A Visual Representation for Knowledge Structures

Michael Travers

MIT Media Lab
Massachusetts Institute of Technology
20 Ames St
Cambridge MA 02139
mt@media-lab.media.mit.edu

ABSTRACT

Knowledge-based systems often represent their knowledge as a network of interrelated units. Such networks are commonly presented to the user as a diagram of nodes connected by lines. These diagrams have provided a powerful visual metaphor for knowledge representation. However, their complexity can easily become unmanageable as the knowledge base (KB) grows.

This paper describes an alternate visual representation for navigating knowledge structures, based on a virtual museum metaphor. This representation uses nested boxes rather than linked nodes to represent relations. The intricate structure of the knowledge base is conveyed by a combination of position, size, color, and font cues. MUE (Museum Unit Editor) was implemented using this representation to provide a graphic front end for the Cyc knowledge base.

INTRODUCTION

Cyc and Knowledge Representation

The Cyc project [Lena88] is an effort to build a very large knowledge base that encompasses a broad range of common-sense knowledge. The knowledge base consists of a network of interrelated *units* (frames), each of which corresponds to a thing to be represented. These can be physical objects, abstract concepts, classes, or anything else of interest. Cyc presently consists of about a hundred thousand units; it is projected to have several million when it is completed[1].

1Some details of Cyc have been simplified in this paper. For a fuller presentation, see [Lena88].

Units contain *slots* that represent properties of the unit. Slots are themselves units; by convention, their names begin with a lowercase letter. Slots have values which are usually lists of other units. Within text or programs units are designated by a prefix of "#%". This prefix may be omitted if there is no question of ambiguity. A typical unit presented as text looks like this:

```
#%People:
#%english: ("The class of all human beings")
#%specs: (#%Workers #%MalePeople #%FemalePeople
   #%USCitizens ...)
#%genls: (#%Mammals #%IntelligentEntities)
#%allElements: (#%MikeTravers #%DougLenat
   #%BobDobbs ...)
#%canHaveSlots: (#%citizenship #%languagesSpoken
   #%countryOfOrigin...)
...
```

Figure1. Textual representation of a Cyc unit.

This unit represents the class of all people, and its slots specify that it has specializations (subsets) such as #%Workers and #%MalePeople, generalizations (supersets) #%Mammals and IntelligentEntities, and elements that include all the individual people that Cyc has reason to know about. The #%english slot contains human-readable documentation. #%canHaveSlots is an example of a slightly more complex slot. It specifies that elements of the class #%People are allowed to have the properties (slots) of citizenship and languages spoken.

MUE, a Navigation Tool for Cyc

Although Cyc is primarily a knowledge base for use by AI programs, the knowledge it contains can and must be accessible directly by humans during Cyc's developmental stage. It thus becomes necessary to treat Cyc as a knowledge exploration environment. The notion of a knowledge environment based on detailed representation was an important precursor to the development of Cyc [Lena84].

A large, densely interconnected structure like Cyc poses formidable navigation problems. When the MUE effort began, the Cyc user interface consisted of a textual browser/editor, and a node-and-line graphical display that was becoming impossibly tangled as the knowledge base grew. MUE began as an effort to design a new graphic browsing interface that exploited the spatial navigation skills of the user by mapping the KB into a simulated physical space.

The original plan for a spatial interface was to use virtual environment technology such as head-mounted displays [Fole87] to present the knowledge base as a museum. The Cyc museum was to have used floors, sections, corridors, exhibits, alcoves, display cases, and other architectural metaphors to effectively spatialize the highly abstract structure of Cyc. This plan was inspired by the museum's traditional role as a place to present broad areas of knowledge; and by theories of architectural space that utilized overlapping hierarchical structures similar to those of Cyc [Alex64].

Creating a virtual space to represent Cyc is not straightforward. Because Cyc is so densely interconnected, it does not map easily into a geometric space. The museum would have to have spacewarps or wormholes to connect related units that might be far removed in the museum space. While Cyc's structures are similar to certain architectural formalisms, in practice they are far too deep, bushy, and tangled to map directly into a navigable architectural space. Cyc's

structures are also rather fluid, in that new intermediate levels are constantly being created. Thus a fixed mapping of units to architectural features seemed impossible to implement.

The solution was to use a uniform metaphor of rooms. Rather than assign some units to larger levels of architecture, every unit representation in the museum space was simply a room. Every unit could appear both as its own room or as an object inside another room. The museum traveller could "jump in" to any visible unit and find herself warped to an entire room that represented that unit. Within a room, objects would appear at fixed places. Certain sets of units that did have a ready mapping into two- or three-dimensional space (*i.e.*, geographic information) could be browsed via ordinary spatial operations.

With this metaphor, in a certain sense, everything is inside of everything else! When you enter a unit, it becomes outermost, and the things that are connected to it become included in it, including the room you just left. For instance, if you enter the Serengeti room, inside it will be areas devoted to inhabitants of the region such as the Masai tribe, gazelles, and tourists. But if you enter the Masai unit, inside there will be an area labelled locales that will include the Serengeti and other places frequented by the nomadic Masai.

The full-scale virtual environment has yet to be implemented. However, the notion of representing a unit as a room and related units as being inside that room was used to create a less ambitious two-dimensional browser. The implemented version does not try very hard to preserve the properties of physical space. Instead, the notion of warping has become the primary navigational technique.

A MUE display is a set of nested boxes, each box representing a unit. For example, the #%Person unit above would be drawn as the set of boxes below.

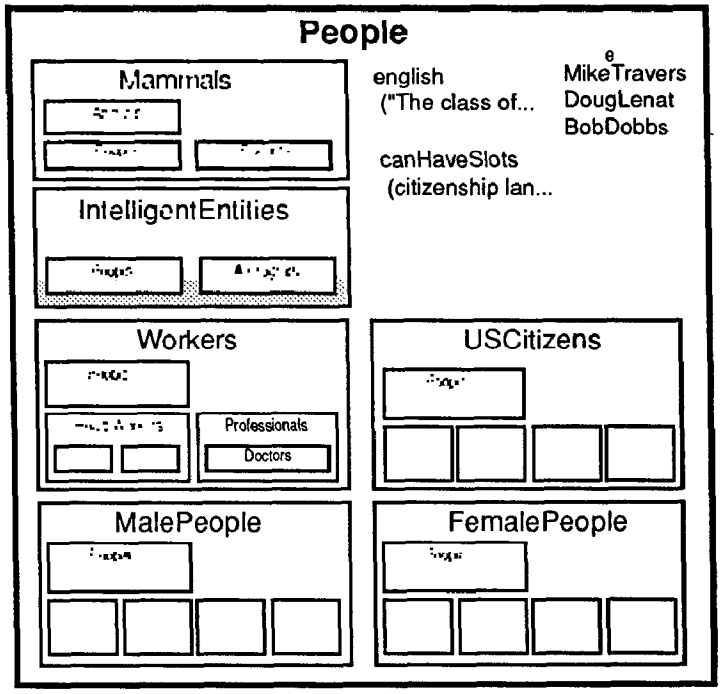

Figure 2. A MUE Display.

Each internal box is recursively subdivided and elaborated as space permits. In practice, a 1024 x 1024 screen permits from three to six levels of recursion (more than can be shown here). The details of the layout algorithms are explained below.

Hypertexts and Semantic Networks

Hypertext and KB-networks share some superficial similarities: they are structured as graphs, and they claim to represent, present, or even constitute "knowledge". But these similarities are rarely acknowledged to be as important as the differences in purposes and styles assumed by the two projects. In fact, their is some low-level hostility between the two groups: Hypertext people believe that the AI project of formal representation is unnecessary, while AI people feel that the hypertext goal of putting text online is unwarranted interference with their more ambitious goal of putting knowledge online in a form that can be used by programs as well as people.

The nature of representation in these two similar yet disparate schemes is a complex philosophical issue that cannot be treated here. For our purposes, it will suffice to simply think of knowledge representation languages as simply a new form of text, albeit a highly dynamic form that can rearrange and represent itself in novel ways. This view will allow us to make use of the highly developed formal methods of AI without having to accept its concomitant semantic theories.

DISPLAY LAYOUT IN MUE

This section explains how various visual elements of the MUE display are related to the Cyc structures they represent.

Units are Boxes

Units are represented on the screen by nested rectangular boxes. The unit's name appears at the top of the box, unless the box is smaller than a fixed size. or the name is too large. In those cases, moving the mouse over the box will cause the full name to appear.

Every MUE unit display is recursively detailed. That is, the neighbors of every unit box are displayed within it up to the limits of screen space.

Relationships are Indicated by Inclusion

MUE's basic display technique is to show relationships by recursive inclusion. If, for instance, #%BlueCollarWorker is a #%spec of #%Worker and has three specs of its own, than this might appear as:

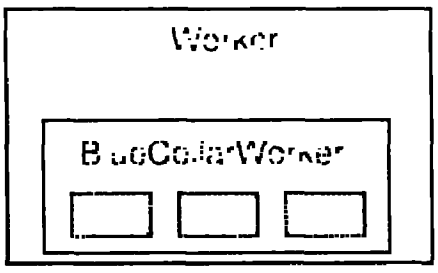

Figure 3. Use of inclusion and color to indicate recursion depth of a relationship.

Color is used to indicate type and nesting depth of relationships

In the MUE display, color is used in two ways: hue is used to indicate the type of relation (*i.e.*, blue indicates #%specs relations) and saturation is used to indicate the depth of the relationship from the current outermost unit. For instance, in the above display of Workers, the blue color becomes more saturated as you go inwards from more general to more specialized units.

Graph structure is made evident through recursive nesting

The Cyc KB as a whole is an unrestricted labelled graph, but certain of its slots are defined so as to form subgraphs that take the form of trees or directed acyclic graphs (DAGs or tangled hierarchies). #%specs and its inverse #%genls are the most important of these, but there are others, such as #%parts and #%partOf. Slots fall into a tangled hierarchy of their own as specified by the #%specSlot and #%superSlot relations. MUE uses these subgraphs as a navigational skeleton.

For MUE's purposes, relations are divided up into classes, based on the information encoded about them in Cyc. The two classes most relevant to the structure of screen layouts are restrictions and loosenings. For example, #%specs is a restriction while #%genls is a loosening. In general, if a slot is a loosening its inverse will be a restriction.

Each unit display is divided up into four or fewer sections: restrictions (usually the bottom half of the box), loosenings (the upper left quadrant), misc slots, and elements (which together occupy the upper right quadrant). See Figure 2.

NAVIGATION IN MUE

The previous section dealt with MUE displays as static visual constructs. This section explores how these elements are used dynamically during the process of navigation through knowledge structures.

The outermost box is the focus unit

The user is considered to be in a particular unit or room. This unit is represented by a box the size of the screen and is referred to here as the *focus*. MUE tries to display units that are in the neighborhood of the focus, as defined by distance along a given set of relationships. It does this simply by recursively elaborating the unit display boxes until they become too small to include any more detail.

Navigation in Cyc is done by mousing on an inner unit display box, which will expand to the size of the full screen and become the focus of the next display.

Movement is via re-rooting

Mousing on a unit display will cause that unit to become the focus of a new display. This shifts the positions and sizes of all of the unit displays present. In the box representation, this can be thought of as turning a display inside-out, since the moused-upon inner box will become the outermost box of the new display, while the former outermost box will become an inner, smaller one. This process may also be thought of as re-rooting a tree by turning an arbitrary node into the root of a new tree:

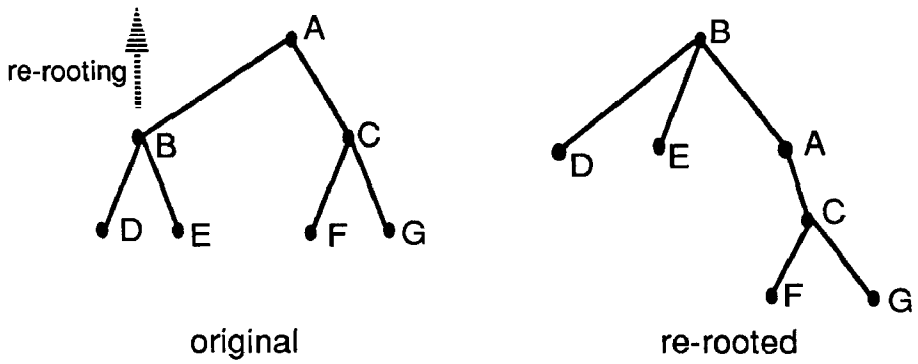

Figure 4. Re-rooting a tree.

MUE never changes the connectivity of the graph that it's displaying (which is part of Cyc), but can re-root it to form different displays. Here is a part of Cyc's class structure, which includes the graph on which Figure 2 is based:

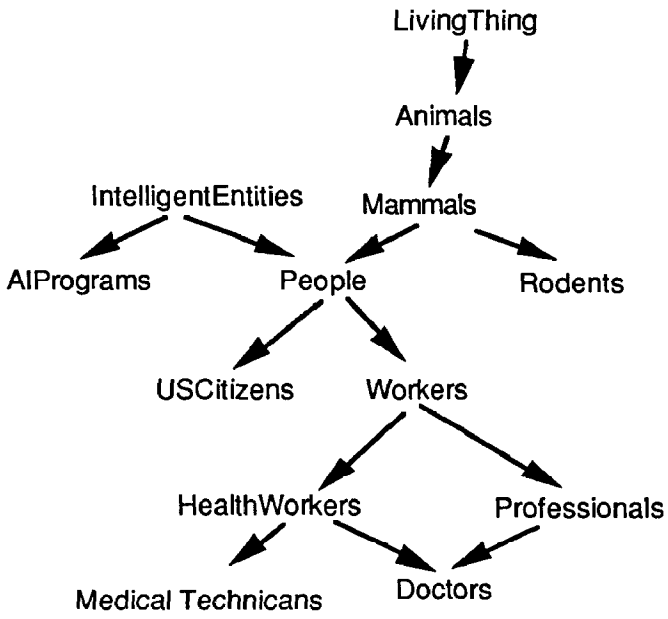

Figure 5. A class graph.

To generate a MUE display, we want to re-root the structure at the unit we are focusing on. For instance, #%People:

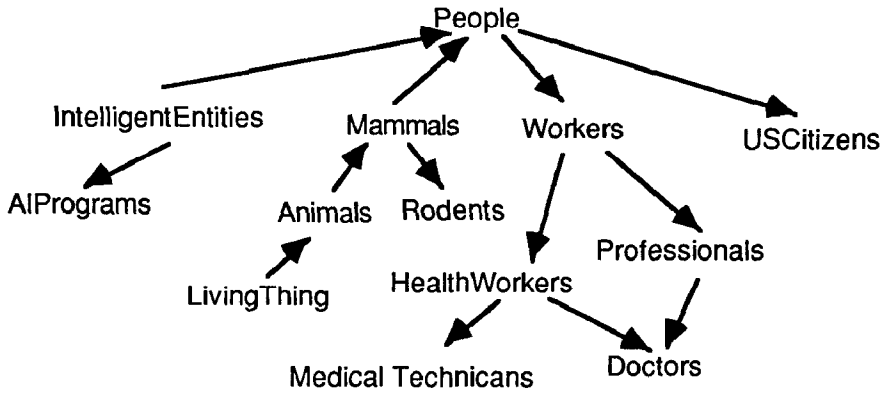

Figure 6. The same class graph, rerooted.

In the above diagram, we have preserved the original relationships and only altered their visual presentation. Some of our directed links now point upwards. To make a MUE display based on this graph, we wish to convert it to a tree-like structure with all links pointing downward (corresponding to inclusion). We can do this by replacing upward-pointing links with their inverses (in this case, replacing #%specs with #%genls). We indicate the #%genls links with shaded lines below (corresponding to the use of color in MUE):

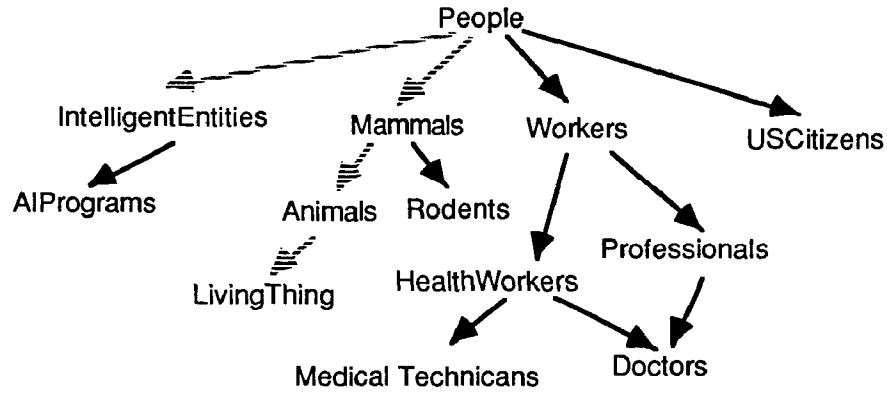

Figure 7. The re-rooted graph relabelled to form a tangled hierarchy.

This diagram could now be used to generate the MUE display in Figure 2, by presenting the relationships as inclusions (see the next section for the treatment of nodes with multiple parents).

Re-rooting allows MUE to pick any unit as the focus by making it the top of a re-rooted tree. Cyc's ability to compute the inverse of a relationship allows arbitrary graphs to be converted into trees and thus displayed via inclusion. The re-rooting process can be animated, but the resulting motion is hard to comprehend, since things are moving in many directions at once. [Lieb89] describes a similar display and animated transform using simpler structures and three-dimensional graphics.

Some units are duplicated

Note that the above diagram is not actually a proper tree: #%Doctors has two parent nodes, #%HealthWorkers and #%Professionals. Since our box diagrams must be tree structured, we can't display this directly. Instead we make two displays for #%Doctors, one inside #%HealthWorkers and one in #%Professionals. To save time, only one of these displays (arbitrarily chosen) will be recursively expanded.

A rear-view mirror metaphor helps make explain the re-rooted display

If it were possible, we'd like to maintain the consistency of the interface metaphor by putting loosenings outside the focus unit. But that would mean putting them outside the screen. Even if the focus were not the size of the screen, there may be more than one loosening of it, and it's impossible to draw two separate boxes surrounding the focus without having them intersect or having one include the other (which would convey false information).

Re-rooting provides a means for turning a neighborhood of some focus unit into a tree structure with the focus as the root and both loosenings and restrictions as subordinate, internal units. But displaying this as a tree, as in the above diagrams, is less than intuitive. We need a means of indicating to the user the difference between units that are beneath the focus in an absolute sense and those that were placed there as a result of re-rooting.

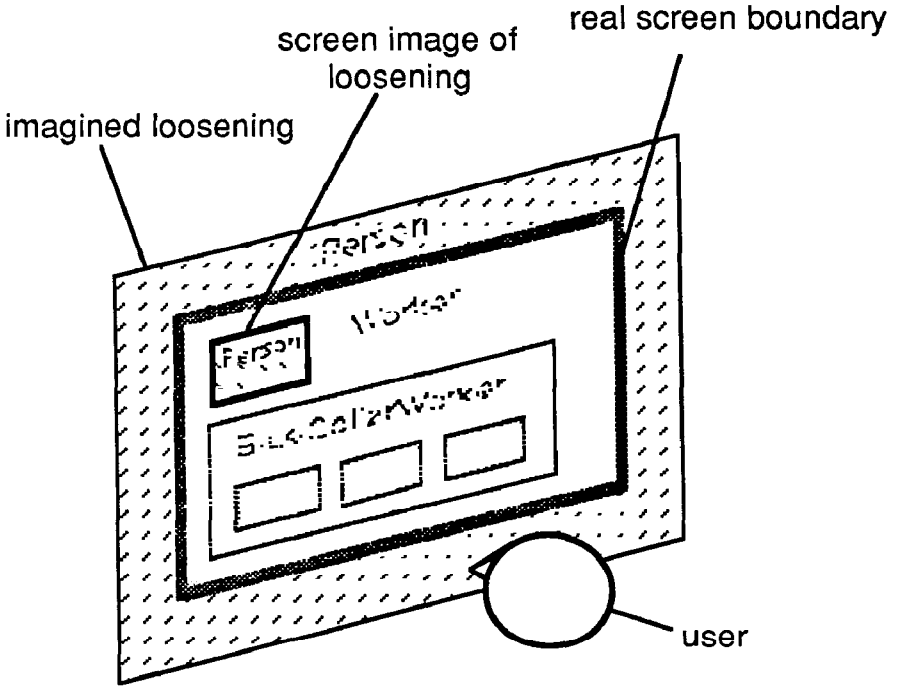

Figure 8. Viewing a MUE display through an imaginary rear-view mirror.

MUE uses position and color cues to distinguish re-rooted loosenings from restrictions. We ask the user to perform a mental inversion, and imagine that the loosenings are made visible through a "rear-view mirror" that brings them into the visible space of the display via a transform.

In Figure 8, the user is focusing on the #%Worker unit, and can see its sole loosening, #%Person, through the rear-view mirror. The rear-view mirror functions recursively as well, so that several levels of loosening can be seen in a typical display.

This layout technique has the disadvantage of having to be explained, and having no natural visual cues to remind the user what's going on. One possible solution is stereo displays that could indicate the true depth of units. Another is using head-mounted displays to generate both stereopsis and visual feedback.

EDITING IN MUE

Cyc's primary editing tool is the Unit Editor (UE) which displays units in a text-based notation (as in Figure 1). The UE can display up to four units and their slots at a time, and allows the user to explicitly edit the contents of any slot.

MUE provides an additional spatial editing capability, which can simplify certain editing tasks involving restructuring of hierarchical parts of the KB. In keeping with its use of inclusion as its primary visual metaphor, editing operations in MUE are accomplished by dragging boxes inside each other. Dragging a box will create a new link between the dragged unit and the destination, and optionally break the old link. These operations can be particularly useful when creating or modifying large class hierarchies.

In such tasks, it is often necessary to interpose intermediate classes between existing general and special-purpose classes. MUE provides a special operation, Subsume, which accomplishes this task, essentially surrounding an existing unit with a new, more general unit (we use the language of specialization and generalization in this description, although the operation can apply to any restriction/loosening type of relationship).

OTHER APPLICATIONS

This section examines the use of MUE for browsing structures other than Cyc.

Hypertext

Text Structures

MUE is useful for browsing graph-like text structures. Electronic mail conversations can form such structures by virtue of recording the links between replies and the messages they are in reply to. When the role that a message plays in a conversation is known (*i.e.*, an agreement, disagreement, or request for elaboration) the structure of the conversation becomes both more intricate and more important. Hypertext argumentation systems can benefit from the truth-maintenance approach that records justifications for facts and can propagate support, contradiction, or believability through a dependency network [Stef87].

MUE has been experimentally adapted to display mail structures based on the reply relationship. While it appears to be of some utility, it falls short of being a good visual representation, primarily because unlike Cyc units, mail messages do not have short unique identifiers to use as box headings.

Browsing Program Structure

The Symbolics Lisp environment includes facilities for recording the relationships between entities of the Lisp language [Symb88]. For a given function (or macro, type, or other named object) you can obtain a list of callers or callees. MUE has been adapted to display these relations, providing the ability to browse the structure of large Lisp programs. Lisp entities do have short unique identifiers and thus MUE's visual presentations are reasonably informative.

CONNECTING CONCEPTUAL AND TEXT STRUCTURES

Cyc has a rudimentary method of connecting conceptual and text structures by means of a special class of units called TextUnits. A TextUnit is simply a kind of Cyc unit that represents a piece of text. An AccessibleTextUnit is a TextUnit that points to text that is accessible online, thus providing an interface between Cyc and arbitrary bodies of text.

TextUnits have been used to create a conceptual index into the text of Melville's *Moby Dick*. At present this simply encodes some very basic properties of the novel. This includes its textual parts such as chapters, its characters (and which chapters they are mentioned in), and significant plot events (and which chapters they happen in and which characters are involved in them). While this effort has so far been rudimentary, this could eventually evolve into an intelligent concordance or reader's guide.

One way to think about using a KB to navigate text is by analogy to the Britannica Propaedia/Micropaedia [Pree79]. The Micropaedia consists of small articles about a great many concepts whose main function is to define and point you to a larger article in the main body of the encyclopaedia (the Macropaedia). Cyc units can fulfill a similar function of indexing text, but because the Cyc units are much more finely detailed and cross-referenced the utility would be orders of magnitude greater.

RELATIONS TO OTHER WORK

SemNet and Fisheye Views

SemNet [Fair88] has extended the techniques of network display to include three-dimensional display and sophisticated techniques for mapping node positions into display space. SemNet makes use of fisheye views [Furn86], a class of display techniques that focus on a particular point of a large structure. More detail is displayed near the focus and less detail as one moves further away. MUE's display is similar to a fisheye view in the above sense, although it does not make explicit use of an "interestingness metric" as the formal fisheye technique does. This technique offers the additional facility of defining some items (landmarks) as having an intrinsically high degree of interest, so that they are visible from a wider range of foci than normal items.

SemNet was designed around a static three-dimensional layout of knowledge and concentrated on the problems of layout and navigation within such a space. This was recognized to be inadequate for complicated networks, and some effort was made towards temporarily moving network nodes to form clusters dynamically. MUE, in contrast, has no fixed mapping into a physical space and instead emphasizes the dynamic creation of layouts based on neighborhoods as defined by the underlying knowledge base.

Nested Hypermedia Structures

Previous systems have used nested boxes to represent hypertext and other browsable structures. [Fein82] used nesting to represent tree-structured documents and could include graphics and animation. Special editors were provided for the various levels of the hierarchy: chapter, page, etc., but had no method for automatically constructing layouts from the underlying structure.

SDMS (Spatial Data Management System) [Bolt79] was an early effort to create a rich spatial browsing environment. The initial version used a tree-structured database, and allowed the user to browse by successively entering and expanding the various nodes of the tree. This was deemed to be too confusing, so later versions abandoned the hierarchical structure for a flat data space. The flat space allows the user a more natural way to use spatial navigation, but does not scale up to very large or inherently structured knowledge bases.

Boxer: Nested Boxes as a Programming Construct

Boxer [diSe86] is an interactive programming system that uses boxes and inclusion as its basic visual metaphor. Designed as a Lisp-like system, it makes use of the close analogy between inclusion of boxes and nesting of parentheses. Boxer has had some success with novice users, suggesting that inclusion is a suitable visual representation for abstraction.

Displaying the Structure and Execution of Programs

[Lieb89] has used the graphic ideas developed for MUE to display the execution of Lisp programs. The program is represented by nested boxes (in 3D) which reconfigure themselves as parts of the program get evaluated. The reconfiguration is similar to MUE's zooming process, but animated in three dimensions using size changes and rotation.

CONCLUSIONS

MUE was the result of an effort to come up with an alternative visual representation for complex knowledge structures. It embodies some new techniques for translating complex graph structure into visual representations. The system is in use by knowledge engineers but has not been evaluated as to its effectiveness in comparison with more traditional node-and-link representations. The same system has been adapted to show text and software structures. A preliminary effort has been made to use the knowledge base as a guide for navigating text.

Directions for future work include: designing richer displays that incorporate pictorial and iconic information; using other sensory cures such as sound to enhance the virtual environment; using animation and stereoscopic displays to convey the navigational processes; and closer interconnection between knowledge bases and hypertexts.

ACKNOWLEDGEMENTS

This work was undertaken as part of the Cyc project at the Microelectronics and Computer Technology Consortium. Doug Lenat and the other Cyclists contributed suggestions, a user community, and a knowledge base worth exploring. Text units were developed in collaboration with Margaret Minsky.

REFERENCES

[Alex64] Alexander, Christopher, *Notes on the Synthesis of Form*, Cambridge: Harvard University Press, 1964.

[Bolt79] Bolt, Richard A., Spatial Data-Management, Arhcitecture Machine Group, Massachusetts Insitute of Technology, 1979.

[diSe86] diSessa, Andrea A. and Harold Abelson, "Boxer: A Reconstructable Computational Medium", *Communications of the ACM* 29(9), p. 859-868, 1986.

[Fair88] Fairchild, Kim M., Steven E. Poltrock, and George W. Furnas, "SemNet: Three-Dimensional Graphic Representations of Large Knowledge Bases", in *Cognitive Science and its Applications for Human-Computer Interaction*, ed. Raymonde Guindon, Lawrence Erlbaum, Hillsdale NJ, 1988.

[Fein82] Steven Feiner, Sandor Nagy, and Andries van Dam, "An experimental system for creating and presenting interactive graphical documents", *ACM Transactions on Graphics* 1(1), p. 59-77, 1982.

[Fole87] Foley, James D., "Interfaces for Advanced Computing", *Scientific American*, 257 (4), p.126-135, October 1987.

[Furn86] Furnas, George W., "Generalized Fisheye Views". *Proceedings of CHI'86 Human Factors in Computing Systems*, p. 16-23, 1986.

[Lena84] Lenat, Douglas B., Alan Borning, David McDonald, Craig Taylor, and Stephen Weyer, "Knoesphere: Building Expert Systems with Encyclopaedic Knowledge", *Proceedings of the IJCAI-84*, pp. 167-169, 1984.

[Lena88] Lenat, Douglas B. and R. V. Guha, The World According to Cyc, MCC Technical Report No. ACA-AI-300-88, Microelectronics and Computer Technology Corporation, 1988.

[Lieb89] Lieberman, Henry, "A Three-dimensional Representation for Program Execution", submitted to *Visual Languages '89*.

[Nels87] Nelson, Theodore H., *Computer Lib/Dream Machines*. Redmond, Washington: Tempus Books, 1987.

[Pree79] Preece, Warren E. (ed) *Encyclopaedia Brittanica (15th edition)*, Chicago, 1979.

[Stef87] Stefik, M, Foster G, Bobrow D, Kahn K, Lanning S, Suchman L, "Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings", *Communications of the ACM*, January 1987.

[Symb88] Symbolics, Inc, *Genera Users' Guide*, 1988.