CMU-ITC-91-101

The Andrew System:

The Role of Human Interface Guidelines in the Design of Multimedia Applications

Maria G. Wadlow

Carnegie Mellon University

Final Draft

To be published in **Current Psychology: Research and Reviews** (Summer 1990)

15 May 1990

Maria G. Wadlow
Manager, Human Factors Design Group
Information Technology Center
Carnegie Mellon University
4910 Forbes Avenue
Pittsburgh, PA 15213-3890

412 / 268 - 6794
412 / 268 - 6787 (Fax)

maria+@andrew.cmu.edu

Abstract

As computer systems are becoming more advanced, the user interface issues involved in system design are increasing in complexity. Multimedia applications add yet another layer of complexity to this already intricate task and guidelines are needed to help ensure consistency, both within a given computer system as well as across systems. The Human Factors Design Group of the Information Technology Center (ITC) at Carnegie Mellon University has developed a set of user interface guidelines for the Andrew system. These guidelines are intended to assist application developers, both within the ITC and externally, in creating application programs whose interfaces are consistent with the look and feel of Andrew.

Introduction

Until recently, computer users have had to rely on text-oriented documents and character-based computer systems to do their work. Newer systems have enhanced the user's ability to represent and communicate ideas through the use of multimedia objects. Most of these systems enable the user to incorporate text, graphics and pictures as objects in their electronic documents. Tables, equations and other object types are currently available, while video and sound are being added to the list of multimedia capabilities that computer systems can offer today.

Such new computer technologies can provide opportunities for users to become more productive and creative. However, users can take full advantage of these opportunities only if they are presented in a way that is easy to understand, learn, and use. Therefore, it is imperative that a computer system have an appealing and coherent user interface because, to the user, the interface *is* the system.

Human factors, cognitive psychology and graphic design principles are beginning to make an impact in the computer community through recent attention to user interface design. However, knowledge about the user interface design process is accumulating at a relatively slow rate. It is difficult to make definitive statements about user interface design, even in areas which have been studied in great detail, such as text-based processing and command-oriented systems. When this territory is broadened to include multiple windows, color, multiple input and output devices, and new object types such as animation and video, the user interface design task can become quite challenging. These challenges are further compounded when compatibility with existing systems is also a concern. While it is impractical to attack all of these issues simultaneously, it can be helpful to explore a few in detail during the design of a system.

Background

Upon examining a number of existing multimedia systems, one may perceive several shortcomings in the level of integration of the multimedia objects in the system. Many systems provide a simplistic model in which multimedia objects are integrated at a somewhat superficial level. In systems of this type, an object is created and can then be viewed in a number of representations. For example, a table of numbers can be inserted into a document and then viewed as a bar graph or as a pie chart. While some attributes of these various representations may be changed, the user may only view one representation at a time. This approach provides the user with access to the multimedia aspects of the system, but allows only a gross level of control over object representation and integration since objects of different types can not be represented simultaneously.

Other systems offer a slightly more complex model, in which a document is viewed by the user as a simple sequence of objects. In these systems, the user creates a textual document and can then insert other objects at any location within the document. This approach provides the user with more flexibility in positioning, in that several objects may be represented simultaneously and a single object does not necessarily occupy the full document width. However, the user must invoke a separate application to edit each of the various objects. When the changes have been made, the user must then update the document to reflect those changes. Editing a document in this type of system is similar to professional paste-up, where the pictures and figures are produced separately from the text and must be inserted by hand. This method of editing is not very interactive or intuitive and adds too much overhead to the document creation process.

Some systems make use of a more sophisticated environment in which objects may be edited in place. Here, the user creates a document and inserts objects into it, as desired. Generally, all documents are considered to be text documents, and multimedia objects may only be inserted into text. Document creation in a system of this type is analagous to having a very sophisticated typewriter, which deals primarily with text, but is also able to edit equations, pictures and other objects. While this system presents a model that is closer to what a user might expect, there are still limitations within the system, such as the special status that text is given over other object types, which contradict those expectations.

The Andrew System

The Andrew system, developed by the Information Technology Center (ITC) at Carnegie Mellon University, includes among its components the Andrew Toolkit (ATK), a multimedia toolkit which is used by developers to build multimedia application programs. One goal in the development of Andrew was to provide developers with the flexibility required to build multimedia application programs while still providing users with a coherent user interface. To an application developer, ATK provides a generic structure within which heterogenous objects may be combined. Several standard objects are packaged with ATK including text, graphics, equations, calendars, tables and simple animations. In addition, several multimedia application programs are distributed as part of the Andrew system. Two of the more prominent of these are *Ez*, a multimedia document editor (See figure 1) and *Messages*, a multimedia electronic mail and bulletin board system (See figure 2). ATK was designed to be extensible, providing developers with the ability to build new objects, which can then be tailored to meet their specific needs. In addition, these new objects may be integrated with the standard objects, in any combination, to create new application programs. The integration of heterogeneous objects, both those supplied with the toolkit and those developed externally, provides a great deal of flexibility for developers.

The Andrew Toolkit uses an embedded object architecture, where objects can be arbitrarily nested within one another at any level. The top level document is not restricted to text, but can be of any type. For example, a user can create a table document which includes graphics and text objects as easily as s/he can create a graphics document which includes text and tables.

Although there are some logical restrictions, theoretically, any object can be embedded within any other object type, to any depth. For example, while it may not be sensible to embed an object within a raster image, one may want to embed a raster image within a table which is itself embedded in a text document (See figure 3). All objects can be edited in place within the parent object. In addition, the rules which govern the behavior of multimedia objects in the editor are also in effect in every other ATK application. This approach is more consistent with the user's mental model of document creation and it helps to reduce the user's awareness of the limitations of the system.

User Interface Issues

The user's ability to adopt a mental model of system behavior is critical to the success of a user interface. Carefully fostered mental models can enhance the user's perception and understanding of the system and how it works. Typically, applications are developed with some type of model in mind. The developer usually has a clear notion of how the application works, which is used to determine whether additional features will be consistent with the rest of the system.

Users also develop a mental model of the system. The user's mental model is modified and refined as the user becomes more familiar with the system. In fact, a well-developed mental model may eventually enable users to predict system behavior as they continue to explore and learn about the system.

Unfortunately, an application's user interface does not always successfully convey the developer's mental model to users of the application. A user whose mental model of the system is inconsistent with that of the developer may find the application confusing and difficult to use. The user's expectations, which are based on his or her mental model of the system, may vary in important ways from the actual functionality of the application. Further, the developer's conceptual model may be too complex and detailed, lacking the appeal necessary to be of assistance to users of the application. In most instances there is no need for users to understand the application to the level of detail and complexity that is required of the developer. Ideally, the user's mental model should be structured in a way that will help users "bootstrap" themselves to greater comprehension of the system as a whole.

One technical goal in the development of ATK was to create a system which could easily be extended by developers external to the ITC. We hoped to provide a framework in which new multimedia objects and applications could be developed. Although we did not want to limit the creativity of these external developers, an important human factors goal was to provide an environment which would foster consistency among new applications. We wanted to create the appearance of a single, unified interface, for although the user may perceive a seamless system, the system is actually composed of many modules which were created by different developers with different purposes in mind. We wanted the users to be able to create a mental model of the system's operations which would remain consistent regardless of the particular application being used or the origins of that application. In addition, we wanted to allow users to transfer skills between different objects and applications. Although the individual components of an application may be seen as distinct modules to the developers, we wanted to blur this distinction for users of the system.

We accomplished this in two ways. We created a set of user interface design guidelines to be used by developers when creating new objects and applications. In addition, we attempted to provide standard objects and applications which were consistent with those guidelines. The goal in developing a guidelines document for Andrew was to focus attention on critical design issues and establish specific design requirements. In addition, it was hoped that these guidelines would encourage further discussion of complex user interface decisions and policies.

User Interface Guidelines

Ideally, the user interface for a system should be simple, consistent, and appealing, encouraging exploration and allowing the user to transfer experiential knowledge between applications. Consideration of guidelines in the development process helps to assure consistency across applications. In addition, the use of guidelines must be supported by a thorough understanding of the user population for which the application is being designed. Designers should keep in mind that accurate samples of the user population rarely consist of application developers. For maximum effectiveness, consideration of the guidelines must take place early in the design process, and prototype testing should be done to ensure good design.

In developing the user interface guidelines for Andrew, *consistency* was our primary goal. We attempted to ensure consistency on several levels, including: consistency in the kinds of objects users see; consistency in the kinds of operations that are possible within applications and in how those operations are accomplished; as well as consistency in the methods used to pass information from user to machine and back. This consistency is crucial in creating a system which appears to its users as a consolidated, working whole, and which in turn enables users to transfer skills and concepts between various objects and applications.

Closely related to consistency is *predictability*, the ability of the user to correctly anticipate responses from the system. We considered system predictability to be vital in establishing the sense of trust that is needed for users to feel comfortable using the system. Surprising behavior might be entertaining in a game but is considered frustrating and annoying in a tool.

In making choices about interface options, priority was given to features which put the *locus of control* on the user, rather than on the computer. It was our belief that users would feel more comfortable and productive if they felt that they were in control. Our goal was to make Andrew a tool which users employ to reach their own goals, rather than a system which appears to have goals of its own.

We wanted the layout of the interface to be inviting, drawing the user in and creating a comfortable working environment. *Visual appeal* is vital to good communication. Complicated, awkward or cluttered interfaces are low on appeal and inhibit productivity. Well-articulated, straightforward, clean designs are appealing because they convey information clearly. Appealing systems encourage exploration, creating an atmosphere which is conducive to learning.

Finally, we believed that users should have the sense that they are dealing *directly* with "real" objects, objects which, though they are only two-dimensional on a computer screen, behave in a manner that is consistent with real world objects. For instance, moving an object on a screen by adjusting numbers of an axis is not direct; moving that object via selection and dragging with the mouse is more direct.

After having developed and tested several objects for the system, we felt that we understood the issues involved in enough detail that we could generalize our knowledge about each of the individual interfaces into guidelines for the design of future objects. Since the existing objects were under our control, we could change their interfaces to match any new guidelines or criteria that we developed.

Although one of the technical goals of ATK was to allow the creation and use of heterogeneous objects, it is difficult to create a standard interface for objects whose types and uses are so different. We categorized the objects into several types: sequential, spatial, and spatial-sequential, to provide a means for meta-level discussions and to anticipate the characteristics and requirements of future objects.

Sequential objects are characterized by the organization and layout of the information that they contain. Text and equations are examples of sequential objects, where one piece of data follows another in a sequential fashion. *Spatial* objects contain data which is related by its spatial positioning. A graphics object would be considered spatial because data is positioned relative to the object's borders rather than to other data in the object. The last category that we considered was *spatial-sequential*. These are objects which possess characteristics of both the spatial objects and the sequential objects. The table object is considered spatial-sequential since it has both sequential and spatial positioning attributes. Upon identifying these meta-level distinctions, our task became much simpler. We could discuss classes of objects without being inhibited by the nuances or idiosyncracies of any particular object.

Each of the objects which existed in the system at this time, and any that we could imagine being built, required specific skills and expertise on the part of the developer. We did not want to restrict the possibilities for interaction within a particular object unnecessarily, but we were concerned about the consistency of interactions between that object and other objects in the system. Thus, we classified the operations with which we concerned ourselves in terms of inter-object operations, which are operations performed on the object as a whole; intra-object operations, operations performed on the contents of an object; and extra-object operations, operations which are outside the realm of the object itself.

Inter-object manipulations that were discussed in the guidelines dealt primarily with operations on the object as a whole. For example, the guidelines described the user interface for object creation, selection, duplication and deletion. In addition, the guidelines addressed operations which dealt with the display of an object, such as the user interface for scrolling and moving an object. It was particularly challenging to design the guidelines which described resizing an object. Depending on the meta-level categorization of the object, its contents may be scaled, cropped or wrapped in order to fit in the available space (See figure 4). Finally, the inter-object guidelines described the changes that occur when a user gives an object the input focus, indicating the desire to manipulate the contents of that object.

Intra-object manipulations describe the operations which can be performed on the contents of an object. Once an object has received the input focus, the behavior that it exhibits and the options that it provides are left to the discretion of the developer. Some operations, such as cut, copy, paste and replace, are available for most objects in the system. The user interface requirements for these and other intra-object manipulations were specified in the guidelines to ensure that they remain consistent between objects.

Extra-object operations dealt with operations which are outside the realm of any individual object. Primarily, these include what we call dialog and response devices. These devices are included within ATK as a service to developers and can be used by all of the objects and applications in the system. Dialog and response devices were divided into four categories:

Action devices - buttons, menus, key and mouse operations

Choice devices - switches, sliders, lists, toggle items

Text Input devices - data entry fields

Feedback devices - dialog boxes, message lines

In keeping with the extensible architecture of ATK, these devices are also considered objects in the system and a developer may replace or create objects of this type as well. The guidelines provided suggestions as to when and how these various devices should be used, in order that their usage could remain consistent among all object types.

In addition to the classes of guidelines mentioned above, we also considered issues involved in cooperation between objects at different levels. For example, the relationship between an object and its parent, or containing object required some amount of detailed discussion and thought. Although the types associated with the parent and child objects are largely irrelevant, there should be some standard way of transferring or inheriting information from one to another. This cooperation is of particular importance when considering the cooperation involved to ensure that the appropriate menus, key bindings and cursors are being used.

Conclusions

We began to compile user interface guidelines for Andrew as an aid to other developers. As this work continued, we found that the guideline development process helped us immensely in understanding the issues involved in user interface design for a multi-window, multimedia, extensible system, such as Andrew. Many of these issues were raised through the development of the standard objects and applications that exist in Andrew, and much of what we learned through this process is reflected in those objects and applications. The guidelines may not provide specific answers to all of the questions that a developer may have, but they can help convey of the flavor of the system and provide some insight regarding the choices that were made in its design.

One of the most important and difficult aspects of designing the Andrew user interface was to determine reasonable default behaviors for our user community here on campus, which varies greatly in its needs, experience, and knowledge. Reasonable defaults are crucial: it is default behavior which users first encounter in a system. Defaults shape how users come to understand and to use a system, and default behavior comprises the mental model which is important for successful interaction with a system. Simple and predictable default behaviors, then, should allow users to do what they need to do to get their work done, while not overwhelming them with complexity or unnecessary information. In many ways, designing a system *is* designing its default behavior.

This work has helped us to learn some valuable lessons about the role of human factors in system design. In particular, it is critical that human factors issues are considered from the beginning of the system design process. Some of the most difficult changes that were made in the system could have been avoided had the issues involved been considered earlier in the design process.

Standards and guidelines are valuable tools for system design. Although it may be difficult to agree on what the standards should be, discussion can help developers focus on problem areas and uncover portions of the architecture which may be poorly specified. Although the basic components of Andrew were developed at the ITC, they were developed over a period of time by an ever-changing group of people with diverse backgrounds and interests. Furthermore, ATK is being used and developed by people around the world. Standards and guidelines can be useful tools for communication within and among these groups.

Notes

The author would like to thank Dr. Christina Haas, Dan Boyarski and Paul G. Crumley, who were involved in the guidelines development and Ayami Ogura for her comments on this paper, as well as other members of the ITC who contributed to the development of ATK and Andrew.

The ITC is a joint venture between IBM and Carnegie Mellon University. This work would not have been possible without the support of these organizations.

Address for correspondence: Maria G. Wadlow, Information Technology Center, Carnegie Mellon University, 4910 Forbes Avenue, Pittsburgh, PA 15213-3890.

References

Human Interface Guidelines: The Apple Desktop Interface. (1987). Addison Wesley.

Borenstein, N., Everhart, C., Rosenberg, J., Stoller, A. (1988). A Multi-media Message System for Andrew, *Proceedings of the USENIX Technical Conference*, Winter 1988, 37-42.

Borenstein, N.S. (1990). *Multimedia Applications Development with the Andrew Toolkit.* Prentice Hall.

Mayhew, D.J. (1988). Basic Principles and Guidelines in User Interface Design, *SIGCHI Tutorial Notes.* ACM Special Interest Group on Computer Human Interaction Annual Meeting: May, Washington, D.C.

Microsoft Windows Software Development Kit: Application Style Guide. (1986). Microsoft Corporation, (Version 1.03).

OPEN LOOK Graphical User Interface Functional Specification. (1988). Sun Microsystems, Inc., (Prerelease Version).

Palay, A.J., Hansen, W.J., Kazar, M.L., Sherman, M., Wadlow, M.G., Neuendorffer, T.P., Stern, Z., Bader, M. & Peters, T. (1988). The Andrew Toolkit - An Overview, *Proceedings of the USENIX Technical Conference*, Winter 1988, 9-21.

Ramsey, R. (1979). *Human Factors in Computer Systems: A Review of the Literature*, Science Applications Incorporated.

Rubenstein, R. & Hersh, H. (1984). *The Human Factor.* Digital Equipment Corporation.

Schneiderman, B. (1987). *Designing the User Interface: Strategies for Effective Human-Computer Interaction.* Addison-Wesley.

Smith, S.L. & Mosier, J.N. (1986). *Guidelines for Designing User Interface Software*, Mitre Corporation (ESD-TR-86-278).

Subasic, K., Robertson, J., Langston, D. & Grantham; D. (1988). *A Guide to Andrew*, Information Technology Center.

Thomas, F. & Johnston, O. (1984). *Disney Animation: The Illusion of Life.* Abbeville Press.

Wadlow, M.G., Haas, C., Crumley, P.G. & Boyarski, D. (1988). *User Interface Guidelines for the Andrew System*, Information Technology Center.

Wadlow, M.G., Haas, C., Boyarski, D. & Crumley, P.G. (1990). Helping the User by Helping the Developer: The Role of Guidelines in a Multimedia Context, *CHI '90 Conference Proceedings*, 1990.

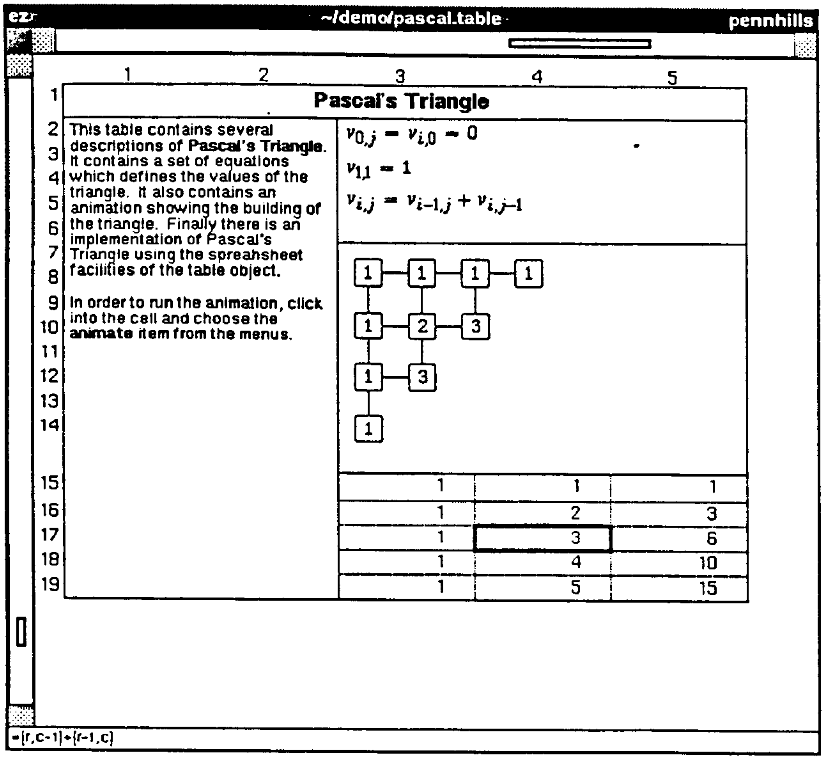

Figure 1. Ez, the multimedia document editor for Andrew.

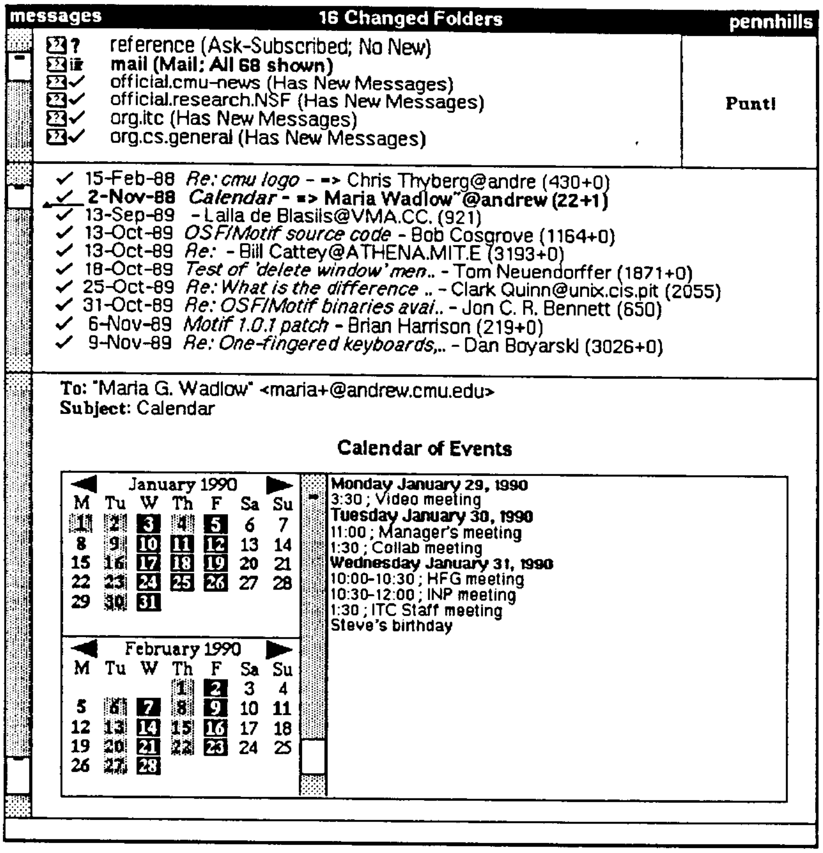

Figure 2. Messages, the multimedia electronic mail and bulletin board system for Andrew.

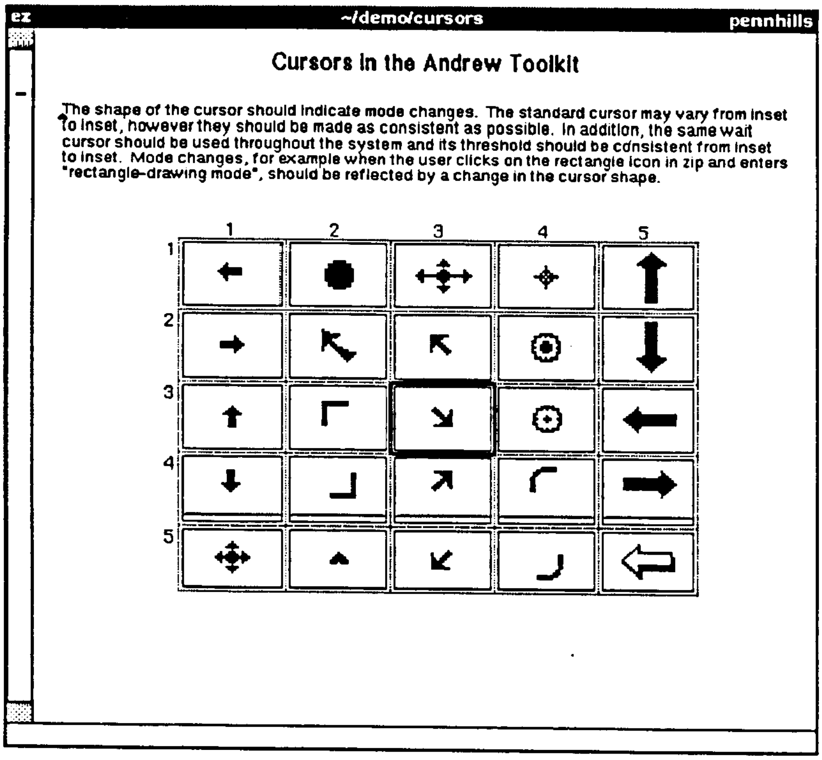

Figure 3. An Ez window displaying a raster image, embedded within a table, nested in a text document.

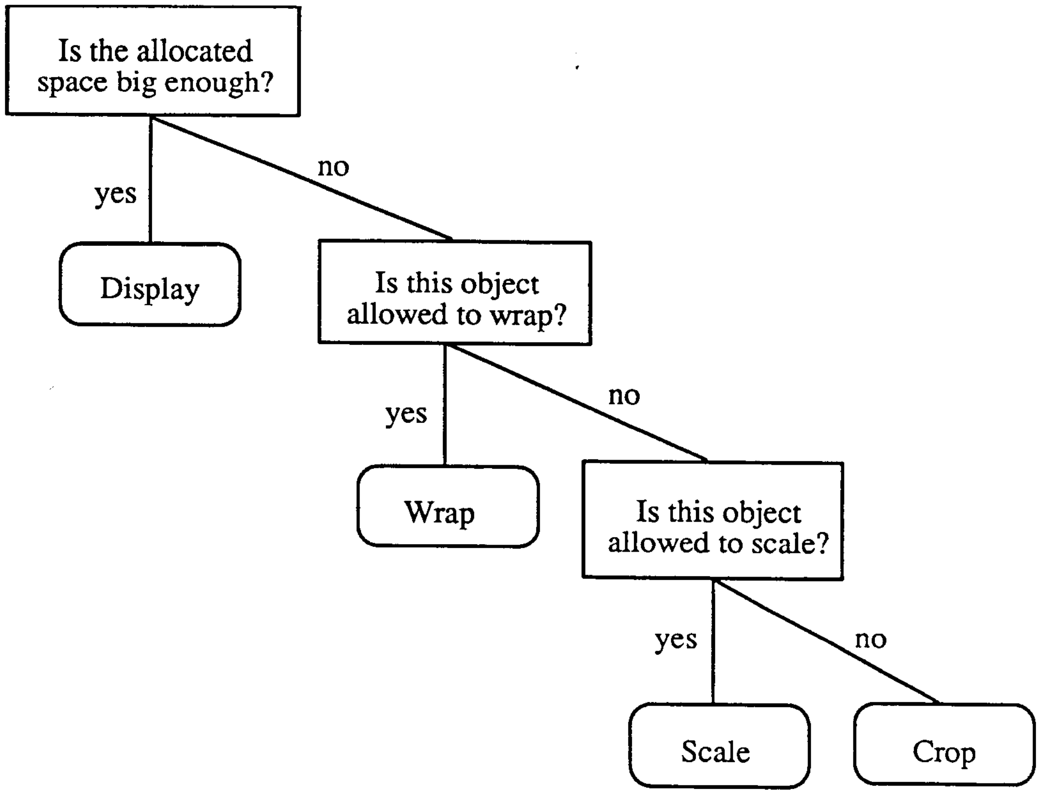

Figure 4. A decision tree to assist developers in understanding the intended algorithm for space allocation.