A GRID-BASED APPROACH TO AUTOMATING DISPLAY LAYOUT

Steven Feiner

Department of Computer Science
Columbia University
New York, NY 10027

Abstract

A research testbed is described for exploring the automated layout of graphical displays. We address the problem of determining the size and position of the objects displayed to a user. Our approach is based on the graphic design concept of a design grid. A design grid is a set of proportionally-spaced vertical and horizontal lines that control the position and size of the objects being laid out. The system first generates a grid intended for a set of possible displays, based on information about the kind of material to be displayed, the user, and the display hardware. The grid is next used, in conjunction with further information about the kinds of objects to be presented, to create a prototype display layout. This prototype display layout determines how each actual set of objects will be sized and positioned in the displays presented to the user.

Keywords: user interface design, graphical layout, design grids

1. Introduction

Recent experiments in interface design have explored the use of direct manipulation graphical editors to lay out an application's displays [HANA80; FEIN82; WONG82; BUXT83; GREE85; OLSE85; HOLL86; MYER86]. Ideally, these systems allow a user to design an interface that has exactly the visual appearance that he or she desires. They have shown some dramatic results in allowing users, both programmers and nonprogrammers, to design certain kinds of interfaces in less time than it would take using conventional methods.

There are several problems, however, with the editor-based approach to interface design. First, most users are unfortunately not experienced graphic designers, let alone interface designers. That the system does exactly what its designer wants is therefore not enough: the designer has to want the right interface to begin with. It is true, however, that by making it easier to create and modify an interface, successive refinement is encouraged. Therefore, the designer may be more willing to change the interface in response to user pressure (or their own attempts to use it). Early editor-based systems provided only the simplest of layout assistance, such as the user-specified layout grids of IGD [FEIN82]. In contrast, more recent projects, such as Peridot [MYER86] and Designer [HOLL86], have begun to address this problem by monitoring the designer's actions and offering rule-based suggestions to help the user craft a better display.

Second, although the information to be presented and its general format may be known in advance, there may be a large, heterogeneous user community and a diversity of situations under which the material is to be viewed. One common approach attempts to take this into account in the initial design phase by providing a fixed set of different display types. Unfortunately, this may still result in grouping users and situations in overly large equivalence classes. Users and situations differ in seemingly small, but important, ways and our systems should be able to adapt to these.

Finally, there are many situations, such as command and control, and maintenance and repair, in which diverse, and sometimes unexpected, information must be presented on the fly. Here, timely presentation of information is essential, yet there may be no time for the presentation to be developed in advance.

In recognition of these problems, a number of researchers have investigated automated generation of both the form and content of graphical presentations, representative of which are [FRIE84; FEIN85a; MACK86]. There is a host of difficult problems associated with tasks such as determining what information is to be presented, when it is to be presented, what format it will be presented in, and what kind of user input will be accepted. In contrast, the system described here concentrates only on the layout of separately generated information. Thus, we assume that the specific material to be laid out will be provided as input.

2. Automating layout

At the core of the graphics layout problem that we are investigating is the task of determining the positions and sizes of a set of graphical objects. In the work described here, we use information about the objects to be displayed, the user, and the display. We intentionally ignore the possibility of interactions between layout generation and content generation.

This work is supported in part by the Defense Advanced Research Projects Agency under Contract N00039-84-C-0165 and the New York State Center for Advanced Technology under Contract NYSSTF-CAT(87)-5.

Our implementation requires that objects to be laid out are nonoverlapping upright rectangles. Even with these restrictions a layout problem may be quite costly to solve. Beach [BEAC85] has pointed out that determining whether an arbitrary set of nonoverlapping rectangular objects can fit on a display is *NP*-complete, as is finding the minimum size rectangle in which the objects can be packed. Although the layout resulting from a space minimization strategy alone may be quite space-efficient, it may also be difficult to use and understand. The challenge is to develop a set of constraints and evaluation criteria that will result in the generation of effective layouts, while simultaneously restricting the possibilities that must be considered. Happily, there are proven approaches to display layout in which arbitrary sizing and placement of objects are expressly prohibited.

3. Grid-based layout

Grid-based layout has provided a particularly influential and effective framework for graphic design [HURL78; MULL81]. In this design technique, the designer divides the design space with a rectangular *design grid*, whose lines are positioned in proportions based on the size of the space, the material being laid out, and the purpose for which the layout is designed. When layout is performed using the grid, objects are typically sized and positioned so that they are aligned with the grid lines and occupy an integral number of grid fields both horizontally and vertically.

Design grids often consist of a set of regularly spaced vertical and horizontal lines that describe a set of equal-sized rectangular grid *fields*. The fields are separated vertically and horizontally by equal-sized spaces and the entire array of fields is surrounded on the display by top, bottom, and side margins. In general, fields need not be of equal size, but there are many designers who follow these restrictions, in some cases further constraining the fields to be square.

Grid-based layout has been used extensively for magazine and newspaper design. In these applications, a single grid is developed for the generic material to be laid out and then is used for each page. The concept of the design grid has been adapted by Friedell [FRIE84] for positioning groups of icons and by Beach [BEAC85] for table layout. In Beach's system, a table is laid out from a user-provided specification of its contents, which includes the column and row position of each item in the table. Given these relative item positions, a constraint-based system determines the position and size of the columns and rows, guided by a style sheet specified for the layout.

Our system generates the original grid itself and completely determines the mapping of objects to positions in the grid. First a grid is created, based on information about the material to be laid out, the display, and the user. In part the information consists of a grammar, discussed later, describing the kind of material to be presented in actual displays. The actual objects that will be encountered in a particular display may be thought of as instances of these general classes of objects whose properties and relationships are input during the grid design phase. The system currently supports pictures, text blocks, and headings, which the user must further specialize by designating limits on their expected size and contents.

Next, the grid that the system produces is used in conjunction with information about the material to generate a *prototype display layout*. This prototype is then used to determine how to lay out input instances of the objects described by the display grammar to form actual displays.

By generating a grid first and using it to produce multiple layouts, we gain one of the important advantages of grid-based design: consistency. Each layout of a set is not optimized as an individual design problem, but bears a visual relationship to the others. Not only do we gain efficiency in not having to redesign each display afresh, but the use of a common layout format visually enforces the relationship between the displays. Furthermore, the regular spacing of the grid, and hence the regular sizes and positions of the objects embedded in it, also helps achieve a coherent, consistent look for an individual display [MULL81].

4. Designing the grid

Our system is a rule-based testbed that embodies a vastly simplified version of one of many possible approaches to display layout. It begins by determining the size of the individual grid field that will serve as the building block from which a grid of uniform-sized fields will be constructed. The field's size is derived from information about the pictures and text to be laid out and the user's position.

The distance at which the display will be viewed is used, in conjunction with legibility rules, to determine the point size to be used to set body text. This in turn determines the leading or vertical space between successive lines of type. The point size and leading for headings is determined similarly. Rules for legibility further determine an appropriate line length and hence the width of a text block. Information about the expected character count of textual material that will be included in the displays allows the system to determine an approximate number of lines, and hence an expected height for the text block.

The input includes a normalized size for the pictures. This normalized size specifies the minimum width and height that the picture must have in order to be understandable when viewed at a set distance. In conjunction with the viewer's distance it determines the actual minimum size at which the picture must be reproduced.

In the uniform-size grid field design scheme adopted, the field size is determined by the size of the smallest picture or text block to be laid out, further constrained so that the field is tall enough to hold an integral number of lines of text. In the layout style espoused by [MULL81], and currently enforced by our system, the ascender of the topmost line of type in a field is set flush with the top of the field, while the descender of the bottommost line of type in the field is set flush with bottom of the field. All lines of type are separated vertically by the previously determined leading. The vertical space between the grid fields must also hold an integral number of lines of text, although it includes leading above and below the first and last text lines respectively. This approach allows a passage of text to span multiple vertical grid fields, while still maintaining the same relationship to the top and bottom lines in each full grid field. The result of these constraints can be seen in the figures presented below.

If a picture does not exactly span an integral number of grid fields, it must be further scaled and/or cropped. In previous work, we explored the automated design of sequences of pictures that explain how to perform actions in a 3D world [FEIN85b]. Each of these pictures included information about the extent that bounds the essential material in the picture that must be displayed. The pictures discussed here are assumed to have this information. Cropping thus involves uniform scaling of the material in the extent if the aspect ratio will correctly span full grid fields or actual expansion of the extent vertically or horizontally. An interesting issue not addressed here is whether additional information or background should be shown by expanding the extent or whether the picture can be generated with an aspect ratio that takes into account knowledge of the grid design.

The horizontal space between horizontally adjacent grid fields must be wide enough to separate objects, such as columns of text, from each other. Furthermore, the array of grid fields containing the text and pictures is offset from the top, bottom, and sides of the display by margins. The display size is given as part of the input to the system. The size of the margins are currently set according to a standard ratio. Since proportionally-spaced fonts are used and the exact text being set is not yet known when the grid is being defined, there is some leeway in adjusting the width of the grid fields in conjunction with the margins and grid field horizontal spacing. Thus, extra slack can be distributed among the horizontal space between fields, the margins, and the width of the grid fields. Figure 1 shows the grid designed for a display whose size is indicated by the outermost rectangle. This is a scaled-down version of a grid designed for an 8½" by 11" display to be viewed at a distance of 20". Note that the lines of the grid are not actually drawn in the finished display.

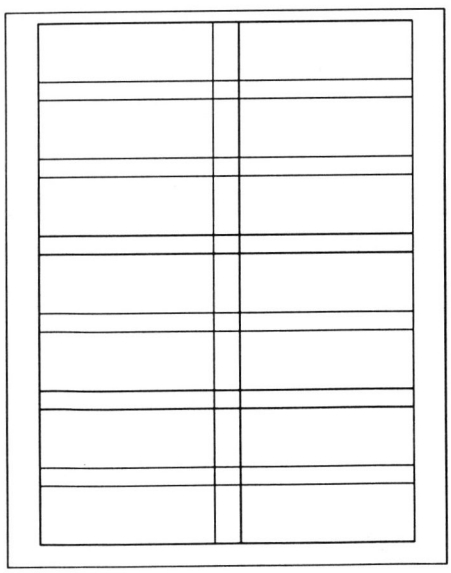

Figure 1: A scaled-down version of a grid designed for an 8½" by 11" display to be viewed at 20". The outermost rectangle defines the display's boundaries.

5. Prototype display grammar

The input used in designing the grid describes subclasses of pictures, text blocks, and headings by specifying bounds on their expected size (number of characters for text blocks and heads and normalized size for pictures). These are prototypes of the objects that will actually be laid out. In addition, these prototypes may be grouped together to define aggregates. Different kinds of groups may be specified, indicating the relationships that hold between their members. Groups are currently constrained to form a single hierarchy with arbitrarily deep nesting. In practice, however, display layouts (as opposed to the complex diagrams that they may contain) do not seem to evidence very deep nesting.

Our current system allows distinguishing whether a group's objects form an ordered or unordered set. For example, a collection of pictures may be an unordered set, while a set of pairs of pictures and text illustrating the steps of a maintenance and repair task may form a sequence. Groups are also used to perform alternation and repetition. Groups thus function as the operators at the internal nodes of a syntax tree whose leaves are the various kinds of pictures, text blocks, and headings from which a display may be constructed.

At the highest level, the entirety of the material to be displayed is organized as a single group. This may be thought of as a *prototype display grammar*, an example of which is shown pictorially in Figure 2. The prototype display grammar defines the underlying logical structure of a class of actual displays whose contents will be input later. The next step is to develop a prototype display layout.

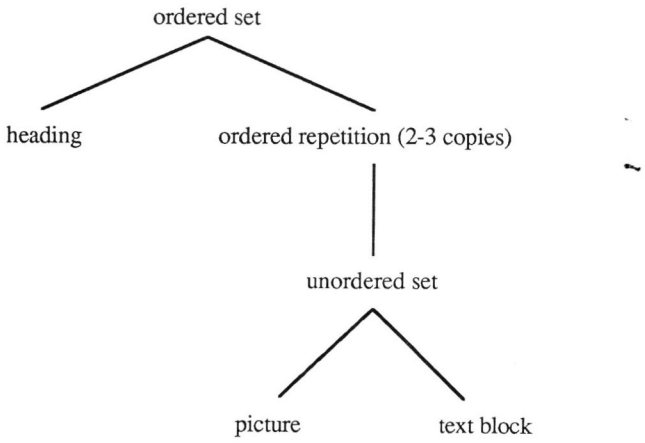

Figure 2: A pictorial representation of a prototype display grammar.

6. Designing a prototype display layout

We have designed a set of rules for each of the different kinds of grouping strategies that determine how their components should be laid out. For example, all of the elements in a sequence will be laid ordered either vertically or horizontally across the page.

The layout algorithm employs a generate and test strategy, traversing the prototype tree bottom-up from the leaves. At each node a set of layout alternatives is generated, based on the layout of its children. The first (and current) algorithm employed initially excludes from generation only alternatives that exceed the bounds of the grid. It generates the entire space of design alternatives before selecting an alternative for each node. The evaluation criteria with which we are experimenting favor designs in which identical elements in a sequence are laid out similarly, and in which horizontal and vertical layout approaches alternate down a branch.

7. Laying out an actual display

Creating a prototype display layout does not produce any graphical output. After the prototype display layout has been created, it can be used to determine the layout of one or more actual displays, based on a description of the input objects of which they are composed. This input consists of a list of object instances, the prototype class with which each is associated, and the actual contents of each instance (which must be consistent with the originally provided descriptions of their prototypes). Each object is then sized and positioned using information generated for its class during the creation of the prototype display layout. Members of sequential groups are processed in sequence to allow the layout of later objects to depend on the positions and sizes of earlier ones.

Figure 3a shows the grid of Figure 1 populated with a set of objects that are accepted by the prototype display grammar of Figure 2. In the current implementation the pictures are grey tone rectangles whose size and placement are determined by the system, while text is represented by hardwired sentences that the system positions and generates in the appropriate font, point size, and leading. Figure 3b shows the display without the grid visible, as the viewer would see it. Figure 4 shows the same set of objects laid out in different sized displays with different grids. All of the figures are generated for an observer at 20" and have been scaled down to ¼ of their actual size.

8. Implementation

The majority of the layout system is implemented in OPS5, a production system language [FORG81]. The drawing routines are written in Lisp and generate a human-readable intermediate file that forms a device-independent representation of the grid and the specific sets of picture and text that are laid out using it. The intermediate file is further processed for interactive display on a bitmapped workstation. It can also be automatically converted to PostScript [ADOB85], which was done to produce the figures included in this paper.

Eventually, we intend to lay out actual pictures and text generated by a companion project [FEIN88]. Therefore, there has been no emphasis on providing other than a program interface for specifying the information needed to build the grid or to describe the contents of a display.

9. Conclusions

We have described the beginnings of a testbed system for investigating the automated layout of graphical displays. The work described here is a preliminary implementation of one part of an architecture for generating both layout and information content automatically [FEIN88]. It has been used to explore the rule-based generation and use of a graphic design grid that governs the display layout process. Because the system is intended to be provided with the actual items to be laid out, it is not responsible for choosing the high-level display design style that determines the identities of these objects.

The current implementation has a number of serious limitations which we intend to address. Many of these are caused by the graphic design rules that the system uses, which are extremely rudimentary and often vastly oversimplified. For example, a number of decisions, such as font choice, are stated a priori. As well, the system has no concept of design basics such as visual balance or rhythm. One area of particular interest is that of design compromises. For example, if the minimum legible point size for a given viewer distance and font causes a block of text to be set so large that it won't fit on a small display, the system currently fails to develop a design, instead of producing an inferior one.

Nonhierarchical layout constraints are not currently provided, so there is no way to indicate that the same layout decisions should be used in disjoint parts of a display. Thus, two groups whose components have identical descriptions may be laid out in totally different ways. As well, the primitives implemented so far must be augmented to include input as well as output primitives.

We are also trying to develop strategies for prototype layout design that involve more careful pruning of the layout alternatives generated, backtracking to avoid the exponential growth of the design search space, and improved criteria for evaluating design alternatives. Although the current system can handle only an extremely small subset of designs, it was intended to provide a framework in which to develop ideas that could help point the way toward future, more powerful systems.

10. Acknowledgements

Mary Jones helped critique some of the system's first layouts and provided valuable suggestions for improvements, most of which still remain to be made. The Hewlett-Packard Company, through its AI University Grants Program, generously donated the equipment on which the testbed is being developed.

References

[ADOB85] Adobe Systems Inc. *PostScript Language Reference Manual.* MA: Addison-Wesley, 1985.

[BEAC85] Beach, R. *Setting Tables and Illustrations with Style.* Ph.D. Thesis, Dept. of Computer Science, University of Waterloo, Ontario, 1985. (Xerox PARC Report CSL-85-3, May 1985).

[BUXT83] Buxton, B. "Toward a Comprehensive User Interface Management System." *Computer Graphics*, 17:3, July 1983, 35-42.

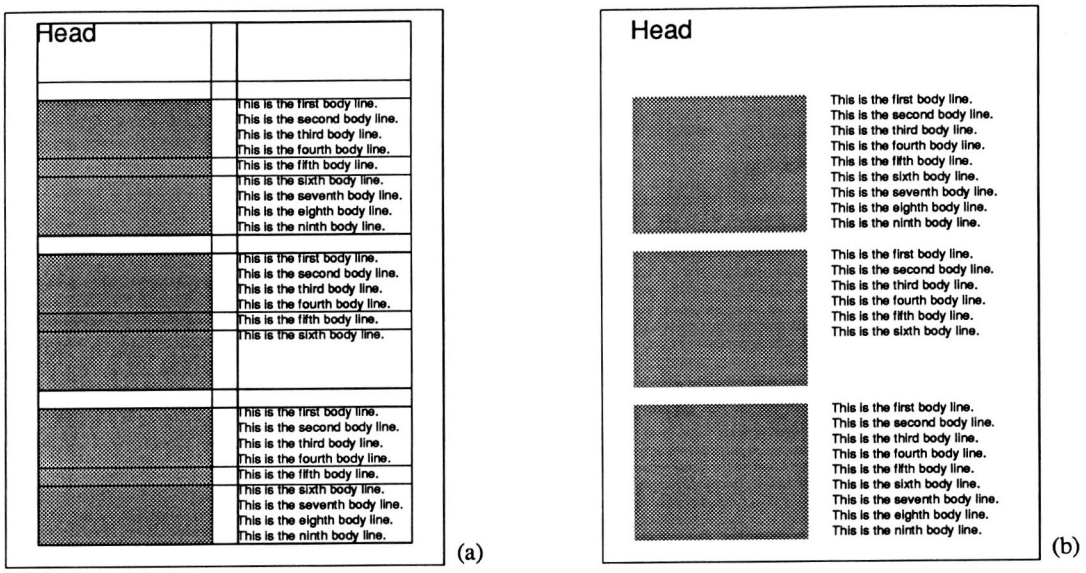

Figure 3: (a) The grid of Figure 1, populated with a set of objects accepted by the prototype display grammar of Figure 2. (b) The display as presented to the user.

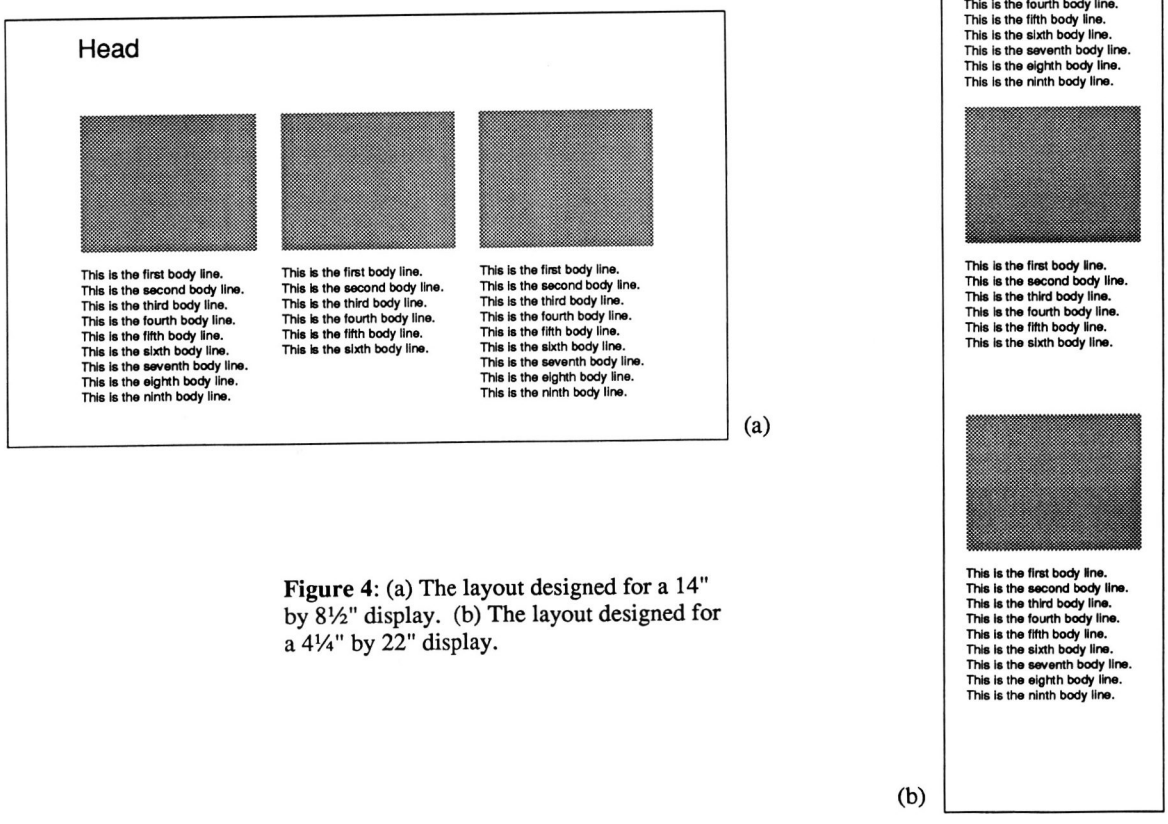

Figure 4: (a) The layout designed for a 14" by 8½" display. (b) The layout designed for a 4¼" by 22" display.

[FEIN82] Feiner, S., Nagy, S., and van Dam, A. "An Experimental System for Creating and Presenting Interactive Graphical Documents." *ACM Trans. on Graphics*, 1:1 January 1982, 59-77.

[FEIN85a] Feiner, S. "Research Issues in Generating Graphical Explanations." *Proc. Graphics Interface '85*, Montreal, May 27-31, 1985, 117-123.

[FEIN85b] Feiner, S. "APEX: An Experiment in the Automated Creation of Pictorial Explanations." *IEEE Computer Graphics and Applications*, 5:11, November 1985, 29-38.

[FEIN88] Feiner, S. "An Architecture for Knowledge-Based Graphical Interfaces." *Proc. ACM/SIGCHI Workshop on Architectures for Intelligent Interfaces*, Monterey, CA, Mar 29 - Apr 1, 1988.

[FOLE82] Foley, J. and van Dam, A. *Fundamentals of Interactive Computer Graphics*. MA: Addison-Wesley, 1982.

[FORG81] Forgy, C. *OPS5 User's Manual*. Computer Science Technical Report CMU-CS-81-135, Carnegie-Mellon University, July 1981.

[FRIE84] Friedell, M. "Automatic Synthesis of Graphical Object Descriptions." *Computer Graphics*, 18:3, July 1984, 53-62.

[GREE85] Green, M. "The University of Alberta Use Interface Management System." *Computer Graphics*, 19:3, July 1985, 205-213.

[HANA80] Hanau, P. and Lenorovitz, D. "Prototyping and Simulation Tools for User/Computer Dialogue Design." *Computer Graphics*, 14:3, July 1980, 271-278.

[HOLL86] Hollan, J., Hutchins, E., McCandless, T., Rosenstein, M., and Weitzman, L. "Graphical Interfaces for Simulation." In W. Rouse, ed., *Advances in Man-Machine Systems*, vol. 3, Greenwich, CT: Jai Press, 1986.

[HURL78] Hurlburt, A. *The Grid*. NY: Van Nostrand Reinhold Co., 1978.

[MACK86] Mackinlay, J. "Automating the Design of Graphical Presentations of Relational Information." *ACM Trans. on Graphics*, 5:2, April 1986, 110-141.

[MULL81] Müller-Brockmann, J. *Grid systems in graphic design*. Niederteufen, Switzerland: Verlag Arthur Niggli, 1981.

[OLSE85] Olsen Jr., D., Dempsey, E., and Rogge, R. "Input/Output Linkage in a User Interface Management System." *Computer Graphics*, 19:3, July 1983, 191-197.

[MYER86] Myers, B. and Buxton, B. "Creating Highly-Interactive and Graphical User Interfaces by Demonstration." *Computer Graphics*, 20:4, August 1986, 249-258.

[WONG82] Wong, P., and Reid, E. "FLAIR – User Interface Dialog Design Tool." *Computer Graphics*, 16:3, July 1982, 87-98.