# RESEARCH ISSUES IN GENERATING GRAPHICAL EXPLANATIONS[†]

*Steven Feiner*

Department of Computer Science
Box 1910
Brown University
Providence, RI 02912

**Abstract**

Advances in computer graphics have led to the ever-increasing ability to depict objects realistically, while work in expert systems has resulted in the capacity to model objects and their interactions well enough to solve some difficult problems. One application in which the power of both disciplines can be coupled is the generation of graphical explanations that depict the performance of actions on objects.

This paper introduces a high-level conceptual architecture for automatically generating presentations. It serves as an organizing framework in which to examine some of the problems that we have encountered in implementing APEX, a test-bed system for creating and laying out pictures that depict actions performed by a problem solver.

**Keywords:** picture synthesis, graphical explanations, expert systems, graphic design.

## 1. Introduction

Much work in computer graphics is concerned with methods for depicting objects with increasing accuracy and efficiency [Cook, Porter, and Carpenter 84; Nishimura et al. 83]. At the same time, artificial intelligence researchers are developing expert systems and problem solvers that model knowledge about how to manipulate objects to solve difficult problems [Hayes-Roth, Waterman, and Lenat 83]. Although there remains a multitude of deep, unsolved problems in both these domains, their potentials are clear enough to suggest one fruitful path of exploration: coupling the decision-making expertise of a problem solver to the graphical output capabilities of modern rendering hardware/software.

Several existing research systems have examined some of the problems engendered by this coupling. AIPS [Zdybel et al. 81] provides facilities for describing and modeling classes of displays that can be generated automatically from information in a knowledge base. The SDMS View Generator [Friedell 84] automatically creates and lays out iconic displays in response to user database queries. GAK [Neiman 82] generates animated vector pictures to accompany a synthesized verbal explanation of how to operate a CAD system. Although the View Generator embodies a sophisticated method for generating certain kinds of iconic displays, it does not address the problems of creating a series of displays as part of an extended explanation. GAK, on the other hand, creates its animations by positioning simple 2D vector representations of objects – the picture is merely the visual side-effect of executing an action, such as translation, on the world model.

## 2. A conceptual architecture for generating graphical explanations

The systems that we would like to build would mediate between an expert system that knows how to manipulate objects in some problem domain and graphics software that knows how to render them. Such systems would produce pictures in the service of explanation, with attention paid to graphic design, both of the individual pictures and of the entire presentation within which they reside. We will call a system that accomplishes this a *communicator*.

Consider a communicator that would show its user how to perform a series of actions on a world of 3D objects. Its input would be derived from a problem solver that is presented with a problem to be solved through the use of knowledge about the state of its world and rules for manipulating it. The communicator would have access to information about this world of objects including their size, shape, position, material, and relationships to each other. It must also be told about the actions to be performed and the changes they effect in the objects. On the other side of the gulf that our communicator must bridge is a graphics system capable of rendering text, simple 2D primitives, and a full range of 3D objects. The graphics system need only be told the precise description of each object to render and the viewing

[†]This work was supported in part by the Office of Naval Research under Contract No. N00014-78-C-0396.

parameters, lighting model, rendering style, and camera characteristics.

The communicator will populate the available display space with graphical objects of various sorts: realistic pictures, diagrams, charts, graphs, text. Some may be static, others moving. One approach is to present information as a sequence of displays, each different from the last, like the pages of a book. Alternatively, we may think of a single display whose elements change either together or individually, by replacement or by gradual metamorphosis, during the course of the presentation. It is the task of the communicator to generate the contents of the display space and determine their positions in both time and space. This job may be thought of as having four parts: presentation design, story-boarding, picture/text specification, and spatial ("page") layout. Note that these are not necessarily sequential and may interact. They are sketched briefly below.

*Presentation design.* Given knowledge about the user, the kinds of tasks to be explained, and the size and capabilities of the display, the system should design a format for the presentation that will determine its general appearance. The presentation design would include rules for the spatial and temporal layout of objects, as well as the style in which they are rendered, all selected to create an effective and consistent presentation [Marcus 84].

*Storyboarding.* As a problem solution is developed, the communicator must decide on the order in which information is to be presented and how much is to be presented at a time. This process may be thought of as a sort of "conceptual pagination" in which the material to be communicated is "chunked" into (possibly overlapping) displayable units.

*Picture/text specification.* Each concept to be presented must be turned into specifications of the appropriate pictures and text in accordance with stylistic rules decided upon during presentation design. The knowledge for specifying each kind of output might be maintained by a group of cooperating expert processes. In the case of picture specification, the communicator must determine the objects to depict; select the viewing parameters, rendering style, and lighting model; and specify graphic devices such as callouts and leader lines. Together these constitute an explicit description that can be handed off to graphics software that will do the actual scan conversion.

*Spatial layout.* Text and pictures generated for the display must be laid out when needed according to the presentation design rules.

Although the tasks described above are all concerned with the creation and layout of output, an interactive system must also process user input. A good communicator must relay the user's actions back to the problem solver to affect its knowledge of the world and allow it to answer questions, tasks fraught with as many difficulties as those of output.

## 3. Research issues

In the previous pages we have sketched out a high-level conceptual architecture for an interface between an expert system and graphics software. Our attempts at implementing some fraction of its picture generation capabilities have presented a number of interesting problems, of which the following are representative. We have divided them into presentation level, display level, and picture level concerns. Although emphasis here is on picture design, many of these problems are similar to those encountered in text generation research [Mann 82].

### 3.1. Presentation level problems

*The structure of an explanation for a problem need not parallel that of the actions which must be performed to solve it.* For example, one might first want to inform the user of how much time a procedure may take and what tools will be required to perform it.

*Superfluous presentation detail should be suppressed.* The system should have criteria to determine the level of detail at which actions should be explained, based in part on a model of what the user knows [Cullingford et al. 81].

*The presentation should be consistent.* Style should be consistent throughout the presentation as well as consistent with other presentations designed by the system. Pictures should be internally consistent – the same techniques should be used to show the same things (e.g., motion). Consistency should be maintained among pictures as well.

### 3.2. Display level problems

*The presentation design affects how much can be said at a time and how it can be said.* A change in the display size should not always result in a simple scaling of the display layout and its contents. For example, if a picture is resized, its design may have to be changed to ensure legibility. Although one picture may suffice to show an object in context on a large display, an additional "detail inset" or a sequence of pictures may be required if the picture is scaled down for a smaller display.

*The system should consider the tradeoffs of using different presentation formats for the same information.* Sometimes it is not a matter of pictures vs. words, but rather what kind of picture to select. [Mackinlay and Genesereth 84] provide examples of one scheme for choosing between alternative diagrammatic languages based on how parsimoniously they can express a set of facts.

### 3.3. Picture level problems

*What objects should be included in a picture?* Suppose a picture is meant to show a person that he or she is to turn a knob on an unfamiliar piece of equipment. One solution might be to include in the picture the knob, the equipment on which it is located, and, perhaps, the hand turning it. If the equipment is the only thing in the room and the knob the only feature on it, this might be appropriate. But, what if the equipment has many knobs and switches and the room contains many similar pieces of equipment? There must be a compromise between the overwhelming detail of showing with photographic accuracy everything that a person might see and the potential ambiguity of showing only those objects that participate in the action. We would like the contents of the picture that we generate to depend on whether the person knows the location of the knob on the equipment or even the location of the equipment itself. If objects are thought of as existing in a 3D hierarchy, then including in a picture objects that are at a level equal to or above those already included can provide context.

*What level of detail should be included in a picture?* Pictures should have unneeded details suppressed and important ones emphasized [Magnan 70]. Whether details are necessary depends upon the user's knowledge of the objects being depicted. Just as adding objects to a picture can provide context, adding details selectively can help disambiguate an object from others with which it may be confused.

*How should motion be indicated?* Although animation is an attractive possibility, especially for complicated actions, actual motion can be needlessly distracting. Consider depicting the simple task of turning a handle for someone who only needs to know the direction to turn. A well-placed arrow pointing in the appropriate direction might be better than an animated movie which included unneeded information on *how* to turn it. There are a variety of other techniques for depicting motion in static pictures [Friedman and Stevenson 80]. These include an object's characteristic posture or context (a bird in mid-air or dust kicked up behind a horse), the same object shown in multiple positions (an arm in different stages of an action), metaphoric blur lines behind a moving object, and abstract lines or shapes representing motion.

*How should other invisible properties or actions be depicted?* A variety of other properties and actions that are not normally visible might be depicted. For example, a broadcast tower in operation might be shown by jagged lines emanating from an otherwise realistically depicted antenna. There is also a large grey area between realistic pictures and schematic representations that includes exaggerated cartoon representations [Perkins and Hagen 80].

*How should the contents of pictures be laid out?* When generating schematic diagrams or adding "meta-objects" such as arrows or callouts to pictures, decisions must be made as to each object's properties and position. [Friedell 84] describes the tradeoffs between reversing the order of constructing and laying out objects in displays. Meta-objects themselves may have to be differentially scaled depending on how they are used. For example, a 3D arrow that looks good in plan view may need to be proportioned differently if it is to be equally effective when viewed from an angle.

Although the proposed architecture assumes that only the problem solver can modify the position of objects that exist in its world, the communicator might suggest to the problem solver that it plan a solution that allows better explanations to be generated. For example, if the problem solver modeled the motions of the person performing a task, it might slightly exaggerate certain movements for pedagogical purposes, as a human instructor might. When alternative methods of performing a task are available, it might choose between them guided by suggestions from the communicator about which will make the explanation clearer.

*The contents of pictures must take into account what has already been explained.* Consider a sequence of pictures that depicts opening a door, performing a number of intervening actions, and opening a second door. If the next picture were to show the closing of the second door it might require less context than if it were to depict the closing of the first.

## 4. APEX: An experimental picture generation and layout system

APEX [Automated Pictorial EXplanations] has as its long-term goal the realtime computer generation of effective graphical explanations. Work on the project grew out of earlier research on systems for editing and

viewing pictorial documents [Feiner, Nagy, and van Dam 82]. APEX is being developed as a test-bed to investigate some of the problems described in the previous section.

The APEX test-bed incorporates the crude beginnings of one model for the creation of certain kinds of displays that show actions, such as pushing, pulling, or turning, being performed on objects. It uses rules to govern each aspect of a picture's composition by determining what objects it will depict, what rendering style will be used for each, what viewing specifications will be used, etc. We have attempted to eliminate unneeded detail, while emphasizing important features. For example, detail is added to a picture where it is determined that it helps disambiguate an important object from others with which it may be confused.

### 4.1. APEX's world

APEX is initially provided with information about a world of objects, the actions to be performed, and what the user already knows.

*Actions:* The actions to be depicted are performed by problem solvers written in micro-Nasl/Frail [Charniak, Gavin, and Hendler 83]. They use rules about the task to be performed and knowledge about the current state of affairs in order to break a high-level task into a hierarchy of lower-level tasks. Each action has associated with it information about the important objects that participate in it and the nature of their roles, any other actions that might have to be performed to accomplish it, when it should be executed relative to the other actions, and those changes that it effects in the environment.

*Objects:* Objects are hierarchically structured as a tree of 3D parts. Leaf nodes are physical objects with properties such as material, color, size, shape, and position, while internal nodes are assemblies (groupings) of leaf and internal nodes. Objects have information about their function and are also characterized by the relationships (such as "on" and "in") that they bear to one another. Figure 1 shows the objects in one world of electronic systems that APEX knows about.

*User knowledge:* Associated with each object and action is information about what the user knows of it (e.g., an object's approximate location).

### 4.2. Depicting an action

APEX considers actions in the order in which they are to be performed. It is told which objects play important roles in each action that it knows how to depict (currently those that manipulate objects directly). These are the objects about which the picture will crystallize. First, APEX adds the important objects to the picture. This involves modifying a data structure in which the picture is represented – no drawing is actually done at this point. APEX next searches near each important object for other objects that are roughly similar to it, and thus ones which the user may potentially confuse with it. These similar objects will be included in the picture. They are compared with the important object and their differences and similarities noted. Information gathered during comparison is used later to determine how the object and its parts will be rendered, with more detail called for when differences are found at a lower level. APEX also looks for "landmark" objects that are sufficiently different from other objects that they may be included in pictures as useful reference points for locating the important objects. Landmarks are also compared with objects to which they are similar so that disambiguating detail may be included if deemed necessary.

An object detail removal process is being developed to speed up the comparison process and make it possible to render objects at different levels of detail. Detail removal associates with each non-leaf object a simplified version of the object properties possessed by its children. This makes it possible to draw an object or compare it with another at a high level without processing its children, selectively progressing down the tree only where more local detail is desired.

As objects are added, viewing specifications are calculated that take into account the kind and extent of the objects included and the position of the viewer.

Other objects may be added to the picture. For example, if an object is supported by another and the supporting object would be visible with the current viewing specifications, then it is included (at its least detailed level).

APEX knows how to indicate motion (specified by the action being depicted), currently by creating and including in the picture an arrow in the direction of motion.

### 4.3. Rendering the picture

The picture is rendered by processing the picture's data structure in conjunction with information about the objects themselves, including that accumulated during comparisons. Objects that have been included in the picture are inspected and rules determine what rendering style should be used for each and whether their children should be inspected. For example, an

object is highlighted if it is one of the important objects directly specified by the action. The result of processing the picture data structure is a precise picture specification in a format accepted by a 3D rendering package [Strauss, Shantzis, and Laidlaw 84].

### 4.4. Examples

Figures 2 and 3 are pictures that APEX generated for opening and closing the drawer of the middle of the three large cabinets shown in Figure 1. Before generating the pictures, APEX was told that the user's only knowledge of the cabinet was that it was among the objects of Figure 1. Figure 2 shows the drawer being opened. The important objects in both figures are highlighted, while auxiliary objects are subdued. Detail has been included when necessary to disambiguate depicted objects. For example, in Figure 2 the drawers and doors of the large cabinets on the side are shown, but the small cabinet on the wall is given no extra detail since it was determined to be relatively too small to cause confusion. An arrow indicates the intended motion of the drawer. In Figure 3, the user is now presumed to know the location of the drawer, so additional objects are not necessary. The cabinet is included, however, since it supports the drawer.

### 4.5. Implementation

The APEX test-bed is written in Berkeley UNIX 4.2 Franz Lisp on a VAX 11/780. Although picture specifications are produced in a device-independent format, the current test-bed allows them to be rendered on a Lexidata Solidview Z-buffered graphics system as they are generated. The pictures in this paper were created by dithering this proof output to create bilevel bitmaps.

The first versions of APEX were implemented using Frail, but rapidly grew too slow to be manageable. For efficiency, the problem solvers are now run ahead of time for a particular problem and the results saved in a file that is read in by APEX before it generates pictures. The solution data structure is designed to allow APEX to incrementally recapitulate it and the changes that it makes in the world in the order in which they were originally performed.

APEX's rules are stored as bits of code that are triggered when certain data is modified or retrieved. For example, whenever an object is added to a picture its viewing specifications are updated automatically to accommodate it.

## 5. Conclusions

A high-level conceptual architecture has been presented for systems that would produce presentations depicting the actions performed by a problem solver. Some research issues were discussed that arose in attempts to implement parts of such a system. APEX, a research test-bed for creating pictorial explanations, models a small subset of some of the graphical design knowledge needed to generate pictures. Among its many current limitations are a restriction to cuboidal objects, and a lack of provisions to ensure that one (meta-)object does not obscure another from the selected viewpoint. Current work is directed toward laying out displays containing the pictures produced.

## 6. Acknowledgements

David Kantrowitz designed several different detail removal algorithms, and Alex Kass implemented an early version of the pagination and layout system; both helped shape the concepts discussed. Many thanks for ideas and suggestions are due to Leif Allmendinger, Barb Meier, Eugene Charniak, Aaron Marcus, Mihai Nadin, and Andy van Dam.

## 7. References

Charniak, E., Gavin, M., and Hendler, J. The Frail/Nasl Reference Manual. CS Tech. Report CS-83-06, Dept. of Computer Science, Brown University, Providence, RI, February 1983.

Cook, R., Porter, T., Carpenter, L. Distributed ray tracing. *Computer Graphics* (Proc. SIGGRAPH 84), 18:3, July 1984, 137-145.

Cullingford, R., Krueger, M., Selfridge, M., and Bienkowski, M. Towards automating explanations. *IJCAI 81,* Vancouver, B.C., August 1981, 362-367.

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.

Friedell, M. Automatic Synthesis of Graphical Object Descriptions. *Computer Graphics* (Proc. SIGGRAPH 84), 18:3, July 1984, 53-62.

Friedman, S. and Stevenson, M. Perception of movement in pictures, in *The Perception of Pictures,* Vol. 1, M. Hagen, ed., Academic Press, New York, 1980, 225-255.

Hayes-Roth, F., Waterman, D., and Lenat, D., eds. *Building Expert Systems.* Addison-Wesley, Reading, MA, 1983.

Mackinlay, J. and Genesereth, M. Expressiveness of languages. *Proc. AAAI 84,* 226-232.

Magnan, G. *Using Technical Art: An Industry Guide,* Wiley-Interscience, New York, 1970.

Mann, W. Text generation. *AJCL,* 8:2, April-June 1982, 62-69.

Marcus, A. Corporate identity for iconic interface design. *IEEE CG&A,* 4:12, December 1984, 24-32.

Neiman, D. Graphical animation from knowledge. *Proc. AAAI 82,* 1982, 373-376.

Nishimura, H., Ohno, H., Kawata, T., Shirakawa, I., and Omura, K. Links-1: A parallel pipelined multimicrocomputer system for image creation. *SIGARCH Newsletter* (Proc. 10th Int. Symp. Comp. Arch), 11:3, June 1983, 387-394.

Perkins, D., and Hagen, M. Convention, context, and caricature, in *The Perception of Pictures,* Vol. 1, M. Hagen, ed., Academic Press, New York, 1980, 257-286.

Strauss, P., Shantzis, M., and Laidlaw, D. SCEFO: A standard scene format for image creation and animation. Brown University Graphics Group Memo, Providence, RI, 1984.

Zdybel, F., Greenfeld, N., Yonke, M., and Gibbons, J. An information presentation system. *Proc. IJCAI 81,* Vancouver, B.C., August 1981, 978-984.

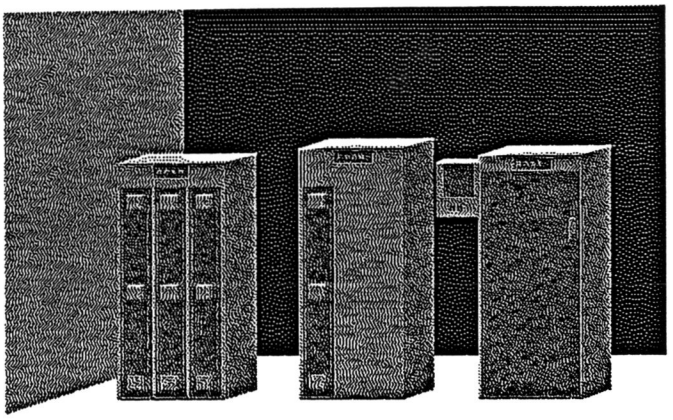

**Figure 1**: A fully detailed view of a set of objects for which APEX can create pictures.

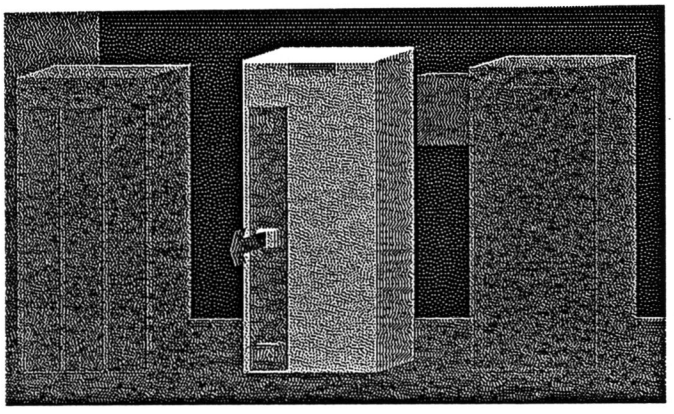

**Figure 2**: A picture created by APEX to represent the command to open the drawer of the center cabinet of Figure 1.

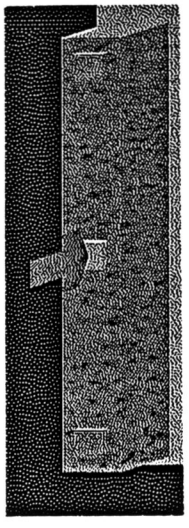

**Figure 3**: A picture created by APEX to represent the command to close the drawer shown in Figure 2.