Use of Computer-Based Rule Systems in Graphic Design

Timothy E. Shea  Bachelor of Fine Arts
California Institute of the Arts
May 1984

Submitted to the Department of Architecture
in partial fulfillment of the requirements
of the degree of
Master of Science in Visual Studies
at the Massachusetts Institute of Technology
September 1986

© Massachusetts Institute of Technology 1986

Signature of Author
Timothy E. Shea
Department of Architecture
August 15, 1986

Certified by
Nicholas Negroponte
Chairman, Departmental Committee for Graduate Students

Accepted by
Massachusetts Institute of Technology
AUG 29 1986
DISCLAIMER OF QUALITY

Due to the condition of the original material, there are unavoidable flaws in this reproduction. We have made every effort possible to provide you with the best copy available. If you are dissatisfied with this product and find it unusable, please contact Document Services as soon as possible.

Thank you.

The images contained in this document are of the best quality available.
Use of Computer-Based Rule Systems in Graphic Design

Timothy E. Shea

Abstract
The number and variety of electronic information sources is growing. In order to manage the vast amounts of data associated with these media effectively, new types of computer intermediaries are being developed. For example, at MIT, experimental magazines and newspapers exist which are no longer written, edited, and designed in the traditional way, but instead are assembled as needed, from scattered sources, by electronic agents acting on behalf of individual readers.

The central question posed by this thesis is: how does one create programs which are capable of formatting this information intelligently? NewsClip is a computer program which uses rules, supplied by a graphic designer, to format an electronic news digest. Its goal is to demonstrate the increased levels of quality attainable when electronic news gathering is combined with intelligent formatting and high quality output devices.

NewsClip represents a new type of graphic design in that it is not an arrangement of visual elements, but a computer program. Its primary output medium is hard copy, although soft copy and voice were tested as well.

This work was made possible by the generous support of IBM, and also with help from HELL, Gmbh.

Submitted to the Department of Architecture on August 15, 1986 in partial fulfillment of the requirements of the degree of Master of Science in Visual Studies

Thesis Supervisor: Muriel Cooper
Title: Associate Professor of Visual Studies
Acknowledgements

I am grateful to the following:

Hal Abelson for intellectual stimulation
Debra Adams for generous support and assistance
Walter Bender for intellectual contribution
Steve Benton for contributions of intellect and time
Pascal Chesnais for stories from NetworkPlus
Muriel Cooper for providing a lab and research support for students like me.

Lois Craig for contributions of intellect and time
Lou Danziger for my design education,
Electronic Publishing Group for intellectual contribution
April Greiman for artistic example
Hell Gmbh. for generous financial assistance
IBM Corporation for generous financial assistance and the generous research assistantship support which made this thesis possible

Russel Kirsch for intellectual stimulation
Henry Leiberman for intellectual contributions to the work
Cheryl Lilley for assistance, patience, devotion, and love
Wayne Lilley for generous technical assistance
Jonathan Linowes for intellectual and research assistance
Andy Lippman for intellectual stimulation
Ron Macneil for intellectual support and assistance
National Bureau of Standards for timely documentation on SGML
Patrick Purcell for intellectual contribution
Mitch Resnick for educational assistance
RichardsonSmith Corporation for generous material assistance
Dorothy Shea for generous support of all kinds
Edward Shea for fatherly support
Skolos, Wedell, and Raynor for generous assistance in production
George Stiny for intellectual stimulation
# Table of Contents

1 Introduction 6

2 The Quantity of Information is Increasing 8
   New Media 12

3 Quality of Information 15
   Strategies for Making Information Manageable 18
   Keyword Search 19
   A Better Search Method 20
   Spatial Data Management 21
   The Electronic Agent 24

4 The Role of NewsClip 30
   The Relevance of Graphic Design 32
   A Programmed Approach to Graphic Design 35

5 The Structure of NewsClip 44
   Structures for Describing the News 47
   The "Generalized Group" 48
   Text 50
   Physical Appearance 54
   Semantic Structure 57
   The Text Object Summarized 64
   Labels 68
   Document as Data Base 71

6 The Operation of NewsClip 74
   NetworkPlus 74
   Reading the Output of NetworkPlus 79
   Assigning Initial Attributes 84
   Mapping Information Onto Pages 86
   Free Space 89
The Simple Case 92
The More Complicated Case 94
The Judge 96
Copyfitting 97
Font Data 98
Translating Page Descriptions to Output Devices 100
Evaluation of Output Devices 100

7 The Impact of New Technology on Graphic Design 106
New Media 106
New Tools 109
The Computer as Conceptual Tool 111

Footnotes 123

Bibliography 125
1 Introduction

NewsClip is a computer program which was developed to test the feasibility of programmed design of electronically gathered news. It is intended to accept input from an intelligent news "agent," which scans a variety of information sources in order to gather stories appropriate to the needs of a particular reader. The news sources include two on-line news databases, television, and a satellite news service. The types of information processed by NewsClip may include text, graphics, and pictures.

The quantity and character of information available in our society is increasing, in part, because of changes in the printing industry, including increasingly digital and automated production techniques, as well as the growth of "desktop" publishing. More fundamental, however, are the changes being brought about through the introduction of new, primarily computer-based, sources of information.

Many of these electronic sources, while able to provide data in large quantities, suffer from unnecessarily low levels of quality. Typically, the user is presented with a vast amount of information which is often poorly organized in terms of his needs, and usually presented with crude graphics. This can be attributed in part to the low quality of current output devices, but also to the fact that in electronic systems human experts are no longer part of the information generation and design processes.

In order to improve this situation, various electronic "agents" have
been devised to help navigate the seas of electronic data. Their role is to locate, retrieve, organize, and display electronically-stored information for the benefit of the reader, thus acting as the electronic analogs of the editor, librarian, and graphic designer. NewsClip attempts to implement a "design agent" which is capable of interfacing to a programmable, intelligent news-gatherer and generating high-quality graphic output.

The primary output generated by NewsClip is hardcopy, produced on a laser printer with type, graphics, and images at a resolution of 300 dots per inch. Other output modes, including soft-copy and voice, were tested and evaluated as well.
The Quantity of Information is Increasing

Figure 2.1  The library at the University of Leiden, the Netherlands, as it appeared in the 16th Century. Books are classified according to subject, and subject names appear on the top of the partitions. Theological texts comprised the largest part of the collection, filling almost six shelves. Mathematics fit on one. Woodcut by Jan Cornelius.

In 1424, Cambridge University's library contained 122 manuscripts. Within 50 years the collection had grown to 330 volumes, reflecting a growth rate of just over 4 acquisitions per year. These figures are in fact quite respectable, considering that the price of a book in medieval times was on par with that of a farm or vineyard.  

Cambridge University's collection now exceeds 2,500,000 volumes. The United States' Library of Congress boasted 54,289,000 items in
The Quantity of Information is Increasing

Figure 2.2 Jost Amman, woodcut illustrations for *Standebuch*, or "Book of Trades", 1568. This sequence of images depicts the various steps involved in printing a book in the 16th Century. Clockwise from upper left: A) Making parchment; B) Making paper; C) Casting molten lead into type; D) Setting and printing type; E) Artist creating images in preparation for woodcut or copper engraving; F) Making woodblock illustrations; G) Illuminating a manuscript with gold leaf and color; and H) Binding the printed books. (Taken from *A History of Graphic Design*, by Philip Meggs.)

1966, including over 13,000,000 books. This collection is growing by over a million pieces per year.

Changes in printing technology over the last five hundred years have made these enormous increases possible. Censorship and other factors being equal, the volume of information available in a society is inversely proportional to the effort required to publish. While it once took an order from Constantine the Great to produce 50 copies of the Scriptures, Johann Gutenberg's invention of movable type in the 15th Century made
the reproduction and dissemination of God's word more routine. The Industrial Revolution brought us the Linotype machine, invented by Ottmar Mergenthaler in 1884, which allowed an operator, working from a typewriter-like keyboard, to cast entire lines of type into a single piece of metal. This meant that people were no longer needed to arrange each letter of a page by hand. The development of photo-composition in the 1950s accelerated the printing process further: a manuscript encoded on paper tape or magnetic media could be marked-up electronically and then sent directly to a photo-typesetting machine. There, a flashing light and a moving photographic negative were used to expose the image of individual letters onto photo-sensitive paper or film. Digital typography did away with the complex machinery associated with photo-typesetting, allowing the text to be manipulated within a computer's memory and printed using laser technology.

Digital transmission of data and images is continuing to automate the printing industry. Word processing not only makes authors' lives easier, but also allows their work to be processed by a computer and sent directly to a typesetting machine, eliminating the re-keying which is necessary when manuscripts are written on paper. Satellite relays are streamlining information flow by eliminating much of the time and cost involved in transporting newspapers and magazines by truck, rail, or air. Publication of the newspaper USA Today, which appears daily in major U.S. cities, is made possible by satellite transmission to regional printing and distribution centers across the country. In this case and others, photographic techniques once used by printers are being replaced by digital processes in which type and pictures are stored in computers, pages are composed electronically, and printing plates are generated
directly (or almost directly) from the electronic representation.

We are also experiencing the growth of another source of information: "personal" or "desk-top" publishing. Authors, using currently-available computer systems, can produce their own books or documents containing type, illustrations, and pictures, which in some aspects rival the quality of traditional printing techniques. Xerographic reproduction allows these documents to be disseminated cheaply. Such systems are being used to produce office reports, technical documentation, and academic papers, including this one.

Many companies such as Xerox, Apple Computer, Interleaf, and others, are producing systems aimed at the various segments of the desktop publishing market.

A current system worthy of mention is Donald Knuth's Tau Epsilon Chi, better known as TEX, a procedural markup language developed to facilitate the production of mathematical manuscripts by their authors. "Procedural markup language" means that the author marks up the manuscript with special codes which describe the way in which various pieces of text are to be printed. The user can create "macro" definitions, meaning that a whole sequence of instructions can be clumped together and referred to by one name.

TEX was developed because mathematical manuscripts are especially difficult to typeset. One reason is that they are typographically complex, due to formulas requiring symbols and arrangements rarely found in non-technical writing. In addition, typographers and printers rarely understand the text of these manuscripts, and therefore are unlikely to catch their own errors. Although Knuth's system was originally developed for the field of mathematics, the necessity for unhampered
manipulation of individual symbols and groups of symbols has resulted in a generalized system of typographic description which can be used on many types of documents. Knuth's system does not deal with images directly, and so papers which contain images have to be assembled later by hand. The intent of Knuth's system is to allow authors to write, design, and produce their own papers, and in this sense it represents a fundamental change in the way we produce information.

Scribe, developed by Brian Reid at Carnegie-Mellon University, is another computer program which helps authors produce documents. A manuscript, which has been suitably marked up, is fed to the Scribe program, which controls fonts, spacing, linebreaks, headings, footnotes, numbering, tables of contents, indexes, and other factors. Various pre-defined formats or "styles" are defined by Scribe, such as a Thesis Format, Technical Report Format, and so on. The format one selects determines the final look of one's document. Scribe was intended primarily for technical writing and, like TEX, does not deal directly with photographs or images.

New Media
A number of new media are appearing which represent a more fundamental development in the way we produce, transmit, and consume information. These media are electronic and computer-based, and they herald a change as important as Gutenberg's movable type. Not only are they capable of storing and transmitting vast amounts of data, but they are also able to combine this data with computational power, thus generating
The Quantity of Information is Increasing

Figure 2.3 This optical disc is capable of holding over 500 million characters of information, with the result that the text of an entire encyclopedia can be stored on less than one side. Because the information is computer-readable, it can be presented to the reader in a wide variety of ways, depending upon the sophistication of the retrieval program.

information systems and computer "environments" with widely varying characteristics.

Information services such as *Dow Jones News Retrieval, NEXIS, The Source*, and *Compuserve* are repositories of vast quantities of information stored electronically and accessed by individual users via telephone lines. Many contain articles from a wide variety of sources, which can be searched by a computer. They also provide a wide variety of consumer services and news. Personal computers equipped with specialized FM receivers get up-to-the-minute news and financial information, including the current stock quotes from Wall Street; satellites carry news from around the world, and inexpensive optical storage
devices such as the video and compact-disc are capable of storing staggering amounts of information in a very small space. Individuals with modest collections of these discs now have access to library-sized quantities of information.
An information system's usefulness is judged not only by the quantity of data it contains, but by its effectiveness in amplifying the part of that data which is relevant to the user. A good system should provide relevant information quickly, organize it properly, and present it legibly.

Many of the newer media present problems in one or more of the above respects. Relevant information is not always easy to find. Irrelevant
information is. Quality problems can be attributed in part to the fact that
human experts are no longer part of the information generation process.
The editor, designer, printer, and librarian perform the tasks of information
organization and presentation in print; in electronic media this is no
longer practical, and quality suffers as a result.

Exacerbating the problem is the fact that most computer output
devices, having descended from terminals intended for computer
programmers, are of low graphic quality. These devices are characterized
by coarse resolution, bad fonts, few or no font or size options, little or
usually no color, and small screens. These systems compete with print by
providing new functionality, not good graphics.

High quality graphics may seem unimportant from a purely practical
point of view. To a certain degree this is true, for we find that as the need
to know increases, people in fact will endure lower levels of graphic
quality2. When a fire is raging, no one really cares about the quality of
spacing between the lines of text on the fire extinguisher, as long as they
can understand its instructions.

On the other hand, good graphics does increase legibility, which is a
necessity, especially for emergency instructions. Thus the opposite point
can be made: as the importance of information increases, so does the
necessity of it being easily understood. The most important information is
in fact most in need of good graphics. This argument further suggests that
good graphics and organization, by improving the legibility of information
in general, and thus promoting that knowledge which is essential to
civilization, has a crucial role to play in the success of our species.

The results of human expertise can be seen in a good magazine,
when viewed in comparison to the typical on-line news database. The
purpose of the specialized journal, of course, is to provide relevant information on a given topic. This requires an editor who solicits, judges, selects, and modifies material from various sources. He or she is primarily making decisions regarding appropriateness and quality. The editor must know the journal's history, in order to provide both the variety and continuity necessary to keep readers well-informed. He or she must be able to make judgements about whether something is "good," "relevant," or "well-written." The editor exercises knowledge about the journal's subject, its goals, readers, and the culture as a whole.

The graphic designer's role at this stage is to understand the information, organize it well, and provide graphics, illustrations, and other material which helps the editor and authors make their point. The designer must also understand the publication's audience.

When the editor, designer, and printer do their jobs properly, the magazine can be an information source of extremely high quality. The resolution is very good, the pictures are in full color, the type is used properly and printed clearly. Lines of text are given a proper length and adequate spacing. Pages are organized, by people, in order to reinforce and illustrate the subject matter. Information which has been groomed in this way is easier and more pleasurable to read. This is the result of human expertise combined with high-resolution color graphics.

After editors and designers have had their say, librarians filter information even further. They acquire that material which is most appropriate to the library's collection and users. They organize it according to standard systems. And they can assist the user in finding what is needed.
Strategies for Making information Manageable

A number of approaches are being tried in an attempt to make electronic information systems more tractable. They include the development of electronic "agents" which act as intelligent intermediaries between the user and data. These agents can be seen as the electronic analogues of print-based human experts such as the editor, graphic designer, librarian, and printer. In attempting to emulate the functions of people, agents suffer the disadvantage of being machines. However they are able to take advantage of many of the possibilities which present themselves when computational power is combined with electronically stored information. Their function is to find, filter, organize, and present information in ways which are relevant, useful, and enriching.

This thesis attempts to demonstrate the increased levels of quality
available when a sophisticated news gathering agent is combined with intelligent formatting and high-quality output devices. The following sections describe a few of the attempts made to solve the various problems presented by electronic information systems. This should establish the context of the current research, particularly as it relates to more sophisticated information-gathering experiments, such as NewsPeek and NetworkPlus, both developed in the Electronic Publishing Group at the Media Laboratory, MIT.

The Keyword Search
The most common example of electronic information management, the "keyword" search, is a method for finding relevant pieces of information from the large bodies of data stored in computer databases. Keyword systems enable one to search electronically-stored news stories for those articles which contain certain words. The premise is that if a set of words can be used to capture the essence of what one is looking for, stories in which these words appear are likely to be relevant, while those that lack them are not.

One problem with this method is that writers often use different language when discussing the same subject. Another is the difficulty posed by synonyms, and words with broad meanings. A search for documents containing the word "chips" may find 5 articles relevant to someone interested in computers, but miss 10 others that were indexed under "integrated circuits." Furthermore, the five relevant articles might be returned mixed with twenty irrelevant documents describing cookies, or other subjects where the word "chips" appears.
In order to alleviate this problem, keyword search systems often provide means for specifying relationships between the keywords, resulting in the ability to find all instances of "Imelda" occurring within ten words of "shoes." Selecting a good set of search criteria is an acquired skill; one always runs the risk of generating huge volumes of irrelevant information if the description is too broad, or of overlooking important articles if the criteria are a little wrong.

A Better Search Method
Recent advances in computer design have made improved search techniques possible. Parallel Processing, in which computers use thousands of processors simultaneously (as opposed to the single processor of traditional computers), can produce very high rates of computation in certain situations. The Thinking Machines Corporation, of Cambridge Massachusetts, has developed an interesting method for document retrieval, using their "Connection Machine" system, which contains over 65,000 processors. A database of documents is stored on the system, with one (or perhaps several) documents assigned to each processor. Once the first "good" document is found (through a conventional keyword search), it is used to create a "pattern." This pattern consists of all the important words and phrases contained in the "good" document. Commonplace words are discarded.

Each word in the pattern is then broadcast to all the computer's processors simultaneously. Upon receiving a word, each processor checks its document to see if it contains that word. If so, that document's "score" is increased. Once the entire set of words in the search pattern
has been exhausted, the document which most closely matches this pattern will have the highest score. This document is then presented to the user as being the one "most like" the original document. Other documents could then be accessed by decreasing level of similarity. This approach exploits the fact that "every document is, in effect, a thesaurus of its subject matter." This method is different than the keyword search discussed previously, for, in effect, the user is asking the system to find another document which is "similar" to the one he or she already has.

This approach seems to be a closer approximation of the way people think, which is probably a good indication that the technique will prove useful. When creating a search process, what we are trying to do, after all, is make a little machine that thinks like we do.

Spatial Data Management
Dr. Richard Bolt, Principal Research Scientist at MIT's Media Laboratory, drawing in part on theories proposed by the psychologist George Miller, and working in conjunction with other members of MIT's Architecture Machine Group, has experimented with a number of solutions aimed at increasing the usefulness of electronic information systems, as well as decreasing the frustration associated with using them. In his book *The Human Interface*, he says:

"People are very good at using the space around them for organizing and storing things. They lose this option, though, when they sit down to work with computers. The opportunities and means to use space in dealing with data
3 Quality of Information

...consider my bookcase. The copy of *Moby Dick* resides on the second shelf from the top, at the far left end. When I reach for it, I do not even have to look in that direction because my action is guided automatically by a mental schema of where it's located, as well as a sort of 'motor memory' whereby my very arms and fingers seem involved in remembering where things are.\(^6\)

Bolt's book emphasizes a number of themes, including new approaches to storing and retrieving data using "spatiality," the combined use of human speech and gesture in communicating to computers, and the use, by computers, of eye movement and direction of gaze as sources of information about the user's area of activity and interest.

One of Bolt's main points is that the schemes used to represent data inside computers, being electronic, are alien to most humans, due to the fact that people have a strong need for the visual, spatial, and tactile cues which they are accustomed to finding in the physical environment. Systems which ignore these needs are likely to be disorienting and difficult to use.

Several computer systems, such as the Media Room at MIT's Architecture Machine Group, were assembled to experiment with these ideas. The Media Room is the size of a personal office, in which one entire wall is a computer display screen. Speech, gesture, touch, eye-tracking, and other methods are used to provide people with a variety of means to communicate with the computer. The screen itself, being extremely large, is capable of providing more information at
once, and better spatial orientation. Representations of data are often visual, consisting of some type of facsimile of the data itself. Any one of several methods are provided to allow users to navigate through this electronic "dataland," including touch, voice, "joystick" input devices, and so on.

The primary intent of this research is to address some of the problems occurring at the "interface" between people and machines, and resolve them in favor of the people, so that computer-based information systems can become richer, more responsive, and easier to use.
The Electronic Agent

The number of media providing us with news and information is growing: from print, to radio, to television, to electronic databases, and others. At the same time, the volume of information being carried on each of these channels is increasing. One method of dealing with this flow is, obviously, to filter it down to a more manageable level.

Professor Nicholas Negroponte of MIT's Media Laboratory has suggested the metaphor of the spouse: you are running late, eating a quick breakfast, while your spouse is reading the paper. You want to know what's in the news. So you ask: "Honey, what's in the news?" Your spouse, knowing your interests, and the context of your request (in a hurry, eating breakfast), is likely to give you an appropriate summary of that news which he or she feels will be of interest to you.7

The electronic "agent" is intended to fulfill this role when searching the seas of electronic and broadcast information. It is a surrogate for one's self, a repository of one's interests, an electronic intermediary. But because it is computer-based, it brings a host of other possibilities to the process. Andrew Lippman, co-director of the Media Laboratory's Electronic Publishing Group, describes his group's research:

"In particular, we investigate systems where broadcasts are directed at computers, and the information thus broadcast is interpreted by the ultimate recipient via interaction with a personal computer. That computer acts as formatter, printer, filter, and general interface between the immediate
desires and interests of the "reader" and the information itself.

"The research specifically addresses innovations made possible by the addition of this computational intermediary -- new publishing forms, styles, and technologies. The presence of processing power acting on behalf and in concert with the information consumer, we believe, has the potential to translate broadcasts into apparent dialogues, introducing personalization into what has generally been assumed to be a highly impersonal, often synchronous informational experience. What has been considered mass media can be redefined."

"This redefinition is significant to both the publishers and recipients of the information. The role of publisher increasingly becomes that of data distributor; the computer readability of the information and associated cross-referencing capability perhaps becoming more important than packaging. In addition, and perhaps more important, a convergence of publishing forms is possible. For example, is a newspaper that contains illustrations drawn from network actualities [i.e. video] a dynamic version of the printed edition or an annotated TV newscast? Most generally, it has the characteristics of each and its ultimate form is as much a result of system design as technological capability." (Andrew Lippman, *Electronic Publishing*, January, 1986)
Electronic agents have intelligence in the sense that they maintain some idea of the reader's interests or needs and peruse news sources accordingly, gathering whatever relevant information they can find. They can accept explicit directives regarding one's interests, or use one's reading "history" as a basis for future decisions regarding the importance of a specific piece of news. In this way, the agent acts as both "filter" and "amplifier." Electronic agents assemble information from a wide variety of sources, including television, optically-stored video and images, wire and satellite news services, and on-line databases.

In addition to providing news, systems like these can organize and deliver personal information, such as the user's electronic mail and messages, lists of appointments, or a reminder of an important meeting. An electronic photography magazine, developed in 1986 at the Media Lab's Electronic Publishing Group by Gitta Salomon, tailors reports on cameras to correspond to that equipment which the reader already owns or is interested in. Current newspapers contain classified ads and cinema listings. Why shouldn't an agent be able to scan electronic classifieds for suitable apartments, or inform one when a favorite film comes to town? Perhaps the system could monitor particular stocks, calling them to the user's attention when they exhibit large changes in price or trading volume. It is easy to imagine many other examples.

There are possible drawbacks to these systems. Probably the most commonly voiced is that an agent might miss news which is important, particularly if it is looking in only those areas for which the reader has previously expressed an interest. The agent is unlikely to exhibit a human sense of appropriateness, or to make the kind of judgements an editor, spouse, or reader would. A fear expressed by some is that the electronic
agent may in fact deny them opportunities that traditional media provide, particularly the serendipitous encounter, where an interesting article is found simply because it happened to be next to another, or on the page one opened first.

However, the electronic agent is not necessarily intended to supplant the role of the traditional newspaper or its editor. The newspaper is a highly developed, well refined information medium which provides a number of functions unavailable in current electronic media. The front page, for example, contains a wide variety of information which arrives "in parallel," meaning it is all available at once, not through a sequence of computer "menus." The reader can peruse several articles before committing to one in-depth. Attempts to mimic the front page of newspapers electronically have been mostly unsuccessful, primarily because computer displays lack the size and resolution necessary to represent the typical newspaper: five to eight stories on the front page, each running 1,000 to 1,500 words before breaking, plus pictures, weather, indexes, and so on.

Electronic information, being represented in forms which people can not read, will always require some form of intermediary. A "dumb" one is not necessarily better. The question is not whether electronic agents should exist, but what forms they should take, how they can be programmed, and what types of behavior are most useful.

Another disadvantage of electronically stored information is that, because it is in a constant state of "publication," access to it can be denied more easily than in print. When one owns a book, one is reasonably certain of having access to the information it contains. In addition, one certainly would not want reading habits monitored, a distinct possibility on
large networks of computers, but not so problematic in the reception of broadcasts or satellite transmissions. Mixing personal reminders with public news, or gathering information from a wide variety of sources raises another question: Is one broad information "funnel" really better than many discrete, specialized sources?

One of the electronic agent's biggest advantages is access to multiple news sources, allowing one to read about an event from US, European, Soviet, Chinese, or Third World news sources, or from the news agencies of the specific countries involved; or one may have a preferred news source for sports, and another for foriegn coverage.

The NewsPeek project, developed by the Electronic Publishing Group at MIT's Media Laboratory, combines a number of these functions into an electronic news "environment." The system provides news from a variety of sources, includes video and sound as output, and presents stories on a touch sensitive video display. Readers read or watch the day's stories, and can interact with them by touching the monitor screen. A touch in one place may move to a related story, while a touch in another may provide a current story in more detail, play a piece of video, or perhaps switch to a different subject entirely.

NetworkPlus, another system developed at the Electronic Publishing Group, was designed to provide background material for television news stories. During the day, NetworkPlus connects to various news wire services and gathers the day's stories. Later that day, when the viewer is watching the evening news, NetworkPlus monitors the closed-captioned text of the broadcast while searching for related stories in its database collected from the wire services. If the viewer is interested in one of the television news stories, NetworkPlus provides instant access to any
3 Quality of Information

related material it has. This allows the viewer to pursue a story in depth, if so inclined.
4 The Role of NewsClip

We are experiencing the proliferation of electronic information sources, as well as the application of computer intermediaries in an attempt to manage this data intelligently. "Magazines" and "newspapers" of a new type will begin to appear, no longer written, edited, designed, and printed in the traditional way, but assembled from scattered sources by electronic intermediaries. Not only must these intermediaries perform the role of information-gatherer, but they must also be capable of organizing and formatting graphic information "on the fly," for the benefit of each reader. This information might be intended for a laser printer, a video monitor, voice and sound generators, or news/information environments, such as NewsPeek. This task can only be done by a computer.

Thus the central problem posed in this thesis is: How does one create programs which are capable of formatting this information intelligently?
The project begins with the assumption that some type of agent has gathered a body of information from a variety of news sources. Currently, these sources include on-line databases, television, and satellite. The degree of organization that the agent might impart to this information is an open question, but (for the moment) is assumed to include an indication of those stories which are related to others, so that groupings of similar stories can be formed. It will probably include some additional scheme for indicating importance. This means news stories will be ranked in terms of their importance relative to other stories of the same subject, and subjects will be ranked in relation to each other. Stories might be ranked in an absolute sense, that is, in isolation from any subject groupings.

The method used to arrive at this ranking is also left as an open question, the answer to which depends on the nature of the agent. Rankings can be attached by an human editor at the information source, they can be derived from the reader's expressed preferences, past history, or in the case of print media, somehow inferred from headline size or page position. In any case, the purpose of the rankings is to indicate which subjects should be emphasized, and which articles within subjects should be brought to the reader's attention first. If no ranking scheme is present, the subjects and stories are simply presented in the order in which they arrive from the information source or agent.

Output devices are assumed to include medium to high quality print, video display, and sound.
The Role of NewsClip

The Relevance of Graphic Design

Graphic design is a somewhat "hazy" profession, in the sense that its borders are not always distinct, wending as they do into the worlds of advertising and commerce, publication design, packaging, fine art, and information design. In any case, what the graphic designer does, for the most part, is work with print. The profession has grown along with the printing industry. The graphic designer knows something about the various printing processes, and at minimum knows how to pull material together and put it into the various forms that the printer needs for reproduction. The graphic designer knows about photography and typography, and he or she usually has a grounding in the visual arts.

Nearly all printed material we encounter, including books, magazines, advertising, billboards, posters, and the like, has passed through the hands of a graphic designer who has molded its visual form according some set of criteria, with the intended effect of increasing sales, communicating an idea or information clearly, or convincing people to do or believe something, as the case may be. Graphic design is a practical profession, in the sense that advertising must sell, books should communicate their information clearly, and magazines must maintain the reader's interest.

Aside from or perhaps in addition to the graphic designer's concern with legibility is an interest in form, not only because form can increase legibility, but for aesthetic reasons. While graphic design is a practical profession, it is also an art; it can engage our aesthetic sense, or its subject matter might, as in the case of a beautiful poster or museum catalog. In some cases, we find designers working as fine artists, with a clientele willing to pay for the designer's aesthetic sense, and the
designer's work being, in greater or lesser degree, primarily an expression of formal or aesthetic concerns.

In regard to this thesis, it is likely that the most practical assistance comes from the field of "information design." The goal of information design is to put information into a form which can be easily understood by people. A manual explaining how to use a computer must be comprehensible if it is to be of any help. This requires that the content of the explanations be clear, that they be presented in the proper order, printed clearly, and found easily when one is in a hurry. Diagrams and pictures might be used to illustrate instructions or processes more clearly. Sometimes information needs to be given in properly-sized "doses" so that the reader is not overwhelmed. Indexes, headings, charts, and coding schemes can assist a reader in steering through a manual and locating what is needed quickly. Information gathered from a variety of sources by the electronic agent, and needing to be assembled into a coherent document or presentation, presents many of the same problems addressed in the field of information design.

Theoretically speaking, the strategy of information design is to create a graphic representation of information which elucidates the inherent structure of the subject matter as much as possible. This is achieved by establishing similarities in appearance between information of the same class, making clear distinctions between information of different types, and adding different degrees of emphasis to information, depending upon its importance in relation to the surrounding information. Adding emphasis through the use of color, changes in type, spatial relationships, and graphics can foster understanding by encouraging text to be read in the order which makes the most sense. A well-designed information system
has consistency, with the result that the reader learns quickly where to go for the particular type of information he needs.

The strategy of NewsClip is to organize news stories from various sources by determining which stories are related, what subjects are most important to the reader, and which stories are most important within each subject. Related news stories are collected in easily-recognizable groups, which are delimited by some graphic means. After this, the parts of each story are located. Then, graphic attributes are assigned to each part in order to emphasize the various parts according to their importance. It was realized at the outset that NewsClip would succeed to the degree that a "programmed" approach to handling information of this type was possible.
Figure 4.3  Gainsborough: Countess Howe. The Countess' right eye is placed according a formula: it is at the intersection of two lines, the first drawn from the lower left corner of the canvas to a point on the top edge which is 1/3 of of the canvas' width. The second line originates at the upper left corner and runs down to a point along the right edge which divides it in a ratio of 1:sqrt(2). Compositional formulas such as these can be found in the work of Rembrandt, Titian, Van Dyck, and others. The "golden section," sqrt(2), 30, 60 and 90 degree angles, and integer ratios were favorite proportions.

A Programmed Approach to Design
Design has a tradition of "systems." These are "programmed" approaches for handling form. In graphic design, the most obvious examples are the modern "grid systems," originating with the De Stijl movement in Holland in 1917, and developed and popularized in various ways, but particularly by the "Swiss designers" of the 1950's. The grid serves a number of
purposes, but the primary one is practical: specifying, in a consistent way, where information of various types is to be placed on the page. In essence, the traditional grid is a system of rules about position.

In a more general sense, grid systems are used to impart order. Although pages may differ from each other in specific appearance, and publications may cover a variety of subjects, basing them on the same grid establishes a common underlying structure, creating a sense of continuity. This similarity may be subtle or obvious, depending on the grid and its use, and it may not always be advantageous. But it is of interest here because it is an effective way of specifying a generalized idea of structure as opposed to a specific instance of it. It is a way of specifying the underlying rules which shape the appearance of a document.

The nature of the grid is another subject of interest. Again, the work of the "Swiss School" figures prominently: their grids, often derived from arithmetic and geometric progressions and functions, are particularly appropriate for computer use. The theory behind this is that people find
certain mathematically defined proportions more satisfying than others, a belief which extends at least back to the ancient Greeks, who used formulas such as the "golden mean," to which the Parthenon was proportioned.

Grids can specify more than position. One of the exciting things about computers is that they make it easy to create "grids" which control other parameters including color, size, and typeface.

Design systems also appear in the corporate world. Many companies commission graphic designers to develop guidelines for their company's graphics. This may involve devising grids for catalogs, manuals, stationery, and inter-office memos. It usually includes specifying proper ways of handling the company's trademark in various sizes and contexts, such as when it appears on products, brochures, or the company's...
Figure 4.6 Proportional systems in Greek architecture.
The Herman Miller company, like many other corporations, uses a set of "design standards." The purpose of these standards is to maintain a level of consistent quality in the company's printed material. As the graphic arts profession becomes increasingly electronic, it makes sense to represent these rules electronically as well, so that they can be integrated directly into the production process.
Figure 4.8 Packages created for Black&Decker by the design firm of RichardsonSmith. It is easy to recognize the design system employed here: the company logo is printed in the black band on top, the product name is located in the upper left, descriptive information always appears in an italic typeface, and so on. The space in the center is reserved for the product itself, which is shrink-wrapped onto the card you see here. Black&Decker uses over 4,000 packages for this product line.

delivery trucks. It may include many other things as well, but the basic intent, again, is to impart order to the large volume and variety of printed material the average corporation produces. These specifications usually take the form of a printed manual, which is then referenced by the corporate designer when creating some new form or document.

Packaging is another example of this phenomenon. A corporation such as Black&Decker, whose packages are presented here, wants the consumer to know that they are buying not just a drill bit, but a Black&Decker drill bit. Furthermore, Black&Decker makes a whole family of products, which the consumer knows are related, because when he or she looks up on the hardware store wall, the products look related, by
virtue of the rules which were used in the design of the packaging.

Why is it important that packages appear to be related? Perhaps it gives the consumer a feeling of confidence in the company, in that if the packaging appears well organized, the production must be too, and thus the product is likely to be of high quality. An attractive package might sell simply because it satisfies some aspect of our aesthetic sense, and we find that pleasurable. Obviously, much of the reaction to packaging is instinctive; perhaps evolution relied on the same responses to help our ancestors pick the ripest fruit from the tree.

The design of these packages follows a number of rules, which reflect the designer's assumptions about which pieces of information are most important to the consumer initially, how the information breaks down by subject, and so forth.

Richardson Smith, the company that designed these packages, ended up producing more than three thousand of them, including over one thousand for drill bits alone. It is easy to see that the packages appear to be of the same "family," in the visual sense. Couldn't a set of graphic rules be created which, when provided with the size of a package and the text, generate an appropriate design? It turns out that the designers involved with the project considered this idea. However, they found that in most cases it was faster and easier simply to take the computer instructions used to generate the text on a digital typesetter, copy them to another computer file, and change the text part of the file to the wording of the new package. The central question in automating processes like package design is, of course, whether the time and effort spent in developing the rules and software is less than that which would be spent using more traditional techniques. However, in the case of personal electronic news
## 4 The Role of NewsClip

### a. Basis

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Technique</td>
<td>31. Written</td>
<td>32. Drawn</td>
<td>33. Composed</td>
<td>34. Some other</td>
<td>35. Combined</td>
</tr>
</tbody>
</table>

### b. Colour

|---------------------|----------|------------|----------|--------------|--------------|

### c. Appearance

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Inclination</td>
<td>41. Upright</td>
<td>42. Oblique</td>
<td>43. Combined</td>
<td>44. Combined</td>
<td>45. Combined</td>
</tr>
</tbody>
</table>

### d. Expression

<table>
<thead>
<tr>
<th>Reading direction</th>
<th>11. From left to right</th>
<th>12. From top to bottom</th>
<th>13. From bottom to top</th>
<th>14. Otherwise</th>
<th>15. Combined</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form</td>
<td>31. Unmodified</td>
<td>32. Modified</td>
<td>33. Projected</td>
<td>34. Something else</td>
<td>35. Combined</td>
</tr>
<tr>
<td>Design</td>
<td>41. Unmodified</td>
<td>42. Something omitted</td>
<td>43. Something replaced</td>
<td>44. Something added</td>
<td>45. Combined</td>
</tr>
</tbody>
</table>

---

**Figure 4.9** Karl Gertner's "morphological box of the typogram."
*(Taken from Designing Programmes, by Karl Gerstner)*

Gathering, the effort of creating this software is always justified, because the content of each person's daily "newspaper" is unique, making traditional design methods impossible.

Karl Gerstner is a graphic designer who has worked with rules and programs in both computer and traditional media. He uses the mathematician Fritz Zwicky's notion of the "morphological box," in which each possible characteristic of something (such as size, color, and font in the case of a piece of type) is listed along with the various values which that characteristic might assume. In this way, a description of the entire range of forms possible for an object within a given system is described. The designer's job becomes one of selection, or choosing the most appropriate values for the various parameters.
Any number of computer programs could be written to do this, in which case the obvious question arises: on what set of criteria does the program base its choices? And the obvious answer: on those the designer supplies. This is what makes the computer an interesting tool for testing these criteria, or "approaches" to creating visual forms. A system built for this purpose should allow a designer to supply any criteria, logical or bizarre. There is no claim here that this is how people make art, or that the output of such a system would be art; it is a mechanistic approach. The potential of this type of system for simulating human decision making is an open question.

The role of programatically-generated forms will be discussed elsewhere in this thesis. The primary intent here is to show the relevance of traditional graphic design methodology in solving the problems of automatic generation of graphic formats for electronically-gathered information.
As a practical demonstration of this thesis, a computer program called NewsClip was developed in order to test the feasibility of programmed design of electronically gathered news and information. This program acts as an intermediary between an electronic news-gathering agent and the reader, taking a collection of news stories and other information and producing high-quality graphic output. Most of the development was done with the aim of producing hard-copy on a laser printer, although some experimentation has been done using a computer-based color graphics display, as well as a voice synthesizer, as will be described later. In explaining the system, it helps to have an end product in mind, so when appropriate, explanations assume output will be printed on paper, unless specifically mentioned otherwise.

The basic operation of NewsClip proceeds as follows. Once NewsClip has received the day's stories from the NetworkPlus, they are organized...
in such a way that NewsClip can begin to work with them. This organization is carried out with the help of a set of organizational rules. Depending on the rules provided, NewsClip is capable of producing wide variety of organizational structures. Although only one set of organizational rules is presented in depth, ways in which NewsClip can be used to create others will be discussed.

After this organization is established, another set of rules is used to assign the initial attributes of each part of the issue. In general, the "initial attributes" are those which can be established independent of page position, such as the size and typeface of headlines, body copy, and so forth. This set of initial-attribute rules, like the organizational rules, forms an independent module. Thus, NewsClip is capable of a wide variety of behavior, depending on the set of initial-attribute rules provided.

Next, the contents of the issue are mapped onto pages, again according to a set of rules. These formatting rules comprise another interchangeable module, providing for the ability to produce formats in one, two, or several columns, any page size, and so forth.

After this step has been accomplished, the resulting device-independent description of the page is translated into an appropriate set of instructions for physical output, such as on a laser printer. The only translator implemented in-depth in this project was one for translation of NewsClip's device-independent pages into the PostScript page-description language, resident on many laser printers, and an emerging standard which is in fairly common use. This process will be described later. Again, this translator forms an independent module, so that output on various devices can be realized.
The hardware used in this project was an IBM PC, equipped with an IBM YODA graphics board and color monitor. Hardcopy was produced on an Apple LaserWriter. Voice output was generated on Digital Equipment's DECTalk. These devices will be discussed later, where appropriate. All software was written in C.
Structures for Describing the News

The task to which NewsClip was initially put was that of designing a personalized "newspaper" for output, in 8 1/2" by 11" size, on a laser printer. Figure 5.2 shows both the index and a few pages from the interior of the paper. Each group of articles forms a "subject," and subjects are numbered. The interior of the paper is designed according to a three column grid. The right two columns are for text, while the left is reserved for visual material such as maps, diagrams, and photographs. NewsClip is capable of representing a large number of other designs, including ones with varying numbers of columns and non-sequential presentation of stories.

Once a series of stories is received from NetworkPlus, it must be organized in such a way so that NewsClip can work with it. Because of the way stories were grouped according to subject, this type of structure suggested itself:

![Diagram of NewsClip structure]

Figure 5.3

Figure 5.3 shows that the issue is divided into two main sections: the
index and the news. The index provides a quick synopsis of the day's stories, while the news presents each story in full depth.

In the index, each subject is numbered, and the stories comprising the subject are listed in the form of "summaries." A story's summary consists of a headline and the first sentence of the story. This works well, owing to the journalist's convention of summarizing stories as quickly as possible. The article's source, author, and so forth may also be given here.

The news section itself follows much the same structure as the index, except that stories are printed in full, and pictures or diagrams are presented when available.

The paper is a typical tree structure. It is organized according to groups: two main sub-categories, the index and the news section, are themselves composed of a number of sub-categories, the "subjects," which in turn are composed of stories. (These subjects may also contain more than just stories. Currently, they also include their subject number, and they may contain a list of keywords describing the subject as a whole.) The stories themselves may be composed of a number of items: text, pictures, diagrams, and captions.

The "Generalized Group"

Computationally, the various pieces of this tree structure are similar. Using the same representation for both a news story as the newspaper as a whole would simplify the program. A computational object was created for this purpose called a "Generalized Group," or "GG" for short. The GG can be viewed as having 3 main parts: a "description," a "parent," and a collection of "sons." Its implementation is relatively straightforward:
The Structure of NewsClip

GGDESC (gg)  
gets a description of what the GG is

GGPARENT (GG)  
gets the GG's "parent"

GGSONS (GG)  
gets a list of the GG's "sons"

The GG's "description" tells whether the object is a story, a group of stories, the news section of the paper, the index, or perhaps the whole paper itself. Keeping track of the GG's "parent" allows us to move back up the tree structure if necessary. If the GG has no parent, such as a GG which represents the paper as a whole, this is indicated by the special symbol "NULL." And of course, the "sons" of the GG may be objects of virtually any type, including other GGs, which gives the GG the flexibility needed to represent the various parts of the paper.

Figure 5.4  
This is a structure composed entirely of "GG"s. Each GG has a list of "sons" as well as a pointer to a "parent." Although only GGs are shown here, a GG's son may be an object of any type. This makes the GG a general-purpose grouping tool.

In terms of rules, the important aspect of the GG is the "description."
This is what allows the rules to identify what part of the paper a particular GG represents, and operate on it accordingly. The nature of a "description" is a subject of enormous interest to this project, and it will be discussed in greater detail later in this chapter.

Text

Effective representation of text in NewsClip is a matter of high importance, considering that many applications, such as creating a newspaper from the material provided by NetworkPlus, involve large amounts of text. Furthermore, the scheme used to represent text must be capable of satisfying two dissimilar requirements.

The first of these is the need to represent text in terms of its structure, as opposed to its appearance. For example, it must be possible to label the headline of a story as something distinct from its body, because the rules which establish the story's appearance will operate according to such distinctions. Obviously this labelling scheme should be as flexible as possible, due to the fact that information can be structured in any number of ways, depending on whether it is a computer manual, newspaper, package design, dictionary, poster, or advertisement.

The second requirement is the necessity of representing the physical characteristics of the text: its size, font, color, the maximum width of lines in paragraphs, leading (space between lines of text), method of justification in paragraphs (e.g. if it is flush-left with a ragged right margin, etc.), and so on. Primitives must be provided to allow the rules to set these attributes. The attributes themselves should be represented in a way which makes it easy to display the text on various output devices.
There are several ways of representing text, none of which is capable of fulfilling both these requirements satisfactorily. The first scheme of representation is common in the world of computer-typsetting, in which text is represented in the form of a computer text file, and its attributes, such as point size and font, are specified by inserting special codes into the file. Using this method, the attributes of any character can be established by reading through the file from its beginning, and noting changes in each individual attribute as they occur. Once the desired character is reached, the combined state of the attributes describes the character's appearance.

This approach is inadequate because, aside from the text itself, the only information the file contains is the instructions for printing the text. For example, in a file containing the text of a magazine, it would be impossible to differentiate between a headline, sub-head, the body text of a story, or a photograph’s caption. Another disadvantage of this method is that it modifies the file containing the text, by the insertion of printing instructions in the form of "formatting codes." The file often becomes illegible as a result. This is poor design, because two very different classes of information, the text itself, and the description of its physical characteristics, are combined into the same form of representation.

It is possible to separate these two classes of information by using other forms of representation. A simple "data structure" can be created which contains a value for each of the various attributes a piece of text might have, as well as the memory location of the text string itself. This representation has the advantage of leaving the text itself unmolested. It is also simple. A typical representation might look like this:
The structure is similar to one that might be used to represent a line, polygon, or other graphic object. Its failing, however, is that it defines attributes globally. This makes it impossible to change something within a text object. For example, a word in the middle of a paragraph can not be italicized. In this respect text is fundamentally different from the simple graphic object. A line or polygon has a fixed number of attributes, such as position, size, and color. The same is true for an individual character (which is really the level of "granularity" most appropriate for text).

However, when representing large bodies of text this approach is obviously inadequate. One would be unable to change to a bold font, or highlight a word by changing its color, for example. Representational schemes for text must be capable of taking its complex and sequential nature into account. For the purpose of this thesis, this method of representation also fails in the same way that traditional typesetting techniques do: it treats only the physical appearance of the text, ignoring its "semantic" structure.

Going back to the traditional typesetting techniques, it is possible to represent the semantic structure of a body of text by inserting a set of
codes into a text file in order to label the text in terms of its parts, (as opposed to its appearance). This is commonly done in manuscript preparation, for example, where "generic codes" are inserted by the author to mark headlines, paragraphs, etc. The Standard Generalized Markup Language, or SGML, is an example of one such system. The designer or typesetter creates a conversion table for these generic codes which can used to process the file, replacing the generic codes with the actual typesetting commands used to print the text.

However, if one wants to simultaneously describe the semantic structure of the text and its physical appearance, this technique would require both classes of codes to be inserted into the same text file. This makes things complicated. In the case of the "generic codes," rules regarding the ability of labels to nest and intersect need to be established (if this capability was desired). In addition, this method still has the drawback that the various types of information, text, formatting codes, and semantic labels, are simply clumped together in the same file. A file is not always the most convenient data structure to work with.

In addition, each program accessing the text file would be required to search through data of all three types: the text itself, the formatting codes, and the labels describing the text's semantic structure. Two of these three would almost always be irrelevant. Not only would this increase search time, but it would require the searching program to perform continual checks to recognize what type of data it was looking at at each moment, which in turn requires that a program which is looking for formatting codes, for example, know at least something about the characteristics semantic labels.

In order to resolve these problems, a new method of representing text
was developed which was more modular, providing separate representations for character data, semantic or structural labels, and formatting codes. This method of representation provided the desired level of functionality, as well as some new capabilities, which will be outlined shortly.

<table>
<thead>
<tr>
<th>text</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
</tr>
<tr>
<td>formatting codes</td>
</tr>
<tr>
<td>&quot;part&quot; labels</td>
</tr>
</tbody>
</table>

Figure 5.6

The text object is composed of three main parts. The first part is, appropriately, the text itself. The characters may reside either in the computer's memory or on disk. Then there is the part of the text which describes its "semantic" structure. In addition, provision is made for recording the formatting instructions which describe the text's physical appearance. In this representational scheme it is usually not necessary to touch the text itself, as the formatting and semantic information are stored separately, referring to specific characters by their location in the text file or string. As the implementation of the string part of the text object is relatively straightforward, we will proceed directly to the description of the other parts of the text object.

**Physical Appearance**

The second part of the text object is the formatting codes. These are the
The Structure of NewsClip

codes which describe the text's physical appearance. Hereafter, they will be referred to as "txc"s, short for "text codes." A "txc" looks something like this:

```
<table>
<thead>
<tr>
<th>t x c</th>
</tr>
</thead>
<tbody>
<tr>
<td>descr</td>
</tr>
<tr>
<td>data</td>
</tr>
<tr>
<td>index</td>
</tr>
</tbody>
</table>
```

Only one Michelangelo painting is known to exist...

Figure 5.7 A diagram of the txc. The numbers along the bottom give the position of characters in relation to the beginning of a file or string.

The txc's "description" tells what type of txc it is. For example, txc's exist to describe text characteristics such as font, point size, leading, maximum line length, justification method, color, line breaks, inserted spaces, and so on. The "data" part of a txc holds values of various types. If the txc represents a font change, "data" contains the name of the font. Likewise, if the txc indicates a change in size or color, "data" would specify specifically what size or color. "Data" is capable of holding information of any type. "Index" refers to the location of the first character which the txc modifies. Once an attribute is set by a txc, it remains in effect until another txc changes it. In this respect, a txc is similar to the traditional computer typesetting code.

Txcs can be formed into lists. This enables virtually any attribute to be set or changed anywhere in the text object, which provides a high degree of flexibility.
Only one **Michelangelo** painting is known to exist...

There are a few drawbacks to the txc approach which should be mentioned. Unfortunately, the flexibility of txcs also presents some problems in the form of ambiguity. When the user specifies a change in leading in the middle of a line of text, for example, does that change manifest itself immediately, with the text dropping or rising in mid-line, or does it take effect after the next line break? What about cases when both maximum line length and justification are changed repeatedly and at random? The reader who follows these changes through (to their effect on the appearance of the text) will understand the difficulty they present. However, this is not entirely the fault of the txc method, but also of other factors, which will be discussed later.

Another problem with the txc approach is that changes are specified with reference to specific locations in the character string. Therefore, when the string is edited, all txcs at and after the editing locations must be adjusted accordingly.

And finally, because txcs are stored in a list, establishing the attributes of any character in the string requires searching the txc list from the beginning, keeping track of attribute changes along the way, until...
character in question was reached. Although this process is hidden from the user, and from the calling routines, it could be slow when dealing with large numbers of txcs. This problem would be alleviated by changing the internal representation of the txc list from a one-way to a two-way linked list. This would allow the list to be searched backwards, resulting in shorter search times. The most recent txcs, being the ones responsible for the attributes of the character in question, would be encountered first. This representation was not implemented because it is more complex, and search times did not prove to be a problem in the cases tested. However, it may be implemented in the future. The interface to the user, or the calling routines, as well as the functionality, would be identical in either case.

The list of txcs can reside either in memory or on disk, so that memory space can be conserved when a text object is not in immediate use.

**Semantic Structure**

Although the "txc" was used successfully to describe the physical appearance of text, txcs contain no information about the text's "semantic" structure. This means that there is no way to determine which part of the text is the headline, and which is the story, both pieces of information needed by the rules which establish the characteristics of each. This need is filled by the third main element of the text object, called the "text label," or "txl." The purpose of the "txl" is to label some portion of text as being a certain "thing."
The Structure of NewsClip

Only one **Michelangelo** painting is known to exist...

![Diagram of the Structure of NewsClip]

The "description" describes the nature of the thing being labelled. For example, if the thing is a headline, it might contain the symbol "headline." The "fc index" marks the first character to which the description applies, while the "lc index" marks the last character which can be considered part of the label. A typical Newspaper story might be tagged like this:
5 The Structure of NewsClip

Figure 5.10 This diagram illustrates a labelling scheme for a simplified newspaper story, with the words along the top representing the txls which delimit the various parts of the text. These txls would probable be stored in a single list. By picking any character from the text, it is easy to tell if that character is part of the "dateline," the "body," or both. This scheme allows text to be organized in ways which have meaning to people, but which machines can understand.

Like the txcs, txls are "modular" in the sense that they do not invade the text file, and in fact, the same text may be used by multiple sets of txls, as in the case where a news story is presented in one place, and its summary is presented in another.

Figure 5.11 This diagram illustrates the fact that a single string of text can be referred to in a number of ways, creating in essence two separate text objects (a and b, above). In this scheme, a text object is really a "reference" to a text string as much as it is the string itself.

Another advantage of txls is that they nest and overlap quite easily.
Consider this case, announcing three movies:

"Rocky IV" Friday 8pm 10-250
"Rambo" Saturday 7pm Kresge
"Barbarella" Sunday 7pm Kresge

(Student Council Film Series for next weekend)

Figure 5.12

Again, the words along the top represent txls. Notice that this information can be indexed in two ways. For example, 8pm is part of the information about "Rocky IV." This is made apparent by placing it in the same column. It is also a member of the class "time," and this is shown by placing it in the same row as the other times. Although this is not the most sophisticated example or design, the point is clear: graphic attributes can be used to elucidate the structure of information, and the txls are the key to defining that structure in the NewsClip system. Because txls can overlap, information can be deduced about class membership of individual words or characters and intersections of classes themselves.

The advantage of txls is that they allow one to view a text object in a number of different ways: Text can be manipulated as a whole, or some part of it, say a headline, defined by a txl, can be treated as its own,
5 The Structure of NewsClip

separate object, passed to functions, referred to by address, and so on. And since txls may nest and intersect, there is no limit to complexity of structure, or the number of "parts," a text object can be described as having. It is a "database" approach, in that the data contained in the text object can appear to have any number of structures, depending upon how it is "looked at."

Another advantage of this representation is that the txl indexing scheme is similar to the txcs'. Therefore, rules which are looking at txls, in order to ascertain the text's structure, and creating txcs, in order to set its physical attributes, can perform both operations easily.

The biggest advantage of the "labelling" method is that it seems to approximate the way graphic designers think about information design. Although it is difficult to say what they do in all instances, typically a graphic designer will divide a body of information into classes, assigning a set of graphic attributes to each class, the goal being to make the underlying structure of the information clear by formatting it according to its perceived structure.

Of course, the internal representation of txls suffers from some of the same problems as the txls: it is list-based and so must be accessed sequentially. When the text itself is edited, the txls must be adjusted. However it should be noted that editing is not too much of a concern at present, as NewsClip is primarily formatting information provided by an electronic agent, and so large changes in the text by a human editor do not occur.

It is fudging the truth to say that txcs and txls do not invade or modify the original text file. Although this was an original goal, it is obviously not possible using the current representation as there are many cases when
text files contain formatting information which must first be filtered out, if the txl labels are to be accurate. For example, stories taken from *Dow Jones News Retrieval* contain line breaks and other information which makes no sense when the article is re-formatted.

Although the txl labelling the headline resides outside the text file, a line break contained in the middle of a headline must be removed before further processing can occur. In fact, NewsClip usually filters a text file to the point where it is simply an uninterrupted stream of characters, with all "structural" information being carried by the txls.
Figure 5.15 If txls can refer to a sequence of character strings, as opposed to a single string, then they can be used to label any text, no matter how jumbled or mixed with formatting instructions. In this way, the original goal of leaving the original text file unmodified can be achieved, at the expense of a more complicated representation of the txl.

More complicated schemes of representation would be able to get around this. For example, instead of containing only one "fc index" and one "lc index," a single txl might contain a whole list of them, in which case it would be capable of specifying an entire sequence of strings in a text file. This sequence, taken as a whole, would then serve to identify the parts of the single "thing" referred to by the txl. For example, a headline with a new line in the middle of it would be referred to in two parts, effectively eliminating the newline code between them, and thus leaving the original file unchanged.

It is easier to modify the text file by simply removing the newline. However, leaving the original text files in a form which is easily readable by people, although this is not critical to NewsClip, may prove important in other applications using a txl-like labelling scheme. For example, in a graphic designer's workstation, where manuscripts are arriving electronically, it would obviously be advantageous to allow the designer
to label the various parts of documents, building an entire description using txls, without changing the document itself. Another advantage of more sophisticated labelling techniques is that it would become possible to work unobtrusively with files already containing other formatting information, appropriate to an entirely different system, such as a word processor or typesetter.

The Text Object Summarized
The text object satisfies the requirements as initially stated, providing representations for the string itself, for formatting information describing the text's physical appearance, and for descriptions of the text's "semantic" structure. The representational schemes developed are flexible, in that any attribute of the text may be changed at any time, and text may be labelled so as to reflect structures of arbitrary type and complexity. In addition, the representation is "modular," in the sense that
the same text file may be shared by any number of text objects, each perhaps with different representations of the structure of the information within the file and different instructions for formatting it graphically. Thus a text object is a combination of a number of things. Its appearance depends not so much on the string itself, or the text's specific structure, but on the way that structure is labelled, the way it is looked at.

As confirmation of this, in NewsClip, the rules which set many of a text object's attributes seldom even look at the text itself, which is often left unexamined on the hard disk. Instead, the rules examine only the text's structure, in the form of descriptions provided by the txls. The txls provide all the information necessary to modify the text's attributes according to a set of rules.

NewsClip provides a uniform set of primitives for making, deleting, accessing, and modifying text objects. Consider the following samples:

```plaintext
    tx = tx_mk(string, txcs, txls)
  error = tx_del(tx)

    font = tx_font(tx, position)
    size = tx_size(tx, position)
    color = tx_color(tx, position)

  error = tx_setfont(tx, position, font)
  error = tx_setsize(tx, position, size)
  error = tx_setcolor(tx, position, color)

    txl = tx_nexttxl(tx, position, description)
  error = tx_addtxl(tx, fc, lc, description)
    list = tx_whatis(tx, position)
```
The first procedures listed are for creating and deleting text objects. The next set of procedures allow one to query the state of any particular attribute at any point. Following this is a set of procedures which allows one to set the state of any attribute at any point. The last group of procedures deal with txls. "tx_nexttxl()" allows one to search for a txl of a given description, starting from a specific position. If one wanted to find the first txl of a given type, one could set the argument "position" to equal 0. "tx_addtxl()" allows one to label a portion of text, starting at the point indicated by "fc" ("first character"), and continuing through point "lc" ("last character"). Many other primitives are provided. "tx_whatis()" returns a list of tags describing what a particular character is, e.g. if it is part of the headline, the body, a keyword and a paragraph, and so on. For complete descriptions, see the project's technical documentation.

One problem encountered with text objects should be mentioned. That is the "sequential" nature of attributes. As described previously, all txcs are indexed to start at a specific character. Any number of txcs may be indexed to the same character, and thus can be thought of as taking effect "simultaneously." This is useful, for instance, when one wants to highlight one word in a body of text by changing both its font and color. A txc describing the font change is created, as well as one describing the color change, and both are set to take effect at the beginning of the desired word.
Only one Michelangelo painting is known to exist...

Figure 5.17

No problem so far. However, suppose one wants to break a line of text at a certain point and change the leading simultaneously, thus immediately increasing the line spacing. If one breaks the line of text first, and then changes the leading, the leading change might not take effect until the next line break, depending on the method employed for processing txcs when printing. NewsClip maintains a history of attribute settings internally to account for this. Thus, the user must be aware that setting attributes in various orders can change the final appearance of the text.

In fairness, it must be said that traditional typesetting techniques, also being sequential, exhibit the same behavior. However, it is not so troublesome there, as the typographer is usually dealing with a human-readable text file directly, and so can see the order of formatting instructions easily. At the lowest levels, programs manipulating txcs must always take into account the fact that changing their sequence could change their appearance.

When implementing a program in NewsClip, a decision must be made as to when to switch from the "GG" representation to that of the individual
text object. Because the text object is capable of changing any of its attributes at any time, and because it provides means for indicating internal structure in the form of txIs, the text of an entire newspaper could conceivably be composed of one, large, text object. Sections, subjects, and stories could all be delimited by txIs. On the other hand, the text of each story could be a single text object. Or each headline and paragraph in each story could be a separate text object. NewsClip is capable of any of these representations. In order to determine which is best, one should decide what size of text object forms the most convenient "conceptual unit" in a particular application. In the case of a newspaper, this might be a "story." From there, GGs can be used for the purpose of grouping these text objects into higher-level objects.

Labels
The reader may have noticed that both "GG"s and "txl"s contain information which has been called a "description," however the specific nature of these "descriptions" was never really explained. In developing the software of NewsClip, the option to use many techniques for object description was left open. In the current implementation, it turns out that both objects use exactly the same method of "description." The role of the descriptive label in NewsClip is an extremely important one; it allows objects to be recognized, classified, and operated on by the various sets of rules. Thus it would make sense to want descriptive labels which are as useful as possible in describing the structure of information.

The simplest way to identify something is, obviously, to give it a name. This name might take the form of a number, sequence of letters, or a
graphic icon; almost any symbol will do. In this way, objects with the same name can be identified as belonging to the same class, and treated accordingly. The same set of graphic rules, for instance, would probably be used to format all of the "chapter-headings" in a text book, while a "paragraph" would be treated differently than a "photo-caption."

If one has enough symbols, any object in a system can be given a unique name, and thus distinguished from its neighbors. Thus, in place of having just a "heading," one could have a "chapter-heading," "section-heading," and a "sub-section-heading." However, even with a large number of symbols, single-name systems such as these are capable of providing only single-level grouping. The problem with this is that objects may have only one of two possible relationships: they are either "the same" or "different."

More complex naming schemes allow subtler distinctions to be made between objects, and thus more sophisticated conclusions to be drawn about them. Simply increasing the number of names used to describe an object increases the variety of groupings which is possible. For example, in looking at the "chapter-heading," "section-heading," and "sub-section-heading" labels mentioned previously, it should be obvious that despite their differences, these objects share one characteristic in common: they are all "headings." A more sophisticated naming scheme enables this characteristic to be recognized by a computer program. This, in turn, allows graphic rules to do more reasonable things. For example, the headings mentioned previously could all be given the same typeface, but assigned sizes which decreased as the headings grew more detailed, thus reflecting their "likeness" in one aspect (the fact that they are headings), as well as their "difference" in another (the fact that each
represents a different level of detail).

The desirability of multiple names can also be seen in analogy: suppose our vocabulary was such that it was possible to replace each English sentence with a single, unique word. This method of representation would be undesirable, due in part to its inefficiency, but primarily because the lack of individual words to describe concepts would prevent us from making connections, recognizing similarities, or using metaphor.

In general, a multiple naming scheme can be implemented for GGs, txcs, or any other object simply by replacing the single name description with a list of names. Then a name recognition package must be devised which provides the primitives to create "object descriptions" and means to test objects against these descriptions. The descriptions themselves contain names and logical operators. In English, typical description might read like this: "the object is a heading, and the object is not a chapter-heading." This can be diagrammed as follows:
Many rules, being primarily "if-then" statements, can use an "object description," along with other boolean operators, in the "if" part of the rule. If the object tests true against the "object description" in the "if" clause, it can be assumed to be of the desired class, and the steps specified in the "then" part of the rule can be carried out.

**Document as Data Base**

The various labelling schemes described so far allow documents to be viewed as "data bases," in the sense that the text of the document can be organized in a number of ways, various types of classifications can be set up, such as "headline," "story," and "news-section," and the objects represented by these classifications can be accessed and manipulated as if they were discrete. From this viewpoint, the actual text of the document assumes minor importance when compared to the labels which define the document's structure. In fact, this "data base" approach can be carried down to the lowest level: characters themselves. In this case, letters, words, and sentences would all be "objects" defined by "labels," just as "headlines" and "paragraphs" are now. When this approach is taken, we see that many of the things we traditionally think of as "characters," such
as word spaces and newlines, are really just delimiters, and so are no
longer necessary, words and sentences already being delimited by the
data base itself. The data base serves as the complete description of the
document. In fact, since the "sentence" is an object which is really just
composed of "words," which in turn are simply composed of "letters," and
since there are only a small number of these letters, the entire body of any
newspaper can be described adequately by a single alphabet ---
provided one has a data base which indexes it properly.

Although this may be taking things to the extreme, representing the
document as a "data base" has a number of advantages in electronic
environments. For example, using txls, a single document can be indexed
in numerous ways, depending on whether one wants an abstract, or
wants to read the document in its fullest level of detail, or wants to read a
"version" of the document which treats one subject in depth while only
summarizing its other parts. In this sense, the txl is the electronic
equivalent of the "highlighting marker" often used by students to mark
relevant material in text books (which suggests another use for electronic
labelling using txls).

As mentioned previously, labelling schemes such as these would be
handy to have at a graphic designer's workstation. Electronically-stored
documents could be marked up with various labels, which could then be
used as a basis for assigning graphic attributes. Once various parts of the
document were tagged (in whatever terminology the designer felt was
appropriate), he or she could say "Change every object which is a
'chapter-heading' to 14 point Helvetica Bold," and the system would. This
allows numerous designs to be explored quickly. The designer could tag
portions of the manuscript for future reference. For example, a label could
be created called "subject-to-further-author-revision." This label could then be applied to the various areas of text which the designer knew the author was thinking of changing. Later on, these portions of text could be called up easily.

The ability to create any arbitrary label and to nest and intersect labels freely allows people to tag and annotate documents with symbols of their own choosing and in a way which seems fairly natural. This means that the document can be structured using parts which have meaning in the human terms, but which the computer can understand and work with, so that automated formatting programs can begin to exhibit more intelligent, sophisticated behavior.
NetworkPlus

NetworkPlus is an experimental news-gathering system developed in the Electronic Publishing Group, at the Media Laboratory, MIT. This is the program which collected news and delivered it to NewsClip for formatting and printing. It is a good example of an electronic "agent," and is similar in function to those discussed in previous chapters. Therefore, it is appropriate to begin the system description with a brief explanation of the NetworkPlus system.

First, NetworkPlus connects to the Dow Jones News Retrieval Service and gather the day's top stories. Dow Jones News Retrieval is a news...
6 The Operation of NewsClip

database which can be accessed on a personal computer via a modem and telephone lines. Each of the stories collected from *Dow Jones* is then passed through a "noise word" filter, which removes common English words, such as conjunctions and pronouns. This leaves a collection of the more meaningful words of the article, which are then counted and saved in a list, the most commonly occurring word first, with the rest of the words following in decreasing order of frequency. This list of "keywords" then serves as the basis for a search through *NEXIS*, another on-line information service, which contains sources such as *The New York Times*, *UPI*, *Reuters*, and *Tass*. For each of the thirty or so stories collected from *Dow Jones*, a group of related stories is gathered, by keyword search, from *NEXIS*. This system appears to work fairly well. News is also received directly from satellite, although this source was not incorporated into *NetworkPlus* at the time this thesis was implemented.

*NetworkPlus* also monitors television news broadcasts, by reading the English-language closed-captioning provided for the hearing-impaired. When doing this, the program is constantly looking for matches between the high-frequency words derived from the *Dow Jones* stories and those appearing in the close-captioned text of the newscast. If *NetworkPlus* can determine that a television news story is related to one of the *Dow Jones* or other stories, the text of the television story is saved and a frame of the newscast is grabbed and stored. These are included in one of the news groups as a related story. Until now, *NetworkPlus* has established a television news story's relevance by counting the number of words in the text of the television story which have matches in the keyword list derived from *Dow Jones*. When this number exceeds a certain threshold, such as 4 or 5, it is assumed that the television story is related.
and a frame of the newscast is grabbed.

Figure 6.2 The frame-grabbing algorithm originally employed in NetworkPlus. In the example presented here, the system has recognized 4 words in the text provided by the close-captioned broadcast. Once this occurs, the frame grabbing process is triggered. This approach proved to be inadequate because the resulting pictures had the appearance of being snapped at random.

One problem with this technique is that once the relevance of the TV story is established, the image acquisition from television news is initiated without regard for what happens to be on the screen. This often results in an image which appears to have been grabbed at random. A picture of Dan Rather announcing a Soviet Nuclear accident does little to elucidate the meaning of the story.

Several approaches to solving this problem have been discussed. One possible method includes snapping a picture every time the program detects a change in speaker. Close captioning often indicates speaker-changes by inserting special symbols in the close-captioned text. Another possibility involves the illustration which usually appears behind the newscaster at the beginning of a story. When this graphic appears, if its edges can be found through use of edge-detection techniques, the graphic itself could be grabbed as an illustration of the story.
An approach which appears promising is one wherein a grid of points is constantly sampled from the television image and analyzed for changes in color. When a certain percentage of the points exhibits a major color change, say 80% of them experiencing a color shift of 20%, it can be assumed that the newscast has switched to another shot or piece of footage, and this image is then grabbed. The proper number of points to sample, the necessary magnitude of color change, and the percentage of changed points required to trigger the image acquisition process are unknown. However, a possible advantage of this method is that large changes in the sampling grid would probably be fairly reliable indicators of editing cuts, which in turn result from decisions made by human experts (editors). News stories consisting of newscaster, on-the-scene reporter, and live or file footage are often fairly well spliced together, and the series of these images, when keyed to text, would provide the reader with a good idea of the flow of the television news story.

Other possibilities for image acquisition include the lookup of images stored on videodisc according to keywords. The videodisc is capable of
storing over 50 thousand maps, photos, and the like.

Figure 6.4

One problem with grouping stories together involves cases of multiple relationships between stories. That is, one news story could be related to more than one group of articles, in which case it becomes unclear where it should be placed. For example, a story on a nuclear accident in the Soviet Union might be grouped with other stories on nuclear power, or with those on the Soviet Union. The Thinking Machines document retrieval method (described at length in a previous chapter), or some similar system, would be of use here because it not only establishes a relationship between two news stories, but it quantifies the closeness of that relationship in terms of a "score." This score can then serve as the basis for determining where to put a story that may be related to a number of other story groups.

All these methods are made easier when the broadcast is first recorded, rather than processed as it arrives, for then time becomes a less critical factor, and multiple passes and filtering of the broadcast are possible.

*NetworkPlus* provides no ranking of importance of stories or story groups. Whenever such a ranking was assumed or needed by NewsClip, it was simulated.
Reading the Output of *NetworkPlus*

When output arrives from *NetworkPlus*, it must be read, understood, and organized into a form with which the graphic rules can begin to work. In order to perform its functions correctly, NewsClip assumes that *NetworkPlus* has provided a number of "subjects." Each of these "subjects" contains one or more "stories." An individual "story" can be composed of text (in the form of one or more text files), images, and graphics.

NewsClip must know how to deal with information from different sources. For example, *NetworkPlus* might be delivering files gathered from *Dow Jones News Retrieval, Nexis*, or *The ABC Evening News*, each of which has a different format. Therefore, a separate "translator" is needed to read each one's files, locate the story parts (such as headlines, dateline, byline, and paragraphs), and remove unnecessary formatting information.

Marcos Wants His Old Job Back;
Reagan Backs Aquino Government

---

HONOLULU - The U.S. is stressing its support for the fledgling Philippine government of President Corazon Aquino.

President Reagan made a courtesy phone call to Ferdinand Marcos Sunday. The deposed leader reportedly
pleaded for help in getting his old job back.

A U.S. official said Reagan "said nothing to encourage (Marcos)."

In Manila, President Corazon Aquino Monday thanked Reagan for reportedly refusing to back Marcos' challenge to her government.

Figure 6.4 A typical news file, as it comes from the wire service.

Figure 6.4 shows the appearance of a typical file, in this case from Dow Jones News Retrieval Service. Notice that a lot of formatting information already exists: the headline is separated from the body of the story by three dashes, each sentence is indented, lines are broken at appropriate lengths, paragraphs are separated by a newlines, and so on. This information was added by Dow Jones to make the story legible. NewsClip will remove it and replace it with its own, more sophisticated, formatting instructions.

The translator may simplify the file before attempting to locate the various parts of the story. The Dow Jones translator begins by removing all occurrences of more than one space, thus eliminating the indentation at each line. Then it replaces multiple occurrences of
carriage-return/line-feed combinations with a special symbol. This symbol marks paragraphs. Next, all single occurrences of carriage-return/line-feed characters are replaced by spaces, thus removing all line breaks, which must be re-calculated later anyway, due to changes in font and format. The program must also remove hyphenation inserted at word breaks, such as "Phil- lippine" (on the second line of the first paragraph of figure 6.4), taking care not to change "valid" hyphenations elsewhere. The translator also removes the 3 dashes under the headline. Notice that Dow Jones News Retrieval uses the same characters for opening quotes as for closing them. The Dow Jones translator changes all Dow Jones' open quotes to "true" open quotes. Finally, the file is "cleaned up" in miscellaneous other ways, such as by removing unnecessary spaces and newlines from the beginning and end of the file.

The next thing the translator does is create a text object, with this file as its source of text. The translator then reads the file, recognizing headlines, the body, and paragraphs, and creating txls along the way to refer to each of them. These txls are added to the text object.

Thus, the structure of the story is established and tagged. Unlike speech recognition, locating the more common "parts" of these documents is often relatively straightforward. This is because many of these parts, such as paragraphs, are delimited by (more or less) standard combinations of characters, such as spaces and newlines, making them easy to recognize. In most cases, rules people use to recognize headlines and paragraphs are fairly simple. Thus it is relatively easy to program a computer to make the same distinctions.

The next operation the Dow Jones translator performs is that of "shrinking" the file, in order to remove any other unnecessary information,
such as the markers which were inserted to identify paragraphs (this is not always necessary, but the translator does it in any case). This results in a file which is essentially one continuous string of characters. Of course, the txls, which are pointing to specific locations within the file, must then be changed. This will be described briefly, because the same process is also performed when the text part of a text object is edited.

In this illustration, the horizontal line represents the sequence of characters comprising the text file. Each letter on the top represents a txl. Remember that a txl has an "fc index" which points to its first character, and an "lc index" which points to its last. These indexes are represented by arrows. The file starts at the left, and ends at the right. $c_1$ refers to the position of the first character removed from the file, while $c_2$ refers to the position of the last character removed. $s$ refers to the string delimited by $c_1$ and $c_2$. In this explanation we assume the existence of three primitive operators: $fc(txl)$ gives access to a txl's "fc index," $lc(txl)$ gives access to its "lc index," and $len(s)$ returns to the string's length. When removing a piece of text, there are only six possible cases to consider:
6 The Operation of NewsClip

```
taxi a: no change in the taxi is necessary
taxi b: \( lc(b) = lc(b) - \text{len}(s) \);
taxi c: \( lc(c) = cl - 1 \);
taxi d: taxi d is deleted, as it no longer points to anything
taxi e: \( fc(e) = c2 + 1 \);
taxi f: \( fc(f) = fc(f) - \text{len}(s) \);
\( lc(f) = lc(f) - \text{len}(s) \);
```

When adding text, there are only three possible cases to consider.

Examine the following illustration:

```
Figure 6.6 Adding text.
```

```
taxi a: no change
ntaxi b: \( lc(b) = lc(b) + \text{len}(s) \);
taxi c: \( fc(c) = fc(c) + \text{len}(s) \);
\( lc(c) = lc(c) + \text{len}(s) \);
```

Case b, however, is problematic because the text inserted is thereafter referred to by any taxis which happen to straddle the insertion point. This may not be what the editor intended. If not, straddling taxis would have to be broken in two parts: the first referring to the portion of the taxi prior to the insertion, the second referring to that portion lying after the
inserted text.

In the future, increasing numbers of information sources will also provide generalized markup of their documents, labeling things like headlines and paragraphs, so that much the process of understanding the structure of information contained in text files will be simplified. SGML, or the Standard Generalized Markup Language, is an example of the type of marking system which would make this possible.

Assigning Initial Attributes
Once the entire structure of the document is created, NewsClip can begin to establish the graphic attributes of the various parts which, when taken as a whole, describe the appearance of the document. In terms of text, NewsClip breaks the process of formatting the document into two steps: 1) assigning initial attributes to the text, such as font, size, leading, color, justification, maximum line length, and so on, and 2) mapping the text onto the actual page.

This can be called the "galley" model of page preparation. It is similar to what many graphic designers do when they determine the appropriate parameters for a body of type (such as font, size, leading, etc.), when ordering it from a commercial typographer. The typographer delivers the typeset text to the designer in the form of "galleys": long pieces of paper, usually with a single column of text running down them. The designer takes the galleys, cuts them up, and pastes them into position on the page. The "galley model" attempts to simplify the process of formatting a document by breaking it into these two main steps. If a set of rules can contain the knowledge necessary to specify the attributes required to
make a "galley," then this galley can be shipped to another process which knows how to place it on the page.

The reader may have noticed that the descriptions of text objects so far contain no mention of position as an attribute. This is because, initially, text objects are defined independent of their position on any page, and decisions about position are made later after the initial attributes are set. This is because a text object which spans many pages, or fills a number of columns will obviously have more than one position. This problem will be treated later.

One of the attributes of the text object is the line break. In this system, a line break is treated as an attribute like any other. If desired, a text object can be broken into lines at the beginning stages of formatting, and these line breaks will be stored as part of the text object's description. Any time the text is displayed thereafter, line breaks do not need to be re-calculated, thus speeding the display process. Of course, line breaks could be calculated at a later stages just as well, such as when "flowing" a body of text into a rectangular area.

One disadvantage of the "galley" method is that because page placement is considered at a later stage, it is more difficult to do subtle copyfitting tricks, such as making changes in leading or point size in order to get a body of text to fit within a prescribed area.

Another problem with the current methods used to format *NetworkPlus* is that they are too sequential. One of the things designers can do is to look at a fully formatted page, recognize problems in it, and go back and make changes. The ability to analyze and revise takes much of the load off the original decision-making process by allowing it to be fallible. Even expert designers make mistakes. Expecting a set of rules to make no
mistakes is impractical.

Donald Knuth's \textit{TEX} does better in this respect, in that it examines a whole paragraph before finalizing line breaks. It uses a notion of "badness," calculating the amount of "badness" inherent in various options and choosing the one which appears best. For example, having to hyphenate a word increases a paragraph's "badness" by a small amount. Knuth explains it best:

"When the end of a paragraph is encountered, \textit{TEX} determines the 'best' way to break it into lines. In this respect, \textit{TEX} gives better results than most other typesetting systems, which produce each separate line of output before beginning the next, because the final words of a \textit{TEX} paragraph can influence how the lines at the beginning are broken. \textit{TEX}'s approach to this problem... requires only a little more computation than the traditional methods, and leads to significantly fewer cases in which words need to be hyphenated."

This approach is more like that taken by the expert typographer: he or she looks at a paragraph after it has been set according to some algorithm, and, when necessary, changes it to improve its overall appearance.

\textbf{Mapping Information Onto Pages}

Now it is time to place the stories, pictures, and graphics onto pages. In \textit{NewsClip}, a "page" is a rectangular area with the origin at the lower left hand corner, and a width and height which increase to the right and up, respectively. A page can keep track of various things, such as a page number, a "current point," and the set of pages to of which it is a member.
A page also maintains a list of the objects on it, including a bounding rectangle for each, in such a way that by looking at a particular point on a page, it can easily be determined if that point is occupied, and if so, what object or objects occupy it. To be more precise, pages keep track of the way in which objects are mapped onto them. For example in the case of text which spans more than one page, the page must know what piece of the text it holds and the position and extent of that text. The same is true for an image which covers two (or more) pages.

In order to represent these types of relationships, NewsClip implements an object called a "chunk." Basically, a chunk is a connecting device which allows a piece of some object to be mapped to a page. From an object's point of view, a chunk contains a reference to a particular page, as well as a rectangle which describes the space which the object occupies on that page. From the page's vantage point, a chunk refers to a specific object, defining which portion of the object is mapped to the page, and giving the physical extent of this mapping in terms of a bounding rectangle.

For example, a text object may have a long list of chunks, one for each place it appears. A text object, or part of one, may also appear simultaneously in two or more places, through the use of multiple chunks. (This also holds for images.) The page, in looking at a single chunk, can access the text object, determine which character is the first to appear on the page, which is the last, and what the extent of the text is, in terms of a bounding rectangle.
This system is quite flexible. For example, if one was examining a picture, one could look at the picture's chunk and, from this, determine the page which the picture occupies. One could then access this page's list of chunks, obtaining a complete inventory of all objects on the page, as well as the position and extent of each. Suppose one chose that object which was closest to the picture, and that this object turned out to be text. One might then wish to look at it, determining what it was as well as what other pages it occupied. Furthermore, by examining its txls, one could tell where each headline, subhead, and paragraph was located, and what other objects were on each of those pages.

People can see. Computers can not. The ability to visualize is obviously of great value in allowing human experts to make designs and evaluate them. Allowing a machine to "see" is important if one wishes it to comprehend the world well enough to make "intelligent" decisions. NewsClip seeks to create a computational model of objects and pages.
which is "well connected," meaning that it is possible for a process to look at an object, and in doing so, glean information about proximity and nature of each of its neighbors. Through the use of "chunks," any object gives access to a page, and thus other objects, and in so doing gives access to the entire document. This results in a large body of information which is available from any level in the system.

One problem with the "galley model" is that it separates the representation of one set of textual attributes (font, size, color, etc.) from information about text's position, which is stored inside the chunks. This distinction is not always helpful. In retrospect, a better approach might be to store all information about text's attributes in one place: the chunks themselves. This would make it more natural to consider all attributes at the same time. It would also make it easier to change various attributes. For example, text which is one size and leading at the beginning of an article might change to another size when it continues at a page somewhere in the back of a document. Although this is possible in the current system, storing all formatting information in the chunk itself would make it easier to do, with less bookkeeping. In this sense, a chunk would be an "instance" of a portion of text, much like it is now, except that it would be the chunk itself which would define all the text's attributes when it appeared on the page. The txc method of attribute coding could still be used, only txcs would be attached to chunks instead of text.

**Free Space**

A page keeps track of the things which are on it, as well as the extent of each of these things, with the result that a page knows where the "used"
space is. Suppose one wants to place something new on the page, how
does one determine all of the places this new thing might fit? And once a
list of possible positions is found, how is the best chosen? Where is a
"good" place to put a picture?

As mentioned earlier in the discussion of the "galley model," NewsClip
divides the process of formatting text into two main parts: establishing the
text's initial attributes, and mapping this partially-formatted text onto the
page. This second stage, placing things on the page, is in turn divided
into a number of sub-steps. The page must be searched for likely places
to put something. If none can be found, another page must be examined
(or created). Once a list of candidates is found, some process must be
capable of picking the "best" among them. Then the object must be placed
there. In the case of text, the object must then be examined, and if there is
still more to be processed, these steps must be repeated until the text is
completely formatted.

The first problem to be tackled is that of finding "free space." A page
can be queried for "used space," in which case it returns a list of
rectangles representing the position of each object on the page. However,
querying a page for "free space" is more problematic.

![Figure 6.8](image)

The problem of finding free space on the page arises out of the
fact that this free space can be represented as any number of
rectangular areas.
Figure 6.8 shows a page with one item on it. The free space on the page can be divided up into a list rectangular areas fairly easily. However, this does not help much, as these rectangles will be of varying size and position, with no guarantee that their dimensions will be appropriate to the task at hand. If one wants to place a new object on this page, it is often necessary to determine which free rectangles are connected to which, so that a contiguous space of the desired area can be found.

In order to solve this problem, NewsClip defines an object called a "grid." In NewsClip, a grid is simply a synonym for a list of rectangles which describes an area (or group of areas) in which it is possible to put something. For example, the NetworkPlus hardcopy is designed on a three column grid like this:

![Figure 6.9 A typical three column grid for text.](image)

If one wanted to insert text onto this page, the text's grid would be supplied to the space-finding process, which would take the page's list of used space, and combine the two to determine where a list of likely places to put the new text would be.
Similarly, one could work with a two or one column grid.

In NewsClip, it is possible to work with pages and grids of any complexity, for the process is generalized to handle lists which contain any number of rectangular areas.

**The Simple Case**

The simplest way to explain the process of finding free space on the page is to first consider only two rectangles, which we will call *used* and *needed*. *Used* will represent some area on the page which is already occupied, while *needed* will represent an area in which we hope to place
We must also add a stipulation. Suppose the rectangle represented by \textit{needed} describes a tall, thin area to be occupied by a column of text. We may want the process to return any rectangular areas within \textit{needed} which are at least as wide as \textit{needed}, but may not be as deep. This could allow us to lay down at least some text in the desired column width. Thus, when supplying the rectangle called \textit{needed}, we must also state whether it is \textit{inviolate} in the x direction, and to keep things generalized, we can also state if it is inviolate in y. For this purpose we will create two variables called \textit{ix} and \textit{iy}. When \textit{ix} is true, it means we are asking for a rectangular area which is at least as wide as the rectangle \textit{needed}. When \textit{iy} is true, the returned area must be at least as high as \textit{needed}. If both are true, this signifies that \textit{needed} is inviolate in x and y, and so we will get a single rectangle of free space back only when the space described by \textit{needed} is completely unobstructed. The case where neither \textit{ix} nor \textit{iy} is true will not be considered.

We will create a process called \textit{getfree1()}, which, when passed this information, will return a list of free areas. If there is no free area, \textit{getfree1()} will return NULL. Observe that there are only 16 ways in which rectangles can overlap.
Certain cases are trivial. If the rectangles do not overlap at all, 
getfree1() returns a rectangle describing the same space as needed. If
the rectangles do overlap, and needed is inviolate in both x and y,
getfree1() returns NULL. If needed is completely enclosed by used, no
matter what the values of ix and iy are, getfree1() returns NULL. Other
cases depend on whether needed is inviolate in x or y. In any event,
getfree1() can produce one of only three possible results: returning
NULL, returning a list of one rectangle, or returning a list of two
rectangles.

The More Complicated Case
Now we will consider the more realistic case where a number of objects
exist on the page, and we are employing a grid of more than one
rectangle in an attempt to find free space. This situation is more
complicated. This explanation will concern itself with two lists of
rectangles which may be of arbitrary length: the *usedlist* and the *neededlist*, which will refer to occupied areas and desired areas of the page, respectively. We will define a new procedure, called *getfree2()*, which will take as arguments the *usedlist*, the *neededlist*, and the two variables mentioned previously, *ix* and *iy*.

*getfree2()* proceeds as follows. If *usedlist* is empty, it returns a copy of *neededlist*. If *neededlist* is empty, *getfree2()* returns NULL. Otherwise, *getfree2()* takes the first rectangle from *neededlist* and the first rectangle from *usedlist* and calls *getfree1()*, with these two rectangles as arguments. This results in list of 0, 1, or 2 rectangles, as described earlier, which are then stored in a list called *freelist*. *getfree2()* then takes the next rectangle from *neededlist* and calls *getfree1()*, checking this rectangle against the first rectangle in *usedlist*, and adding the resulting list of free rectangles to the *freespace* list. This process is continued until *neededlist* is exhausted. At this point, one can be sure that *freelist* contains only those rectangles derived from *neededlist* which are not violated by the first rectangle in *usedlist*. This process is then repeated on each rectangle in *usedlist*, but now using the rectangles in *freelist* as a description of what is "needed" on the page. If at any time *freelist* becomes empty, it can be concluded that there is no free space available, and *getfree2()* terminates, returning an empty list (i.e. NULL). Otherwise, *getfree2()* continues to the end of *usedlist* and returns the *freelist*. Because we started with a good description of the space we wanted in the form of a grid, *getfree2()* returns a description of free space on the page which is appropriate to the type of space which is needed, and is thus usable by the formatting rules.
The Judge

Once this process is completed, the freelist is passed to another process, which uses a set of criteria to determine if the freelist contains any space which is appropriate to the particular task at hand. We will call this procedure the judge. For example, a page may contain some free space along the bottom, but this space might not be an appropriate place to begin a story, because the headline or the first sentence might have to be broken before being continued on the next page.

The judge must be capable of making determinations of this type. And the criteria vary depending on what is being formatted. A headline may have different spatial requirements than a caption, a picture, or the body of the news itself. In all cases, the function of the judge is to determine whether any "good" space exists on the page, given the requirements of the type of object being formatted. As one of its arguments, the judge takes an interchangeable set of criteria describing what is considered a "suitable" space for each of the objects in the system. If a suitable space is found, the judge returns it in the form of a list of rectangular areas. If nothing can be found, the judge returns NULL. In this case, another page must be found or created, and the entire search and judgement process repeated.

In the case of simpler formatting needs, such as pages composed of one column of text, this complicated process is usually unnecessary. A page with a "current point" can be used to keep track of what space is used, and what is not (NewsClip provides support for this as well). However, as the situation grows more complex, as in a newspaper with a six column grid and stories which are starting and stopping in many places, a single "current point" is no longer sufficient. The method
described here is much better in this case, for it is capable of "looking" at a page, deciding what space is taken, what is not, and using an interchangeable set of criteria, determining if the page provides the type of space which would be appropriate for the kind of object being formatted.

Copyfitting
In addition to these procedures for finding free space, NewsClip provides various copyfitting primitives, many of which make use of the same grids as the free space routines. The first procedure, \texttt{copyfit1()}, takes a text object, and a rectangular area on a page, and "flows" the text into the rectangular area. \texttt{copyfit1()} will determine and use a text object's current attribute set, which includes font, size, leading, and justification method, and then return a chunk describing the result. If the text does not fit entirely within the supplied space, \texttt{copyfit1()} will format as much of the text as it can, and return a chunk describing the result, including the bounding rectangle of the newly formatted piece of text, and an index of the last character formatted, allowing formatting to be easily continued at another place.

A second procedure, \texttt{copyfit2()}, is capable of "flowing" text into an arbitrary number of rectangles, each of which may have any shape (so long as it can accomodate at least one line of text). \texttt{copyfit2()} returns a list of chunks which describe how the text object is mapped to the page in various places, as well as the index of the last character formatted, so that if there is any text left, formatting can be continued easily at another place.

A procedure called \texttt{copyfit3()} formats text by distributing it evenly
along the tops of a list of rectangles. This list may be of any length. This is
the formatting method employed in the first design for the NetworkPlus
hardcopy. Because text objects may change leading at any point,
copyfit3() can not simply count the number of lines of text and divide it by
the number of rectangles, but it must calculate all the line breaks ahead of
time, then determine the total depth of the text object, taking into account
any leading changes. It then determines where to break columns so that
the text is distributed along the tops of the rectangles as evenly as
possible. Other copyfitting schemes are possible, but have not been
implemented.

Font Data
Obviously, copyfitting requires that one have access to accurate data on a
font, including widths of each of its characters. This information must be
available from the outset, for NewsClip uses it in making many formatting
decisions. In the case of the Apple LaserWriter, which was the primary
output device, it was necessary to retrieve and store this information for
each font, so that it would be available to NewsClip at runtime. Although
the LaserWriter does provide input/output primitives, it was decided that
accessing the LaserWriter from within NewsClip as the program was
running would be too slow and complicated. Therefore, information about
all the LaserWriter's fonts was collected ahead of time by another
program, and stored for the use of NewsClip.

In order to keep track of this information, a separate class of objects
was created which could be accessed easily and quickly from within
NewsClip. The main object of interest is known as an LWfont, which is the
repository for all the information about a LaserWriter font of a particular size.

The LaserWriter stores its fonts in outline form at the size of 1 point. Thereafter, if one wants to write a sentence in 48 point Helvetica, for example, one must first create a font of Helvetica characters in that size. In order to do this, a copy of the 1 point font outlines and data is made, which is then scaled linearly to the desired size.

I say it is scaled "linearly" because in traditional typography this is not the case. The characters of 48 point Helvetica are not simply scaled-up versions of the 6 or 8 point size, but are separately-designed letters. They have a different shape. Small letters, usually used in the body of stories and articles, are meant to be as legible as possible at small sizes, given the characteristics of our vision and limitations of printing technology. Large characters, usually found in headlines and in lesser quantities, have to satisfy different functional requirements.

In any event, the LaserWriter scales the characters linearly. This makes font data much easier to compute. The font package developed for use with NewsClip does the same thing, storing fonts at 1 point, and then creating a table for a given font of the requested size, and returning this table in the form of an object called an \textit{LWfont}. This object is provided with primitives which allow one to determine the font's name, size, the width of any of its characters, or the width of a string of characters written in that font. Unlike the PostScript primitives, it is also possible to get values for the descender height, x-height, and ascender height for that font. Fonts can be created in any size, including non-integer ones, such as 10.312. This package provides a good set of primitives, which are fast, functionally "abstract," and easy to use.
Translating Page Descriptions to the Various Output Devices

Once a document has been formatted by NewsClip, it resides on the computer in a form which is independent of any particular output device. The size and location of text and images is specified completely in terms of abstract "pages," and all line breaks have been calculated independently and marked. At this stage, NewsClip shifts from a content-oriented representation of the document to page-based representation. The content of the issue, in terms of "subjects," "stories," "headlines," and the like, ceases to be of importance, while the description of the physical appearance becomes essential. The document is now viewed simply as a series of pages, each with a collection of objects on it which must be printed or displayed. In order to accomplish this, these descriptions must be translated into a series of instructions which some output device can carry out.

A separate translator is required for each output device, and so the translator itself is considered a modular part of the NewsClip system.

Evaluation of Output Devices

The primary output device used in this thesis was the Apple LaserWriter printer. IBM's YODA graphics package was also used, as was Digital Equipment's DECTalk speech synthesizer.

The Apple LaserWriter is a laser printer which contains a PostScript interpreter. It produces black and white hardcopy on paper up to 8 1/2" inches in width, at 300 dot per inch resolution. It can be accessed by either through an RS-232 serial communication port, or through the
AppleTalk local-area network. In this project, all connections were made from the IBM PC through the LaserWriter's RS-232 port.

PostScript is an interpreted computer language designed to be a modern standard for electronic printing. The language had its beginnings in 1976 at the Evans and Sutherland Computer Corporation, where it was developed as part of a research project using an interpreted language for building three-dimensional graphics databases. The major ideas behind the language are attributed to John Gaffney. What is now known as PostScript was further developed at the Xerox Palo Alto Research Center, where it gave rise to another printing protocol, Xerox's Interpress. When Chuck Geschke and John Warnock formed Adobe Systems in 1982, they developed the language further, naming it "PostScript." PostScript is currently used as a two-dimensional page description language, and usually resides as an interpreter in the hardware of various laser printers. ¹⁰

Although PostScript was designed primarily with graphics in mind, it provides most of the features of "standard" programming languages, including the ability to perform arithmetic operations, define variables, evaluate conditional statements, run loops, and define procedures.

Basically, NewsClip has a separate translator, which takes a description of a page, and translates it into a PostScript program, which is then sent to the Apple LaserWriter, interpreted, and printed.

PostScript provides primitives for constructing arbitrary shapes from lines, curves, and splines, as well as procedures which allow shapes to be drawn in any line thickness, filled with shades and textures, or used as a "clipping path" to crop other graphics.

An advantage of PostScript is that it treats text characters like any
An advantage of PostScript is that it treats text characters like any other graphic shape, with the result that they can be operated on by any of the graphics primitives used to manipulate "standard" graphic objects. PostScript supports the full set of geometric transformations (rotation, scaling, and translation) for all graphic objects, including text. In addition, PostScript contains a number of specialized text functions, making it fairly easy to set lines of text, create font outline descriptions, calculate character dimensions, and so forth.

The YODA graphics display device is a 640 x 480 x 8bit frame buffer, with a 640 x 480 viewable resolution. Because it is capable of representing 256 colors at a time, it can support anti-aliased text and graphics. It is not a commercial product, having been developed at IBM's Yorktown Heights research facility, hence the name "YODA," meaning "Yorktown Display Architecture." The YODA device comes with a number of graphic primitives, including anti-aliased text support. It contains its own microprocessor and is micro-codable, resulting in very fast graphics. The YODA represents a level of quality, including color pictures and text, which we will soon be seeing on general purpose computing systems, but which would be especially helpful for data retrieval and display, the text being of much higher quality than that found in typical computer output.

Interfacing with the YODA system, however, is more problematic than interfacing with PostScript, primarily because it does not provide PostScript's extensive library of primitives. Nor does it provide the generality of PostScript, in the sense that in YODA text, being anti-aliased, is treated quite differently from other graphic objects, making translation of abstract document descriptions into displayable images difficult. For example, the YODA represents characters as images, not
suitable for display at only one size. A complete set of bitmaps must be stored for each size of each font. Therefore, it is impractical to store more than a few sizes. This makes the electronic display of a document difficult, because the 50 or 60 fonts stored on the system are unlikely to correspond precisely to the sizes and styles necessary to format a document at any given scale factor.

In YODA, fonts can be scaled, of course, by scaling the individual bitmaps of each of the letters, but this results in an unacceptable loss of quality, as well as slow performance, and a great deal of complexity being added to graphics operations. In addition, fonts which have been scaled up look blocky, while letters which are scaled down tend to look fuzzy and uneven. Even in this case, YODA's primitives are only capable of scaling in integer increments (1/4, 1/2, 1, 2, 3, etc). Rotation is limited to angles which are multiples of 90 degrees. At MIT, the Visible Language Workshop has implemented a limited set of primitives for providing the standard geometric transformations (without rotations, however). The YODA is still a poor place to do serious document formatting, primarily because the bitmap-oriented character representation inhibits accurate scaling of text.

In addition, the font routines which are provided are rather quirky, in that the standard typographic metrics are not supported.

Another problem involves the anti-aliasing of fonts. In the YODA, anti-aliased characters are typically composed of 16 colors: a foreground color, a background color, and a series of 14 intermediate colors, derived by linear interpolation between the foreground and background. Unfortunately, this means that the background color of each character is fixed, which makes it impractical, for example, to place a line of text on top
of a picture, or to run a line of text from an area of one background color into an area of another.

A better way to implement text is to store it in outline form, which can then be scaled to any arbitrary size, rotated, and so on. In the case of medium resolution CRT output devices, however, anti-aliasing must also be dealt with, a factor which the typical laser printer, with only one color and higher resolution, can safely ignore. Perhaps higher spatial resolution will make anti-aliasing less necessary, but in the meantime, the ability to perform anti-aliasing against backgrounds of arbitrary and varying types would be helpful. Lack of this capability imposes unacceptable restrictions on the designer when compared to print media.

Lack of an adequate interface to softcopy output makes it difficult to translate arbitrary page descriptions into display instructions. There are simply too many things which are difficult or impossible to do. This in turn requires that the formatting program and its rules know the eventual output devices (such as the YODA) intimately, destroying much of the generality which is one of the primary advantages of NewsClip's design.

In addition to the LaserWriter and the YODA display device, some experimentation was done with producing voice output. This capability might be useful to the handicapped.

Basically, the DECTalk takes an ascii file and reads it aloud, interpreting special imbedded codes which allow it to change voice, tone, stress, speed, and inflection. In producing speech output, NewsClip can direct the DECTalk to change these parameters, producing, in effect, "spoken typography," in which headlines are announced in one voice, stories are read in another, and quotes recited in a third. The interface to the DECTalk is relatively straightforward, and the job of translating news
The Operation of NewsClip

stories into a form appropriate to the DECTalk is minimal. The txl system of text tagging allows any part of a text object, such as a "headline," or "keyword," to be read aloud. The voice sounds artificial, but it is understandable, the primary problem being the mispronunciation of various words. This will be a difficult problem to solve completely, and the best that NewsClip could do at the moment is provide a dictionary of mispronunciation, which would store words with which the DECTalk was known to have trouble with, providing a set of misspelled, but phonetically more accurate, representations.
Computer-related technologies will change the practice of graphic design in two primary ways: through the development of new media, and through the introduction of new tools. This chapter covers both of these points broadly.

**New Media**

The first point to make is that there are many new media, some of them more related to the field of graphic design than others. For example, electronic "markup languages," in which the various parts of a document are tagged with descriptive labels and then processed according to a set of rules or "style sheet," relates directly to document preparation and current typesetting technology, and thus graphic design. This is a natural area for graphic designers to employ computer technology. They will soon be using systems such as these to create standardized formats for documents, which can be stored electronically and used repeatedly to create high-quality reports, manuals, documentation, and the like.

"Automated" or "rule-based" techniques for design and formatting, such as the system implemented in this thesis, are necessary to handle output generated by a "personalized" information retrieval systems. Because it is impossible to design this information using traditional techniques, some sort of design program is required which can be
Impact of New Technology on Graphic Design

integrated into an electronic information environment.

As the workplace becomes further computerized, the graphic quality of its electronic communications will become increasingly important. The availability of display devices capable of generating higher-quality graphics will increase, and designers will find work creating electronic formats and design standards for various electronic media.

However, other computer-based technologies are very different than print. A primary difference is the ability of some media to combine a wide variety of information types simultaneously, including text, images, cinema, voice, sound. Evaluating the "impact of new technology on graphic design" in light of more exotic media is problematic because many of the people working in these new media will obviously not be graphic designers. They will find themselves designing voices, not layouts. They will think not only about the aesthetic of form, but the aesthetic of dialogue. Therefore, addressing the question of "the impact of new media on graphic design" is probably looking at things from the wrong viewpoint. It is more realistic to speculate on the nature of the new professions which will be the electronic analogs of graphic design.

In addition to providing information of a number of types, computer media are capable of something even more profound: interactivity. One stumbling block for designers who wish to take part in these new media is that the problems are often unfamiliar, and the technology is not understood. An interactive news database, for example, presents many options which simply do not exist in print, including interaction on the part of the reader, and a body of information which is dynamic, meaning that it can be viewed in various ways, according to the reader's needs and preferences. The graphic designer who considers his job to be a good
screen design is really missing the point.

The field of electronic information design uses principles from many other media, including computer science, product design, film, and human factors. It includes the problem of user-interface design. When moving into new media, particularly those in which the user is expected to interact, the graphic designer will find that a number of problems arise, not just because he doesn't fully understand the medium, but because no one else does, either. There are few established conventions. We all know to read books from left to right, top to bottom, and the graphic designer can count on his audience being familiar with the conventions of print. In newer media this is less true.

When solving interface problems, the designer is often confronted with two main options: the first is to do something people are already familiar with. The second is to do something new, and hope the user will get the hang of it.

"Doing something which people are already familiar with" is exemplified by the use of metaphor in computer interfaces. In this approach, we find pictures of office-like file folders representing data, and trash cans symbolizing the file deletion function. In essence, the designer is attempting to describe something new by metaphorical reference to those things with which the user is likely to be familiar.

The second option is to pick some more or (hopefully) less arbitrary approach, and by using it consistently (or perhaps just frequently), establish it as a convention in the user's mind. This is the "brute force" technique. It has the disadvantage that working methods represented in this way are harder to learn. There is nothing in the user's experience to "hook" them to. In reality, these options are not distinct choices, but
New Tools

New tools are finding their way into the design studio. This is important to graphic designers for economic reasons, but also because it is likely to change the way designers work and think. The relationship between the tool and the artist is cyclical. A need of the artist or culture inspires the development of a new tool. The character of the tool, the things it suggests, makes possible, or precludes, often influences what the artist thinks of to do with that tool. Therefore, designers will want to make sure that they have good tools.

Many of the designer's tools are likely to change simply as a matter of practicality. Computer imaging, layout, and typography are already proving beneficial in the printing industry, where production techniques are becoming increasingly digital. Similar developments are likely to take place in the design profession, not only because they speed things up in the studio, but because designers who can interface directly to modern printing plants' digital production techniques will have an economic advantage.

Simulating high quality print is as expensive in terms of materials as it is laborious. Obviously, one of the primary areas where computers can be of assistance is in producing "comprehensives," simulations of the printed article which are used to get a realistic idea of what a given design will actually look like, or to finalize it with a client. Systems such as Lightspeed's "Quolor" allow one to produce comprehensives quickly. As the systems become cheaper and better, more designers will use them.
Currently, laser printing devices are making it possible to get almost-print-quality type at a cost which is equivalent to xerography. The quality of this type is high enough for most conceptual or "rough" design work. As the number of fonts available on these machines increases, so will the number of design studios who purchase and use them.

The ability to do electronic retouching and image processing will allow designers to manipulate pictures in ways which used to be expensive, and hence would not have been considered a few years ago. Except for the most expensive in-plant production systems, the resolution available is not yet on par with print. Designers will use these systems in increasing numbers if resolution and cost continue along current trends.

These trends, however, are not entirely to the benefit of the design profession, for they are making it possible for individuals and companies to produce their own high-quality documents. The field is known as "desktop publishing." This means that a certain amount of work that currently goes to a design studio will now be kept in-house. The graphic quality of this information may not be as high, but I would not over-estimate the value of that quality in the heart of the cost-conscious manager, nor would I underestimate the ability of the intelligent non-designer to produce a reasonable-looking layout. In fact, in the near future, the layperson may be assisted by machines with a limited amount of graphic "intelligence," put there by a graphic designer. This represents a fundamental change in our culture: it is becoming increasingly feasible for individuals to write, design, and publish their own books in respectable quality, for moderate cost, and without outside help.

Perhaps the design profession will shrink as a result of increasingly efficient and automated information production techniques. On the other
hand, our conversion from an industrial to an "informational" economy will make communication skills and graphic quality increasingly important. The net gain or loss of employment opportunity in the graphic design profession is difficult to predict.

**Computer as Conceptual Tool**

The previous discussion focused on the use of computer-based tools in the performance of tasks which are, for the most part, already possible using current graphic arts technologies, the main advantage of the new devices being that they make some process faster, easier, cheaper, or possible for the designer to perform himself, on an in-house basis. However, this alone is a limited view of the computer as a tool. The primary characteristic of the computer, its ability to carry out a program, can make it valuable in the hands of a designer/programmer who wants to experiment with automated form generation, modeling a particular "style" or approach, or solving problems using prescrived or "parameterized" methods. There has been a fair amount of work done in this area, much of it without the benefit of computers.

"Programmed methods of form generation are nothing new. They are encountered in the rules of certain very ancient games and spatial puzzles, in various types of ornamentation, in order-based architectural styles, and the iconographic canons of painting. They are found in the proportional systems that establish a modular coordination between elements, from the classical golden section to the
Corbusier Modulor...

"Sometimes designers, artists, and architects talk in a logically organized way about the role of a conceptual programme in their work: the celebrated "Five Principles" of Le Corbusier are recalled, for example. It is natural to suppose that certain conceptual programmes arise from a designer's analytical research, that others result from lengthy practical experience... We need to be aware that these conceptions can be expressed not merely through words, but also graphically, mathematically, in specialized machine languages, or through sundry other forms of model." (V. F. Koleichuk, of the Research Sector for the History and Theory of Design, All-Union Research Institute of Technical Aesthetics, Moscow, USSR. From "Programmed Form Generation in Design", Environment and Planning B, 1980, volume 7)

Figure 7.1 Programmed form generation. The three on the left are Hepplewhite-style shield-back chairs attributed to Samuel McIntyre. On the right is a new Hepplewhite-style chair generated according to a grammar devised by T. Weissman Knight.
Programmed form generation. From the upper left: Greek vase (Kylix) with harmonic analysis (from *The Diagonal*, Yale University Press); nautilus shell (photograph, Wendigen); architectural proportions; Botticelli, *Birth of Venus*, 1485.
George Stiny [1980], W.J. Mitchell [1980], and others have developed the notion of the "shape grammar," particularly as it applies to the field of architecture. A clear example of this work was executed by two students of Stiny's, H Koning and J Eizenberg in the form of a parametric shape grammar describing the style of Frank Lloyd Wright's prairie-style houses. The grammar was capable of generating a number of convincing houses.

1 \* \rightarrow (0, 0, 0)

(a)

(b)
Figure 7.3 Graphic Illustration of an architectural grammar.
"Consistency in grammar is therefore the property -- solely -- of a well-developed artist-architect. Without that property of the artist-architect not much can be done about your abode as a work of Art. Grammar is no property for the usual owner or occupant of the house. But the man who designs the house must, inevitably, speak a consistent thought-language in his design. It properly may be and should be in a language of his own if appropriate. If he has no language, so no grammar, of his own, he must adopt one; he will speak some language or other whether he so chooses or not" (Wright, 1954, pages 182-183).

T Weissman Knight created a grammar which generated floor plans for Japanese tearooms [1981]. All tearoom plans, and Japanese floor plans in general, are based on the ken grid. The ken is a unit unique to Japanese architecture, and is approximately 6 feet long. The location of the hearth, entrance, and other elements follow a well defined set of rules.

![Diagram of a Japanese tearoom plan]
Computers can be used for modelling an "approach" to creating an image or building. A rigorous description of the formal properties of an object of a particular "style," when represented as a computer program, is capable of generating a range of objects within that style. This can be called a "shape grammar:" a body of rules about forms and the ways in
which they can be put together.

In this sense, the designer is concerning himself directly with the approach to creating an image, or a class of images, rather than the specific image itself. The creative act occurs when specifying the form-generating entity. Or, when having copied an existing style, modifying it in some way. To further remove onself from the image, it might be possible to model and approach to modeling and approach to generating imagery. Or an approach to modeling and approach to modeling...

In the world of graphic design, and more specifically in the world of typography, one can consider an object such as a piece of type to have a number of "parameters," meaning attributes which change in value. For example, suppose one is designing a page, and the only thing to be put on it is a piece of type. There are a number of attributes which could be varied, such as font, size, color, position, whether it will appear in one line or two, and if two, whether it will appear flush-left, flush-right, justified, or center-stacked, whether it will be rotated, distorted, and so on. Say there are about ten such attributes, and each attribute, on the average, can assume any one of ten possible states. This is essentially the idea of Gerstner's "morphological box of the typogram."

Rather than specify these things directly, the designer might create a program which changed each attribute systematically, one at a time, until all possible configurations had been exhausted. The designer could then be certain of having explored the complete range of possibilities before deciding upon the one that was best.

One advantage of the approach is that the program, even within this relatively limited problem, would probably stumble upon one or more
7 Impact of New Technology on Graphic Design

solutions which wouldn't have occurred to the designer. The disadvantage of the method is that it results in $10^{10}$, or 10 billion, possible configurations. There are two important aspects of this approach:

1) it is extremely stupid and slow

2) the complete works of Shakespeare (as well as every weird variation upon them) could have been generated this way.

One can see that when one has a very small number of parameters and a small number of states for each parameter, the number of possible solutions remains reasonably small and envisioning them is within the grasp of the average person. As soon as the number of parameters, and the number of possible states for each parameter begins to increase, the number of possibilities explodes. Specifically, the number of possibilities exhibit an exponential order of growth equal to $v^p$, where $p$ is the number of parameters, and $v$ is the potential number of values for each parameter. This can be thought of as an n-dimensional space in which each parameter is represented by a dimension, and a graphic arrangement is represented as a point corresponding to a particular collection of parameter settings.

Because of the large number of possibilities involved, the designer may wish to define a "strategy" for searching this n-dimensional space for graphic solutions. This strategy could be in the form of a set of rules regarding the values each parameter could assume. It could specify a simple set of allowable values, or functional relationships between parameters, such that as $x$ varied, $y$ would vary as a function of $x$, or as a function of $x$ and $z$, and so forth. In this way it is possible to create an n-dimensional "solution space" which defines the boundaries of the class of solutions which are considered (nominally) acceptable. This is the
opposite of the commonly held notion of design as starting with "nothing," and ending up with "something." Instead, it can be thought of as starting with "everything," and narrowing down to a number of likely "somethings."

One of the characteristics of this model of form generation is that it deals exclusively with the parameters of already existing objects, not the creation of new ones, and so in this respect (if not others) is probably an unrealistic model of the designer's actual thought processes. Therefore, it may be necessary to devise rules governing the creation of objects as well. For example, in producing hard copy output of *NetworkPlus*, not only was it necessary to assign attributes to each story, but new objects had to be created. This included objects such as type, rectangles, and lines, which served as delimiters, highlights of specific pieces of information, and the like.

Figure 7.5  Jere Donovan, *Action*, Herbert Bayer Design Class, 1939. Is it possible to describe, in a machine language, this aesthetic of asymmetry and dynamism?
One problem with this type of approach is that once the grammar underlying any form generation scheme is detected and fully understood, it becomes predictable, and thus boring.

In addition, there are good and bad subjects for programmed design. Computer-generated forms in "fine art," for example, are usually failures. One reason is that, although formal grammars are capable of dealing with "meaning" in a limited sense (the "function zones" in Eizenberg and Koning's prairie-house grammar, or the txl labelling scheme in NewsClip), they can not really create or understand human meaning. They work better in the realm of form. When we see "pure shape" generated by a machine, it almost never succeeds artistically, because it lacks the meaning, or "resonance" necessary to engage human intellect and emotion.

On the other hand, systems of form generation seem to fare better when buried beneath something else, when used as structure, as in the fields of architecture and graphic design, where the product has immediate function as well as aesthetic value, and the generation of forms serves not as the entire message, but as the foundation for other, more important messages. In this sense, form is used to impart an underlying sense of beauty, order, or proportion.

The process of modeling an approach to creating visual imagery obviously has educational value. In fact, some of what we do in a design education is precisely that: getting students to evaluate specific "styles" or image-generating approaches and model them in their own work. This increases their visual vocabulary, makes them critically more adept, and teaches them about the the use of various approaches in specific contexts. Once a style has been analyzed and modeled, it can then be
modified, thus creating a number of offshoot systems of form generation. This doesn't necessarily require a computer, and in fact in educational situations, work might proceed faster without one, as the necessity of learning specific languages and devices is eliminated in favor of concentration on the conceptual coherence and description of simple form-generation algorithms.

It is easy to be beguiled by the ideas of rule based design, but anyone who has tried to implement such a system knows the difficulties involved and the limitations of mechanical approaches in creating truly meaningful work. This does not mean that there is no fascination or value in formal systems of shape generation, only that they are better suited to some types of problems than others.

Problems involving a high degree of repetition with minor variation, such as the design of a large number of packages for a family of related products, are probably likely candidates for automated layout. Proposed systems such as those which "keep track of" company design standards, and which reprimand the designer for making "non-standard" design decisions are a reprehensible use of technology. They represent a repressive view of design and the role of the worker.

After having established the feasibility of the rule-based approach, the next logical step will be the development of specialized languages and interfaces to allow designers to work electronically and begin to model grammars and constraints more easily, and so integrate them directly into their working processes.
Footnotes

1. Philip Meggs,
   *A History of Graphic Design*,

2. I must give credit for this idea to Lou Danziger,
   from whom I first heard it.

3. This example taken from:
   Thinking Machines Corporation,
   *Thinking Machines Technical Report 86.14: Introduction to Data Level Parallelism*,

4. Thinking Machines Corporation,
   *Thinking Machines Technical Report 86.14: Introduction to Data Level Parallelism*,

5. George Miller,
   "Psychology of Information",
Footnotes

6. Richard A. Bolt,
   *The Human Interface: Where People and Machines Meet,*

7. I must give credit for this idea to Nicholas Negroponte,
   from whom I first heard it.

8. Fritz Zwicky,
   *Die morphologische Forschung,*
   Kommissionsverlag, Winterthur, 1953.

9. Karl Gerstner
   *Designing Programmes,*

10. Adobe Systems, Incorporated,
    *PostScript Language: Reference Manual,*
    (Introduction by John Warnock)

11. Adobe Systems, Incorporated,
    *PostScript Language: Reference Manual,*
    (Introduction by John Warnock)
Bibliography

Abelson, Harold and Gerald Sussman with Julie Sussman.  
Structure and Interpretation of Computer Programs.  

Adobe Systems Incorporated.  
PostScript Language: Tutorial and Cookbook.  

Adobe Systems Incorporated.  

Barron, Stephanie and Maurice Tuchman, eds.  
The Avant Garde in Russia: 1910 - 1930.  
Los Angeles County Museum of Art, Los Angeles, California, 1980.

Bolt, Richard A.  
The Human Interface: Where People and Computers Meet.  

Gerstner, Karl.  
Compendium for Literates.  
Gerstner, Karl.
Designing Programmes.

Ghyka, Matila.
The Geometry of Art and Life.
Sheed and Ward, New York, 1946.

Gross, Mark D.
Design as the Exploration of Constraints.
PhD. Thesis, Department of Architecture,
Massachusetts Institute of Technology, 1986.

Hill, Ann, ed.

Hofstadter, Douglas R.
Godel, Escher, Bach: an Eternal Golden Braid.

Klee, Paul.
The Thinking Eye: Notebooks, Vols 1 & 2.
Bibliography

Kepes, Gyorgy.
Language of Vision.

Koning, H. and J. Eizenberg.
"The Language of the Prairie: Frank Lloyd Wright's Prairie Houses."

Knuth, Donald E.
TEX and METAFONT: New Directions in Typesetting.

Lippman, Andrew.
Electronic Publishing.
Media Laboratory, MIT, 1986

Meggs, Philip G.
A History of Graphic Design.

Moessel, Ernst.
Die Proportion in Antike Und Mittelalter.
C.H. Beck'sche Verlagsbuchhandlung, Munich, 1926.
Bibliography

Muller-Brockmann, Josef.
Grid Systems.
Arthur Niggli, Ltd., Jeuten, Switzerland, 1981.

National Bureau of Standards.
Information Processing - Text and Office Systems -
Standard Generalized Markup Language.
ISO/DIS 8879.

Peterson, Ivars.
"Computing Art".

Rand, Paul.
Thoughts on Design.

Reid, Brian K. and Janet H. Walker.

Richter, Irma.
Rhythmic Form in Art.
Bibliography

Ruder, Emil.
Typography, a Manual of Design.

Stiny, George.
Algorithmic Aesthetics:
Computer Models for Criticism and Design in the Arts.

Stiny, George.
The Grammar of Paradise:
On the Generation of Moghul Gardens.

Stiny, George.
Introduction to Shape and Shape Grammars.

Thinking Machines Incorporated
Introduction to Data Level Parallelism.
Thinking Machines Technical Report 86.14
Thinking Machines, Incorporated, Cambridge, Massachusetts, 1986.

Thomas, Brian.
Geometry in Pictorial Composition.
Wascher, Susan.  
Content and Form:  
Method for Optimizing the Spatial Layout and Temporal Sequencing of Information for Viewer Comprehension.  

Weissman, Knight T.  
"The Generation of Hepplewhite-style Chair-back Designs".  

Weissman, Knight T.  
"The Forty One Steps: The Language of Japanese Tea-Room Designs".  