIL?

Traditional electronic mail systems have been limited to sending only textual data. While text-based store and forward communication systems are useful, the old adage that "a picture is worth a thousand words" is often true. For example, it is nearly impossible to concisely describe a complex architectural drawing by writing a textual description of it. Many applications involve the use of graphical information and it is extremely inconvenient to transport this information using existing electronic mail systems.

Furthermore, the mode of information delivery most familiar to humans is direct voice communication. The work of Rohr [Rohr 86] indicates that some concepts are inherently graphical while others are inherently verbal. Other research indicates that in many applications, concepts are understood better when presented in a "mixed media" mode [Guastello 89].

While computer-based electronic mail systems enjoy many benefits over paper mail, they continue to lag behind in many respects. With paper mail systems, users can post basically any type of document which can be printed on paper. This includes images and formatted text, all possibly with annotations marked on the document.

With a more flexible electronic mail system it should be possible to combine the benefits of both systems: the convenience and speed of existing computer systems with the flexibility of paper mail systems. Furthermore, it should be possible to use the dynamic aspects of computing technology to open up mail systems to media which have previously gone unused in a store and forward environment, such as audio and video.

DESIGN

Our design for the Montage system was influenced by several desired goals. Foremost, we wanted to create a mail system which implements our model of multimedia messages. Secondly, to be widely used, the system had to be built on top of existing protocols and standards where ever possible. Finally, to meet unforeseen needs, the system had to be extensible by end users.

Requirements

Our design was constrained by several factors. First, we wanted to distinguish between multimedia messages and more complex general multimedia "documents." Montage is not a hypermedia system. Section 4, *A Model for Multimedia Messages* , ellaborates on this.

With this design decision, we were able to greatly simplify implementation. Montage messages proceed along one basic stream of control, unlike hypermedia messages which have a potentially large number of paths through the information.

Another design constraint was that the system should function cleanly over existing low-level transport mechanisms. We do not require that Montage must deliver its messages over TCP/IP connections. Since mail messages may be transferred between several intervening machines which may or may not run Montage, the messages should look for all intents like ordinary Simple Mail Transport Protocol (SMTP) [Crocker 82][Stallings 87] messages. Montage does not rely on any protocols higher than SMTP for transport.

Finally, we deemed that the system must be as extensible as possible. Since we cannot anticipate the needs of all users, we must provide a mechanism to dynamically add support for new media types, without the need to alter and recompile Montage.

Platform

We chose our platform based on availability and portability. While the Montage mail format may be implemented on a variety of platforms, including PCs and Macintoshes, our prototype implementation is for Unix workstations running the X Window

System. This platform ensures a reasonably high degree of portability for the system.

The interprocess communication abilities of Unix and the high level of flexibility in X also greatly simplify the implementation of some parts of Montage, as we shall see.

Specifically, our implementation of Montage was done on a Sun SPARCstation-1/GX machine running SunOS 4.0.3 and X11R4. The interface was built using Hewlett-Packard's widget set for X Windows. Audio support is native on the SPARCstation, video support is provided by a RasterOps video frame buffer board.

The hardware dependent parts of the Montage implementation (audio and video) are localized to simplify the porting process. It should be possible to run Montage on any Unix-based platform with X support. The audio and video functionality can be ported if necessary, but Montage will function perfectly well without audio and video support. In other words, Montage will function as a formatted text and image mailer if these facilities are not present.

Extensibility
It is impossible to know *a priori* all the things users may want to mail with a multimedia mail system. Almost by definition, multimedia mail systems should be able to send and receive a wide array of media.

Therefore, it was a primary goal that Montage be highly extensible by its end users. Montage allows users to modify the behavior of the program at runtime, without the need to recompile the system. Support may be added for new media types and new message transport techniques.

The facilities available for user extensibility will be addressed in following sections.

A MODEL FOR MULTIMEDIA MESSAGES

It is our belief that unlike more general purpose multi- or hypermedia documents, mail messages typically tend to be short and attempt to convey one central idea from the author. This belief has driven our design for the Montage message format.

Hypermedia documents tend to present the user with a large body of information in various media. The user then "navigates" through the document along a path that he or she chooses. Hypermedia systems generally tend to be quite interactive--the user is presented with almost all relevant information about a topic and then browses the document to find interesting information [Nielsen 90].

Contrast this model to a typical mail message in which the author is generally trying to convey some small number of central tenets. Unlike more general hypermedia documents, mail tends to be less interactive. The author has an idea to present and in some sense defines the reader's path through the message at composition time.

We believe that by limiting the generality of our message format we can achieve greater levels of usability, simplicity (for both the users and the implementors), and understandability of messages.

Still, however, since various media will be combined within a single message there must be some way to navigate through the packaged information. We have taken a simple approach to this task without having to resort to the less-constrained approach of a full-blown hypermedia system.

In Montage, all media are one of two classes: static or dynamic. The defining characteristic of a dynamic medium is that the information presented changes with time. Static media do not have this temporal component. Examples of dynamic media include video and audio clips. Examples of static media include simple text, formatted (rich) text, and still images.

Obviously it is of great interest to be able to combine various media freely within a single message. The differing characteristics of our two media classes make this packaging difficult. It is natural to think of a message window containing text and image data freely interspersed. Similarly it is natural to think of video and sound data being played at the same time. This mixing of media is common and is something users are used to experiencing on a day-by-day basis.

But does it make sense to combine static and dynamic media? Consider a message containing only text. When the message is opened a window appears that contains the text of the message. Now consider a message containing only voice. When the message is opened the voice clip is replayed. But what about a message containing both voice and text? Does one begin playing the voice as soon as the text window is opened? What about video and text? Does one open two windows, one for the text and one for the video? How does the user know which window to focus attention on? What if the moving video is placed within the text window? The video would become unviewable if the text were scrolled.

In both of these circumstances the mode of interaction would become one where the user's attention is focused on one media while the other is ignored. Some means must be provided to let users control the playback of the dynamic media so that they can decide when to focus attention on the various media. In addition, in our example above where video is presented in one window and text is presented in another, the user has no idea which contains the "central" part of the message. In other words, the user does not know which component of the message to focus on first.

Research supports the idea that users' interactions with multimedia systems tend to be largely "media-modal." That is, they segregate the information presented to them based on the medium of interaction [Laurel 90].

Primary Media Classes

To overcome these difficulties, Montage introduces the concept of a *primary media class*. Each message in Montage has a primary media class. This class is either static or dynamic. The primary media class is the type of the media presenting the "main thrust" of the communication from author to reader. Various media with the same media class (either static or dynamic) as the primary media class may be combined freely. For example, if a message's primary media class is dynamic, the author may freely intersperse audio and video in a message.

All of the components of the primary media class which compose the principal part of the message are collectively called the *primary component*.

The primary component is presented to the user in a single window. The window contains mechanisms for controlling the presentation of the primary component contained therein (in the case of dynamic media, the controls may be buttons for play, pause, fast forward, and reverse; in the case of static media the controls may be scrollbars to view various parts of the message).

We use the concept of the primary component to project a single "path" through the message, much as in existing mail systems. The primary media and primary component concepts also serves to give the author a means for expressing the central ideas in a message.

If Montage were restricted to using only media with types the same as the primary class, the system would not be very flexible. There is an obvious need for the incorporation of any type of media within a message, regardless of the primary media class.

As we shall see, the Montage model allows the use of any arbitrary media in the form of attachments.

Attachments

To augment the power of the system, Montage also makes use of *attachments*. Attachments may be placed on any message of a given primary media class. Attachments themselves are basically submessages and may be of any media type, regardless of primary media class.

Attachments may be considered to be "margin notes" that are not central to the message but still convey useful information. Since they are not presented to the user immediately when the window is opened they are, in some sense, secondary in importance to the information contained in the primary component.

Attachments give us a way to convey additional information while retaining our easy-to-play back, easy-to-understand main message thread. Without attachments, messages would degenerate to a conglomeration of mixed media within a single window which would in many cases be unmanageable.

In Montage, attachments are presented along side the primary component window in the form of icons. The image of the icon represents the type of media in the attachment. Furthermore, attachment icons are "connected" to a particular location in the primary component of the message. A use of attachments may be to provide annotation or supplementary information to the information contained in the primary component.

In the case of static primary media, attachments are connected to a certain physical point in the message. Thus, a voice attachment may be connected to a particular line number in a text message. In the case of dynamic media, attachments are connected to a certain time range in the primary message. Thus, as the dynamic message "plays back," the various attachments are presented to the user during the time they are relevant. These attachments may be selected as they appear to give additional information provided by the author.

Consider two examples of the use of primary media and annotations to construct a message. A geographically distributed group may be collaborating on a document. The primary component may be the document in question. Attached to this at various points may be audio annotations requesting changes, image data to be considered for review (image data may also appear within the document itself), and video message clips to the various members of the group.

As another example, consider a researcher mailing a video of a presentation to a colleague. The primary media class here is dynamic, and the primary component of the message contains the video and associated audio of the talk. Attached to this at appropriate points are textual and audio annotations (the playback of the presentation may be stopped at any point to review the annotations), image data of the slides used in the presentation, text of papers, program source, and so forth.

Summary of Model

To summarize, the Montage model projects a single path of message traversal. This primary path is reflected in the primary component of the message. The primary component may contain mixed media, but the media it contains must be either all static or all dynamic to focus reader attention.

Additional information may be placed in a message in the form of attachments. An attachment is a piece of annotational information which is secondary to the primary flow through the message. Attachments are not restricted to being of the same class as the primary component.

IMPLEMENTATION NOTES

We have nearly finished a prototype implementation of Montage. This section covers the details of our implementation, including our assumptions and design decisions, the format Montage uses for message interchange, and the user interface to the system.

Overview

It was our desire that Montage be as flexible as possible in the types of media it can use. Therefore, we were determined to provide support even for media with very high bandwidth requirements, although the actual use of such media may be awkward on today's hardware because of pragmatic constraints.

Montage provides full support for interchange of text, still images, audio, and video. Local area networks commonly found today, such as Ethernet, have bandwidth in the 10 Mbps range. Even with Ethernet, applications will almost never see the full theoretical bandwidth available in the system.

The bandwidth provided by such a network is sufficient for text and small image interchange, workable for audio and large image interchange, and unwieldy for video interchange. Nevertheless, network speed and capacity will improve, so it is important to lay a software groundwork for applications which can make use of these faster networks.

Fortunately, the batch-processing style of mail makes full utilization of network resources less important. Whereas a real-time video conferencing system would require a guaranteed portion of the communication bandwidth to function, mail systems can be much less picky. As long as the message arrives in what the user perceives to be a "reasonable" amount of time the system is meeting its goals.

Montage relies on the underlying mail transport agents for actual mail delivery across the network. So as newer mail delivery agents that can efficiently transport large message across a network become available, Montage will be able to make use of these systems.

Interchange Format

While Montage mail headers conform to the SMTP standard, Montage makes use of a custom format for the bodies of electronic mail messages. This format was created because we felt that existing standards were either inflexible in the types of media they allow, or were too unconstrained to present media in an understandable format appropriate for mail messages.

The limitations of existing mail transport agents require that the contents of Montage mail message consist entirely of printable ASCII characters. Although there are some experimental binary mail transport agents, far and away the largest number of mailers available on Unix platforms only support ASCII transfer.

Because of this restriction, we are forced to encode the bodies of Montage messages into ASCII. Our reference implementation uses the standard Unix *uuencode* program to accomplish this. The ASCII-encoded file's size is expanded by 35% after this encoding process. To compensate for this, the message body is first compressed before it is mapped into ASCII.

But what is actually contained in this message body? In Montage, the message body actually consists of several discrete units, called *chapters*. When mail is received by Montage, the body is separated from the header, it is converted to binary from its ASCII format, uncompressed, and then broken down into its components. In our implementation, each of these chapters is stored as a separate file. There is one special chapter, called the *table of contents*, which contains information on the layout of the entire message. The table of contents references and connects all other chapters in the message.

Our first implementation uses existing Unix tools in an effort to rapidly produce an operational version of Montage. We use the *tar* Unix archive program to combine the various chapter files upon message creation, and to break out the chapters upon message receipt.

Montage adds one line to the SMTP mail message header which specifies the version number for the encoding scheme. Since the above method is only one of many possible encodings, future Montage mailers will be able to determine the encoding method by the version number in the header. Note that this is the only addition we need to make to the header to be able to send multimedia messages.

X.400

In 1984, CCITT released a set of standards for electronic mail systems. These standards do not deal with the user interfaces of mail systems, but rather they specify the services available for sending messages across the network.

The X.400 model defines two agents that make up an electronic mail system: the User Agent (UA) and the Message Transfer Agent (MTA). The UA provides the user interface for the system and may interact with other UAs. UAs hand messages off to MTAs for transport. X.400 specifies the interactions between UAs and MTAs, but does not specify the interactions between UAs and the users [Cunningham 84][Cunningham 85].

Montage implements a subset of X.400. Montage provides most of the header fields which are used in communication between UAs and MTAs, but does not provide the inter-UA communication facilities.

It should again be noted that Montage is largely independent of any underlying mail transport agent. The system can be configured to use an administrator-defined mail agent. On our prototype system this is the Unix *sendmail* program. While we have not tried using a full X.400 mail transport agent with Montage we believe that it should function properly.

Chapters

As stated before, a message body consists of one or more message chapters along with a table of contents. The table of contents specifies the relations between the various chapters (relative placement within a message, chapter format, etc.)

Each chapter contains a "section" of the message. Each chapter is represented in one medium. The type of this medium must be specified in the table of contents so that the recipient mailer will know how to "play back" the chapter.

A chapter is basically a single file containing a single message component. Chapters may contain either part of the primary component, an attachment, or the table of contents.

Note that the principle component of the message may be composed of many chapters, as long as the media in those chapters is of the same class as the primary media class (either static or dynamic). Thus, it is possible to have text interspersed with images in the primary component of the message.

Montage treats each chapter as raw data. That is, it associates no real semantic information with each chapter. Instead, the chapter data is handed to a playback module, or handler, which may be internal or external to Montage. The handler which is invoked on a particular chapter depends on the media type of the chapter. These handlers are specified by the users of the system, and Montage provides a mechanism for users to specify external programs which they can use to play back and record message chapters.

Media types are identified to Montage by *tags* which are associated with each chapter via the table of contents. Montage does not predefine any tags and indeed does not even associate any meaning with tags. Tags are defined by the users of the system. When a chapter is encountered the catalog of tags is searched and the user-specified handler (which may be an external program) is invoked on the chapter. The mappings from tags to handlers may be completely specified by users in a per-user database (with "sensible" defaults provided in a system-wide database).

Thus Montage is completely runtime extensible by the user in the media domain.

This ability provides Montage with a great deal of its flexibility. Users can choose their favorite tool for "recording" text (i.e., they can use their choice of editors (or word processors or spreadsheets) to enter text into the mail system). Similarly, they can use their choice of tools for viewing received audio and video chapters. Montage provides some simple means for playing and recording certain simple media. But because the system is not limited to those media which have built-in handlers, users can automatically add support for new media by specifying external programs to be used for media playback and recording. Additionally, this mechanism allows a great deal of user customizability and support for individual user preferences, by allowing users to use their choice of mechanisms for playback and recording of media.

Table of Contents

The table of contents (or *TOC*) is a special chapter that specifies the relations of the other chapters to one another. The format of the TOC is relatively simple, reflecting our simple model of mail usage.

We have chosen a line-oriented ASCII format for the table of contents. Since most TOCs will be relatively small, we felt that the space savings accomplished by encoding the information in a machine readable format would not be as important as the simplicity and ease of debugging provided by a human-readable format.

Each TOC must have at least 3 records. Each record gives some piece of information and each appears on a separate line. The required records are:

1. **`TOCVersion`** The version identifier for the TOC format.
2. **`Class`** The class of the primary message component, either static or dynamic.
3. **`Primary`** The name and type of the message chapter containing a part of the primary component of the message.

In addition to these required records, there are several optional records:

1. **`Attachment`** The name, tag, and position of an attachment. Position is given as a line number in the primary component if the message is static, or as a time offset in the primary component if the message is dynamic.
2. **`Author`** The mail address and (optional) name of the sender.
3. **`CreationDate`** The date the TOC was created.
4. **`Subject`** The subject of the message.
5. **`ID`** A unique identifier for the message on the machine it was created.
6. **`Comment`** Signifies that the rest of the line is to be treated as a comment (ignored). This is primarily used as a debugging tool.

Since it is possible to combine several chapters within the primary section of the message (as long as these media are either all static or all dynamic), there may be several `Primary` fields in the TOC. These chapters will be presented sequentially to the user.

Additionally, some media (for example, a proprietary format for a general purpose multimedia document) may mix static and dynamic media within a single window, even though this mixing is something that "native" Montage does not allow. In such a case, the media in question may reference files which it expects to contain the various media components it needs. These external files may be bundled together with the rest of a Montage message, and are referred to as "subchapters" since they are not used directly by Montage itself, but rather by one of the media that Montage is dealing with. The media which uses these external files must be responsible for managing them. Subchapters are not specified in the TOC since individual applications, rather than Montage itself, deal with them.

Here is a sample table of contents for a Montage message. This message consists of a simple text primary component, with three attachments, a μ-law sound clip (with tag "CODEC88"), another simple text segment (with tag "SimpText"), and an X Bitmap image (tag "XBM").

```
Author: Keith Edwards <keith@cc>
CreationDate: 18 Oct 90 10:18:52 PDT
Subject: Notes from the meeting
ID: 2848.AA08665
TOCVersion: 1.0
Class: static
Primary: txt.2848.txt SimpText
Attachment: snd.2848.snd CODEC88 27
Attachment: txt.2848.txt SimpText 30
Attachment: xbm.2848.xbm XBM 49
```

Interface

We built the interface for our implementation on top of the X Window System. We perceived two primary reasons for using X. Most importantly, X is accepted as a standard throughout the workstation environment we were targeting as our audience. A primary reason for this acceptance is the portability of X and applications developed for X. As a result of this portability, the interface portions of Montage should recompile cleanly on any workstation which supports X Windows.

Secondly, X provides certain general mechanisms for user customization at runtime. We were able to make use of these mechanisms to create a highly configurable mail system--not only is the interface of Montage configurable, but the actual types of media the system can handle can be changed at runtime. It would have been possible to get this degree of configurability without using X, but we were able to prototype the system much more rapidly by using the facilities already available to X applications.

Our front end is built using a set of user interface objects (*widgets* in the X parlance) provided by the Hewlett-Packard company. We choose the HP widget set based on its perceived completeness and orthogonality. A secondary consideration was that the HP widget set is freely available, unlike some widget libraries (such as Motif and OpenLook) which are available only to sites which pay a licensing fee.

Montage makes use of a multiple-window interface. The first thing a Montage user sees is the *Control Palette*. The Control Palette presents the user with a group of on-screen "buttons" representing the major options available in the system. The primary options available from the Control Palette are

- **New Mail** check in any newly received mail into the Montage system. New mail must be "checked in" before it can be read.

- **Info** presents the user with a "pop-up" window, giving information about the program's origin.

- **Messages** presents the user with a scrollable list of messages. Users may then view, forward, save, or delete these messages.

- **Compose** is an interface to message creation. Facilities are provided for composition in several media types. Support for new media types may be added dynamically.

- **Iconify** reduces all the Montage windows to a single small window. This is convenient for when the user is not working with the system at the moment.

- **Preferences** brings up a panel which allows the user to customize his or her default preferences.

- **Quit** quits the system.

We will now go into some detail on the mechanics of the Montage interface.

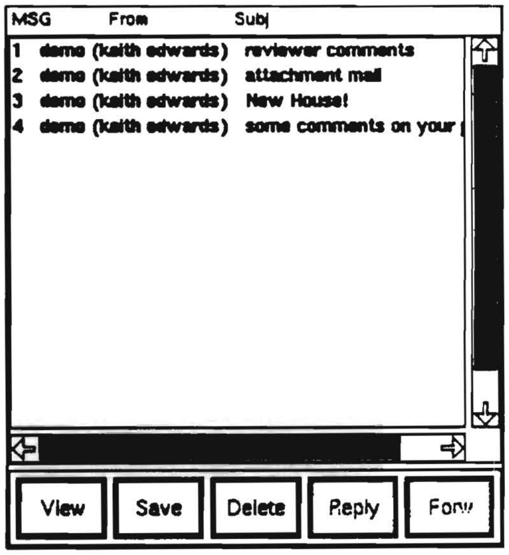
Checking In New Mail Clicking the New Mail button causes any mail in the user's mail to be "checked in" to Montage. Whenever new mail is checked in, the system reads in all new messages from the user's mail spool file. As the messages are read in, the headers are parsed and a mail cache file is built in the user's home directory. The cache file contains important header information such as the message sender and subject, and the number and format of any attachments. The system also calculates the byte offset of the start of the actual message body so that the message data can be separated from the header. After the cache file is updated, the individual messages are stored in a subdirectory of the user's directory, on a one-message-per-file basis. The system spool file is then removed to free up system disk resources.

Users are notified when new mail arrives by the playback of a sound file. Users may also enable an automatic mail check-in function via the preferences panel (see **Preferences**, below).

Basic Message Control The most commonly used Control Panel items are **Messages** and **Compose**. These two items give the interface to message creation, viewing, deletion, and most other common functions.

When the user clicks the Messages button to view the list of current messages, the system scans the mail cache file to retrieve the header information. This keeps Montage from having to open, scan, and parse all the message files individually. A new window is opened and the user is presented with a scrollable list of the current messages.

From the *Messages* window (see Figure 1) users have the option to view messages, save messages to a file, delete messages, or reply to or forward messages. All actions on messages are accomplished by highlighting the desired message or messages, and then clicking the button to perform some action on those messages. Since viewing is perhaps the most common operation, users may view messages by double clicking on the message.



MSG	From	Subj
1	demo (keith edwards)	reviewer comments
2	demo (keith edwards)	attachment mail
3	demo (keith edwards)	New House!
4	demo (keith edwards)	some comments on your

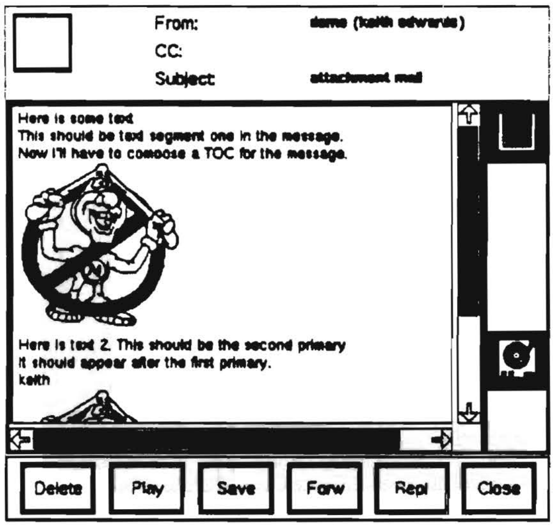
Figure 1: The Message Window

Saving a message brings up a window which prompts for the filename in which to save the message. Currently, messages are saved in their native, bundled format. We will be adding support for saving individual message components.

Replying to a message puts the user into the *Compose* window (to be discussed shortly). When replying, the user is not restricted to using the same primary media class as the original message.

Forwarding simply allows the message to be resent to a new destination.

Message Viewing When a message is viewed, the *View* window (Figure 2) appears on the user's display. The message is automatically unpacked and the primary component of the message is presented in the main portion of the window. Any attachments are displayed along the side of the window and are represented by icons which depict the types of the attachments. There is a single scrollbar attached to the primary window and attachment list.



From: [illegible] (keith edwards)
CC:
Subject: attachment mail

Here is some text
This should be text segment one in the message.
Now I'll have to compose a TOC for the message.

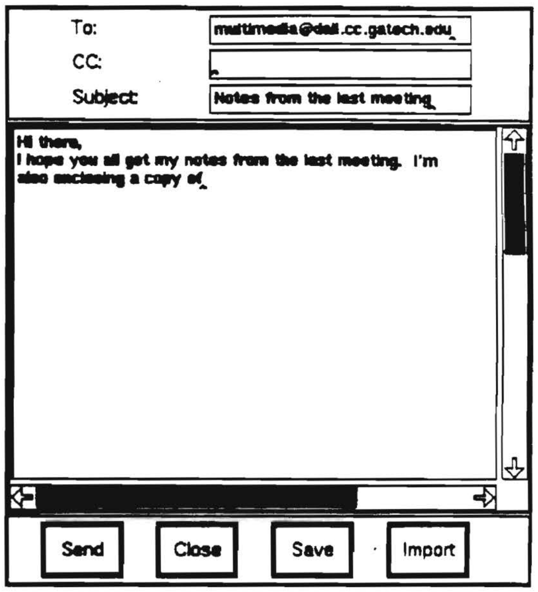
Here is text 2. This should be the second primary
It should appear after the first primary.
keith

Figure 2: The View Window

What is actually displayed in the primary component window depends on the media themselves. Typically, static media will be presented in a scrolling window. The handling of dynamic media is more complex, however. Users have control over sound via a control panel much resembling a tapedeck. Video messages are controlled by via a window with a set of controls much like a video cassette recorder. As the dynamic media are played, any attachments scroll by the user to the side of the control panel. If desired, users may stop playback and view attachments relevant to the portion of the primary component they are viewing.

The *View* window also gives users options to delete, save, forward, and reply to messages. These functions work the same as those presented in the *Messages* window.

Composing New Messages Clicking the Compose button brings up the *Compose* window, which is an interface to message creation (see Figure 3). Users fill in the necessary components of the message header (recipient, subject line, and any carbon copies) by typing in the appropriate fields.



To: multimedia@dali.cc.gatech.edu
CC:
Subject: Notes from the last meeting

Hi there,
I hope you all get my notes from the last meeting. I'm also enclosing a copy of

Figure 3: The Compose Window

The main portion of the *Compose* window is taken up by a region where the primary component is created. Users may type into this window directly, or they may import files of various types via the **Import** button.

The **Import** button brings up a window which allows users to insert external files as either attachments or as parts of the primary component. The user must specify the type of the inclusion so that Montage may build the table of contents for the message correctly.

Because of the ability to type directly into the *Compose* window's primary component area, the input of text is simple. However, it is desirable to be able to record other media as easily as text. It is often cumbersome to have to use tools external to Montage to record some message component into a file and then import the file into a message. Thus the *View* window provides a menu for **Compose Tools**. These tools allow users to incorporate media into messages almost as easily as text.

For example, Montage supports a sound recording tool from the **Compose Tools** menu. This tool (resembling a tapedeck, see Figure 4) allows users to record and edit sound clips. A similar video tool allows the recording and editing of video messages. Users may configure Montage to add new items and actions to the tools menu.

Users click the **Send** button to pass the message on to the mail transport agent. The *View* window contains buttons to close the window, and save the message to a file.

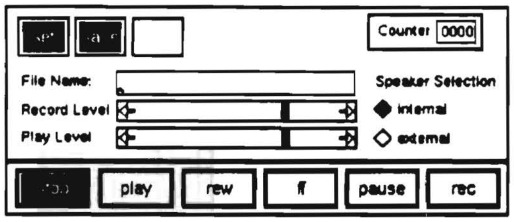


Figure 4: A Sound Tool

Hiding Montage Montage uses a multiple-window interface. Each window in the system may be shrunk or "iconified" individually.

It may be desirable at times to iconify all the windows of the application at once to effectively hide the application. This may be accomplished by the **Iconify** button on the Control Palette. This button reduces all the Montage windows to a single small window. Any windows which have been iconified individually are "joined" into the Montage icon. All windows will be returned to their original states when the icon is clicked.

User Preferences Montage supports user-configurable preferences via the **Preferences** button. This button brings up a control panel which allows users to select and modify a variety of system parameters, including changing the speakers to which audio output will be routed, setting audio play and record levels, setting whether or not message headers will be displayed, and so on.

PERFORMANCE

Overall mail system performance depends on many variables: the underlying network used for mail transport, the size of the messages, and even the habits of the users (some users may want to send only sound clips while others may be content to send simple text most of the time).

Designers of mail systems must be concerned with two aspects of performance: mail delivery latency and message size. Mail delivery latency is the actual time required for transmission of a typical message across the network. Message size is the amount of physical storage required to actually store the mail message.

The delivery latency is dependent on the speed of the underlying network which is in turn dependent on the mail delivery agent chosen for message transport. Since Montage is usable with any RFC-822-compliant mail transport agent, we are not directly concerned with network latency and performance may be characterized by one metric: message size. In most cases message size will be proportional to delivery latency for a given network and transport agent, so message size is an effective measure of performance.

Sending a message via Montage will add somewhat to the "native" size of the information being transmitted. Essentially the native size of a Montage message is the sum of the sizes of the attachments, along with the primary message component and the message header. When the message is sent, a table of contents is added. Furthermore, all of the message body undergoes a packaging process which adds some to the size of the message.

Our goal is to minimize the increase in size by (1) optimizing the packaging process for size, and (2) making the table of contents as small as possible. We will address each of these issues in turn.

The packaging process basically consists of "bundling" the various message components into one file. To minimize the size of the resulting file, it is compressed using adaptive Lempel-Ziv coding [Welch 84] as implemented by the Unix *compress* program). The amount of compression obtained by using this algorithm depends on the size of the input and the distribution of common substrings. Typically, for English text, the compression will be in the range of 50% to 60%. Some files, such as µ-law sound files, are already encoded and cannot be compressed further.

Since the result of this compression is a binary file, it must be mapped into ASCII characters for transmission via RFC-822 mailers. This mapping process expands the file's size by 35% (3 bytes become 4, plus some control information is added). Thus, for fairly long text messages, Montage can achieve message sizes smaller than those for messages containing the same, but unencoded text (original message size × .55 × 1.35 = .74 of the original message size).

Of course added to any encoded message is a table of contents. This table of contents specifies the "layout" of the message. Our format for the table of contents is ASCII-based. We deliberately chose this format for simplicity in implementation and to ease debugging. Usually the table of contents for a given message will be very small.

The minimum size for a table of contents is variable. Since the table of contents contains entries for things such as sender name and subject there is no fixed minimum size. But in general, a simple table of contents will be on the order of 300 bytes. Each attachment will add something on the order of 60 bytes for information regarding attachment type and placement. So we see that the contribution of the table of contents to the size of the total message is negligible.

STATUS

The design and a first-pass implementation of Montage have been completed. Currently the system allows static primary media, and attachments of either text, audio, or several image formats. Facilities have been completed to allow easy composition of messages by typing or importing text, importing images, and recording sound via a simple sound editor mechanism.

Work is progressing on the addition of full dynamic primary media capabilities. We hope to soon have store-and-forward video transmission along with synchronized sound output.

Montage is currently running as a complete functional mail system which can serve as a replacement for existing Unix mail systems. That is, Montage may already be used to replace users' text mail composition and reading programs; the remaining work lies in the area of support for additional media.

CONCLUSIONS, CAVEATS, FUTURE DIRECTIONS

Our work on Montage was based on a set of assumptions which we believe greatly enhanced the usability of the system, as well as simplified the implementation. The initial response within the local research community at the Software Engineering Research Center and outside sponsors has been favorable.

In the near future, we plan to polish the implementation of the system to the point where we have a robust, workable mail system which may then be integrated into an actual work environment. Further work will be done on enhancing the video capabilities of the system and integrating the mailer with existing tools and media.

Much work still needs to be done on increasing the usability of the user interface. An online, context-sensitive help system is also needed.

We feel that Montage represents a flexible, workable system for the exchange of multimedia documents while presenting a friendly, intuitive interface to users.

ACKNOWLEDGEMENTS

The research and development of this system were supported by a grant from BellSouth Corporation.

REFERENCES

Crocker, David H. [1982] Standard for the Format of ARPA Internet Text Messages, Internet Request For Comment (RFC) 822, August 13, 1982.

Cunningham, I. [1984] Electronic Mail Standards to Get Rubber-Stamped and Go Worldwide. *Data Communications*, May 1984.

Cunningham, I., and I. Kerr. [1985] New Electronic Mail Standards. *Telecommunications*, July 1985.

Guastello, S., M. Traut, and G. Korienek. [1989] Verbal Versus Pictorial Representations of Objects in a Human-Computer Interface. In *International Journal of Man-Machine Studies*, July 1989, Vol. 31, No. 1, pp. 99-120.

Laurel, Brenda, Tim Oren, and Abbe Don. [1990] Issues in Multimedia Interface Design: Media Integration and Interface Agents. In *ACM SIGCHI Proceedings, 1990*. (Seattle, Washington April 1-5). pp. 133-139.

Nielsen, Jakob [1990] *Hypertext and Hypermedia*. Academic Press Inc., San Diego, CA, 1990.

Rohr, G. [1986] Using Visual Concepts. *Visual Languages*, S. Chang, T. Ichikawa, and P. Ligomenides, eds., Plenum Press, New York, 1986, pp. 325-348.

Stallings, W. [1987] *Handbook of Computer-Communications Standards, Volume 3: Department of Defense (DoD) Protocol Standards*. New York: Macmillan, 1987.

Welch, Terry A. [1984] A Technique for High Performance Data Compression. *IEEE Computer*, vol. 17, no. 6 (June 1984), pp. 8-19.

Appendix 3

Montage: An X-Based Multimedia Mail System

This paper describes the current generation advanced prototype Montage system.

Montage: An X-Based Multimedia Mail System

W. Keith Edwards

Multimedia Computing Group, GVU Center
Georgia Tech
Atlanta, GA 30332-0280
keith@cc.gatech.edu

1.0 Abstract

This paper describes an extensible multimedia electronic mail system called Montage which is based on the X Window System. Montage supports the composition, transmission, and viewing of structured documents consisting of virtually any type of medium. Further, users can at runtime extend the system easily to support new document types, including text, images, audio, video, executable programs, and commercial file formats.

2.0 Introduction: Why Multimedia Mail?

In the past decade, the proliferation of fast, inexpensive, networked computer workstations has produced an explosion in the use of electronic mail. Electronic mail systems have traditionally been limited to the transmission of simple textual information.

More recently however, as computer workstations have increased dramatically in power and as the use of windowing interfaces becomes more widespread and standardized, the capabilities have emerged for the composition, transmission, and reception of complex multimedia electronic mail messages consisting of voice, imaged, video, and other media (in addition to plain text of course).

The chief advantage of a multimedia electronic mail system is the increased bandwidth between the users of the system. Research indicates that some concepts may be most appropriately expressed in certain media. [7, 8, 10] A good multimedia mail system should allow people to communicate with one another as freely and without restriction as conventional "paper" mail systems do. With paper mail, users can compose documents, jot notes onto them, and then seal them along with any other enclosures into an envelope and expect prompt delivery. Multimedia mail systems would offer similar flexibility.

Furthermore, since the most basic goal of any electronic mail system is to expedite and enhance the communication between people, care should be taken to conform to mail transport standards. An electronic mail system which does not interoperate with a broad range of systems denies its users communication with those systems.

This paper presents a multimedia electronic mail system called *Montage* which has been developed at the Georgia Tech Multimedia Computing Group (a part of the Graphics, Visualization, and Usability Center under the direction of Dr. James Foley) [6]. We believe that this system provides a flexible and convenient means to send and receive complex multimedia documents. This system is built strictly on top of lower-level mail transport standard and thus should be portable to and interoperate with many systems. Our current implementation is on the Unix operating system and X11, specifically Motif 1.1.

3.0 Design Goals and History

When work was started on this project late in 1990 [5], we began with several principles that we hoped would lead to a powerful and flexible system. This section describes our three major design goals.

3.1 Extensibility

First, and perhaps most importantly, we wanted to build a system that would not be a "closed box." That is, we wanted a system which could be easily extended by its users at runtime to support arbitrary media. Too many email systems support only a restricted range of media. However complex these media may be, there is still no provision for extending the system.

At the time we began our work, there were several multimedia mail systems already available for Unix systems, the best-known of these being BBN Slate, the Andrew Message System, and NeXTmail. Each of these suffers from its own set of problems. For example, the BBN Slate system provides users with a number of tools for composing complex mail messages. BBN Slate messages can even contain spreadsheet data, but users are restricted to working with the built-in, integrated Slate spreadsheet rather than the tools there are most used to.

Another system, the CMU Andrew Message System, was built using the Andrew tools provided by CMU [1]. This mail system can send any type of construction which can be expressed by the objects in the Andrew environment. Thus, the system is extensible but it can not interoperate well with systems not built with Andrew.

A third system, NeXTmail, bundled with all NeXT computers, could compose and send fairly complex documents consisting of formatted (rich) text, images, sounds, and typed files. There were (and are) a number of problems with the NeXT mail system though. First, the system is built to a large extent on primitives provided by the NextStep environment which are not likely to be found on other systems. Second, the specification for the mail encoding format (specifically the header lines used and required by the system) is not publicly available.

3.2 Mail Transport Protocols

In addition to our goal of extensibility, we wanted to construct a system which was built on top of existing standards. For our platform, this meant the Simple Mail Transport Protocol (SMTP) as a base [4, 11]. SMTP is widespread among the Unix community. As long as our mailer spoke SMTP its messages would be transmittable by the large number of mail transport agents in the world that speak SMTP.

One of the limitations of SMTP is that it only supports the transfer of non-binary data. Thus, Montage must perform a "packing" and "unpacking" process to convert the message body data to a form which can be transmitted via SMTP.

We will discuss our approaches to the problems of mail transport protocols and packing formats shortly.

3.3 Message Presentation Format

A third requirement of our system was that it should present its messages in an intuitive and easy-to-understand format. It was our belief that, unlike more general hypermedia documents, mail messages are typically generated by their authors to convey some small number of important ideas or data. Thus, whereas hypermedia documents which are reader-driven[9], mail messages are typically author-driven. We reasoned that a hypertext-like presentation format may not be the most useful or intuitive for a mail system.

While we did not want our messages to be full-blown hypermedia documents, we did acknowledge that there was a need for some interactiveness within a message. One common example of this may be a document which is being distributed for review by a number of commentors or coauthors. One would like to be able to view the original document as well as selectively viewing annotations by the various reviewers. We felt that some degree of interactivity would empower the mail system and its users.

As we shall discuss later, we feel that the restriction against generalized hypertext has had a simplifying effect on the design and implementation of the mail system. We also feel that the interactiveness provides a great deal of the system's power.

The next three sections address our solutions to these goals as they are currently embodied in Montage.

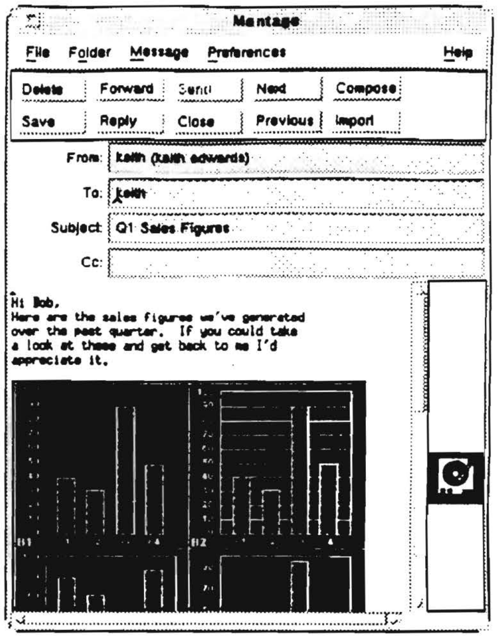
4.0 A Model for Multimedia Messages

We mentioned that one of our design goals was to develop a presentation format for multimedia messages that was (1) somewhat more restricted than hypermedia systems to facilitate the type of communication common in electronic mail, and (2) allowed some degree of interactivity, especially in support of annotations.

Our model for multimedia mail messages essentially consists of two parts. First, all messages consist of a main body (called the *primary part*). The primary part consists of any number of components (called *chapters*) which may themselves be of any media type. All of the chapters of the primary part appear in linear order, just like a single paper document. Montage presents the primary part in a scollable window.

In addition to the primary part, a Montage message may also have zero or more *attachments*. Attachments are analogous to margin notes or "Post-It" notes in a paper document. They allow the author to attach supplemental information which refers to or supports the original document. In Montage, attachments appear as small icons on the border of the primary part. The image in the icon denotes the type of medium in the attachment; the attachment is activated or opened by clicking on it with the mouse. At composition time the author chooses the spatial location of the attachment so that, for example, comments to a document can be located across from the portion of the document they refer to. As the main body of the message is scrolled, the attachments scroll so that they keep their relative position to the part of the main body they refer to. Just like chapters in the primary part, attachments can be of any media type. Figure 1 shows a sample Montage mail message in the system mail viewer. The text and graphics compose the primary part of the message. The small phonograph icon denotes an audio attachment which is spatially located across from the charts.

There are a great number of uses for this scheme of attachments to messages. One of the canonical examples which


Montage
File Folder Message Preferences Help
Delete Forward Send Next Compose
Save Reply Close Previous Import

From: keith (keith edwards)
To: keith
Subject Q1 Sales Figures
Cc:

Hi Bob,
Here are the sales figures we've generated over the past quarter. If you could take a look at these and get back to me I'd appreciate it.

FIGURE 1. An example Montage message

we have already referred to is coauthoring and commenting of written documents. In Montage, a document could be exchanged in mail as the primary part of a message. Various authors can then attach comments, rewrites, graphics, audio annotations, or even video clips to the document at appropriate points. Research supports the fact that some types of revisions are most appropriate in nontextual media [9].

Also, since attachments provide a means of having an *active* message which can interact with the user, several uses for this model which are not based on simply annotation of documents come to mind. For example, a mail message may contain a large number of spreadsheets, say, one for every operating month of a company's history. The recipient of this mail message may not want to view data from every month, so the preferred format may be a message with an attachment for every spreadsheet which may then be opened at the reader's discretion.

Another example may be an attachment which consists of a shell script to upgrade some software package on the user's machine. The main body may have the message "Click here to upgrade to release 2.0." Across from that is the attachment which will perform the upgrade when clicked. (We are ignoring the security problems involved in sending actual executable programs through the mail for now. Obviously you would not want to execute a program sent to you through the mail unless you trusted the source of the message and could verify that indeed the message had come from that source).

The important thing is that the author has the choice at composition time to decide on the layout of the message. We feel that this format gives us a substantial amount of power in a relatively easy to express (and implement) fashion.

5.0 Extensibility and Configurability

Ideally, a multimedia mail system should be able to transport virtually *any* medium. This includes not only text and graphics, but also various dynamic media (such as audio, video, animation, and even executable programs), and various commercial file formats (used by spreadsheets, desktop publishing packages, and so on).

Obviously it is not feasible for the builders of the mail system to have to compile support for these media into the mailer itself. This requires a great deal of work on the part of the system builders to maintain support for all of these various formats, and also means that if the system builders don't choose to support a given format, the users of that format will be out of luck. Further, the mail system is always a "step behind" the rest of the applications world: the mail system builders are continually playing "catch up" to build in support for new formats as they become available.

Perhaps the best solution to the extensibility problem is to have a computing environment in which applications can communicate with one another, and application objects can be shared among and embedded in different applications. Some strongly object-oriented environments (NeXTstep comes to mind) do support this type of behavior, but it is not available yet in the X world.

Since this solution wasn't available to us, we had to take another approach to solving the extensibility problem. Our solution is to externalize as much of the work of interpreting and handling the various media outside the mail system as possible. We use external programs (called *handlers*) that "understand" the various media to display and edit them. In the Montage model, the mailer itself is very simple; it is basically just a framework which has the responsibility of parsing the mail messages, providing the basic mailer functionality (folders, aliases, and so on), invoking the external handlers, and creating a nice presentation for the overall message. The work of understanding what a particular medium "means" and then doing the right thing with it is solely the responsibility of the external handlers.

Media types are identified by *tags* which are simple ASCII strings that are sent along with the message components when it is transmitted (see the section Transport Protocols and Packing Formats). Montage itself associates no meaning with the media tags; instead it decides which handler to invoke on a particular component by looking up its tag in a per-user database which maps tags to handler programs chosen by the user.

This design has several benefits. First, users can make use of the applications most familiar to them to view and edit message components. Vi and Emacs users will be able to choose their favorite editor to compose text components. Secondly, if a work group begins to use a new application for its work, it is easy to enable support for the new application's data format by simply assigning it a tag and putting an entry in the database that specifies the program the be run when a message component with this tag is encountered.

Because of the presentation format we are using (described in the previous section), message components belonging to the primary part of the message are presented "in-line" (that is, they visually and structurally form a single document, rather than being presented in different windows), while attachment components are presented outside the main body of the message in their own windows when they are activated.

The different requirements of displaying message components in-line and outside of the main flow of the document require us to have the notion of several classes of handlers. The basic handlers are:

- **Editor** The program which will be invoked when the user wishes to compose a message component of a given type.

- **Renderer** The program which will be invoked when a message component needs to be displayed in-line to a message.
- **Viewer** The program which will be invoked when a message component needs to be displayed outside of a document.

The editor and viewer handlers operate as expected. They are the normal applications found on a system which are used to display and edit application-specific data (such as Lotus 1-2-3 or FrameMaker or a paint program). These applications, by default, create their own windows as children of the X root window. Thus, when invoked by Montage they will start up and appear as top-level windows "outside" the message itself. Thus, the viewer handler is the program which will be invoked whenever an attachment is opened.

But what about message components which should be displayed in the main body of the message? By default, most any applications will create their own top-level windows when they start up. We need to be able to display message data inside the main body of the message as well as in separate windows. Our solution to this is the notion of renderers. Renderers are programs which know how to draw (or "render") the media type *inside* the main body of the message.

Since this is a rather unusual requirement, most systems will not have renderers already on them waiting to be used (as is the case with viewers and editors, which are conventional off-the-shelf applications). We are working to build several renderers for common formats, and hope that if Montage ever reaches some degree of popularity there will be no shortage of publicly-available renderers.

The mechanics of how a renderer performs its job, that is, how it draws inside the main body of the mail message, are somewhat difficult. We have investigated two possible solutions. In the first potential solution, the renderer draws the medium into a pixmap and then returns the identifier of the pixmap to Montage which determines the geometry of the pixmap and copies it into the message display area. This effectively disallows any type of dynamic medium in which the contents of the displayed are subject to change.

In the second solution, which is the one we are currently working with, Montage launches renderers with a command line argument which is the window ID into which the renderer should draw. This allows fully dynamic media, but has several drawbacks. One is that the renderer must continue running as long as the message is displayed even if it is rendering a static medium. This is so that it can handle exposures in the window. Another drawback is that most existing widgets don't perform well when their windows are resized from some external controlling process. We are investigating writing a new widget which exhibits the proper behavior.

Note that if the X Toolkit provided support for forcing an application's top-level window to be specified on the command line or via some other mechanism, then we could actually embed running applications in the mail message itself. Unfortunately no X toolkit that we are familiar with provides this capability.

While the invocation of an external program for every type of media provides a great deal of flexibility, it is not the most efficient way to work with very common media which will be used on a day-to-day basis. For this reason, Montage has a few very simple handlers built in to it. Users can specify "PrimaryTextRenderer" in the configuration database to tell Montage that a tag represents simple text and that the system should use its own internal text renderer to display it. There is also a "PrimaryImageRenderer" built-in handler that can understand and display a good number of image formats internally (including Sun raster images, PBM, PPM, PGM, Gif, Faces, XWD, Group 3 FAX, MacPaint, XPM, XBM and a few others). Similarly, there is a "PrimaryTextEditor" which tells the system to use its built-in-line text editor so that users will not have to open another window to simply compose a text message.

These built-in handlers provide a certain common ground of media types that Montage can handler "out of the box" without requiring any sort of external mechanisms. Essentially they are an escape hatch around the requirement that users must have external handlers for all the media types that they wish to mail or view.

6.0 Transport Protocols and Packing Formats

As we mentioned, at the time we began work on Montage there were widely accepted standards for multipart multimedia mail transport. We knew that we must base our system on SMTP to allow interoperation with the majority of existing Unix mail agents, but beyond that there were few accepted standards. The current implementation of Montage uses a message transport format developed in-house to support our model of multimedia mail messages. In the past year, however, a format called MIME [2] (Multipurpose Internet Mail Extensions) has gained considerable acceptance and generated quite a bit of interest. We shall describe each of these formats in turn.

The MIME format is quite similar in many regards to the current Montage format and so we plan to convert the system over to MIME in the near future.

6.1 Current Format

To support the message format we wanted, we developed our own transport format based on SMTP. In this format, each individual message component was compressed, bundled, and converted to ASCII (since SMTP doesn't support binary message transmission). The conversion of the individual message components into a single transmittable block of data is called *packing*. This block of data then was transmitted as the body of a message, with the appropriate SMTP headers placed on the front of the message. Along with the message components themselves we transmit a *Table of Contents* (or TOC) which describes how to *unpack* the message body into its individual components, and the relationship of those components to one another.

In the current implementation of Montage, the TOC is a simple ASCII file which has a single line per component, and specifies the component name, whether it is a primary or attachment component, the relative position of the component in the mail message, the medium type, and compression type. The system packs and unpacks messages transparently to the user.

6.2 MIME

In many regards the MIME format is quite similar to the current Montage format and, since it will be supported on many more platforms than the current Montage format, we plan to convert the system to MIME in the near future.

MIME provides support for multipart messages in which each part contains the data for that part and specifies the type and encoding format for the data. The information in the MIME subparts is sufficient to allow for unencoding and decompressing Montage components, but there is no provision in MIME to specify any type of structural information about messages, such as component layout. Therefore we will transmit the Montage TOC as a separate MIME subpart so that Montage mailers can display the messages with all their structural connections, while other MIME-compliant mailers will display Montage messages in a linear layout.

7.0 Implementation Notes and Status

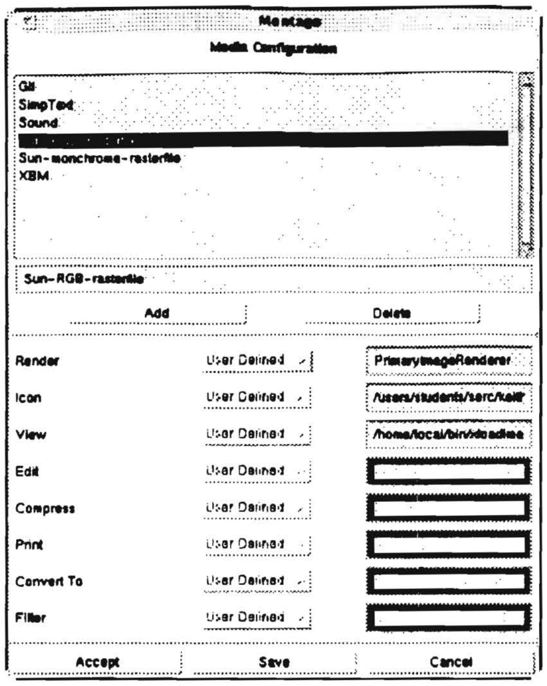
The current version of Montage is implemented using X11R4 and Motif 1.1. The code is approximately 40,000 lines of ANSI C.

This project was begun in late 1990 and resulted in a prototype implementation (based on the HP Widget Set) that demonstrated the basic concepts of the system (extensibility, external handlers, and so on) but was somewhat lacking in the features necessary to convince users to use the system on a day to day basis. In September of 1990 we began a reimplementation of the system based on Motif 1.1 and added a number of useful features.

Most of the features found in Montage are, of course, those found in any conventional non-multimedia mail system, such as folders, support for aliases, saving messages, replying to messages, and so on.

There are several other features specific to Montage have were incorporated into the Motif version though. Perhaps most important is the on-line configuration system. The prototype version of Montage used the X Resource Database to define media tags and map them to handlers. We wanted to separate the configuration of the system into appearance customizations (which would be handled by the X Resource Database mechanisms) and the more Montage-specific tag/handler mappings. One of the primary reasons for this was that we wanted users to be able to establish new mappings between tags and handlers without having to modify their X resource defaults.

Thus, we defined a format for Montage configuration files that contains information about tag-handler mappings. One side benefit of this choice is that we can provide a function in Montage to automatically rewrite the configuration file; this would have been awkward had we contin-



Sun-RGB-rasterfile

Render User Defined PrimaryImageRenderer
Icon User Defined /users/students/serc/keith
View User Defined /home/local/bin/xloadimage
Edit User Defined
Compress User Defined
Print User Defined
Convert To User Defined
Filter User Defined

FIGURE 2. The Media Configuration Panel

ued to use the X Resource Database because rewriting resource files would have lost any comments contained in the files (and thus customizing Montage would have erased comments on other applications' defaults).

Figure2 shows a Montage configuration panel for changing tag-handler mappings. Clicking the Accept button changed the preferences for the current session only. Clicking Save causes Montage to rewrite its own configuration file. These configuration files are, by the way, simple ASCII files and are very human-readable. The automatic configuration mechanisms simply provide an easy-to-use tool for customization by novice users.

In addition to the basic handlers mentioned earlier (editor, viewer, and renderer), Montage also keeps some other information in its configuration database on a per-tag basis. These include:

- **Compressor** Denotes the type of compression used on the tag in question (for example, LZW for text components, JPEG for images).
- **Icon** The file to use for the icon image when a medium of the specified tag is used as an attachment.
- **Printer** A handler for printing the specified medium.
- **Filter** A handler for filtering the medium into text for display on a dumb terminal.
- **ConvertTo** The name of a tag to try to convert this medium into whenever it is encountered.
- **Converter** A handler which is used to perform the conversion to the new media type.

This information allows users to change the behavior of Montage in a number of ways. The method of compression can be changed to suit the particular medium being sent, icons can be chosen on a per-medium basis, handlers can be specified for printing and filtering, and so on.

A note on the converter mechanisms. Some media which are received may not be in a format which the reader can view. The converter mechanisms provides a means for Montage to automatically perform type conversions to a new medium which can be viewed (that is, for which a renderer or viewer is defined). Converters can be chained to any arbitrary level, and Montage can detect and break cycles in the converter graph.

8.0 Conclusions, Caveats, and Future Directions

We feel that the current implementation of Montage serves to illustrate some useful concepts that are important for generalizable, flexible electronic mail systems. We view the current system as an "advanced prototype;" that is, the system implements a number of nice features but it is not commercial-quality software.

Currently, Georgia Tech is involved with licensing negotiations with several companies which wish to take Montage and turn it into a commercializable product. These companies are interested in adding features (such as security) which we are ill-equipped to do because of time and money constraints. Nevertheless, we would like to see Montage released into the community as freely distributable software even if the negotiations succeed.

In the area of future directions, we have several goals for Montage. The most important goal is support for the MIME standard for Internet multipart mail messages. We are also interested in exploring the domain of dynamic mail messages to a greater degree. In the current system, attachments are spatially collocated with the primary components they reference. We are interested in possibly exploring a time-based connection in which the main message body would be a dynamic component, such as video, and the attachments would be tied to a certain time point in the video and would scroll by at appropriate times.

While the concept of renderers provides a powerful extensibility mechanism, the current implementation leaves much to be desired, both in terms of flexibility and ease of implementation. We would like to experiment with more complex Montage-to-renderer protocols, perhaps using some sort of RPC-based mechanism. This would provide a greater degree of renderer control from within Montage (to support, for example, VCR-style controls on a video component).

We are also interested in the use of extension languages which could be bundled with Montage to give it even greater power. Such a system would allow high-level interpretted components to be sent as message components.

9.0 Acknowledgments

Thanks to our sponsors for this work, BellSouth and the Georgia Tech Advanced Technology Development Center. This work would not have been possible without their generous support.

I would also like to thank Tom Rodriguez and Jens Kilian for the significant amounts of time and energy they have devoted to the development of this system.

10.0 References

[1] Nathaniel Borenstein. A Multimedia Message System for Andrew, in *Proceedings of USENIX Winter Conference*, February 1988.

[2] Nathaniel Borenstein, and Ned Freed. *MIME: Mechanisms for Specifying and Describing the Format of Internet Message Bodies*, Internet Draft.

[3] Barbara Chalfonte, Robert Fish, and Robert Kraut. Expressive Richness: A Comparison of Speech and Text as Media for Revision, in *Proceedings of ACM SIGCHI Conference*, 1991.

[4] David Crocker. *Standard for the Format of ARPA Internet Text Messages*, Internet Request for Comment (RFC) 822, August 13, 1982.

[5] Keith Edwards, *The Design and Implementation of the Montage Multimedia Mail System*, Technical Report GIT-SERC-90/04, April, 1990.

[6] Keith Edwards, The Design and Implementation of the Montage Multimedia Mail System, in *Proceedings of IEEE Conference on Communication Software* (TriComm), April 1991.

[7] S. Guastello, M. Traut, and G. Korienek. Verbal Versus Pictorial Representations of Objects in a Human-Computer Interface, in *International Journal of Man-Machine Studies*, July 1989.

[8] Brenda Laurel, Tim Oren, and Abbe Don. Issues in Multimedia Interface Design: Media Integration and Interface Agents, in *ACM SIGCHI Proceedings*, 1990.

[9] Jakob Nielsen. *Hypertext and Hypermedia*, Academic Press Inc., 1990

[10] G. Rohr. Using Visual Concepts, in *Visual Languages*, S. Chang, T. Ichikawa, and P. Ligomenides, eds., Plenum Press, 1986.

[11] W. Stallings. *Handbook of Computer Communications Standards, Volume 3: Department of Defense (DoD) Protocol Standards*. Macmillan, 1987.

Appendix 4

Montage Beta Release User's Guide

This document is the draft Montage user's guide. It is not yet complete, but does describe the basics of Montage configuration.

Montage Beta-Release User's Guide

W. Keith Edwards

Multimedia Computing Group, GVU Center
College of Computing
Georgia Institute of Technology
801 Atlantic Drive
Atlanta, GA 30332-0280

This guide should provide information on installing and using the beta release of the Montage multimedia electronic mail system. The Montage release is currently in beta testing which means that you may find some bugs. This document contains instructions on reporting bugs to the Montage development group. Please note that this beta release is an opportunity for us to learn from you; any comments, suggestions, or gripes which you may have which would help us improve this software would be most helpful.

General requests for information should be directed to the electronic mail address `montage-info@multimedia.cc.gatech.edu`. See the section Reporting Bugs and Suggestions for information on how to pass other information on to the Montage development team.

Before you start using Montage be sure to check the section Caveats and Future Directions for details on known bugs and parts of Montage which are not yet implemented.

Introduction

Montage is a multimedia electronic mail system which allows the composition, transmission, and reception of complex compound documents consisting of text, graphics, executable programs, and even audio and video clips (on workstations which support these media).

Understanding the differences between Montage and some other electronic mail systems may help you to use and understand the system better. This section describes some of the design philosophy behind Montage and highlights some of the more important features.

Message Format

Several characteristics distinguish Montage from other multimedia electronic mail systems. First, Montage uses a fairly rigorous document format. Montage messages consist of a "main body" which may itself be made up of several components of various media types, and zero or more "attachments." Attachments allow Montage users to annotate documents which are exchanged in electronic mail. They are much like PostIt notes or margin notes on a physical paper document, and allow users to exchange, say, a report in progress and comment on it without changing the underlying document itself. In this case, the report would be the main body of the message, and users could annotate it with attachments of various media types (including textual rewrites of paragraphs, audio annotations, supporting charts and graphs, and so on).

We have tried to emulate paper documents in this aspect of Montage. Rather than allowing an arbitrarily complex hypermedia document which may be difficult to compose and interpret, Montage restricts messages somewhat to (hopefully) achieve greater simplicity for both the author and the reader of the message.

Extensibility

A second characteristic of Montage which distinguishes it from other systems is its extensibility. One of the primary motivating goals of Montage was to produce a system which wasn't a "closed box." Many other electronic mail systems on the market restrict the types of media which you can transmit. Some of the more complex of these systems allow you to send, for example, spreadsheet data. The catch is that the data is the format supported by the spreadsheet built in internally to the mail system; often you cannot use your favorite word processors, spreadsheets, and other tools in conjunction with your mail system.

Montage overcomes this limitation by allowing complete runtime extensibility by end users. Users can compose and view message components in virtually any medium, as long as they have a program on their system which "understands" that medium internally. Adding support for new media is as easy as popping up a configuration panel on Montage and specifying some parameters for the system to use in dealing with the new medium.

Montage accomplishes this by basically externalizing all medium-specific handling out of the mail system. The base mailer itself is very simple and only provides the fundamental support for packaging, unpackaging, viewing, composing, and organizing mail messages. A Montage mail message is transmitted as a group of "components" along with a "table of contents" which tells Montage the types of the components and their relation to one another in the message. Montage itself associates no meaning with the type information in the table of contents. Instead, the type is used as a key into a user-definable database which maps media types to external programs, or "handlers," which "know" how to interpret the medium in question. In this way, Montage can support new document formats, data types, and even executable programs without having to be recompiled to "build in" support for the new medium.

Transport Independence

A third interesting characteristic of Montage is that it is built on top of existing mail transport agents, which are themselves fairly high level. In this way, Montage is largely transport-protocol independent. This beta release of Montage functions over existing Unix mail transport (SMTP) links, which means that to the underlying mail transport system, Montage messages "look" just like normal Unix text messages. This allows Montage messages to be transmitted by a wide variety of systems and networks. This network transparency also means that when sending a message from host A to host B, only A and B need to have the Montage software present to interpret the message; to any intervening hosts the message simply appears to be a standard mail message and is thus transmitted through untouched.

Installation

To be included

Configuration

Since one of the main goals of Montage was configurability, care must be taken to ensure that the system is correctly configured when installed. Configuration is controlled by several factors at runtime:

- Environment Variables
- Configuration File(s)
- Resource Database

Montage uses several environment variables to control relatively simple personal configuration options: location of the config file, alternative mail spool files, and so on. The Configuration file itself controls how Montage maps media types to handler programs. The Configuration file is the "heart" of Montage's extensibility since it is in this file that support for new media can be enabled. The Resource Database is a standard X Windows application defaults file which governs the appearance characteristics of Montage: color of windows, fonts, and so on.

This section provides information on how to configure Montage for your system.

Environment Variables

Montage makes use of a number of environment variables to bootstrap the rest of the configuration process. These environment variables are explained below. Each of these variables affects the behavior of Montage on a per-user basis. Thus, if you want any of these variables to take a value other than they're default, you must set them either at the command line or in the login initialization script for the particular shell you are using.

To set an environment variable in csh, do

```
% setenv ENV_VAR_NAME new_value
```

To set an environment variable in ksh or sh, do

```
$ ENV_VAR_NAME=new_value; export ENV_VAR_NAME
```

The environment variables used by Montage are described in Table 1.

TABLE 1. Montage Environment Variables

NAME	DEFAULT	DESCRIPTION
MONTAGERC	$HOME/.montagerc	Specifies the location of the user's configuration file.
MMMAILDIR	$HOME/MMMail	Specifies where Montage will store mail which the user has "checked in."
SPOOLDIR	/usr/spool/mail/username	Specifies where Montage will look for incoming mail.

Most users will not need to set these environment variables to anything other than their default. Note that at the current time, Montage does not support a system-wide configuration file. Thus, each user must either have a personal configuration file in his or her home directory, or must set the MONTAGERC variable to point to a centrally-stored configuration file.

You may wish to set the SPOOLDIR variable to someplace else to avoid interfering with your normal mail processing (that is, so that Montage will not check in your "real" mail...remember that this is Beta-release software!)

Configuration File

The Montage configuration file maintains user preferences and the mapping between media and compression types and programs that "understand" how to handle these types. This information is stored in a configuration file so that it is saved between invocations of Montage.

Future versions of the software will support a centralized system-wide configuration file which will be set up by a system administrator for a particular site. Users will then be able to keep their own personal configuration files that override and augment settings in the system-wide configuration file. The current version of Montage does not support this however; only one configuration file can be read in at start-up. Thus, it is probably best if each user keeps a personalized version of the configuration file in his or her home directory.

The configuration file is an ASCII text file which has three sections: a preferences setting section, a compressor declaration section, and a medium declaration section. The preferences section saves user preferences between runs of Montage. The compressor and medium declaration sections establish the mapping between type names and handlers.

The exact format of the configuration file is beyond the scope of this document, and in fact, users should hopefully never have to modify the configuration file "by hand." Rather, Montage provides some fairly sophisticated tools for manipulating and automatically generating the configuration file via the Preferences menu on the Montage browser window (see the section Using Montage). Users may alter and add to their personal configuration files and Montage will automatically rewrite the file itself, without the user ever having to invoke a text editor on the file.

If Montage ever becomes hopelessly confused however, you can edit the configuration file by hand. Its syntax is fairly regular and hopefully self-explanatory.

Resource Database

Since Montage is based on the X Window System and, in particular, Motif 1.1, the standard X configuration system can be used to modify the appearance and (to a lesser degree) the behavior of Montage. A full explanation of the X Resource Database mechanisms is beyond the scope of this document. If you are familiar with resource databases you are welcome to experiment with new settings in the Montage resource file. If not, you may want to just leave this file alone.

The Montage resource file which comes with the distribution is called Montage.ad (the "ad" is for application defaults). If you want this resource file to be used system-wide you can install it in your X11 app-defaults directory (probably in /usr/lib/X11/app-defaults or /usr/local/lib/X11/app-defaults). If you don't do this, you will probably want to either append the file to your .Xdefaults file or use xrdb to load the file "by hand" from the command line (see the manual page for xrdb for details on doing this).

Using Montage

To be included

Tips for Beta Testers

To be included

Caveats and Future Directions

Since this software is in Beta release there are obviously problems with it (otherwise we would be in final release!) This section should give you some idea of things in Montage that are either not implemented or are known not to work. Also we will go into some of the future directions for this project.

Implementation Holes

There are several "features" in Montage which, while they may appear on menu items, do not work yet. Please be aware of these so that you don't inadvertently lose a mail message by selecting a feature which is not implemented!

File Menu
The New Browser and Read MBox File menu items are not implemented.

Folder Menu
The Rename Folder and Write MBox File menu items are not implemented yet.

Message Menu
The Print and Write MBox File menu items are not yet implemented.

Preferences Menu
The Address Book and Iconify menu items are not implemented yet.

Help Menu
No online help is currently available.

Configuration Options
The UnknownIcon, ShowHeaders, LogFolder, LogOutput,, and EncryptOutput configuration options are not implemented (thus changing these options in the configuration file or via the Options panel will have no effect).

Preferences Panels (Media, Compressors, and Options)
The behavior of the Option Menus on these panels is not completely finished. The menus can be used to set preferences but will not change in response to user input in other areas of the panels.

The File Selector popups in the Options Panel don't work yet.

Known Bugs

This list is only the bugs known since we began our official bug list. There are probably many more not on this list. Please submit a bug report if you find any others.

- If the UseTrashcan option is enabled but the trashcan folder doesn't exist, Montage crashes whenever a message is deleted. The "correct" behavior should be to create the trashcan folder if it doesn't exist.
- If two messages with the same name (message number) are deleted from two different folders, then the first message with that name in the trashcan gets overwritten. Message numbers in the trashcan should have their own ordering and behave just like other folders.

Future Goals

Obviously, we hope to gain many ideas for future directions from you, our beta testers. There are some fairly short-term goals we have in mind for the system however. Please let us know how important these goals are to you to help us gauge where to expend our resources.

- **MIME support.** The current implementation of Montage uses a non-standard message transport format which is not readable by other multimedia mail systems. The reason for this is that at the time work on Montage was started there were not accepted standard for the transmission of complex multipart messages. Recently however, a proposed standard called MIME (Multipurpose Internet Mail Extension)-has emerged. We have been in contact with the developer of MIME and plan to support it in a near-future release of Montage. This would allow interoperability with any MIME-based mailers (including the Andrew Message System).
- **FAX/Telephony support.** The primary goal of Montage is to support and enhance the effectiveness of asynchronous communication (as opposed to realtime communication, such as a face-to-face meeting or telephone communication). To further this end, we are looking at adding FAX I/O and telephone answering capabilities to Montage (with additional hardware support of course). This would allow you to examine your incoming faxes and phone messages in the same "mail box" in which you receive your other electronic communication. The system would also allow faxing of outgoing Montage messages to people who don't have access to a Montage-compatible electronic mail system.
- **Address books.** Since the goal of Montage is to support interpersonal communication as flexibly as possible, an "Address Book" mechanism that maintains user preferences about mail recipients may make electronic communication more powerful. In addition to providing basic mail alias capabilities, Address Books would keep user profile information, such as preferred mail reader programs, media types that users may not have access to, and so on. The Address Book would allow you to customize Montage so that outgoing mail to a user would be automatically converted into the most appropriate format for that user. Of course, the Address Book could also manage other information, such as phone numbers and addresses, which could be used by a fax subsystem.
- **Features, features, features.** Montage is fairly weak compared to most electronic mail systems in the day-to-day features count. This is because we have been devoting most of our time to building the framework of the system. There is probably an endless list of helpful features which could be added into the system. Send us your list of suggestions.

Reporting Bugs and Suggestions

Electronic communication is the best way to reach us to report bugs and suggestions. Please send any bug reports or suggestions that you may have to the electronic mail address `montage-bugs@multimedia.cc.gatech.edu` (Montage-format mail neither required nor desired for bug reports).

Any requrests for basic information should be directed to the electronic mail address `montage-info@multimedia.cc.gatech.edu`.

If you simply want to contact the development team, you can reach all of us at the address `montage-builders@multimedia.cc.gatech.edu`.

As a last resort, you can send us physical mail at the following address:

Montage Project
c/o Keith Edwards
Multimedia Computing Group, GVU Center
College of Computing
Georgia Tech
Atlanta, GA 30332-0280