DOES VISION NEED A SPECIAL-PURPOSE LANGUAGE?

VISION FLASH 50

by

Scott Fahlman

September 1973

Massachusetts Institute of Technology

Artificial Intelligence Laboratory

Abstract

This paper briefly discusses the following questions: What are the benefits of special-purpose languages? When is a field ready for such a language? Are any parts of our current vision research ready?

Work reported herein was conducted at the Artificial Intelligence Laboratory, a Massachusetts Institute of Technology research program supported in aprt by the Advanced Research Projects Agency of the Department of Defense and monitored by the Office of Naval Research under Contract Number N00014-70-A-0362-0005.

Under certain conditions, a special-purpose language can be an exceptionally powerful programming tool. Such languages have been used to good effect for graphics, natural language parsing, and for all types of pattern matching. Micro-Planner and Conniver have made it far easier to program certain types of problem-solving systems and to organize perceptual programs along heterarchical lines. Even LISP can be thought of as a special-purpose language, since it provides extensive facilities for processing list-structured data, at some expense to its efficiency in handling arrays and numbers. This paper briefly addresses the question of when it is useful to create a special-purpose language and, more specifically, whether any part of the vision/robot problem could benefit from the creation of such a language at this time.

First, we should try to figure out what we are talking about: The term "language" has been used in many different ways. It is sometimes maintained that a language must have its own peculiar syntax or at least a separate interpreter; otherwise it is just a collection of functions in some other language. For the purposes of this paper, however, a broader definition will be used: A group of related facilities will constitute a language whenever they provide the user with an essentially new and coherent level upon which to think, regardless of syntax or details of implementation. If the language is to provide a clear organizational level for its author, and if it is to be useful at all to anyone else, it must be fairly well polished, reasonably complete, and above all well documented.

Our consideration will not be limited to procedure-defining languages alone. Systems of data-description can form new conceptual levels in ways very anologous to the levels of programming languages: The difference between "2 x 3 x 5 bricks" and the equivalent collection of

faces and vertices is closely parallel to the difference between CONS and the corresponding set of machine instructions. The same requirements of neatness, completeness, and documentation apply.

What does a language do for us? Of course it provides a set of facilities that are useful in the given domain, but far more important is its value as a commitment to a certain way of doing those things that must be done often. The user is not only spared the labor of repeatedly writing out a lot of details, but he is also spared the demands of these details upon his attention. This is what creates the new level of organization, thought, and communication. What before were matters requiring conscious decision, now are completely implicit, perhaps even unknown to the programmer.

Of course, a commitment cuts two ways. By turning control of basic stylistic decisions over to the language designer, the user relinquishes much of his freedom and flexibility. If the decisions embodied in the language are consistently those that the user himself would have made, or if the choices are equally good, nothing is lost. But the danger is great that an elegant language based upon premature decisions will seduce large numbers of users into unprofitable ways of thinking. This process is all the more insidious because the critical decisions are hidden below the surface level and may never be considered by the user at all. Most languages have some mechanism for handling exceptions, for escaping to a more general language in cases where the old commitments are not desirable, but it is part of the value of special-purpose languages that these options are not normally considered. Special-purpose languages inherently limit the imagination; that is their job.

All of this should give us a pretty good idea of when a special-purpose language should be created. The field in question must be mature enough that the basic underlying decisions have been made and tested, and that the remaining work consists mostly of applying the mechanisms that have been decided upon. To borrow a metaphor from Thomas Kuhn (The Structure of Scientific Revolutions-highly recommended) a language is the embodiment of an established paradigm in the field. It is of inestimable value for "normal research" but its acceptance dictates

that major changes in approach can only be achieved by means of a traumatic "revolution" in the field.

We now turn to the question of whether any of the vision group's current activities fall within a well-established paradigm and are thus ready for the imposition of a language. In general, my answer will be "No." Many vision activities seem still to be in a pre-paradigm stage of more or less unstructured groping, while other areas seem to be in the midst of major paradigm shifts as attention is turned from the Blocks World to more interesting problem domains. Nothing seems to be in stable production-mode at the moment.

We are, of course, thinking of language-making in the tool-building sense; that is, as the consolidation and packaging of a pre-existing paradigm. The creation of the paradigm itself is the highest level of creative research and should not be hidden under the heading of language development unless the language itself is the goal and object of study (as in, say, the creation of FORTRAN.) We will now examine the various areas of current vision group research and see what the prospects are.

1) The area of low-level image processing appears to be far too unsettled at the moment for a language to be beneficial. There is as yet no general consensus as to the best way to find and verify areas, edges, and vertices; how to deal with color, texture, non-uniform lighting, and motion; and how much (if any) initial pseudo-parallel processing of a scene should occur before the content directed heterarchical processes take over. Indeed, if the current trend toward full heterarchy down to the lowest levels continues, a special low-level language could be quite detrimental, since it would tend to create an hourglass barrier between functions written in this low-level language and the rest of the system. The lower-level routines should of course be neat and well-documented, but they should not form a separate grouping that differs significantly from the higher-level routines.

If, in the future, we should become more interested in doing some

amount of parallel, non-directed processing of incoming images, we might well want to look into the work that has been done or parallel image-processing languages. We might, for instance, need an easy way to specify operations in which stored images are added, subtracted, blurred, shifted, scaled, threshholded, and the results stored as new images. I suspect that such a language might have to be hand-compiled for efficiency, but it might still be useful as a medium for thinking and communication.

2) Waltz-type line and vertex labeling does constitute a well-defined and generally accepted paradigm, and Waltz's notation might well be viewed as already constituting a data-description language, though it is not optimally packaged for general use. It would appear, however, that this paradigm is not likely to be used much in the future, at least without radical changes. Waltx's particular problem was not very open-ended and it would appear that he has exhaustively solved it. There are a number of areas into which a Waltz-type approach might extend, but these extensions will require significant (and as yet unspecified) changes to the current Waltz notational system and to the programs using this system. It thus seems to be too late to create a language embodying the current Waltz conventions and too early to create a language for the extensions of Waltz's system. It may be that in this particular area, progress is inherently jerky, so that there will never be a good opportunity to develop a useful language.

3) The question of how to describe shapes and other high-level aspects of visual input is wide open at present. For curved objects there do not even seem to be many good ideas, let alone a consensus. Obviously, we need to develop a good visual description language, but this is a long-term research goal, not a case of tool-building.

4) It might seem attractive to have a special language for specifying hand and arm motions of the robot. It is currently believed, however, that even the low level hand motion functions should be heterarchically intergrated with the robot's systems. As was the case with low-level vision, there is considerable danger that the grouping of the hand routines into a separate language would interfere with this integration.

5) Finally there is the topic of the heterarchical control and communication mechanisms that underlie most of the vision and robot programs, both present and projected. I believe that CONNIVER, WIZARD, BUILD, and perhaps some of Freuder's ideas about advice and suggestions together form the beginnings of a paradigm, but that more experience is needed before we will be ready to cast these ideas into a solid, coherent language. There are just too many ideas around at present, and only natural selection will produce the winners. Probably in this area the best course would be to procede incrementally for a year or so, before making a final judgement as to what combination of facilities should be locked into a new language. Specifically, various control mechanisms, canned loops, and disciplines for creating and maintaining plans, hypotheses, suggestions, comments, and so on should all be generalized, documented, and kept in a library for various users to consider and try out. Only after some consensus is reached should there be an attempt to form the winning collection of features into a language that works well as a whole.