

National Center for Geographic Information and Analysis

Visual Interfaces to Geometry

By

Werner Kuhn and Max Egenhofer

National Center for Geographic Information & Analysis
Department of Surveying Engineering
University of Maine, Orono ME 04469

Technical Report 91-18

July 1991

Simonett Center for Spatial Analysis
University of California
35 10 Phelps Hall
Santa Barbara, CA 93106-4060
Office (805) 893-8224
Fax (805) 893-8617
ncgia@ncgia.ucsb.edu

State University of New York
301 Wilkeson Quad, Box 610023
Buffalo NY 14261-0001
Office (716) 645-2545
Fax (716) 645-5957
ncgia@ubvms.cc.buffalo.edu

University of Maine
348 Boardman Hall
Orono ME 04469-5711
Office (207) 581-2149
Fax (207) 581-2206
ncgia@spatial.maine.edu

Preface and Acknowledgements

This report presents the results of a two-day workshop on "Visual Interfaces to Geometry" which was conducted by Werner Kuhn and Max Egenhofer at ACM's CHI'90 Conference on Human Factors in Computing Systems. From the perspective of the National Center for Geographic Information and Analysis (NCGIA), this workshop provided an opportunity to share and discuss results of NCGIA's Research Initiative #2, "Languages of Spatial Relations" with computer scientists and cognitive scientists from around the world and to prepare for Research Initiative #13, "User Interfaces for GIS". The report consists of an edited collection of summaries and research direction papers written by the participants after the workshop. A shortened version of the report has been published in the April 1991 issue of the SIGCHI Bulletin. Reprints of the pre-workshop position papers are available upon request from the NCGIA site at the University of Maine.

I The editors gratefully acknowledge the workshop participants' interest and active involvement; Jeff Jackson's assistance in preparing the meeting and compiling report contributions; Robert Cicogna's editorial help; support from the-CHI'90 organizers, particularly the workshop chair, John Thomas; and funding from the National Science Foundation (NCGIA grant No. SES 88-10917) and Digital Equipment Corporation (grant No. TP-765536).

INTRODUCTION

Workshop Goals

The purpose of this interdisciplinary workshop was to explore and integrate advanced approaches to the visual representation and interactive manipulation of geometric information. The approach taken was to establish desirable properties of interfaces, identify problems in achieving them, and suggest new approaches to solve these problems, in order to improve the communication between users and systems when dealing with geometric information.

Geometric data are used in many areas, such as Computer Aided Engineering, Geographic Information Systems, Computer Graphics, and Graphic Arts. While data structures for their representation and algorithms for their manipulation have received ample attention in order to minimize memory requirements and processing time for systems, there has been little emphasis on user concerns: how can geometric information be presented and manipulated at the user interface, while optimizing the user's "memory" requirements and "execution speed."

Workshop Organization

Eighteen participants from five countries and three continents attended this two-day workshop. They had been selected based on position papers (see appendix). Five participants came from industry, the rest from academia. The range of application areas represented included Computer Aided Design (CAD) and Engineering (CAE), Geographic Information Systems (GIS), Image Processing, and Computer Graphics in general. The workshop participants and their affiliations were:

Bruce Ballengee, Andersen Consulting, Dallas
Leone Barnett, 3M Software & Electronics Resource Center, St. Paul
Meurig Beynon, University of Warwick, Computer Science
Eric Bier, Xerox PARC
Shi-Kuo Chang, University of Pittsburgh, Computer Science
Max Egenhofer, University of Maine, NCGIA and Surveying Engineering
Susan Finger, Carnegie Mellon University
Andrew Frank, University of Maine, NCGIA and Surveying Engineering
Annette Herskovits, Wellesley College and University of Pennsylvania, Linguistics
Jeffrey Jackson, University of Maine, Surveying Engineering
Deepa Krishnan, University of Tokyo, Computer Science
Werner Kuhn, University of Maine, NCGIA and Surveying Engineering
Clayton Lewis, University of Colorado, Cognitive Science
David Maulsby, University of Calgary, Computer Science
Lil Mohan, Purdue University, Computer Science
Hans-Joachim Novak, IBM Stuttgart, Linguistics
David Pugh, Carnegie Mellon University, Computer Science
Raul Ramirez, Ohio State University, Center for Mapping
Piyawadee (Noi) Sukaviriya, George Washington University, Computer Science
Martin Tuori, Alias Research, Toronto.

The workshop opened with presentations by three guest speakers, setting the stage from different perspectives. Annette Herskovits offered a cognitive scientist's perspective in her talk on "Finding Spatial Relations in the World to Match our Prepositions"; Shi-Kuo Chang spoke, as a computer scientist, on "An Algebra for Symbolic Image Manipulation and Transformation"; and Andrew Frank (University of Maine) took an engineer's point of view in "An Engineer's Vision of Geometric Interfaces and Their Construction."

The rest of the time was devoted to free-ranging discussions and some video presentations on a broad range of related topics which had been motivated by the position papers. Among the discussion topics were

- task requirements in GIS, CAD, and industrial design;
- formal task analyses;
- fighting "featurism" in user interfaces;
- editing paradigms and their application to different concepts of space;
- spatial and other metaphors: navigating, zooming and panning, apprentices;
- communicating by sketching;

- the representation and manipulation of (spatial) context;
- multiple representations of geometry at the interface;
- multiple representations of geometry in the system;
- representing uncertainty of geometric information;
- learning systems and learning to use systems;
- formal models of user interfaces and their complexity;
- limitations of input and output devices;
- different styles for different user communities;
- relations between geometric and other (e.g. functional) descriptions;
- how designers work now vs. how they could communicate with a system;
- constraints and their visualization.

Report Organization

As an attempt to bring some structure into this variety of issues, the organizers proposed a conceptual framework for the discussions which also serves to organize this report. The interdisciplinary spirit of the workshop required a structure which was not biased toward special disciplines, but nevertheless related to actual issues in application domains. One way to achieve this goal was to regard interaction with geometry-processing systems as a communication about geometric models: User and system are partners in a dialogue about their respective "internal representations" or models of a task. While they communicate in a variety of languages, this workshop dealt specifically with languages which have visual components.

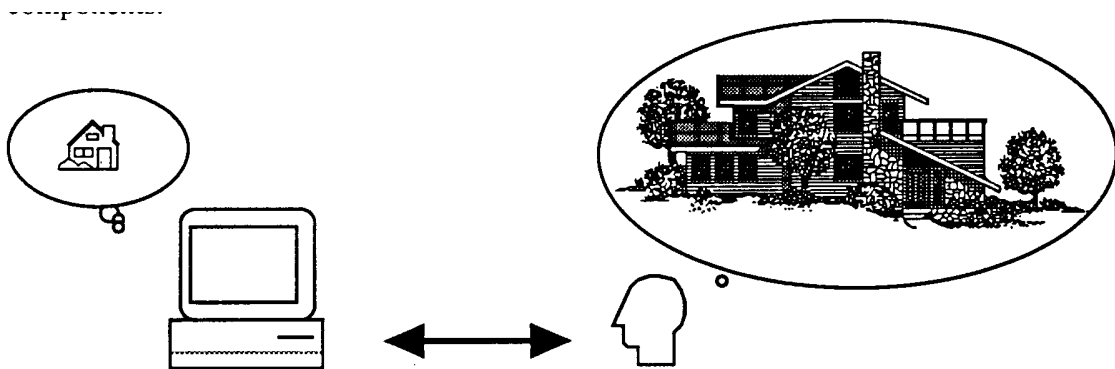


Figure 1: Interaction as communication about models

The communication is obviously asymmetric, since the two sides have different skills and the system is supposed to support the user in solving tasks rather than the other way around. Nevertheless, such a conversational view supports a more balanced approach to interaction than a system-biased understanding of user interfaces as abstract machines. It emphasizes that for each issue on the system side there is a dual issue on the user side and vice versa. Consequently, the topics of discussion and of this report have been partitioned into three major groups:

- Task requirements;
- User (or cognitive) issues; and
- System issues.

Despite the fact that most interface issues, by definition, do not fall in just one of these categories, the structure has helped to put them into perspective and to separate concerns.

Most participants have contributed summaries of their thoughts and of the workshop discussions on one or more of the discussion topics. The remainder of the report contains slightly edited versions of these contributions, grouped into task, cognitive, and system issues. The appendix contains the pre-workshop position papers.

TASK REQUIREMENTS

Specifications of what users want to achieve are the motivating force and ultimate criteria for all improvements to human-computer interaction. They were discussed from a general engineering perspective and with a focus on two application domains: Computer-Aided Design (engineering as well as industrial) and Geographic Information Systems.

An Engineer's Vision of Interfaces to Geometry

Invited presentation by Andrew U. Frank

An engineering perspective on the workshop topic could come from two different points of view:

- that of using geometry-processing systems as an engineer-,
- that of designing such systems with engineering methods.

Given the stated application-driven goals of the workshop, Andrew Frank started with the user's perspective by discussing aspects he considered typical for engineering applications as well as established work habits of engineers. Some fundamental requirements for interfaces to geometry emerged from this discussion:

1. Engineers constantly deal with spatial or geometric concepts. Most of the objects engineers construct are physical and possess, among others, geometric properties which must be designed. This spatial aspect is what makes engineering different from many other professions.
2. Complex objects need to be visualized from different points of view, at different levels of detail, and at different points in time. Good engineering often depends on seeing the "big picture," thus, overview and abstraction tools are needed. An interface to geometry must provide smooth transitions between these various views and support integration with views of non-geometric (physical, aesthetic) models.
3. Engineers not only want to see things, but to express their thoughts visually. Sketching, understood as a process rather than a static diagram, is constantly used by engineers to communicate. Machine understanding of sketches is not trivial because it is difficult to analyze automatically what is intended versus what is coincidental. Nevertheless, effective interfaces to geometry must allow for sketching as an input mode, including the accompanying gestural and possibly verbal signals.
4. Interfaces must, at a reasonable level of detail, document the design process, the assumptions made, and the conclusions drawn. Legally and morally, engineers are responsible for the artifacts they construct. Documenting the process of engineering design not only ensures the obligation of responsibility, it also allows for cooperation.
5. Interfaces have to support group work, differentiating work done by different persons within a team, and allowing reaction to and integration of these. Most engineering work today is a team effort, each member of the team possessing a specific set of responsibilities and focusing on specific parts of a whole. The interface must allow the group to work together at varying levels of detail, integrating the work of individuals to create a composite artifact.

From the second perspective of the engineer as the system designer, Frank's presentation emphasized the need for better engineering methods and tools. Interface design has so far been more of an art than an engineering science. Strengthening the engineering component will help to produce systems with the required characteristics reasonably on time and within budget limits.

An engineering approach to interface design requires means for a high-level specification of functionality. Existing specification methods tend to deal with "how it is done" rather than "what needs to be provided." Based on preliminary research results, Frank hypothesized that algebraic specifications may help in this respect, as they allow one to describe functionality independently of how it is achieved. Obviously, designs have to encompass aspects beyond functionality, such as usability. Unfortunately, these factors are less understood.

After requirements have been specified, an engineer would ask for factors to optimize and for their relative weight in the system. Throughout the workshop, participants pointed out that the time users spend to learn and use a system should be emphasized over the utilization of computing resources.

Frank concluded that we are still far away from a reasonably complete set of engineering methods, let alone tools, applicable to interface design in general and for geometry- processing systems in particular.

An Overview of Industrial Design

Martin Tuori

To understand industrial design as it relates to the topic of this workshop, we must begin by understanding what it is, and how it differs from engineering design. Industrial design is concerned with the functional, ergonomic, economic, and aesthetic qualities of a product. Designers who are concerned only with the aesthetic qualities of a design are usually referred to as stylists. Engineering is typically less concerned with aesthetics, and more with detailed specification and analysis to determine manufacturability, reliability, and the cost and methods of production. Nonetheless, the best engineers have a good sense of aesthetics.

In the competitive marketplace, it is not sufficient that a product function correctly, while being reasonably priced and robust. The form given to a product, whether is a piece of jewellery, an automobile, or an airplane, directly and strongly affects the consumer's willingness to buy and use the product.

While industrial design and engineering are quite different, a design team typically includes representatives of both disciplines, as well as marketing and manufacturing staff. Together they translate marketing objectives and strategies into attractive, highquality product designs, and communicate those designs to engineering and manufacturing groups responsible for creating the finished goods.

The product design process traditionally involved time-consuming, sequential steps-concept sketches and hand renderings, detailed drawings and cross-sections from various views, model making using clay, foam, or other materials, refinement of the model, and finally measurement and drafting for communicating the design downstream. At each stage in this process, the best alternatives are selected from as many possible designs as time permits. Changes may be required as a result of subsequent analysis, so the design process is a series of loops, with progressive refinement over time. A late change that affects decisions at an early stage in the process can result in significant and costly delays and reworking.

The term CAID (Computer Assisted Industrial Design) was coined to emphasize the specialized requirements of industrial designers. Most CAD systems are focused on the tasks of drafting, detailing and analyzing carried out in engineering, whereas CAID systems provide features to support earlier phases of the design process. The primary issues in CAID systems are ease-of-use (industrial designers do not generally have the training or temperament of engineers), support for sketching and free-form shape design, access to the best possible photorealistic rendering and animation, and the creation of accurate, engineering-compatible data for communicating with other groups.

The use of CAID changes the traditional approach in several important ways. CAID allows the designer to work directly in 3D, creating curves, surfaces and objects, rather than drawings. A 3D model can be used to generate a series of renderings, animation of the object in motion, prototyping through NC tooling or stereolithography, and allows other members of the design team simultaneous early access to 3D data. Simultaneous engineering improves the team's ability to detect problems early and carry out necessary changes. As part of an overall design and manufacturing strategy, CAID also alters and improves communication with corporate management, by allowing them to view realistic renderings and prototypes more quickly than with traditional methods. Full-size prototypes in clay or foam may still be required, but they can be a side-branch of the process, with 3D data flowing directly from design to engineering. The flexibility to make modifications to an existing design allows the designer to explore a wider range of possibilities, which in turn results in a better final design. From the corporate viewpoint, the benefits of CAID can be measured in financial terms-improved time-to-market, improved quality, and improved design team productivity.

The changes underway in the field of industrial design are based on careful attention to the work-flow and special needs of the design team. The process is changed by the introduction of new methods and it demands continued study and analysis, especially in realistic settings. Only then can these users' needs, with respect to visual interfaces to geometry, be fully understood and met.

Selecting Geographic Information System Software

Bruce Ballengee

"A sea peril next awaited them--the passage between Scylla & Charybdis... It was a frightful ordeal and six of the crew lost their lives there." Homer

One of the greatest challenges to any organization implementing a Geographic Information System (GIS) is successfully navigating the software selection process. This stage is fraught with risks that require thoughtful analysis and management. In addition to correctly communicating the organization's GIS requirements, the specification must sail a true course between two clashing objectives:

The specification must be tight enough to limit the number of finalists to a manageable number. After all, virtually every selection process is bounded by time and budget constraints.

The specification must be open enough to promote a level playing field where the contestants can compete based on functionality and cost, and be judged on common characteristics. Even when not required through regulations, as in the public sector, it is good management practice to have a competitive process.

A classic solution to this age-old problem is to plot a thin line between the two demands by inundating the perspective candidates with a plethora of specific, detailed functional and technical requirements ranging from the number of line thicknesses supported within the symbology language, to the clock speed of the central processing unit, to the vertical acceleration of the pen plotter head. In theory, this marvelous myriad of requirements collapses the channel down to a narrow chokepoint so that only a few hardy candidates survive the gauntlet. In practice, the classic approach often leads to selecting a solution that fails to fulfill expectations.

An alternative route can lead to a more successful outcome. Focusing on how the function is to be accomplished, rather than on what the function is, moves naturally toward a viable solution that revolves around the GIS visual interface. The visual interface encompasses more than just the look and feel of the GIS, it is the principle mechanism for empowering the user's desires and serves as the chart by which the user navigates through the system's functions and features.

A GIS visual interface should manifest the following qualities:

The look and feel must be consistent with other applications with which the target user will be interacting. This goes beyond compliance with user interface standards such as X windows or Presentation Manager. The interface must be tailored to the point where the user can make an easy transition from one application to another. This is especially important in the GIS and CAD environments where the typical users will spend half or more of their time performing other automated functions-most often office activities. A high degree of compatibility and seamlessness greatly improves user acceptance.

Functions must be easily accessed and executed. GIS systems that are rich in functionality become useless in the hands of all but the most expert users if they rely on a command line interface that requires user knowledge of command names, parameters and sequence. Menu systems that deluge the user with a torrent of cascading options are little better. The degree of complexity in the user interface is the determinant factor in the amount and quality of training and ongoing support that are required for assured implementation success.

A good rule of dead reckoning is that 90% of the user population requires only 10% of the GIS functionality. This typically reduces to a core of 15 to 20 basic functions. Identifying these functions and devoting extra attention and emphasis to them throughout the selection process increases the probability of success. A 10% shortfall in expectations is usually more correctable than one of 90%.

Finally, the user interface should empower the users in their work. It must be extensible as requirements change and user understanding grows. It must put powerful data manipulation tools at the user's fingertips, allowing access to the underlying database while sharing the data simultaneously with fellow workers in a cooperative environment.

By correctly specifying desirable visual interface characteristics and verifying these features during benchmarking prior to final selection, an organization can dramatically improve the likelihood of implementing a truly successful GIS, while still satisfying the need to select from a short list of qualified candidates.

Interacting with Geometry in GIS

Raul Ramirez

From the point of view of using topographic data in Geographic Information Systems for design and analysis, two of the most desirable properties of visual representation and interactive manipulation of geometric information are: (1) better means to observe the terrain model from different locations in the space without loss of resolution and (2) efficient means to manipulate the terrain model in a "natural manner" with a minimum of computational effort.

Current methods of display are too "artificial" and too slow due to the large amount of data needed for accurate topographic representation and the state of the technology. They are limited to a single full resolution terrain view per display unit at any time. Of course, additional display units can be used for additional terrain views, but the result is not better than looking at static pictures of the terrain. The other major problem with the current methods is the relative loss of resolution when the observer moves toward the terrain surface.

Manipulation of elements of three-dimensional terrain representations is very cumbersome today; there is no easy way to locate or change specific elements on the terrain. When a large number of elements constitutes the terrain model, as is usually the case in topographic terrain representation, it is necessary to display a limited area of the surface on the display unit to locate or change information. This may disorient users due to the loss of resolution and the restriction of view, or at least slow down their work. Another problem is the identification of a three-dimensional element with a two-dimensional input device. This is a frustrating and unproductive method of interfacing with a three-dimensional terrain model.

The ideal representation of the terrain is a true three-dimensional model in which one can move around and look at it from any angle or location. In this ideal representation one would like to have the equivalent of a magnifying glass to allow close-ups of specific portions of the model without loss of resolution. One would also prefer to have a pointing device which works in three-dimensional space, and can be used to locate and manipulate terrain elements.

There are still major problems that need to be solved on the way to this ideal, but current research in computer graphics and holography are promising. The workshop showed that research efforts in diverse fields can be combined towards this goal.

COGNITIVE ISSUES

Cognitive issues in user interfaces to geometry are closely linked to task requirements: they are concerned with how people think and communicate about tasks, particularly about the geometric aspects of them. For example, interface designers are challenged by such issues as how the meaning of spatial relations can be formalized, how humans perceive space and deal with it in various contexts and at different scales, how powerful means of communication like sketches and pictorial representations can be employed, and how the complexity of using a system can be managed.

Finding Spatial Relations to Match Prepositions

Invited presentation by Annette Herskovits

Annette Herskovits' talk addressed the understanding of prepositional phrases, i.e., the process of finding spatial relations which correspond to the prepositions we use in everyday language (e.g., "across," "over," "around"). Not only is this one of the core problems in natural language understanding requiring geometric information, but the semantics of prepositions have direct implications for the design of any geometry processing system, particularly for the definition of spatial terms in its interaction language. Additionally, natural language manifests spatial cognition, thus bringing out the conceptual geometric categories relevant to the design of visual interfaces in general.

Other cognitive semanticists have provided fine-grained descriptions of the meaning of prepositions as geometric schemas built from cognitive and perceptual primitives. Herskovits' recent work is directed toward extending these analyses in several important ways: (1) it gives an account of how prepositional meanings interact with different structurings of the spatial world, structurings geared to different spatial tasks; (2) it explains certain extended uses of the prepositions, showing that preposition selection and interpretation involve maximizing a "goodness of fit" measure between a prepositional schema and selected aspects of the situation to be described; (3) it points to the existence of conventions beyond the meaning(s) of prepositions proper—a particular preposition must be used for some standard type of situation defined as a functional interaction rather than in strictly spatial terms.

In order to understand a prepositional phrase, the schema for the preposition has to be matched onto the specific situation. This process involves two stages, first the extraction of geometric elements from the scene and second, in some cases, the relaxation of schema constraints. The major problems faced are that prepositions have several meanings (e.g., around 50 for "across") and that there are multiple ways geometric information can be extracted from real objects.

In dealing with these problems, steps are taken toward resolving a mystery central to the functioning of language: the notorious fluidity of lexical meaning. Communication between humans and computers will benefit from this, particularly if the resulting definitions of semantics can be formalized.

Schema and Pictorial Representations for Understanding

Hans-Joachim Novak

Apart from prepositional phrases, another natural language understanding problem requiring geometric information stems from the understanding of dimensional adjectives like "high," "long," "broad." Lang has proposed object schemas for their understanding in [Mark et al. 1989].

In the discussions at this workshop, natural language came up again when Andrew Frank spoke about Geographic Information Systems (GIS). He said that navigation is one of the major uses of GIS and this is where multimodal interaction comes

into play. One of the possible scenarios is a GIS that can answer questions about routes, i.e., "How do I get from A to B?" or, even further away from present reality, it gets a natural language description of a scene and then answers the question: "Where am I?"

In our own research at IBM Germany we investigate text understanding. We use Lang's object schemas and pictorial representations for understanding dimensional adjectives and prepositional phrases. A depiction is a two-dimensional sketch of an object seen from the top. Depictions are implemented as cell matrices, i.e., an $N \times N$ grid where the different points in the grid can be either on or off. Prototypical depictions are associated with spatial objects and can be evoked to construct depictions of a spatial layout defined by a natural language description. When answering questions about locations of objects or prepositional relations the system can use the depictions, besides the prepositional representations, to answer the questions by "looking" at the originally described scene. This "looking" process has been formalized and is called inspection. Inspection can, for example, expand regions in a depiction and determine whether another object is in a certain region, in order to answer the question of whether the object is "in front of" another one.

It became clear during the workshop that in order to construct natural language interfaces to GIS or other geometric systems we will have to enhance the existing representations of the systems. Research topics that came up, among others, were: What requirements do natural language interfaces impose on the underlying representations of geometry systems? Do we know all the requirements for a language describing the physical world? How fine-grained should a geometric representation be in order to be suited for language purposes? Should we incorporate coarser representations like depictions into geometric systems?

Appropriate Editing Paradigms for Differently Sized Spaces

Eric A. Bier

At a Specialist Meeting of the National Center for Geographic Information and Analysis, David Zubin proposed that people think about objects and spaces differently depending on how large the space is relative to the observer [Mark et al. 1989]. He proposed four qualitatively different sizes of space. Type A spaces are small enough that you can pick them up and rotate them around to see all sides (e.g., a coffee cup). Type B spaces require scanning with the eyes to take them all in (e.g., the wall of a long hallway). Type C spaces require walking around to see them all (e.g., the rooms of a house). Type D spaces are so large that one would consider a vehicle and aids such as a map to explore them (e.g., a city or territory).

If a system is really to be general-purpose, it should support editing operations that involve, at one time, objects of different sizes. At this workshop, we asked ourselves if the size range of objects and spaces has an effect on the operations that we would feel comfortable using to *edit* such objects or spaces. In particular, when faced with editing objects of different sizes, does one need different operations for selecting, adding, deleting, or moving objects and their parts?

A first observation is that the absolute size of the real objects has no effect on editing. The computer allows us to treat the exterior of a house as a type A object, rotating it around as easily as we would pick up and rotate a coffee cup in the real world. However, the *relative* size of objects does have an effect. If the scene contains both very large and very small objects, the user can only see, in a single view, either the details or the big picture, but not both at the same time. This leads to a taxonomy for editing similar to Zubin's taxonomy, but involving the ratio of object sizes rather than their absolute sizes. Below, I describe some of the editing problems and techniques that we discussed at this workshop ordered by the qualitative size ratio of the scene components involved in a particular editing task.

"Type A" editing

We call an editing sequence "type A" if all of the objects involved are of approximately the SAME size. In this case, all relevant aspects of editing can be seen in a single view.

Selection techniques based on pointing (i.e. with a mouse or tablet) work well on type A tasks. Direct manipulation techniques, such as snap-dragging, can be used to position objects. Operations that can rotate the whole scene are useful so that the user can get a good view of the objects from different points of view.

"Type B" editing

We call an editing sequence "type B" if the objects involved differ in size enough that if the camera zooms in so that the smallest objects are clearly shown then some of the larger objects will not fit on the screen. The difference in size is small enough, however, that by panning back and forth it is possible to see all of the larger objects.

Type B editing requires more sophisticated viewing operations. For instance, if the user is trying to rest a long beam on two tiny boxes, one at each end, the user may wish to have a split-screen view of the scene, with one view on each end of the beam. On the

other hand, if the user wishes to move a tiny box from one end of a long beam to the other, a single view would suffice so long as that view can be panned from one end of the beam to the other. For this translation operation, it would be helpful if the user could place the tiny box in a "pocket" and carry it from one place to another. The pocket might be a small rectangle on the screen that stays with the screen when the view is panned.

Also, if the user will need to go back and forth repeatedly between the ends of the beam, it would help to be able to place special "transporters" or "trap doors" at each end of the beam. Clicking on these special objects would allow the user to jump directly to a previously specified location. Alternatively, the screen could show two views of different scales, one large enough to view the largest object (the beam in our example), and one detailed enough to show the smallest objects. The user could point in the large view, which acts as an overview, to change the location displayed in the detailed view.

"Type C" editing

We call an editing sequence "type C" if the objects involved differ in size enough that it would be necessary to "walk through" the larger objects in order to see all of the smaller ones. Examples of type C tasks include moving a lamp from one room of a house to another, ensuring that all of the ceiling fixtures in a house are properly wired, or selecting all of the light switches of a house and coloring them white.

Like type B tasks, type C tasks are aided by multiple views and overviews. For some type C tasks special overviews can be used. For instance, if the largest object involved in a task is a house, the user could zoom out to a floor plan to move from room to room.

"Type D" editing

We call an editing sequence "type D" if the objects involved differ in size so much that powerful navigational tools would be needed to navigate the largest objects while keeping track of the smallest objects. Type D editing should be relatively rare since it is usually possible to structure editing so that large objects are dealt with at a large scale and small objects at a small scale. However, type D situations do come up. A whimsical geographical example would be placing a county line so that Mr. Smith's pine tree is to the north of it. A more common example occurs in integrated circuit design where the placement of a large component such as a programmable logic array or data path might interfere with a small special-purpose logic gate that had been previously placed.

Type D editing is hard. VLSI designers have built many special tools that automate operations where large but detailed structures must be laid out. However, interactive operations are not entirely hopeless. Zooming in and out in several stages can help solve some problems. Multiple views at different scales are also useful.

There are also techniques that aid some generally useful operations. Consider, for instance, the problem of placing a very long line on a plane that contains many tiny shapes. If you wish to place the line so that it has some desired relationship to one or more of the tiny shapes, you might distort the view of the scene so that the objects near the line are magnified perpendicular to the line, allowing fine relationships to be seen. This is a special case of the fish-eye views idea [Fumas 1986].

In conclusion, the Zubin spaces provide a useful thought experiment for those designing general-purpose interactive geometry systems. In the CHI'90 workshop, we suggested a few examples of editing styles that might be used in a system tuned for editing in all four types of Zubin spaces. Hopefully future designers will use this framework to improve the design of interactive geometric design systems.

Multiple Representations

Lil Mohan

One of the key areas of discussion in the workshop was the topic of representation. From the most general perspective, representation corresponds to how the visual object is captured within the computer (at various levels of abstraction). One of the points that emerged from the discussion was that multiple representations of the same geometric object (or scene), for viewing the same geometric object from different perspectives and at different levels of resolution, would be desirable for geometric interfaces. With multiple representations the users have the freedom to explore various options about how they want to interact with the system. Multiple representations of the same artifact may be necessary from another point of view. There are really two separate types of people interacting with the system—the person developing the software and the person who is going to use the software. The interface developer has to make the design decisions as to what are the semantically relevant units about the geometric objects that need to be captured. Having done so, he then has to figure out what facets of the semantics can be captured by which types of representations. The most important research issue is how these various representations can be combined and indexed.

Communicating by Sketches

David Pugh

One way to improve the productivity of a CAD system is to make it easier for the designers to describe their mental models to the computer. This might be done by looking at how users present their designs to other users and to incorporate some elements of this communication into the dialogue between the user and the computer.

An element that seems both to be widely used and to have potential for use in CAD systems is the creation of sketches. The potential is due to the speed with which humans can create and interpret rough sketches. Creating a sketch is normally a very "natural" process: simply add or remove features until the sketch looks "right." Typically, these sketches have multiple interpretations and are not drawn accurately. Even so, humans seem to have little trouble "seeing" the object represented by a sketch.

Suppose that computers could interpret sketches with the same level of skill humans do. The practical effects would be two-fold. First, CAD would probably be used far earlier in the design process than it is now. Currently, users often create pencil and paper sketches prior to using a CAD system. The cause seems to be that it is easier to modify a sketch than it is to modify a model on a CAD system. As a result, users often will wait for the design to "stabilize" before creating the corresponding model on a CAD system. Second, the training requirements for the CAD system will be reduced (at least the amount of time required before a user can become productive), because most people have a great deal of experience creating sketches and the procedures in a sketch-based interface are more intuitive than the equivalent procedures in a conventional interface.

Implementing a sketch-based interface requires solving two separate problems. First, a mechanism must be found for generating a "reasonable" interpretation of a change to the sketch. While there are a variety of algorithms for generating 3D interpretations of 2D sketches [Sugihara 1986], none are particularly good at generating "reasonable" interpretations. A variation of these techniques is to force new interpretations to be "similar" (by some metric) to the previous interpretation, on the basis that most changes to the sketch tend to make only minor changes to the 3D interpretation. Second, the user must be able to make the sketch more accurate as the design process advances. An obvious way to do this is to allow the user to place constraints on objects. [Nelson 1985] describes a system for placing and solving systems of non-linear constraints in two dimensions. One characteristic of this system is that it uses the current position of the points when searching for a solution to a new system of constraints, on the assumption that the positions will not change "much" as constraints are added or removed. The techniques described above are a partial solution to the problems of creating a sketch-based interface to a CAD system. The main problem is that neither the sketch interpreter nor the constraint solver work well when the user makes a major change to the sketch. The effect of this problem is reduced if the user is given both continual feedback and mechanisms for "cuing" the system to find the correct interpretation and solution.

Cognitive Walkthrough Analysis of a GIS

Clayton Lewis

The cognitive walkthrough method [Lewis et al. 1990] is a technique for analyzing the strengths and weaknesses of a user interface by examining each user step required to accomplish representative tasks. The analyst examines each step, asking whether a user could be expected to choose the correct action, given the user's knowledge of the task and prompts and feedback provided by the system. The analyst is guided by a structured list of questions on a walkthrough form which are designed to help identify common problems, such as the presence of an incorrect action which might look appropriate at a given step.

While the walkthrough form is designed for analyzing highly-prompted interfaces that require minimal background knowledge from users, so called "walk-up-and-use" interfaces, the underlying logic of the method can be applied to other kinds of interfaces. [Bell et al. 1991] adapted the method to aid in the design of a graphical programming system which provided few prompts and assumed considerable background knowledge on the part of users.

In this note I follow this precedent in applying the walkthrough approach to the user interface of the popular ARC/INFO Geographic Information System [Morehouse 1989]. First, I selected a sample task for which an appropriate sequence of user actions was available. I then examined each step in the sequence, noting what knowledge users would need in order to choose the correct action, and what planning or problem-solving they would need to carry out.

The Task

The task is adapted from a laboratory exercise developed by the National Center for Geographic Information and Analysis. I modified the exercise to remove purely tutorial aspects. The user's goal is to determine the impact on runoff of developing a parcel of open land. Maps of soil type, precipitation, and land use are assumed available, together with a program implementing a runoff model.

The table at the end of this note shows appropriate steps for accomplishing this task using the ARCANFO system. Commands and names of ARCANFO entities are shown in upper case. Following each step in the table are notes generated in examining each step.

Walkthrough Analysis

The walkthrough analysis identifies a number of general features of the interface which can be expected to pose difficulties for new or occasional users. First, as a command interface the system relies on users to determine how to specify nearly every operation. Online help provides some support, but commands are listed alphabetically, so the user must have some idea of what an operation is called in order to locate it. Once a command has been identified the help system does aid in getting the parameter specifications right.

Another problematic aspect of the interface is that commands are often broken into pieces so that a first command specifies an object to be operated on by subsequent commands. While this saves some respecification of parameters it adds to the inference process needed to connect a goal to the corresponding action. The user must determine not only the command that accomplishes a desired effect but also may have to find out what command is used to prepare the ground for the desired command. The sequence SELECTRESELECT in Steps 14 and 15 illustrates this; only the RESELECT command is directly linked to the user's current goal.

The interface is strongly moded in that a given command can be issued only when the system is in certain states; the correct one of several subsystems must be active. This means that users must not only determine the right command for some purpose but also which subsystem they must enter in order to issue that command.

Besides adding to what users must learn and understand in order to use the system, this modedness complicates the planning users must do. Steps 4-7, in which the user produces a map, illustrates this. Step 7 actually draws the map, and must be issued in ARCPLOT, the drawing subsystem. But the command requires a lookup table which cannot be constructed within ARCPLOT; creating this is Step 2. So the user must plan out the map drawing steps in sufficient detail to detect the need for the lookup table before actually issuing any of the commands to draw the map. One problem with this is that the plan for drawing must be worked out mentally before executing any of it, a difficult reasoning process. A second problem is that some of the planning must be repeated when the time comes actually to issue the commands, since it is unlikely that the user will remember all details of the steps from the first planning episode. A third problem is that the creation of the lookup table is liable to be forgotten until after ARCPLOT has been entered, since issuing the ARCPLOT command looks fully appropriate to the user's current goal of drawing a map.

The fact that the modes of the interface correspond to distinct subsystems brings in another difficulty: the same information may be referred to differently at different times. For example, the polygons in the composite cover COMP86 are indicated in Step 8 by the cover name COMP86 and the feature class POLYS. But in Step 14 where the information about some polygons is to be changed, the user must specify a file name with the tag PAT. Thus the user must learn two ways to refer to the same information

In Steps 17 to 21 the user is changing the information to be processed by the RUNOFF program. This is done by actually viewing and changing the program, which seems undesirable in four ways. First, the user must know that this change must be made, rather than being prompted by the RUNOFF program. Second, the user must read or be told enough about the internal structure of the program to make the change. Third, the user must learn to manage a specialized editor to make the change. Fourth, a number of separate steps are required. All of these difficulties would be avoided if programs could prompt for this kind of parameter; this may be possible but how to do it is not readily apparent.

Discussion

The walkthrough analysis identifies a number of potential problems with this interface. Many of these are reflections of the underlying division of the software into subsystems, and would require considerable reworking to address. Nevertheless, the results of the analysis might be useful in planning a restructuring of the interface, or in comparing the effectiveness of this interface with competing alternatives.

Further benefit might be gained from extending this analysis to cover a range of representative tasks. This would not only cover more aspects of the interface but would also permit the description of what [Bell et al. 1991] calls the "*doctrine*" of the system. Doctrine is the body of knowledge of the system that users must have to apply it effectively to a range of problems.

For ARC/INFO the doctrine would include facts about the division of the system into subsystems, different means of referring to objects within the subsystems, mapping of frequently-occurring task elements onto system commands, and the like. 'Me ARC/INFO documentation does a good job of presenting some of this prerequisite knowledge, but an explicit representation of doctrine could be useful in finding holes in the coverage of the documentation. It would also permit comparisons among competing

designs to be made on the basis of required doctrine: a design requiring less doctrine, or simpler doctrine, would be preferred in a comparison.

Table

This is an application of the general approach of the cognitive walkthrough to a lab exercise on using ARC/INFO prepared by NCGIA. Because the lab exercise relies on quite a bit of advance scaffolding (e.g., creation of lookup tables) this does not cover all aspects of use that would be involved in really performing this job.

I have modified the example to concentrate just on the task goal of running the model on modified land uses, omitting some stuff that is specific to the lab exercise.

Initial goals: select a parcel with area between 80,000 and 300,000 m² that is not already in land use "urban". Reassign it to land use "urban" and calculate runoff for changed use using program RUNOFF.

Plan:

Use ARC/INFO to:

find suitable parcel

a parcel is to be made up of pieces each of which has one land use,

soil type, and precipitation level

do this by displaying a map of possible pieces

change land uses

run RUNOFF program for various conditions

prereqs: have soil, precipitation, and land use data in one cover

Steps:

First, get into correct context (in ARC, right directory)

(1) UNION SOIL PREC TEMP

(2) UNION TEMP LAND86 COMP86

Note: must know names of covers that contain relevant information. Must know that UNION will combine information in such a way as to produce the necessary subregions of the whole. Must figure out how to use TEMP to combine more than two covers.

(3) create a lookup table (in INFO), say SHADELU, to translate land use codes into shade numbers

Note: must know to do this; must do it before entering ARCPLOT to draw map; must look up appropriate shades in default shade set to do this; must know how to do it, including that entering INFO is needed and how to get out of INFO (4) ARCPLOT Note: must know to do this to produce a map

(5) DISPLAY 9999

Note: must know to do this after ARC PLOT

(6) MAPEXTENT COMP86

Note: must know to do this to set size of map from cover to be shown

(7) POLYGONSHADES COMP86 ALUCODE SHADELU

Note: must know this is the way to see map shaded by land use

(8) IDENTIFY COMP86 POLYS

Note: this permits user to select a region with the mouse cursor. Must know to do this; must put in cover name and class of feature to select. Must know to press a key to select.

(9) select a region and note its area and record number

Note: must know to record the record number for future use

Note: must know what shade corresponds to urban use to avoid these

(10) Must repeat the IDENTIFY command and action until have accumulated an appropriate package of pieces

(11) QUIT

Note: must know to do this to make changes to cover

(12) COPY COMP86 NEW86

Note: so can make changes nondestructively

(13) INFO

Note: must know to do this to manipulate tabular data

(14) SELECT NEW86.PAT

Note: must know to use this command to operate on file; must also know PAT component of cover is the right one

(15) RESELECT ... record nos. of pieces selected

Note: choose records for subsequent modification

(16) CALCULATE ALUCODE=code for urban landuse

(17) CHANGE RUNOFF

Note: to modify program to refer to new cover

(18) L 10028

Note: this is the line that must be changed; if did not know would have to type runoff and read it

(19) R

Note: ask to replace selected line

(20) SELECT NEW86.PAT

Note: in response to prompt "enter text". Use of PAT can be copied from former version of line

(21) SAVE

Note: must know care and feeding of this editor

(22) RUN RUNOFF

Note: to examine runoff predictions. Will prompt for antecedent moisture and inches of precipitation.

SYSTEM ISSUES

System issues are concerned with finding those internal representations of geometry which optimally support the communication between users and a system while not neglecting additional system requirements. For example, a system should be able to perform spatial reasoning, support multiple representations of geometry, provide spatial and logical context, and learn from its users.

Representations for Spatial Reasoning with Images

Invited presentation by Shi-Kuo Chang

This presentation focussed on data structures for pictorial data, concepts of how to derive spatial information (spatial reasoning), and algorithmic aspects thereof. The term spatial was understood as purely *geometric* and no meanings of geometric objects were considered.

Initially, Chang defined three terms: a *visual language* uses visual expressions, such as drawings, gestures, and icons, to convey meaning; *visualization* is the use of visual expressions to illustrate something; and *visual programming* expresses programs in visual expressions. He argued that *spatial reasoning* combines geometric processing with visual interfaces and referred to the navigation problem where someone wants to get from A to B avoiding some obstacles. *Visual reasoning*, on the other hand, emphasizes visual exploration in the problem space. As an example, he mentioned a very large library in which a user wants to find a specific book.

The remainder of the talk centered around data structures for spatial objects to support spatial reasoning and the acquisition of spatial knowledge, by presenting results of Chang's recent work on a knowledge structure for pictures. This work describes an algebra which is based on symbolic projections including a limited set of relational operators [Chang et al. 1990, Jungert and Chang 1989]. The motivation has been given by queries in pictorial information systems which frequently include constraints on spatial relations, such as *left of*, *above*, and *behind*. In order to represent a picture, symbolic projections for the extrema of object boundaries onto lines ("cutting edges") parallel to the x- and y-axes are used. Objects in a two-dimensional space are encoded as strings indicating the sequence of objects along the cutting edges. Operations detecting spatial relationships are, therefore, reduced to operations on strings. The outcome is an image algebra based upon symbolic projections. It can be extended to deal with a multilevel hierarchy of representations as well as aggregates of objects. The results show that this algebraic approach to image manipulation and transformation constitutes a powerful means for many different applications.

The subsequent discussion clarified that the method is not invariant under rotation, but it can approximate rotations. Furthermore, the model does not include metric relations like near and far, but such extensions are possible. Another attractive extension could lead to a formalization of schema-like constructs (see above sections by Herskovits and Novak).

Multiple Internal Representations

Deepa Krishnan

No single representation is ideal for a particular purpose. For example, in CAD/CAM, a feature-based description of a machine part is more suitable for applications such as design and process planning whereas a boundary representation (B-rep) is more convenient for examining topological adjacency relationships or to perform local modifications such as tweaking on an object. In GIS applications, both raster and vector representations have their advantages. For example, it is easier to derive buffer zones around an object in a raster representation, while vector representations facilitate higher-level tasks such as navigation. Another issue involves hierarchical representations. Both in GIS and CAD, it is desirable to represent objects at different levels of detail.

Thus, either all representations are maintained or one is maintained and the others have to be derived when necessary. However, in a case where two or more equivalent representations of an object or a scene exist simultaneously, the problem of consistency arises. When one of these representations is modified, the other has to be altered too, to reflect the changes made. This is complicated and requires efficient algorithms to convert from one representation into another. When a single representation is maintained in the database, the interface should be well-equipped to derive a variety of external views from the data structure. This may impose a burden on the interface designer though consistency problems will be eliminated from the database.

Supporting Context

Leone Barnett

Use of the notion of "context" as a means to provide clarification of meaning, or a more focused interpretation of some potential information, has an intuitive appeal. In general, workshop participants agreed that it is important to incorporate contextual

information into computer systems, especially those systems concerned with representation of spatial information. However, it was noted that it is not at all clear how to do this.

First of all, context means different things to different people. Thus many things can be classified as some type of context. In a sense, a context is used to bring the appropriate information to bear on a given situation. What is appropriate? How is it that context helps us apply the right rules and filters at the right time? What is meant by a context both in general and in specific instances?

An important first step is to develop a formal description or definition of context. Not surprisingly, context itself might be defined in different ways "depending on the context" in which a definition is sought. One way to view context concerns machine learning. Four types of context have been described in machine learning. They include:

- *state*-the relevant aspects of history;
- *goal* decomposition-used in goal-oriented problem solving;
- *bias*-the representation allowed by a language;
- *focus of attention*-pertains to the selection of objects within the bias.

Another "foursome" indicating a different view of context came from Zubin's four cognitive spaces (see Eric Bier's section above). It is interesting to think about these four editing and navigation paradigms in terms of information space as well as physical space. This leads to the problem of multiple representations, such as form and function, that may be useful to visualize for an object. Context is often needed to support such multiple representations.

It is not clear how to specify the conditions under which a given aspect of context applies or does not apply to a specific task or need. For example, if context is to be useful in geographic queries, how would you specify a meaningful context when some type of map must be produced? How can the system infer the context from the query? The system should be able to infer or default to some acceptable notion of context, but this implies some pre-existing assumptions about context. What should these assumptions be? What is the difference in the types of assumptions used in geographical queries vs. other application areas? Can anything generalize across application domains? Despite any default assumptions about context, a user may wish to alter them. Thus, a lot of context specification may depend on the user's intentions.

Another area where context support could be useful concerns feature based systems with high learning curves such as CAD systems. In this case, context could relate to some model of the user and their expected proficiency in using a complex system. Context could be used to help hide complexity, but not necessarily require the removal of complex features. This could shorten the learning curve on a complex system.

With respect to both examples above, it may be useful to view context as the set of transformations required to collect relevant data from a base store of data and organize it into a focused, meaningful presentation.

Context Models for Predictive Interfaces and Learning Agents

David Maulsby

Fundamental to a user's mental model of an activity, such as editing a geometric design, is the association of context with action. A context is a set of conditions that determines a choice, as when selecting an object or deciding which operation to apply. The ultimate context is the user's intention, a hierarchy of goals. An interactive system that captures context/action associations can perform much more of the work than one that provides only a set of primitive tools for manipulating data. The problem is that a user's intention is utterly private and only partially and indirectly expressible-even through the medium of natural language. It is possible nonetheless to observe or infer some aspects of a context, which may be sufficient in many cases to form associations that are correct most of the time, and which may be confirmed or refined through dialogs that are much simpler than explicit programming.

This section of the report describes a framework for mental models of geometric editing tasks based on inferring context/action associations. A specific model has been used to implement Metamouse, a user-interface agent that learns graphical constraints and editing procedures from example traces performed by the user [Maulsby et al. 1989].

Relation to other mental models

In order for user-system interaction to support the mental models described previously in this report, the system must react to contexts expressed in the user's behavior. A predictive model observes user actions (including object selections) and reacts by performing associated actions. Associations may be established by observing a sequence of user behaviors or by explicit programming. The use of multiple representations of data is mediated by the user's goal or preferences as context. Again, the association between context and representation may be derived from observing the individual user's choices or from a design-time user

study. A system that adapts its constraint operators, output representation and feedback to editing at different scales may have hard-wired associations between these and (say) distance moved, but should react to user preferences, especially when tuning borderline cases. A responsive (reactive) document records many types of association between elements of a picture, other data bases, constraints, editing procedures and representation methods. Most of these are specified by the user and are therefore candidates for automatic capture.

As indicated above, another application of the model is in designing user-interface agents that learn procedures.

Elements of the model

A computer model of context/action pairings requires a system capable of inferring and performing them. Methods of debugging or refining associations not based on inference are not discussed here (see [Maulsby et al. 1989]). The system's basic elements are its inputs, outputs, bias, focus of attention, inference mechanism, conflict resolution strategy, and interactions with the user.

Inputs

The inputs are the user's behavior, application context and environmental conditions. Behaviors are sequences of user inputs to the application program, such as mouse movement, keystrokes, and menu selections. Application context is essentially program state, in particular the attributes of and relationships between objects the user is editing. Objects include files, windows, and data structures, typically edited as pictures.

Outputs

The outputs are context/action pairs, where the context is a set of conditions under which an action is performed, and the action is an operator (performable by the user) and its parameters. It is convenient to treat context/action pairs as production rules. The lefthand side, context, comprises program state (described above) and task state, which is an ordered subset of previously fired rules. The right-hand side is the action and resulting program state. The model is unfortunately complicated by the fact that actions need to be parametrized with constraints: for instance, the intention of moving a line so that its endpoint touches the lower-left corner of some box cannot be captured without including the touch relation as a constraint on the action. Selecting an action depends on the ability to meet such postcondition constraints, as when deciding whether or not to continue an iteration; hence they also belong in the context.

Typically, the system will express its model by matching context with actual situations and then performing the associated action.

Bias

The system's selection of attributes and relations for use in context requires a trade-off between completeness and computational complexity. Attributes may be limited to those features of objects explicitly represented in the application's data structures and its user interface, for instance, a text item's font and point-size. Relations are problematic; many interesting ones will not be explicit in the application, and a search through the space of all possible relations between a set of objects is intractable. Hence, a vital component of the system is its vocabulary of primitive relations (such as touch) and logical operators (not, and, or) for composing them. The system must also include restrictions on the complexity of expressions formed to describe relations (e.g., allow conjunctions of up to 3 relations). This combination of vocabulary and restrictions is the system's bias, because it restricts the space of descriptions the system can form (and need search). Bias can be partitioned into a sequence of language schemas so that search is ordered on the complexity of descriptions [Raedt and Bruynooghe 1989].

Focus of Attention

Another means of restricting the search for relationships is focus of attention; the system observes only part of the application program's state, for instance, only the attributes and relations of objects adjacent to the cursor. Focus of attention may be treated as a hard-wired "sensory" limitation, as in *Metamouse* (which observes only touch relations between objects and a turtle icon or the object it is carrying) or as a means of assigning levels of importance to sensations, as in [Agre 1988].

Generalization

Given its observations of program and task state, the system must select those that are relevant context for the next action. This amounts to generalizing conjunctive expressions by dropping irrelevant items. An alternative approach is to prioritize context items and employ weighted partial matching when identifying a context with an actual situation. In either case, two methods of generalization apply—knowledge-based and similarity-based.

Knowledge-based generalization selects or prioritizes elements of context according to a set of rules that describe important features of the application. In a geometry-based drawing program, constraints expressed by touch relations are important. Touch relations can be ordered by their degree of constraint; for example, a vertex-to-vertex touch is more constraining than an edge-to-edge. Thus, if both occur, there is reason to propose the vertex -to-vertex touch as the more important constraint, hence to include it in the context rather than the edge-to-edge. The choice of objects is also important, and may be ordered on the likelihood that a particular one is chosen. For instance, if a rectangle selected in the current step was also selected in some previous context, it is probable that this is intentional, and hence an equivalence between variables should be included in the context. Rules for ranking touch constraints on drawing actions and for relating variables are given in [Maulsby et al. 1990].

Similarity-based generalization selects or gives priority to features that occur repeatedly over multiple examples of a context. It may be used in place of knowledge-based generalization or as a means of further refinement. No background knowledge of the application is required, unless individual features are to be generalized by reducing their inherent constraint, for instance, replacing "vertex" by "any-point-on-edge" in some touch relation. If a context fires erroneously, it can be specialized by adopting constraints that knowledge-based generalization had discarded. The similarity-based version space methodology [Mitchell 1982] can be used to search the lattice of all descriptions permitted by bias and focus of attention, by generalizing and specializing until only 0 (failure) or 1 description covers all examples seen so far. By matching contexts and predicting actions, the system can generate its own test examples in order to collapse the version space more quickly.

Knowledge-based generalization makes a quick guess and therefore is useful for giving the user feedback regarding the system's model of context. Similarity-based methods are more reliable in the long term but require at least two and potentially many examples. Hence, in a practical application of context modeling the user must be able to push the system toward the correct generalization, by pointing out important or irrelevant features or by stating desired constraints.

Conflict Resolution

An actual situation may match more than one context. If the system is to predict only one action or to offer choices in an efficient order, it needs additional information. One approach is to prioritize context/action production rules according to recency or frequency of correct firing, or the ratio of correct to erroneous firing, or a benefit/cost ratio. Such a conflict resolution strategy also addresses the problem of eliminating useless or incorrect rules.

Interactions

In a practical application, a mental model structured as described above would determine the behavior of a predictive user interface, for instance an instructible agent like Metamouse. A critical question arises: how does the model affect interaction? Choices of bias, focus of attention and generalization method will affect not only computational complexity but also the structure and complexity of interactions with the user. For instance, enlarging the bias and focus of attention enable the system to search for relationships that the user would have to construct as a procedural test otherwise, but might also require more examples to zero in on a useful generalization, implying that the user must wait longer for assistance, or respond to numerous test examples produced by the system. Next to no research has been done on these problems from an ergonomics perspective.

Summary and Research Directions

This section has described the elements of a framework for mental models of contextual constraints on activity in a geometric editing system. The basic inputs are user behavior and system state, the outputs are associations between context and action. Context is a generalization of state, and action is a generalization of behavior. For feasibility, the model requires restrictions on its description of state; linguistic restrictions are its bias and sensory limitations are its focus of attention. Inputs are generalized by two methods: knowledgebased, which selects features likely to be important according to rules about the editing application; and similarity-based, which progressively generalizes a description as more examples are seen. Since many generalized contexts may cover a given real situation, the model includes conflict resolution strategies based on probability and recency. Choices made regarding bias, focus, generalization, and conflict resolution affect interactions between a system and its user; this aspect of the framework is undeveloped as yet.

Other directions for research include analysis of computational and interaction complexity implied by bias and focus of attention. The practicality of describing domains for knowledge-based generalization, and its merits relative to similarity-based, are open questions. The framework should be tested against a variety of models of context in graphical and textual editing applications. The problem of deciding which situations to match with each other when generalizing (the "clustering" problem) should be added to the framework. Weighted associations and partial matching also deserve consideration.

Implications for Software Design

Noi Sukaviriya

Several problems related to user interfaces of existing geometry design software were discussed in the workshop. One of the problems mentioned is the incompatibility between computer entity models and entity models used by designers in the design process. A particular example discussed in the workshop is CAD software which normally requires designers to operate on points and lines to create a design object, while industrial designers actually work on a design by starting from a loose shape, then gradually constraining shape properties and manipulating aspects of the design such as curvature, slant edge, surface, and dimension, to finish the design. Another problem discussed is insufficient use of media by design software. Engineers emphasize sketching and visual thinking to communicate, while most software offers text only or a limited combination of graphics and text, thereby limiting the nature of interactions which could facilitate informational exchange, both between designers and software as well as among designers. These problems indicate how software models, which include both user interfaces and software functionality, can be incompatible with designers' conceptual models and design processes. They also indicate that the interaction aspect of software is still too limited for its effective use.

The implication of a software model incompatible with a designer's conceptual model suggests the need for revisions in the software design process. A software model should be based on entities with which designers are familiar, before being extended by other entities which take advantage of computer-aided design software. The same can be stated about operations which the software should offer to operate on the entities. The software model should be designed separately from its actual interface so the model will not be dominated by the underlying graphics packages, as was the case with most currently existing CAD software and its base entities of points and lines. It is essential to maintain the integrity of the software model when concentrating on the implementation of how software functionality can be accessed and on the "naturalness" of different interface designs.

The scope of the incompatibility of models problem discussed also includes the restricted ability of semantic-dependent functionality in design software to aid in sophisticated entity manipulations. This is mainly because of the lack of semantic information available on entities (being created on the screen)--essential information which, if it existed, would allow meaningful functional operations to be performed on screen entities. An example discussed is the case where simulation of 3-D objects is required after their creation. Such a feature would render design software more useful beyond being a basic 3-D drawing tool. Specification of semantic information related to design objects should be incorporated in the software model, and connecting such information to functional operations of entities is desirable. Though adding semantic information is very domain-dependent and may not be solved in a generalized fashion, it is this feature which designers envision useful in the design process. Efforts to incorporate entity-specific semantic information in the software model are therefore encouraged.

The group also discussed the lack of appropriate media to communicate with design software. A non-computer related aspect of this problem, such as the need for proper inputs to express spatial terms or for languages to express design changes envisioned in designers' mind, is based on the lack of appropriate, unambiguous words, utterances and terminology to express design goals. While this level of the problem requires further studies regardless of software models, a computer-related aspect of the same problem is the limited ways to express design goals. This suggests the use of multiple input devices and a combination of interaction techniques--gestures, text, speeches and/or graphics--to access functionality in the software model and to elaborate design parameters. Solving this aspect of the problem centers around the interface design of design software and confirms the criticality of separating the interface design from the software model design.

Poor documentation can also be an obstacle in extending the use of geometric design software. One problem is the non-coherence of off-line documentation and the actual software. Another problem relates to how manuals often strictly document individual software functions and lack task-oriented guidelines, which were discussed as being more useful for designers to match their expectations to the capability of the software. Though a well-known (yet never achieved) solution is to improve the documentation process, a more viable solution is to generate on-line help from the actual running software thus guaranteeing parallelism between underlying software and help information. Research in this area has considered using software specifications, which represent the conceptual model of software, to reason about tasks and operations executable in the software [Senay et al. 1989, Sukaviriya and Foley 1990]. Changes in the specifications immediately result in changes in on-line help presentation, hence parallelism between help and the actual software is ensured. The use of graphics and graphical animation in on-line assistance [Cullingford et al. 1982, Feiner 1985, Sukaviriya 1988, Sukaviriya and Foley 1990] also adds to the usability of graphics-oriented design software in ways which, obviously, benefit from the use of the computer as a design aid.

CONCLUSIONS

This workshop has been successful in raising the awareness among all participants for different perspectives on their own problems in other disciplines, as well as for new applications related to their research. However, it clearly produced more questions than answers--which is what could reasonably have been expected from such an interdisciplinary meeting.

An important result from the perspective of application domains like engineering or geography was the realization that the human-computer interaction community has a keen interest in non-traditional applications. The situation is somewhat reminiscent of the "discovery" of non-traditional applications by the database community in the mid-eighties. Research in human-computer interaction is rapidly approaching a point where disciplined analyses and designs of interfaces beyond text editors are feasible. At the same time, the state of the art of user interfaces in complex application areas urgently calls for applying such approaches.

The discussion about user interfaces to geometry will continue in various formats. A special issue of the *Journal of Visual Languages and Computing* with articles from workshop participants is in preparation. Some of the issues raised at this workshop have been and will be pursued at other meetings. Cognitive and linguistic aspects of geographic space were the topics of a NATO Advanced Study Institute held in July 1990 [Mark and Frank 1991]. The National Center for Geographic Information and Analysis (NCGIA) has announced a new Research Initiative on "User Interfaces for Geographic Information Systems." It will address human-computer interaction methods and related issues in the design and implementation of user interfaces for GIS's and other geographical software packages. A "Specialist Meeting" for this initiative is planned for the Buffalo area in June 1991. Special Interest Group meetings in preparation for this conference are planned for the CHI'91 Conference in New Orleans.

References

- Agre, P.E. 1988. The dynamic structure of everyday life. MIT Artificial Intelligence Laboratory. Technical report (Ph.D. thesis) No. 1085.
- Bell, B., J. Rieman, and C. Lewis. 1991. Usability testing of a graphical programming system: Things we missed in a programming Walkthrough. Proceedings, CHI'91 Conference on Human Factors in Computing Systems.
- Chang, S.-K., E. Jungert, and Y. Li. 1990. The Design of Pictorial Databases Based Upon the Theory of Symbolic Projection. In *Design and Implementation of Large Spatial Databases*. Edited by A. Buchmann, O. Gunther, T. R. Smith and Y.-F. Wang. 303-324. New York, NY: Springer-Verlag.
- Cullingford, R.E., M.W. Krueger, M. Selfridge, and M.A. Bienkowski. 1982. Automated Explanations as a Component of a Computer- Aided Design System. *IEEE Transactions on System, Man and Cybernetics*.
- Feiner, S. 1985. APEX: An Experiment in the Automated Creation of Pictorial Explanations. *IEEE Transactions on Computer Graphics and Applications*.
- Furnas, G.W. 1986. Generalized Fisheye Views. Proceedings, Human Factors in Computing Systems (CHI86). Edited by M. Mantei and P. Orbeton. 16-23.
- Jungert, E., and S.K. Chang. 1989. An Algebra for Symbolic Image Manipulation and Transformation. In *Visual Database Systems*. Edited by T. L. Kunii. 301-317. NorthHolland.
- Lewis, C., P. Polson, C. Wharton, and J. Rieman. 1990. Testing a walkthrough methodology for theory-based design of walk-up-and-use interfaces. Proceedings, CHI'90 Conference on Human Factors in Computing Systems. 235-242.
- Mark, D.M., and A.U. Frank. 1991. Cognitive and Linguistic Aspects of Geographic Space. Dordrecht, The Netherlands: Kluwer, in press.
- Mark, D.M., A.U. Frank, M.J. Egenhofer, S.M. Freundschuh, M. McGranaghan, and R.M. White. 1989. Languages of Spatial Relations: Initiative Two Specialist Meeting Report. National Center for Geographic Information and Analysis; Santa Barbara, CA. Technical Report 89-2.
- Maulsby, D.L., I.A. Witten, and K.A. Kittlitz. 1989. Metamouse: Specifying Graphical Procedures by Example. Proceedings, SIGGRAPH '89. 127-136.
- Maulsby, D.L., I.A. Witten, K.A. Kittlitz, and V.G. Franceschin. 1990. Inferring graphical procedures: the complete Metamouse. Dept. of Computer Science, University of Calgary. Research report 90/388/12.
- Mitchell, T.M. 1982. Generalization as search. *Artificial Intelligence* 18 (2): 203-226.
- Morehouse, S. 1989. Architecture of Arc/Info. Proceedings, Auto-Carto 9. Edited by E. Anderson. 266-277.
- Nelson, G. 1985. Juno, a Constraint- Based Graphics System. Proceedings, SIGGRAPH '85. Edited by B. A. Barsky. 235-243.
- Raedt, L.d., and M. Bruynooghe. 1989. Towards friendly concept learners. Proceedings, IJCAI 89. 849-854.
- Senay, H., P. Sukaviriya, and L. Moran. 1989. Planning for Automatic Help Generation. Proceedings, Working Conference on Engineering for Human Computer Interaction.
- Sugihara, K. 1986. *Machine Interpretation of Line Drawings*. Cambridge, MA: The MIT Press.
- Sukaviriya, P. 1988. Dynamic Construction of Animated Help from Application Context. Proceedings, ACM SIGGRAPH Symposium on User Interface Software.
- Sukaviriya, P., and J.D. Foley. 1990. Coupling a User Interface Framework with Automatic Generation of Context- Sensitive Animated Help. Proceedings, ACM SIGGRAPH Symposium on User Interface Software and Technology.

APPENDIX: ADDRESSES OF PARTICIPANTS

Bruce V. Ballengee

Andersen Consulting Manager
1950 Stemmons Freeway, Suite 5002
Dallas, TX 75207 USA

Email :

Fax : (214) 339-7316, (214) 761-8181

Phone : (214) 761-8151

Meurig Beynon

Dept. of Computer Science
University of Warwick
Coventry CV4 7AL UK

Email : wmb@CS.WARWICK.AC.UK

Fax : +44 203 525714

Phone : +44 203 523089

Shi-Kuo Chang

Dept. of Computer Science
University of Pittsburgh
322 Alumni Hall

Pittsburgh, PA 15260 USA

Email : chang@cs.pitt.edu

Fax : (412) 624-8465

Phone : (412) 624-8423

Susan Finger

Carnegie Mellon University
3318 Doherty Hall
Pittsburgh, PA 15213 USA

Email : Susan.Finger@ISL1.RI.CMU.ED

Fax : (412) 621-5477

Phone : (412) 268-8828

Annette Herskovits

Wellesley College
currently at:
Computer and Information Science
200 South 33rd Street
Philadelphia, PA 19104-6389 USA

Email : hersk@linc.cis.upenn.edu

Fax :

Phone :

Leone Barnett

3M / Software & Electronics Resource Center
3M Center, 260-6A-08
St. Paul, MN 55144 USA

Email : lbarnett@3m.com

Fax :

Phone : (612) 736-3144

Eric Bier

Xerox PARC
3333 Coyote Hill Rd.
Palo Alto, CA 94304 USA

Email : Bier.parc@Xerox.com

Fax :

Phone : (415) 494-4439

Max Egenhofer

NCGIA and Dept. of Surveying Engineering
University of Maine
107 Boardman Hall

Orono ME 04469 USA

Email : max@mecan1

Fax : (207) 581-2206

Phone : (207) 581-2114

Andrew Frank

NCGIA and Dept. of Surveying Engineering
University of Maine
107 Boardman Hall

Orono ME 04469 USA

Email : frank@mecan1

Fax : (207) 581-2206

Phone : (207) 581-2174

Deepa Krishnan

Dept. of Information Science, Faculty of
Science
The University of Tokyo
7-3-1, Hongo, Bunkyo-ku

Tokyo 113 Japan

Email : deepa@is.s.u-tokyo.junet

Fax :

Phone :

Werner Kuhn

NCGIA and Dept. of Surveying Engineering
University of Maine
107 Boardman Hall
Orono ME 04469 USA
Email : kuhn@mecan1
Fax : (207) 581-2206
Phone : (207) 581-2118

Jock MacKinlay

Xerox PARC
3333 Coyote Hill Rd.
Palo Alto, CA 94304 USA
Email : Mackinlay.pa@Xerox.COM
Fax :
Phone : (415) 494-4335

Sean Philip McElroy

Cognition Corp.
755 Middlesex Turnpike
Billerica, MA 01821 USA
Email : sean@dandelion.ci.COM
Fax :
Phone : (508) 667-7900

Hans-Joachim Novak

Wissenschaftliches Zentrum IBM
Institut für wissenschaftliche Systeme, Wissens-
und Sprachverarbeitung 2
Postfach 80 08 80
Stuttgart 80 D-7000 BRD
Email : novak@ds0lilog
Fax :
Phone : (0711) 6695-447

J. Raoul Ramirez

Center for Mapping
The Ohio State University
1216 Kinnear Road
Columbus, OH 43212 USA
Email : RAUL@mapvxa.cfm.ohio-state.E
Fax : (614) 292-8062
Phone : (614) 292-1600

Clayton Lewis

Dept. of Computer Science and Institute of
Cognitive Science
University of Colorado
Campus Box 430
Boulder, CO 80309 USA
Email : clayton@sigi.colorado.edu
Fax : (303) 492-2844
Phone : (303) 492-6657

David L. Mulsby

Knowledge Sciences Laboratory, Department of
Computer Science
The University of Calgary
2500 University Drive NW
Calgary, Alberta T2N 1N4 Canada
Email : mulsby@cpsc.ucalgary.ca
Fax : (403) 284-4707
Phone :

Lil Mohan

Knowledge Based Systems Lab
School of Electrical Engineering
Purdue University
Box 111, EE Building
West Lafayette, IN 47907 USA
Email : lil@ec.ecn.purdue.edu
Fax : (317) 494-6440
Phone : (317) 494-0724

David Pugh

Dept. of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3890 USA
Email : david.pugh@cs.cmu.edu
Fax :
Phone : (412) 268-7555

Piyawadee (Noi) Sukaviriya

Dept. of Electrical Engineering and Computer
Science
The George Washington University
801 22nd Street, N.W. - Room T607
Washington, DC 20052 USA
Email : noi@seas.gwu.edu
Fax :
Phone : (202) 994-5912

Martin Tuori

Alias Research Inc.

110 Richmond Street E., 5th Floor

Toronto, Ontario M5C 1P1 Canada

Email : martin@alias.uucp

Fax : (416) 362-0630

Phone : (416) 362-9181

Paul Wilson

2956 Joaquin Miller Road

Oakland, CA 94602 USA

Email :

Fax :

Phone : (415) 530-0456