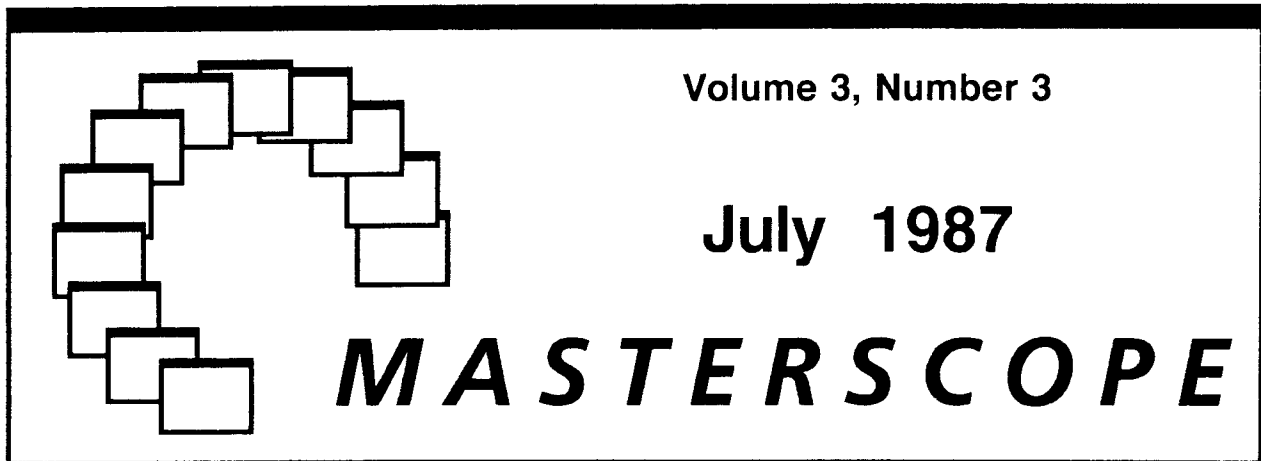




Letter from Xerox

Dear Users:

The Xerox Artificial Intelligence Environment (XAIE) has given Xerox AIS customers the tools necessary to develop a large number of applications in decision analysis and support systems. These systems provide expert guidance in design, planning, and diagnostic situations; in natural-language interface systems that expand the usefulness and accessibility of management information and database systems; in intelligent computer-aided engineering design and manufacturing systems; in real-time monitoring systems; and in a growing host of others in finance, industry, education, the military and professional practice.

This is a special edition of *Masterscope* in that it includes summaries of some of the applications that Xerox Artificial Intelligence System users will be demonstrating in the Xerox booth at AAAI '87 in Seattle, Washington. The customer applications to be shown are only a sampling of how our customers are using the powerful set of tools in XAIE to develop innovative and practical business applications at hundreds of installations around the world.

We hope that you will all be able to join the Xerox Artificial Intelligence Systems team at our booth to view the demonstrations and also to join us and other users in our hospitality suite and User's Group meeting.

Xerox Artificial Intelligence Systems

Table of Contents

Letter From Xerox	1
Xerox at AAAI	
Battelle Memorial Institute	
CSRL: A Language for Expert Systems for Classification	3
Stanford University Medical Center	
ONCOCIN: Therapy Planning System	4
OPAL: Graphical Knowledge Acquisition Interface	4
NEOMYCIN: Medical Diagnosis Program	5
HERACLES: Tool for Building Knowledge-based Consultation System	5
University of Colorado at Boulder	
ARITHPRO: A Computer Simulation of Children's Arithmetic Work-problem Solving	5
University of Colorado at Colorado Springs	
CHALLENGER: A Domain-Independent Intelligent Training System	6
IntelliCorp	
KEE: Expert System Development Shell	6
Honeywell Bull	
PERMAID: Expert Systems for Predictive Maintenance	7
Rockwell Graphic System	
PRESS LINE-UP ADVISOR	7
5thGen Technology Inc.	
FACTORIES: Factory Simulator	8
Mystec Associates, Inc	
AURORA: Expert System Line	8
Brooks Air Force Base	
AI AND EDUCATION: An Intelligent Tutor for the Space Domain	9
Martin Marietta Denver Aerospace	
AVDT: A Knowledge-based Interface to an Aerospace Vehicle Design Tool	10
Illation, Inc.	
OPS5: On the Xerox Series 1100	11
Georgia Institute of Technology	
INNER BUDGET	12
Articles	
A Virtual Notebook for Biomedical Researchers	
Dr. G. Gorry, Baylor College of Medicine	13
The Beta-Test Process from the Tester's Point of View	
Arun Welch, Ohio State University	17
Wizard's Shelf	18
The University Setting	
Guy Lapalme, Universite de Montreal	28
News and Announcements From Xerox AIS	
Educational Services Class Schedule	29

Xerox at AAI

CSRL: A Language for Expert Systems for Classification

Battelle Memorial Institute
Columbus, Ohio

CSRL (Conceptual Structures Representation Language) is a language to facilitate the development of classification expert systems based on a paradigm of "cooperating classification specialists." It was developed at the Ohio State University Laboratory for Artificial Intelligence Research under a Battelle Memorial Institute funding grant, and is used at OSU and Battelle for the implementation of knowledge-based systems for which classification is an appropriate problem-solving strategy. Battelle is currently commercializing CSRL in an industrial joint venture. Diagnosis, catalog selection, and certain types of simple planning are examples of tasks for which classification is well-suited. CSRL is intended to speed up the knowledge acquisition process by facilitating the input of knowledge in a form close to the domain experts' conceptual level and to make it easier to enforce consistency between the expert knowledge and its implementation.

The usual approach to building knowledge-based systems is to emphasize a general knowledge representation structure and different problem-solvers that use that knowledge. One difference in the CSRL approach is that the organization of knowledge is not intended as a general representation for all problems, rather it is tuned specifically for classification. By limiting the type of problem to be solved, a specific organizational technique (classification hierarchy) and problem-solving strategy (established-refine) can be used to provide focus and control in the problem-solving process.

A classification structure in CSRL is composed of a collection of specialists, each of which corresponds to a potential hypothesis about the current case. They are organized as a classification hierarchy, e.g. a classification of machine faults. The classificatory knowledge is embedded within the specialists, so that each specialists can be viewed as an individual problem solver with its own knowledge base. A top-down strategy called establish-refine is used, in which either a specialist establishes and then refines itself, or the specialist rejects itself, pruning the hierarchy that it heads. CSRL is a language for representing the specialists of a classification hierarchy and the classification knowledge within them. The classification knowledge is encoded at various levels of abstraction: message procedures, which describe the specialists' behavior in response to messages from other specialists; knowledge groups, which determine how data relate to features of the hypothesis; and rule-like knowledge, which is contained within knowledge groups.

CSRL is a significant tool for the engineering of expert systems that make use of classification as a problem-solving strategy. CSRL bridges the gap between simple rule-based tools which are limited in the scope of their application an general AI programming environments which require specialized skills for their use. Powerful computational techniques are effectively integrated in CSRL for application to a specific class of problems, making it easy for the system developer to understand the use and operation of the CSRL constructs.

ONCOCIN: Therapy Planning System

Stanford University Medical Center
Stanford, California

ONCOCIN is an advanced expert system for cancer treatment (oncology). It is designed for use after a diagnosis has been reached, focusing instead on assisting with the management of cancer patients who are receiving chemotherapy. Because anticancer agents tend to be highly toxic, and because their tumor-killing effects are routinely accompanied by damage to normal cells, the rules for monitoring and adjusting treatment in response to a given patient's course over time tend to be complex and difficult to memorize. ONCOCIN integrates a temporal record of a patient's ongoing treatment with an underlying knowledge base of treatment, aborting cycles, ordering special tests, and similar management details. The program uses such knowledge to help physicians with decisions regarding the management of specific patients.

A major lesson of past work in clinical computing has been the need to develop methods for integrating a system smoothly into the patient-care environment for which it is intended. In the case of ONCOCIN, the goal has been to provide expert consultative advice as a by-product of the patient data management process, thereby avoiding the need for physicians to go out of their ways to obtain advice. It is intended that oncologists use ONCOCIN routinely for recording and reviewing patient data on the computer's screen, regardless of whether they feel they need decision-making assistance. This process replaces the conventional recording of data on a paper flowsheet and thus seeks to avoid being perceived as an additive task. In accordance with its knowledge of the patient's chemotherapy protocol, ONCOCIN then provides assistance by suggesting appropriate therapy at the time that the day's treatment is to be recorded on the flowsheet. Physicians maintain control of the decision, however, and can override the computer's recommendation if they wish. ONCOCIN also indicates the appropriate interval until the patient's next treatment and reminds the physician of radiologic and laboratory studies required by the treatment protocol.

OPAL: Graphical Knowledge Acquisition Interface

Stanford University Medical Center
Stanford, California

OPAL is a graphical environment for use by an oncologist who wishes to enter a new chemotherapy protocol for use by ONCOCIN or to edit an existing protocol. Although the system is designed for use by oncologists who have been trained in its use, it does not require an understanding of the internal representations or reasoning strategies used by ONCOCIN. The system may be used in two interactive modes, depending on the type of knowledge to be entered. The first permits the entry of a graphical description of the overall flow of the therapy process. The oncologists manipulates boxes on the screen that stand for various steps in the protocol. The resulting diagram is then translated by OPAL into computer code for use by ONCOCIN. Thus, by drawing a flow chart that describes the protocol schematically, the physician is effectively, programming the computer to carry out the procedure appropriately when ONCOCIN is later used to guide the management of a patient enrolled in that protocol.

OPAL's second interactive mode permits the oncologist to describe the details of the individual events specified in the graphical description. For example, the rules for administering a given chemotherapy will vary greatly depending upon the patient's response to earlier doses, intercurrent illnesses and toxicities. etc. It permits the entry of an attention schedule for an agent based upon the patient's white count and platelet count at the time of treatment. Tables are generally found in the written version of chemotherapy protocols. Thus OPAL permits oncologists to enter information using familiar forms displayed on the computer's screen. The contents of such forms are subsequently translated into rules and other knowledge structures for use by ONCOCIN.

NEOMYCIN: Medical Diagnosis Program

Stanford University Medical Center
Stanford, California

NEOMYCIN is a medical diagnosis program that expands on MYCIN's knowledge base by including competing diseases that could be confused with meningitis, and by cleanly separating the domain knowledge from the control knowledge. This reconfigured knowledge base facilitates explanation and student modeling and provides an opportunity for teaching diagnostic strategy.

HERACLES: Tool for Building Knowledge-based Consultation Systems

Stanford University Medical Center
Stanford, California

The HERACLES heuristic classification shell is the generalization of NEOMYCIN. It includes the diagnostic procedure and relational language which can be adapted for developing other diagnostic or classification expert systems. GUIDON2 is a set of tutoring modules for any program based on the HERACLES shell. The GUIDON2 programs provide the user with tools to browse the knowledge base and to observe and control the reasoning strategy. GUIDON-WATCH allows the user to watch the expert system solve a diagnostic problem and to observe the nature of the search process. GUIDON-MANAGE allows the user to control the reasoning process by specifying high-level tasks for the expert system to perform.

ARITHPRO: A Computer Simulation of Children's Arithmetic Word-problem Solving

University of Colorado
at Boulder

ARITHPRO is a computer simulation of children's arithmetic word-problem solving behavior. It is an instantiation of a cognitive model, developed by Kintsch and Greeno (1985), of the procedures and knowledge required to solve such problems. The program solves arithmetic word problems by comprehending the text in which the problem is embedded and then representing numerical information as set of objects. Superstructures are then built from these sets, specifying the logical relations between the sets. Once the set relations have been made explicit, ARITHPRO then uses a counting procedure to derive the answer to the problems.

The primary goal of this project was to test the sufficiency of the Kintsch and Greeno model as a viable model of problem solving. Kintsch and Greeno argued that children's problem-solving behaviors result from an interaction of text comprehension processes and arithmetic problem-solving strategies. Therefore, a child's solution strategy is determined by how well he or she understands the problem described by the test. The model solves standard word problems at the level of a third grade child. A second goal of the project was to explain regularities in the data gathered from children's solution attempts, particularly in the area of frequently occurring errors. Most errors are not arithmetic errors, but the result of mis-reading of the problem test. ARITHPRO has produced an explicitly coded theory of children's cognitive processing during problem-solving attempts. ARITHPRO's programming architecture includes a lexicon written in Loops, and a production system, written in Interlisp-D. The program currently runs only on a Xerox 1100 series workstations. Embedded in this architecture are the main components of the Kintsch and Greeno model, knowledge concerning sets and their logical relations, arithmetic counting strategies, default solution strategies, a goal stack, a short-term memory buffer (STM) with a capacity of 5 units, and a long-term memory (LTM). The main data structures created are propositional frames (Winston & Horn, 1981) which constitutes a single unit in short-term memory.

CHALLENGER: A Domain-Independent Intelligent Training System

University of Colorado
at Colorado Springs

A team of researchers, Bob Gattis, Ray Kolbe, and Terry Trout, under the direction of Dr. Marijke Augusteijn, has developed a domain-independent Intelligent Training System prototype, called CHALLENGER, that is currently used to teach in the domain of orbital dynamics. CHALLENGER is implemented on a Xerox Artificial Intelligence workstation, using the object-oriented software development tool LOOPS. Domain-independence is achieved through modular design. The Domain Model is the only module requiring replacement when the instructional domain is changed. It is based on a cognitive learning model in which knowledge is acquired incrementally in different phases of learning. This model has been adapted to serve as a hierarchical knowledge representation of the Domain Model. The structure serves two purposes. It stores the instructional material and in addition, serves as an expert system that can solve problems in the instructional domain.

The intelligent features of the prototype are adaptation on the individual needs of the student with respect to level of understanding, learning style, pace, and preferences. The system also provides coaching during problem solving and analysis of a student's misconceptions. A Student Model aids in decisions concerning the tutorial flow. This model is essentially a pointer structure "overlaying" the Domain Model and is entirely system generated. It also stores information concerning learning style and a history of a student's performance in problem solving. The system has no prior knowledge about a new student. The initial interaction is menu driven. The system observes the student's selections and runs various discrimination nets once it has gathered enough data. There is a net for learning style and one for learning pace. After a student's style and pace are known to the system it adapts its instruction accordingly. A third net serves to change previous decisions, if necessary.

The system coach can be involved by the student in the problem solving phase. It has various review options and can also provide the next step toward the correct solution of the problem that the student is working on. The misconception analysis of formula based problems is domain = independent. The basic mechanism is comparison of a system generated expression tree representing the expected calculation with an expression tree formed according to the student's computations. Inferences concerning a student's misunderstandings are drawn from the differences. The techniques employed are general with respect to any formula based instruction.

KEE: Expert System Development Shell

IntelliCorp
Mountain View, California

The Knowledge Engineering Environment (KEE) is an object oriented, multiple paradigm, expert system development shell. The KEE system supports object oriented programming using a frame structure, LISP methods and active values. Tightly integrated with the frame structure is a rule system supporting forward and backward chaining, a multiple worlds facility and an assumption based truth maintenance system (ATMS). Interface design and construction are facilitated through KEEpictures, an object oriented graphics package. Developers interact with the KEE system development environment through a sophisticated mouse and menu interface that facilitates interactive system construction and rapid prototyping.

The IntelliCorp Trucking Company system is a demonstration application constructed by IntelliCorp. The system uses the tasks performed in creating schedules or a fleet of trucks to show a use of several advanced features of the KEE system (KEEworlds, ATMS) and their integration with the KEE rulesystem.

PERMAID: Expert System for Predictive Maintenance, Trouble Shooting and Recovery for Disk Subsystems

Honeywell Bull
Phoenix, Arizona

PERMAID (peripheral recovery and maintenance aid) is an expert system that is being used in predictive maintenance, trouble shooting and recovery for disk subsystems at HONEYWELL BULL customer sites. PERMAID runs on an XEROX 1185 workstation within the LOOPS environment. PERMAID can be used locally on the 1185 or may be remotely accessed through the HONEYWELL BULL telecommunications network.

PERMAID embodies a separate knowledge base for each of the three areas, predictive maintenance, trouble shooting, and file and media recovery. The knowledge bases consist of Directed Acyclic Graph (DAG), scripts, and rules. Other key components of PERMAID are the knowledge base update assistant and the inference engine. The purpose of the knowledge base update assistant is to assist the domain expert to update the knowledge engineer. The inference engine is based on abductive logic. It includes facilities for backtracking, certainty and impact factors, and the ability to suspend a session for later resumption.

A number of LOOPS features were used in the design of the inference engine. An example is the ability of one object to send a message to another object in the troubleshooting DAG. Within the knowledge base update assistant the lattice browser is used to build the DAG, assign values to instance variables, and build functions that will be used as methods for each of the objects in the DAG.

PRESS LINE-UP ADVISOR: Expert System for Configuration of Printing Presses

Rockwell Graphic Systems
Cicero, Illinois

The Graphics Systems Division of Rockwell International, a major designer and manufacturer of printing presses and associated electronic controls and software for newspapers and commercial products, has designed and implemented an expert system which configures the image plates on large, high-speed presses to print multi-section newspapers.

A printing press is composed of several printing units each of which can print up to 16 pages on the paper web passing through it. There may be 12 or more units in a press. Frequently more than 100 pages, which can include a variety of color usage, and sometimes as many as 160 pages are printed in an edition.

The placement of printing plates on the press in the correct order to produce the newspaper product with the proper section and page sequences, requires knowledge and expertise of a wide spectrum of processes and publisher's operational rules. To accomplish these complex processes, an expert, who often needs years of apprenticeship in the disciplines, is required.

The Press Line-Up Advisor analyzes the product and other user input information. By utilizing the expertise designed into it, a suggested configuration for placement of the printing plates on the press is provided. The graphic line-up produced can be sent to the printing production and layout areas for use in setting up the press and can be transmitted to remote printing sites for use with those presses.

The Los Angeles Times is currently field testing version of the Advisor, which utilized expertise provided by the Times, with good success.

FACTORIES: Factory Simulator

5thGen Technology Inc.
Long Lake, Minnesota

5thGen's FACTORIES is an event driven simulator for modeling discrete-batch manufacturing. FACTORIES is a top-down simulator that generates or accepts customer and forecast orders as input. The system will then schedule factory and purchase orders by running an MRP that supports a modular "bill of material" with any number of levels of components and subassemblies. The MRP has several user selectable scheduling algorithms and features expedite priority, "build to order" and "assemble to order" options.

The software is object oriented and makes extensive use of AI inheritance hierarchy to relate "Parts", factory "Machines", and manufacturing "Process Plan" steps. This technique allows the conservation of a great deal of user input time as well as enabling FACTORIES to make decisions on the factory floor that improve schedule and cost performance. Lots, tooling transporters, standard and actual costs, yields, inventory, setup time, drop rate, and vendor processing, are some simulator features.

Users - manufacturing engineers, industrial engineers, materials control people, and eventually marketing personnel, need no knowledge of Artificial Intelligence, conventional programming, or even a simulation language. Data and operation of the product are maintained in a form that is close to the natural form of the factory. With a moderate amount of experience the FACTORIES simulator will allow the user to track an actual factory fairly closely. Eventually the simulator may take input directly from the corporate Master Production Schedule and or MRP and be capable of indicating improved schedules; reduced part, inventory and operation costs; optimum shift and overtime allocation; and more efficient capital equipment investment to achieve better throughput, quality and costs. The system should be invaluable in refining standard costs and lead times, indeed it will be possible to compute dynamic lead times that reflect the near real time condition and potential of the factory.

5thGen began the product configuration of FACTORIES in the fall of 1986. The principals had worked the previous two years on the application of AI to the manufacturer of fiber optics and electronic connectors. This work was leveraged by the experience of several years of AI applied to electronics CAD at a large midwest computer company.

5thGen's FACTORIES simulator is in a demonstrable form with all basic features in place. The software is undergoing performance testing and tuneup. Documentation and product verification will begin in July. Plans are for a Beta site in August and product release in the fall.

AURORA: Expert Systems

Mystec Associates, Inc.
Mystec, Connecticut

MYSTEC designed and developed an expert system to isolate faults detected in the complex missile test equipment FR/PTS Test Console. The expert system is based on a Xerox LISP machine utilizing LOOPS and LISP. A rapid prototype rule base, a man-machine interface, and an inference engine were developed using a LISP-based processor. Faults were isolated down to a subsystem level, and the prototype system was tested and validated by system experts, designers, and an independent test team. An advanced expert system was developed from the prototype to isolate faults to the component or minor assembly level.

As part of the Small Business Innovative Research (SBIR) program, MYSTEC has completed Phase I of an expert system development project for the Naval Training Systems Center (NTSC) in Orlando, Florida.

This project researched the feasibility of utilizing an expert system to assist DOD personnel in the training task selection process. The study, which was divided into five major tasks, concluded that it is feasible to develop an expert system decision aid for the training task selection process. During Tasks I and II, the task selection process and criteria were examined. In Task III, task selection methods were designed to be incorporated into the decision aid. The decision aid interface mechanisms were designed during Task IV, and a limited task selection prototype was developed and demonstrated during Task V. The prototype system was developed in Golden Common LISP on a IBM PC compatible, transferable to the NTSC Zenith 248 hardware. The prototype successfully showed the limitation of the high level subjectivity found in the task selection process, promoting more efficient analysis techniques. Results also proved that less experienced personnel could use the system to more effectively select tasks for training.

THE AURORA PRODUCT FAMILY

Mystec Associates, Inc. is currently releasing its new AURORA line of low-cost Artificial Intelligence products. AURORA is a state-of-the-art generic inference engine designed for the development of small to large scale customized knowledge-based expert systems. All products in the AURORA product family incorporate the use of advanced menus, multi-windowing, a standard three button mouse, and high resolution animated graphics to significantly reduce the development time required to construct sophisticated user-friendly interfaces. The AURORA expert system line also features extensive uncertainty management capabilities that can be modified by the knowledge engineer, or even by the user operating the expert system software, as desired. AURORA also uses both forward and backward chaining techniques through the use of an intelligent process controller. The expert system's knowledge base is constructed using a modular set of English-like IF-THEN-ELSE statements with user-defined probability factors associated with each rule. Finally, extensive on-line help/data retrieval facilities with a complete "audit trail" of all transactions have been provided to help the user learn the expert's reasoning process and to assist the AI programmer's review of the internal logic during the development and testing phases.

AI AND EDUCATION: An Intelligent Tutor for the Space Domain

Human Resources Lab
Brooks Air Force Base
San Antonio, Texas

In the Intelligent Training Worlds project at Brooks Air Force Base is developing an intelligent tutoring system designed to teach students how to "deduce" a satellite's orbital elements by looking at a graphic display of it's ground track. The system also teaches problem-solving skills by monitoring a student's ability to collect, organize, and understand data.

In order to monitor and plan for satellite missions, an Air Force crew must be able to read and understand ground tracks. Ground tracks are two dimensional displays that show the portion of the earth that a satellite covers in one orbit. The "shape" of the ground track is a direct function of orbital elements, so proper understanding of these variables and of the interactions between these variables is critical for anyone interested in satellite operations.

One way to teach students how to deduce orbital elements from a satellite's ground track is to present the various mathematical formulas that are used to compute the orbital elements. In contrasts to this approach, ID's in-house researchers have provided student with a microworld, or simulated environment, that allows them to manipulate various orbital elements and observe how each of the parameters affects the shape of the ground track. If the student is not making satisfactory progress toward understanding the lesson's goal, then the computer intervenes and offers advice, displays example, or prescribes tasks.

There are several on-line tools that the student can use to explore the ground track world, These include a History tool that allows the student to overlay previously generated ground tracks and note

relationships between parameters; an Orbit Window that displays a two-dimensional representation of the orbit; and a Dictionary that allows students to obtain information concerning terms, etc. The student may select these tools whenever he/she desire and is encouraged to use special tools whenever the Tutor detects a misconception.

As indicated above, the Tutor is organized to provide a range of guidance for the student. The system is designed to keep track of the student's progress toward, a) the mastery of the relationship between orbital parameters and ground tracks, and b) the demonstration of appropriate investigative behaviors. The system intervenes whenever the student is making erroneous predictions or when he/she is not using the tools effectively. A series of "critics" of rules reside in the system to monitor the student's behavior i these two areas.

The initial prototype is now complete and IDI will be testing the tutor with students from the Space School at Lowery AFB and from the Air Force Academy. This data will then be used to improve the diagnostic portion of the Tutor. Project completion is scheduled for Summer 1987.

AVDT: A Knowledge-based Interface to an Aerospace Vehicle Design Tool

Martin Marietta Denver Aerospace
Denver, Colorado

Motivation: Aerospace Vehicle Design Tools (AVDTs) are complete design programs often using a combination of analytic and simulation methods to provide feedback on vehicle performance, cot, and configuration given various input parameters. These tools are typified by their many numeric input parameters and their fragility with respect to the values of these parameters.

AVDTs procedurally embody substantial knowledge about the types and "legal" combinations of inputs to the tool. For this reason, unless one is fairly experienced in the use of such a tool, one can expect to be bewildered by the tool's behavior given inconsistent inputs. What appears to be needed is assistance in supplying the tool with consistent input parameters. Recognition of this need led to the development of a knowledge-based interface to one such AVDT on a Xerox 1186 workstation.

Workstation Selection: The selection of a prototyping environment was mainly driven by a demonstration deadline requirement, not by the availability of particular development environments (our laboratory has a variety of development environments available, including Sun, Symbolics, Xerox, IBM XT/AT, and Macintosh workstations). Given the tight demonstration deadline, we needed an environment with the following characteristics:

- A rich set of menu and icon construction tools
- Availability of an interpretive language for menu and icon display. Such a language would facilitate form and content change to the interface without a time-consuming edit-compile-link development cycle.
- Availability of some sort of "expert system shell" language
- The ability to interact with the "FLY-IT" AVDT

Given these desired characteristics, we selected a Xerox-1186 workstation. The workstation has a powerful set of menu and icon construction tools available within the Interlisp-D language. Interlisp-D code may be run either interpreted, compiled, or mixed interpreted/compiled. Xerox Quintus Prolog is available to serve as both the "expert system shell" and a relational database, and interfaces easily with Interlisp-D code. Xerox Loops can be used to describe the AVDT in an object-oriented manner. The workstation also comes with an optional PC emulation board plus software, allowing programs developed on the PC (such as FLY-IT) to be run on the board under Interlisp-D's control.

In retrospect, the selection was a wise one. We were able to prototype the interface with two people in a little under one month. This was primarily due to the rich set of built-in icon and menu construction/display functions.

Function Allocation: FLY-IT's knowledge base is managed by Loops and Prolog. Loops describes the objects related to FLY-IT, their visual aspects, and the methods invoked for consistency checking. Prolog stores all the facts and rules necessary to provide FLY-IT with a consistent set of input values; facts about how menus should be populated and displayed to the user; and facts and rules related to FLY-IT input file formats.

Interlisp-D and Loops are responsible for interacting with the user. Loops methods control the sequence of menus, icons, and graphs to be displayed, and intercept and filter user utterances. Input parameter values obtained from the user via the menus are then passed to Prolog, where consistency checking is done. If consistency errors are found, Prolog asks Loops to display corrective information and retrieve the corrected values.

Editors' Note: A full description of AVDT will appear in the September issues of *Masterscope*

OP55: On the Xerox Series 1100

Illation, Inc.
Bloomfield, Connecticut

OP55 is the most widely-used language in the production system language family, a family of languages designed to simplify rule-based programming. Production-system computations tend to be different in style from computations performed by programs written in procedural languages, applicative languages, object-oriented languages. The major difference is the production system's use of data-sensitive un-ordered rules rather than sequenced instructions as the basic unit of computation. A production system is appropriate when the knowledge base naturally occurs in rule form, when a program's control is exceedingly complex, or when a program is expected to be modified over a long period of time.

The power of OP55 is most evident when applied to large, ill-structured problems for which it is difficult to provide detailed functional specifications. When the problem solving environment is complex, and when responses to it must be diverse and based on attention to many factors, OP55 is an appropriate choice as a programming language.

OP55 is a general purpose language based on one of the most widely used models of knowledge representation and application, the "production system model." The inference engine in OP55, consisting of the three algorithms for match-rules, select-rules, and execute-rules, directly supports only forward-chaining, with match-rules matching the condition elements against working memory; select-rules choosing one "dominant" rule and execute-rules firing the rule by executing the actions on the right hand side of the rule sequentially.

While the inference engine is a forwardchaining engine, backwardchaining problemsolving strategies can be implemented. OP55 imposes no constraints on the type of application program that can be written. In the past, however, OP55 implementations tended to lack a productive development environment. Particularly lacking were good rule editors that were capable of handling OP55 syntax.

Illation's latest version of its implementation for the Xerox 1100 Series Workstations include the following features:

- Full OP55 syntax is supported, making applications portable to other OP55 environments
- A new rule editor that can check OP55 syntax from within the editor. Multiple edit sessions may be run concurrently.

- A window oriented user interface, integrated with the Xerox AI environment.
- A full set of dynamic debugging aids, including conflict set tracing, display of working memory changes, and a trace of rule firings.
- The ability to access the full features of the Xerox AI environment, including calls to both Common Lisp and Interlisp-D functions.

INNER BUDGET

Georgia Institute of Technology
Atlanta, Georgia

INNER BUDGET has been in prototyping and extension phases since mid 1984. The current version, which operates on the Xerox 1186 within Lyric Common Lisp, engages the PC/DOS emulation window hosting a Lotus 1-2-3 spreadsheet. The BUDGET.WKS template is designed to create a budget and support overhead worksheets for project-level sponsor review and funding of projects for the Georgia Tech Research Institute and the Academic Departments. Distribution of Version 1.1 of the BUDGET.WKS spreadsheet will reach approximately 500 research engineers and scientists at Georgia Tech and perhaps 300 academic faculty by July 1987.

In INNER BUDGET, the Lotus spreadsheet forms the "outer budget" to be interactively analyzed on the 1186 within the PC/DOS emulation window. INNER BUDGET makes use of Spreadsheet Scanner Button, dual forward and backward-chaining inference engines, and a carefully tailored knowledge base. Graphic interpretation of the knowledge base is also available.

The 1186 provides a set of transparent reasoning services to the spreadsheet user, including (1) instant scanning of selected spreadsheet numerics to check them against historic parameters and internal dis-proportionalities; (2) classifying of the project by major types using backward inferencing techniques; (3) providing of narrative explanations of invoked policy-rules from the knowledge base that suggests changes needed to budget line-items (using a forward chaining inference process) and, (4) displaying of fired rules to permit browsing of their structure, content and warrants.

The carefully designed rules within the current knowledge base have been prepared using proven knowledge engineering techniques. Senior accounting and finance staff have been intensively debriefed using hardcopy budgets generated from BUDGET.WKS for real-time proposals. Expertise in interpreting the budgets has now been embedded within the Lyric environment for immediate use as numbers are passed from the surface of the spreadsheet into the inferencing environment (constituting an embedded Expert System).

Additional features of INNER BUDGET environment of the 1186 currently under development include overview windows involving an analysis "onion" of the inner budget structure and Lotus generated graphics extending the spreadsheet analysis with the PC/DOS window itself.

Articles

A VIRTUAL NOTEBOOK FOR BIOMEDICAL RESEARCHERS

G. Anthony Gorry, Ph.D.
Vice President for Information Technology
Baylor College of Medicine
Houston, Texas

INTRODUCTION

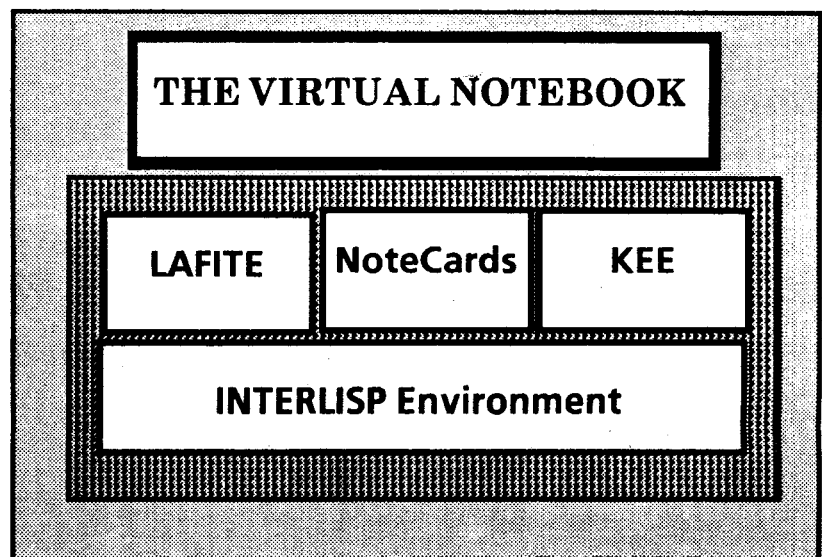
The popular conception of scientific work generally centers on a single scientist engaged in a lonely pursuit of new knowledge. In biomedicine, however, the technical complexities of the research enterprise dictate an entirely different mode of investigation. The typical basic or clinical research undertaking in biomedicine is a group effort, and success hinges on the solution of a variety of task definition and management problems. Members of the group pursue specific tasks to achieve an encompassing objective - the acquisition of some new knowledge in a given scientific domain. The group needs ways to integrate the particular findings or observations of the members, and it needs a way to articulate the new knowledge as it emerges. While there are many differences among them, successful groups create environments in which the generation of ideas is facilitated and in which information gathering and integration can take place efficiently.

Advanced information technology can help biomedical research groups in the information management and task coordination that are so vital in their work. For example, the use of electronic mail in the research setting has generated considerable enthusiasm and many anecdotal testimonials to its value. Generally electronic mail loosens time and space constraints on communications among group members. Despite its virtues, simple electronic mail is inadequate in itself for the varied and complex information exchange activities of a biomedical research group. For instance, electronic mail retains the linear form of conventional written messages, and it therefore limits the extent to which a group of users can easily integrate information.

We are developing a technologic framework, the Virtual Notebook, for improving information sharing and knowledge articulation for biomedical research groups. The principal features of the Virtual Notebook are:

- a mechanism for sharing ideas among members of the group
- automated procedures for importing relevant information into the group from external sources such as libraries
- a procedure for facilitating task assignment and control in the group
- a paradigm for expressing emerging group knowledge in an active form
- an integration of the above functions through the use of a few common representational concepts
- a highly interactive graphical interface presented to the user by the underlying system

Figure 1: Components of the Virtual Notebook



THE ELEMENTS OF THE VIRTUAL NOTEBOOK

To create the Virtual Notebook, we will knit together several commercial products, the NoteCards and Lafite systems from Xerox and the KEE system from IntelliCorp. NoteCards will serve as the underlying data structure for our framework. Lafite is an electronic mail system that we will use to pass information among group members. KEE is an "expert system shell" that we will use in two ways - first as the organizing scheme for emerging group knowledge, and second as the system in which we will develop our highly interactive graphical interface. In addition to the substantial task of integrating these products, we will develop a number of new functions that are important to group communication and information management.

NoteCards is a system for organizing text and graphics in a nonlinear fashion in what is called a hypertext system. The elements of NoteCards are the notecards themselves, data structures for representing information such as notes and sketches much as on conventional paper cards. As needed, these cards can be displayed as windows on the screen of the Lisp workstation. The NoteCards system provides tools for creating cards of different types, for linking cards together, and for organizing cards into so-called file boxes. With these tools, a user can create an interconnected intellectual structure for a body of information. NoteCards also provides special browser cards for graphically displaying the structure of a given set of cards as implied by the links that the user (or the system) has inserted.

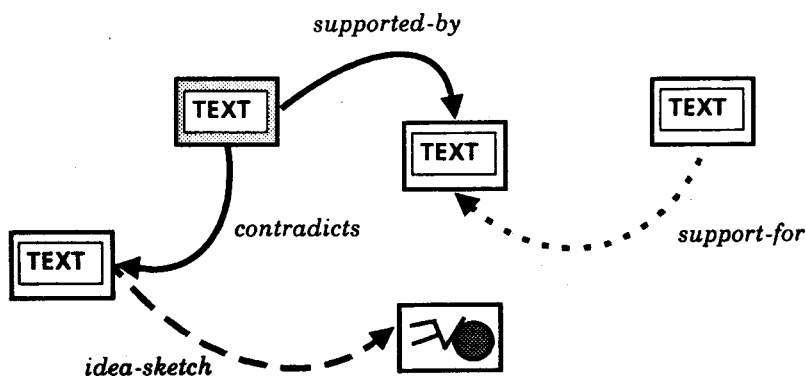


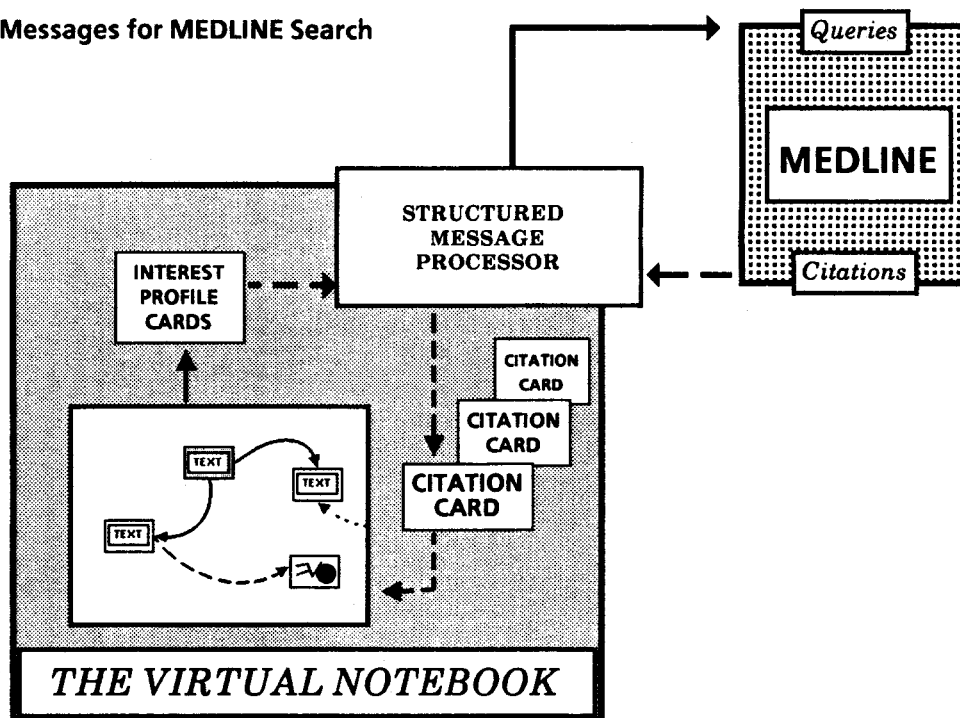
Figure 2: Schematic NoteCards Cards and Links

We will use the basic features of NoteCards in information management and integration in the Virtual Notebook, and we will also blend NoteCards with Lafite and KEE to provide entirely new facilities.

Lafite is an electronic mail package developed by Xerox for use in the Interlisp-D environment. It provides a graphical interface for the preparation of mail and a set of folders to help the user organize incoming mail. We have modified Lafite so that it accepts notecards for mailing and can use NoteCard file boxes as the repositories for incoming mail. The ability to mail notecards (and with some more work on our part, portions of networks of cards) is important, because the members of a research group using the Virtual Notebook will thereby be able to share the information structure for ideas through ordinary electronic mail. That is they will be able to exchange hypertext structures that embody ideas, comments, critiques, requests, etc., rather than linear text alone. As group conversations develop, the framework can maintain the structure of the discussion. A newcomer could employ the browser tool of NoteCards to pursue threads of the previous conversation

To further expand the information-handling capabilities of the Virtual Notebook, we have used KEE in conjunction with Lafite and NoteCards to build a prototype of a processor for semi-structured messages. (See Malone, T.W., et.al., "Semi-structured messages are surprisingly useful for computer-supported communication" in the Proceedings of the Conference on Computer Supported Collaborative Work, December 3-5, 1986, Austin, Texas, pp. 102-114) The basic idea underlying this processor to use a structured vocabulary to code various aspects of a message (sender, subject, degree of urgency, etc.). The recipients of a given message are then determined automatically by a match of the coding of the message with previously filed user interest profiles. These profiles consist of sets of KEE rules which are applied as filters to messages by the Virtual Notebook. Messages that selected by the rule sets for a user are filed automatically in designated file boxes. Thus, in a simple case, a user could indicate an interest in any messages (ideas, seminar announcements, visitor announcements etc.) pertinent to a given subject that might be posted with the system. Take for example the interchange of ideas through semi-structured messages. Suppose that a group member wishes comments on a set of related ideas such as that which might be represented by the schematic in Figure 2. By specifying the key idea(s) using a defined vocabulary, the user could "post" the portion of the network. Other group members who have stated appropriate interests to the Virtual Notebook would automatically receive the "message." A respondent could add new cards or links to the portion of the network, and the growing collection of cards and links would represent a dialogue about the original idea or suggestion.

Figure 3: Semi-Structured Messages for MEDLINE Search



A more elaborate use of semi-structured messages involves the conceptualization of a bibliographic service such as Medline as set of messages. Medline is a computerized data base of citations (titles and abstracts) of biomedical articles. About 30,000 citations per month are added to Medline. Each citation is keyed with terms selected from a structured vocabulary of 18,000 medical terms. To approach the problem of accessing Medline, we can view each citation as a message that is posted in the data base to be "forwarded" automatically to users of the system (and stored in designated file boxes such as "Current Citations") when their specified rule sets indicated a potential interest in the citation. We have already implemented a graphical interface to the MeSH and permuted MeSH vocabularies that permits the user to construct such an interest profile and we are writing programs that generate the KEE rules required to filter the citations and store the selected ones in designated file boxes. Through the Baylor Information Network, we can connect our Lisp machines to the Medline system at the Texas Medical Center Library. The basic technology, therefore, is in place for us to complete this portion of the Virtual Notebook.

LINKING KEE AND NOTECARDS

Some of the information-handling facilities of the Virtual NoteBook will be obtained by blending KEE and NoteCards. These facilities will complement those provided by the integration of NoteCards and Lafite sketched above. In general, the purpose of linking KEE and NoteCards is to provide users of the Virtual Notebook a consistent information management paradigm (NoteCards), but at the same time allow rule-based reasoning and analysis functions (KEE) to manipulate portions of the stored information.

We are working to link NoteCards and KEE for knowledge representation. Using structured notecards as representations of knowledge in a group, we can support knowledge engineering applications of KEE within the context of the group work. We intend to apply the Virtual Notebook to the knowledge management problems in two sites - the Baylor-Methodist Organ Transplantation Center and the Baylor Molecular Biology Information Resource, and in both case, one purpose of the group is to capture new knowledge about an area of biomedicine in an active, computer-based form. In brief, each group hopes to build an "expert system", the first concerning the use of cyclosporine in the management of transplantation patients, and the second involving the determination of the folding of proteins. An integrated information sharing (NoteCards/Lafite) and knowledge engineering (KEE) environment would be a great help to both groups, and that is what the Virtual Notebook is intended to be. In brief, as Figure 3 suggests, we wish to make it possible for KEE to extract rules and facts from NoteCard structures, or alternatively, for NoteCard networks to be derived from KEE knowledge bases.

Both the Transplantation Center and the Molecular Biology Resource are important scientific endeavors at Baylor, and to the extent that our knowledge management framework facilitates the work of these groups, it will be an important contribution to the College. Further, if we demonstrate the value of the Virtual Notebook in a basic research setting and in a clinical research setting, it is very probable that it could be fruitfully used in many other scientific groups in the College. Should the framework prove useful in the Transplantation Center, we will use it as the basis for an advisory system for cyclosporine therapy that will be implemented at another transplantation center outside the Texas Medical Center.

The Beta-Test Process From the Tester's Point of View

Arun Welch
Laboratory for AI Research
Computer Information and Science Department
Ohio State University
2036 Neil Avenue Mall
Columbus, Ohio 43210-1277

Ohio State University was one of the sites chosen to beta-test the Lyric release of Xerox Lisp, and it struck me that people might be interested in what the process is like, from the testee's point.

It all started last October, when we asked to be a beta site. We were asked why we should be chosen as one of the sites. Since the whole idea of beta testing is to have a small group of sites use the software and interact closely with the developers, Xerox needs to screen the applicants to make sure that those chosen:

- a) represent a good cross-section of the user community
- b) will use the software heavily enough to make sure that bugs will be found, and,
- c) are knowledgeable about the system in general.

The Lyric beta-test was divided into two parts, one group of people started in February and worked with the development group in Palo Alto; the other group started in mid-March and worked with the support folks in Pasadena. Essentially, the first group was sort of an external-alpha testers, who were checking out the system in an environment different from Xerox's. The second group tested the system in the form that it would finally be released to customers.

We were part of the first group. In early February, John Sybalsky and Annette Adler from Palo Alto and Thu Le from Pasadena arrived in Columbus with a box full of floppies and documentation. Our first task was to install the Lyric SystemTools on our machines. With that finished John talked to us a little about what the new system was like, what some of the major changes were and what our duties as beta-testers were. Basically, we were to use the system as we normally would and report any problems. Non-fatal problems (MP breaks, non-fatal bugs, things that we just plain thought were screwy) were to be reported immediately, in the form of a bug report, or a phone call to the hotline if the problem was fatal. John, Annette, and Thu stayed around for another day, just to answer any questions that might arise as we started to use the system.

The first thing we noticed was that Lyric is a fairly major change from earlier releases of Xerox Lisp (the name has changed from Interlisp-D to Xerox Lisp or Xerox AI Environment) since it now contains more than just Interlisp-D. While it's still possible to remain in a mostly old-interlisp-D world, there are a whole bunch of new features that make the new environment pretty neat. Spending some time reading the release and implementation notes was one of the first things that we did, and something that anyone converting to Lyric really should do. Some aspects of the system are different enough that if one doesn't check things out first, strange things could happen.

Every two to three weeks we received a floppy or two containing the latest patches. Patches were loaded up as soon as possible so that if we had problems the folks at development knew the problem was in the latest version of the system. In mid-March we received the Beta-2 set of software and then one more sysout at the end of April.

The first systems were kinda buggy (there were days in which I submitted 5-10 bug reports), but with each patch or sysout the number of bugs were reduced. The last one which has been pretty much bug-free (my last couple of attempts at submitting bug reports have been pointed out as errors on my part).

Beta-testing was quite a learning experience for me, since I learned to read disassembled code and some of the techniques involved in using some of the debugging tools like Teleraid. I suspect that it was a learning experience for Xerox too, as they discovered just how different the user's environments can be from theirs.

The testing is over now, and by the time you read this you should have received your copy of Lyric. Motown is now in development, and will probably go into testing early next year. Maybe one of you will be a beta-tester next year now that you know what's involved.

WIZARD'S SHELF

Tips, Tweaks and Miscellaneous Information

This section is devoted to interesting pieces of code, patches to bugs, and requests for the like. It's also for software questions that users want to ask other users. While I've tried to test most of the submitted code, there's no guarantee that it will work on your system. Some of the tips might be unorthodox, or may cause your system to crash.

The "rules" for this section are as follows:

- 1) If you submit software or hardware patches, try not to submit things that are copy-righted, untested, or that you know will cause systems to behave strangely.
- 2) If you would like to answer questions in this issue, then feel free to send the answer to Arun Welch and I will forward it to the requester and include it in the next issue.
- 3) Neither Xerox, nor I, nor the original authors take any responsibility for what these bits of code do to your system (I have to put this in, obviously). The understanding is, however, that people who submit things for this section will make an attempt to debug them, since they would like others to do the same...
- 4) If you have questions you'd like to include in the next issue, send them to me. Or, if you have software that you'd like to make available to other people, send me short description, along with instructions for obtaining it. **Note:** This is not a forum for free advertising, but rather a way of distributing knowledge among users. Distribution is restricted to public domain software (software that you charge distribution costs only, not for software costs). See instructions at the end of this issue on how to submit information.

- ❑ **N Functions from the folks at Ford Aerospace: Dennis Heher, Ron Siemens, Tihamer Toth-Fejel, and Jay Ferguson (now at Carnegie Group)**

Function: **CENTER.BM**

A function for centering bitmaps into a window

```
(CENTER.BM BM WIN CLRFLG)
calls:    CLEARW,BITBLT,IQUOTIENT,IDIFFERENCE,
          WINDOWPROP,BITMAPWIDTH,BITMAPHEIGHT
blinds:   X,Y
globals:  BLACKSHADE
```

```
[LAMBDA (BM WIN CLRFLG)
  (* T.TOTH-FEJEL ''14-Feb-87 00:58'')
  (* * CREATED J.FERGUSON ''30-JUL-85 13:15'')
  (PROG (X Y)
    (IF CLRFLG then (CLEARW WIN))
    SETQ X (IQUOTIENT (IDIFFERENCE (WINDOWPROP WIN
      (QUOTE WIDTH))(BITMAPWIDTH BM))
      2))
    SETQ Y (IQUOTIENT (IDIFFERENCE (WINDOWPROP WIN
      (QUOTE HEIGHT))(BITMAPHEIGHT BM))
      2))
    (BITBLT BM 0 0 WIN X Y NIL NIL (QUOTE INPUT)
      (QUOTE PAINT)
      BLACKSHADE])
```

Function: CENTERGRAPH
Centers graphs into a window

```
(CENTERGRAPH GRAPH WIN LBF MBF)
  calls: WRESET,CLEARW,DSPXOFFSET,DSPYOFFSET,
        DSPCLIPPINGREGION,LET,CENTER.REGION,
        GRAPHREGION,WINDOWPROP,PLUS,
        CREATORREGION,DIFFERENCE,TIMES,SHOWGRAPH
  blinds: REGION,WREGION,BORDER
  fields: LEFT,BOTTOM,WIDTH,HEIGHT
```

```
[LAMBDA (GRAPH WIN LBF MBF)
  (* t.toth-fejel ''14-Feb-87 00:59'')
  (* * CREATED BY j.ferguson ''5-May-86 15:49'')
  (LET (BORDER WREGION REGION)
    (WRESET WIN)
    (CLEARW WIN)
    (SETQ REGION (CENTER.REGION (GRAPHREGION GRAPH)
      WIN))
    (SETQ WREGION (WINDOWPROP WIN (QUOTE REGION)))
    (SETQ BORDER (WINDOWPROP WIN (QUOTE BORDER)))
    (DSPXOFFSET (PLUS (fetch (REGION LEFT)
      of REGION)
      (fetch (REGION LEFT)
      of WREGION)
      BORDER)
      WIN)
    (DSPYOFFSET (PLUS (fetch (REGION BOTTOM)
      of REGION)
      (fetch (REGION BOTTOM)
      of WREGION)
      BORDER)
      WIN)
    (DSPCLIPPINGREGION (CREATEREGION 0 0
      (DIFFERENCE (fetch
      (REGION WIDTH)
      of REGION)
      (TIMES 2 BORDER))
      (DIFFERENCE (fetch
      (REGION HEIGHT)
      of REGION)
      (TIMES 2 BORDER))))
    (WINDOWPROP WIN (QUOTE MUCKED.WITH)
      T)
    (SHOWGRAPH GRAPH WIN LBF MBF])
```

Function: CHANGE.WINDOW.TITLE.FONT
Changes the font of the title of a window

```
(CHANGE.WINDOW.TITLE.FONT NEW.FONT WIN NEW.TITLE GLOBALCHANGEFLAG)
  calls: LET,DSPFONT,FontP,ERRORSET,WINDOWPROP
  binds: OLD.FONT
  uses free: WindowTitleDisplayStream

[LAMBDA (NEW.FONT WIN NEW.TITLE GLOBALCHANGEFLAG)
  (* t.toth-fejel ''14-Feb-87 01:16'')
  (LET(OLD.FONT NEW.FONT)
```

```

        (SETQ OLD.FONT (DSPFONT NIL WindowTitleDisplayStream))
        (SETQ NEW.FONT (DSPFONT (if (FONTP NEW.FONT)
                                     then NEW.FONT elseif [SETQ NEW.FONT (FONTP
                                     then NEW.FONT)
WindowTitleDisplayStream))
(if WIN then [if (NULL NEW.TITLE)
then
    (SETQ NEW.TITLE (WINDOWPROP WIN (QUOTE TITLE)
(WINDOWPROP WIN NEW.TITLE))
(if (NULL GLOBALCHANGEFLAG)
then

```

Function: FFIND.ALL

Given a nested list, recursively finds all atoms that pass a filter

```

(FFIND.ALL LIST FILTER)
  calls: APPLY*,MAPCONC,FFIND.ALL,LIST
  called by: FFIND.ALL
  binds: ELEMENT

```

```

[LAMBDA (LIST FILTER)
  (* t.toth-fejel ''14-Feb-87 01:02'')
  (* * CREATED j.ferguson ''14-May-86 16:36'')
  (for ELEMENT in LIST join (if(LISTP ELEMENT)
    then
      (FFIND.ALL ELEMENT FILTER)
    else
      (if (APPLY* FILTER ELEMENT)
        then
          (LIST ELEMENT]

```

Function: LIST.TO.STRING

Returns the elements of a list as a string

```

(LIST.TO.STRING LIST-TO-CHANGE)
  calls: GNC,GLC,LET,MKSTRING
  binds: NEW-STRING

```

```

[LAMBDA (LIST-TO-CHANGE)
  (* t.toth-fejel ''21-Jan-87 12:40'')
  (* * RETURNS STRING OF ELEMENTS OF LIST, NOT INCLUDING
    PARENTHESES, IE. (QUOTE (A B C))
    -->'A B C')
  (* * CREATED BY D.Heher ''25-Jun-86 17:50'')
  (LET (NEW-STRING)
    (SETQ NEW-STRING (MKSTRING LIST-TO-CHANGE))
    (GNC NEW-STRING)
    (GLC NEW-STRING)
    NEW STRING]

```

Function: BG.BEF

Background function for inspecting various aspects of windows and other such beasts

```

(BG.BEF)
  calls: WINDOWP,SELECTQ,DECODEBUTTONS,CACHED.MENU,INSPECT,
        WHICHW,GETPOSTION,WINDOWPROP,INSPECT/PLIST,RECORDACCESS,
        SYSRECLOOK1,GW,IDD,BKSYSEBUF,CONCAT,MOVEW,CLOSEW,LET,
        printout,PRINTOUT,RESETFORM,OUTPUT,PLAYTUNE
  binds: WIN
  globals: USERBACKGROUNDMENU

```

```

[LAMBDA
  NIL
  (* t.toth-fejel ''13-May-87 10:59'')
  (SELECTQ
    (CAR (DECODEBUTTONS))
    (LEFT
      (SELECTQ
        (CACHED.MENU
          (QUOTE BG.LBM)
          (QUOTE (create MENU ITEMS<--
            (QUOTE (Inspect% Window

```

```

Inspect% UserData
Inspect% Menu
Inspect% DisplayData
Inspect% Stream
    Get% Region
    Move% Window
    Close% Window
Close% Stubborn% Window))
CENTERFLG(--T)))
(Inspect% Window (INSPECT (WHICHW (
GETPOSITION))
    (QUOTE WINDOW)))
    (Inspect% UserData
    (WINDOWPROP
    [INSPECT/PLIST
    (RECORDACCESS (QUOTE USERDATA)
    (WHICHW (GETPOSITION))
    (SYSRECLOOK1
    (QUOTE WINDOW]

    (QUOTE TITLE)
    'USERDATA OF A WINDOW YOU SELECTED'))
(Inspect% Stream
    (INSPECT (WINDOWPROP (GW)
    (QUOTE DSP))
    (QUOTE STREAM)))
(Inspect% DisplayData (IDD))
(Inspect% Menu
    (INSPECT (CAR (WINDOWPROP (GW)
    (QUOTE MENU)))
    (QUOTE MENU)))
(Get% Region
    (BKSYSBUF (CONCAT '(QUOTE'
    (WINDOWPROP
    (GW)
    (QUOTE REGION))
    '))))))
(Move% Window (MOVEW (GW)))
(Close% Window (CLOSEW (GW)))
[Close% Stubborn% Window
    (LET (WIN)
        (SETQ WIN (GW)
            (if (WINDOWP WIN)
                then
                    (WINDOWPROP WIN
                    (QUOTE CLOSEFN)
                    NIL]
                NIL))
    (MIDDLE (if USERBACKGROUNDMENU then
        (CACHED.MENU (QUOTE BG.BBM)
        USERBACKGROUNDMENU)
        else
        (PLAYTUNE (QUOTE((500 . 2000))))
    NIL]

```

Function: ROTATE.BITMAP

Rotates bitmaps, with or without chopping off the corners. **WARNING: This function is SLOW!**
And visual Harmonics can creep in to uglify bitmaps.

```

(ROTATE.BITMAP BITMAP DEGREES NOCHOPFLAG)
calls: EQP, GREATERP, BITBLT, LET, BIPMAPWIDTH, BITMAPHEIGHT, QUOTIENT,
    BITMAPCREATE, BITMAPBIT, PLUS, MINUS, TIMES, SQRT, DIFFERENCE,
    SLANTPRINT, ARCTAN, IPLUS, COS, SIN
called by: SLANTPRINT
binds: WIDTH, HEIGHT, XORIGIN, YORIGIN, SCRATCH.BIT.MAP, ROW, COLUMN,
    MAGNITUDE, ALPHA, SCRATCH.BIT.MAP2

```

```

[LAMBDA
    (BITMAP DEGREES NOCHOPFLAG)
    (* t.toth-fejel '18-Nov-86 15:53')
    (LET
        (WIDTH HEIGHT SCRATCH.BIT.MAP XORIGIN YORIGIN
        MAGNITUDE ALPHA SCRATCH.BIT.MAP2)
        (SETQ WIDTH (BITMAPWIDTH BITMAP))
        (SETQ HEIGHT (BITMAPHEIGHT BITMAP))
        (SETQ XORIGIN (QUOTIENT WIDTH 2.0))
        (SETQ YORIGIN (QUOTIENT HEIGHT 2.0))
        [if (EQP DEGREES -90)
            then

```

```

    (SETQ SCRATCH.BIT.MAP (BITMAPCREATE HEIGHT WIDTH))
    [for COLUMN from 0 to WIDTH do
      (for ROW from 0 to HEIGHT do
        (if (EQ 1 (BITMAPBIT BITMAP COLUMN ROW))
          then
            (BITMAPBIT SCRATCH.BIT.MAP ROW (PLUS
              -1 WIDTH (MINUS COLUMN)))
          1]
    elseif
      (EQ DEGREES 90.0)
    then
      (SETQ SCRATCH.BIT.MAP (BITMAPCREATE HEIGHT WIDTH))
      [for COLUMN from 0 to WIDTH do
        (for ROW from 0 to HEIGHT do
          (if (EQ 1 (BITMAPBIT BITMAP COLUMN ROW))
            then
              (BITMAPBIT SCRATCH.BIT.MAP (PLUS -1
                HEIGHT (MINUS ROW))
                COLUMN 1]
          1]
    else
      (SETQ SCRATCH.BIT.MAP (BITMAPCREATE (TIMES 2 WIDTH)
        (TIMES 2 HEIGHT)))
      (for COLUMN from 0 to WIDTH do
        (for ROW from 0 to HEIGHT do
          (if (EQ 1 (BITMAPBIT BITMAP COLUMN ROW))
            then
              [SETQ MAGNITUDE
                (SQRT (PLUS (TIMES (DIFFERENCE
                  COLUMN XORIGIN (DIFFERENCE COLUMN XORIGIN)
                  (TIMES (DIFFERENCE ROW
                    YORIGIN)
                    (DIFFERENCE ROW YORIGIN)
                    (SETQ ALPHA (SLANTPRINT.ARC TAN
                      (DIFFERENCE ROW YORIGIN) (DIFFERENCE COLUMN XORIGIN)))
                  (BITMAPBIT SCRATCH.BIT.MAP
                    [IPLUS XORIGIN
                      (TIMES MAGNITUDE (COS (PLUS
                        DEGREES ALPHA]
                    [IPLUS YORIGIN
                      (TIMES MAGNITUDE (SIN (PLUS
                        DEGREES ALPHA]
                    1]
              (if NOCHOPFLAG then SCRATCH.BIT.MAP else (SETQ
                SCRATCH.BIT.MAP2
                (BITMAPCREATE WIDTH HEIGHT))
                (BITBLT SCRATCH.BIT.MAP XORIGIN YORIGIN SCRATCH.BIT.MAP2
                NIL NIL WIDTH HEIGHT (QUOTE INPUT)
                (QUOTE REPLACE))
                SCRATCH.BIT.MAP2]

```

Function: SLANTPRINT
Prints text to a window at slants

```

(SLANTPRINT EXPRESSION POSITION CHOSEN.WINDOW DEGREES
FONT WHERE ROTATE.LETTER.FLAG
  calls: BQUOTE, FONTP, GEQ, LESSP, MAPC, CURSOR, BITBLT, MOVETO, PRIN1, LET, EVAL,
          DSPFONT, CONS, DSPXPOSITION, DSPYPOSITION, FONTPROP, TIMES, SIN, COS,
          NCHARS, IDIFFERENCE, SELECTQ, IQUOTIENT, IPLUS, REVERSE, UNPACK,
          ROTATE.BITMAP, GETCHARBITMAP, LIST, PLUS, CREATEPOSITION
  binds: LETTERHEIGHT, XPOS, YPOS, SIN.OF.DEGREES, COS.OF.DEGREES,
          STRINGLENGTH, STRINGWIDTH, LETTER
  globals: WAITINGCURSOR
  fields: XCOORD, YCOORD

```

```

[LAMBDA
  (EXPRESSION POSITION CHOSEN.WINDOW DEGREES FONT WHERE
  ROTATE.LETTER .FLAG)
  (* t.toth-fejel '12-Dec-86 15:44')
  (LET (LETTERHEIGHT STRINGLENGTH STRINGWIDTH XPOS YPOS
    SIN.OF.DEGREES COS.OF.DEGREES)
    (** WHERE CAN BE ABOVE, BELOW, CENTER, RIGHT, OR LEFT
    OF POSITION.DEFAULT IS RIGHT FROM THE CURRENT POSITION)
    (if (FONTP FONT)
      then
        (DSPFONT FONT CHOSEN.WINDOW)
      elseif FONT then (DSPFONT (EVAL FONT)
        CHOSEN.WINDOW)
      else
        (SETQ FONT (DSPFONT NIL CHOSEN.WINDOW)))

```

```

[if (NULL POSITION)
  then
    (SETQ POSITION (CONS (DSPXPOSITION NIL
      CHOSEN.WINDOW)
        (DSPYPOSITION NIL
      CHOSEN.WINDOW]
  (if (NULL DEGREES)
    then
      (SETQ DEGREES 0))
    (SETQ LETTERHEIGHT (FONTPROP FONT (QUOTE ASCENT)))
    (SETQ XPOS (fetch (POSITION XCOORD)
      of POSITION))
    (SETQ YPOS (fetch (POSITION YCOORD)
      of POSITION))
    (SETQ SIN.OF.DEGREES (TIMES 1.3 (SIN DEGREES)))
    (SETQ COS.OF.DEGREES (TIMES 1.3 (COS DEGREES)))
    [if WHERE then (SETQ STRINGLENGTH (TIMES LETTERHEIGHT
      SIN.OF.DEGREES (NCHARS EXPRESSION)))
      (SETQ STRINGWIDTH (TIMES LETTERHEIGHT
      COS.OF.DEGREES (NCHARS EXPRESSION)))
      (SETQ XPOS (IDIFFERENCE (SELECTQ WHERE
        ((ABOVE BELOW CENTER)
        (IDIFFERENCE XPOS (IQUOTIENT
      STRINGWIDTH 2)))
        (LEFT (IDIFFERENCE XPOS
      STRINGWIDTH))
        (IPLUS XPOS (IQUOTIENT
      STRINGWIDTH 2)))
        (IQUOTIENT LETTERHEIGHT 2)))
      (SETQ YPOS (IDIFFERENCE (SELECTQ WHERE
        (ABOVE (IPLUS (IQUOTIENT
      STRINGLENGTH 2) YPOS))
        (BELOW (IDIFFERENCE YPOS
      STRINGLENGTH))
        (IDIFFERENCE YPOS (IQUOTIENT
      STRINGLENGTH 2)))
        (IQUOTIENT LETTERHEIGHT 2]
    [for LETTER in (if (OR (GEQ DEGREES 90)
      (LESSP DEGREES -90))
      then
        (REVERSE (UNPACK EXPRESSION))
      else
        (UNPACK EXPRESSION))
      do
        (if ROTATE.LETTER.FLAG then (CURSOR WAITINGCURSOR)
          (BITBLT (ROTATE.BITMAP (GETCHARBITMAP (EVAL
      (BQUOTE (CHARCODE %, LETTER)))
      FONT)
        DEGREES T)
        NIL NIL CHOSEN.WINDOW XPOS YPOS NIL NIL (QUOTE
      INPUT)
        (QUOTE PAINT))
        (CURSOR T)
      else
        (MOVETO XPOS YPOS CHOSEN.WINDOW))
        (PRIN1 LETTER CHOSEN.WINDOW))
        (* * (STRINGLENGTH LETTER FONT))
        (SETQ XPOS (PLUS XPOS (TIMES LETTERHEIGHT
      COS.OF.DEGREES)))
        (SETQ YPOS (PLUS YPOS (TIMES LETTERHEIGHT SIN.OF.DEGREES]
        (CREATEPOSITION XPOS YPOS]

```

Function: SLANTPRINT.ARCTAN

An ARCTAN2 used by SLANTPRINT and ROTATE.BITMAP that doesn't bomb when xrun = 0 and gives accurate results when YRISE = 0

(SLANTPRINT.ARCTAN YRISE XRUN)
 calls: EQP, GREATERP, LESSP, ARCTAN2
 called by: ROTATE.BITMAP

```

[LAMBDA (YRISE XRUN)
  (*t.toth-fejel '22-sep-86 10:35')
  (if (EQP XRUN 0.0)
    then
      (if (GREATERP YRISE 0.0)
        then 90.0 elseif (LESSP YRISE 0.0)
        then -90.0 else 0.0)
    else
      (ARCTAN2 YRISE XRUN]

```

Function: STRING.TO.LIST
Returns a list from a string of atoms

```
(STRING.TO.LIST STRING)
  calls: SKIPSEPRS, FRPLACD, CLOSEF, LET, OPENSTRINGSTREAM, READ, LIST
  binds: STR, RESULT

  [LAMBDA (STRING)
    (*t.toth-fejel '21-jan-87 12:41')
    (* * TURNS A STRING INTO A LIST, IE. 'A B C'-->(A B
c))
    (*CREATED BY j.ferguson '8-May-86 11:27')
    (LET (STR RESULT)
      (SETQ STR (OPENSTRINGSTREAM STRING (QUOTE
INPUT)))
      (SETQ RESULT (while (SKIPSEPRS STR)
collect
(READ STR)))
      (CLOSEF STR)
      RESULT])
```

Function: TABLEPRINT
Prints on a datastream a list of entries into a table

```
(TABLEPRINT LIST DATASTREAM LEFT RIGHT CENTERFLAG
FORCED.NUMBER.OF.COLUMNS)
  calls: GREATERP, GEQ, DSPRIGHTMARGIN, FRESHLINE, TERPRI,
  CENTERPRINTINLINE, PRIN3, SPACES, LET, IDIFFERENCE, LINELENGTH,
  NCHARS, CONCAT, STRINGWIDTH, DSPFONT, ITIMES, IQUOTIENT, SUB1,
  IPLUS, ADD1, FQUOTIENT
  binds: MAXNCHARS, LONGEST.WORD, WORD, MAXSTRINGLENGTH, FONT, SPACEWIDTH,
  COLUMN.NUMBER, COLUMN.WIDTH, TOTAL.WIDTH, COLUMN, COLUMNLEFT,
  COLUMNRIGHT

  [LAMBDA (LIST DATASTREAM LEFT RIGHT CENTERFLAG
FORCED.NUMBER.OF.COLUMNS)
    (* t.toth-fejel '12-May-87 17:17')
    (* * TURNS THE ENTRIES OF A LIST INTO A NICE TABLE AND
PRINTS THEM IN A DATASTREAM)
    (* * WARNING - USING FORCED.NUMBER.OF.COLUMNS WILL PROBABLY
RESET THE LINELENGTH AND THE RIGHT MARGIN)
    (LET (MAXNCHARS LONGEST.WORD MAXSTRINGLENGTH COLUMN.NUMBER
COLUMN.WIDTH TOTAL.WIDTH FONT SPACEWIDTH)
      (if (NULL LEFT)
        then
          (SETQ LEFT 0))
      (if (NULL RIGHT)
        then
          (SETQ RIGHT (IDIFFERENCE (LINELENGTH NIL
DATASTREAM) LEFT)))
      [SETQ MAXNCHARS (NCHARS (SETQ LONGEST.WORD (CONCAT
(for WORD in LIST largest (NCHARS WORD)
finally
(RETURN $$VAL))
'M')]
      SETQ MAXSTRINGLENGTH (STRINGWIDTH LONGEST.WORD (SETQ
FONT (DSPFONT NIL DATASTREAM)
(SETQ SPACEWIDTH (STRINGWIDTH ' ' FONT))
[if FORCED.NUMBER.OF.COLUMNS then (SETQ COLUMN.NUMBER
FORCED.NUMBER.OF.COLUMNS)
(SETQ COLUMN.WIDTH MAXNCHARS)
(DSPRIGHTMARGIN (ITIMES MAXSTRINGLENGTH COLUMN.NUMBER)
DATASTREAM)
(LINELENGTH (ITIMES COLUMN.NUMBER COLUMN.WIDTH)
DATASTREAM)
else
(SETQ COLUMN.WIDTH (IQUOTIENT (SETQ TOTAL.WIDTH
(IDIFFERENCE RIGHT LEFT))
(SETQ COLUMN.NUMBER (IQUOTIENT
TOTAL.WIDTH MAXNCHARS)
(* * (CREATEPOSITION (DSPXPOSITION NIL DATASTREAM)
(DSPYPOSITION NIL DATASTREAM)))
(bind (COLUMN <-- -1)
(COLUMNLEFT <-- (IDIFFERENCE LEFT
COLUMN.WIDTH))
(COLUMNRIGHT <-- LEFT)
WORD until (NULL LIST))
```

```

first
(FRESHLINE DATASTREAM)
do
(if (GEQ COLUMN (SUB1 COLUMN.NUMBER))
then
(TERPRI DATASTREAM)
(SETQ COLUMN 0)
(SETQ COLUMNLEFT LEFT)
(SETQ COLUMNRIGHT (IPLUS COLUMN.WIDTH
LEFT))
else
(SETQ COLUMN (ADD1 COLUMN))
(SETQ COLUMNLEFT COLUMNRIGHT)
(SETQ COLUMNRIGHT (IPLUS COLUMNRIGHT
COLUMN.WIDTH)))
(if CENTERFLAG then (CENTERPRINTINLINE (pop
LIST) DATASTREAM COLUMNLEFT COLUMNRIGHT T)
else
(PRIN3 (SETQ WORD (pop LIST))
DATASTREAM)
(SPACES (FQUOTIENT (IDIFFERENCE
MAXSTRINGLENGTH (STRINGWIDTH WORD FONT))
SPACEWIDTH)

```

Function: IDD
Inspects DisplayData of a Window

```

(IDD WIN)
calls: GW,INSPECT,RECORDACCESS,WINDOWPROP,SYSRECLOOK1
called by: BG.BEF

```

```

[LAMBDA (WIN)
(* dva '14-Jan-86 16:10')
(if (NULL WIN)
then
(SETQ WIN (GW)))
(INSPECT (RECORDACCESS (QUOTE IMAGEDATA)
(WINDOWPROP WIN (QUOTE DSP))

```

Function: CACHED.MENU
(CACHED.MENU MENU.NAME DEFINITION Window position releasecontrolflg nestedflg)

```

calls: WINDOWP,EQUAL,PUTPROP,WINDOWPROP,CREATE.CACHED.MENU,CONS,COPY,
GETPROP,MENU
called by: BG.BEF
binds: MENU

```

```

[LAMBDA (MENU.NAME DEFINITION Window position
releasecontrolflg nestedflg)
(* j.ferguson '7-Nov-85 08:32')
(PROG (MENU)
[if (WINDOWP Window)
then
[if (NULL (SETQ MENU (WINDOWPROP
Window MENU.NAME)))
then
(SETQ MENU (CREATE.CACHED.MENU
(WINDOWPROP Window MENU.NAME (CONS
MENU (COPY DEFINITION)))
else
(if (NOT (EQUAL (CDR MENU)
DEFINITION))
then
(SETQ MENU (CREATE.CACHED.MENU
(WINDOWPROP Window MENU.NAME (CONS
MENU (COPY DEFINITION)))
else
(SETQ MENU (CAR MENU)
else
(if [NULL (SETQ MENU (GETPROP MENU.NAME (QUOTE
MENU]
then
(SETQ MENU (CREATE.CACHED.MENU

```

```

DEFINITION))
      (PUTPROP MENU.NAME (QUOTE MENU)
       (CONS MENU (COPY DEFINITION)))
    else
      (if (NOT (EQUAL (CDR MENU)
                       DEFINITION))
          then
            (SETQ MENU (CREATE.CACHED.MENU
                        (PUTPROP MENU.NAME (QUOTE MENU)
                                   (CONS MENU (COPY DEFINITION)))
                        else
                        (SETQ MENU (CAR MENU]
              (RETURN (MENU MENU position releasecontrolflg
nestedflg]

```

Function: CREATE.CACHED.MENU

Comments:
 (edited: 18-Apr-86 09:07)
 (CREATE.CACHED.MENU DEF OPTIONS)
 calls: BQUOTE,EVAL,LIST*,LIST
 called by: CACHED.MENU

```

[LAMBDA (DEF OPTIONS)
  (* EDITED: '18-Apr-86 09:07'')
  (if (OR (EQ (CAR DEF)
              (EQ (CAR DEF) (QUOTE CREATE))
              (EQ (CAR DEF) (QUOTE create)))
      then
        (EVAL DEF)
      else
        (EVAL (BQUOTE (create MENU ITEMS<-- (QUOTE %, DEF)
                                CENTERFLG <-- T MENUFONT <--
(QUOTE (CLASSIC 14 BOLD]

```

Function: GW

Puts up crosshairs and waits for you to mouse on a window you want returned. ONE OF OUR MOST-USED FUNCTIONS!

(GW)
 calls: WHICHW,GETPOSITION
 called by: IDD,BG,BEF

```

[LAMBDA NIL (* jcf '2-May-84 14:41'')
  (WHICHW (GETPOSITION)

```

Function: VMEMDISP

Sets up a window that constantly updates current virtual memory size

(VMEMDISP)
 calls: BQUOTE,WINDOWPROP,DSPFONT,LET*,CREATEW,GETBOXREGION,FUNCTION,
 ADD.PROCESS,LIST*,LIST,DSPFILL,CENTERPRINTINREGION,VMEMSIZE
 binds: WINDOW
 uses free: WHITESHADE

```

[LAMBDA NIL (* edited '18-AUG-86 10:04'')
  (LET* (WINDOW)
    (SETQ WINDOW (CREATEW (GETBOXREGION 85 35)
                          (VMEMSIZE)))
    [WINDOWPROP WINDOW (QUOTE CURSORINFN)
      (FUNCTION (LAMBDA (WINDOW)
                  (DSPFILL NIL
WHITESHADE (QUOTE REPLACE) WINDOW)
              (CENTERPRINTINREGION
(VMEMSIZE) NIL WINDOW]
              (DSPFONT (QUOTE (HELVETICA 12 BOLD))
WINDOW)
              (ADD.PROCESS [BQUOTE (WHILE T DO
(CENTERPRINTINREGION (VMEMSIZE)
  NIL %, WINDOW)
              (BLOCK (CONSTANT (ITIMES 1000

```

```
(QUOTE NAME)
(QUOTE VMEMDISP)
(QUOTE RESTARTABLE)
t]
```

Function: WINDOWPROMPT

Sets up a temporary promptwindow for a window and gets rid of it after a response

```
(WINDOWPROMPT WINDOW PROMPT)
  calls: RESETLST, REMOVEPROMPTWINDOW, GETPROMPTWINDOW, RESETSAVE,
        TTYDISPLAYSTREAM, TTYIN
  binds: PWINDOW, TEXT

[LAMBDA (WINDOW PROMPT)
  (j.ferguson '7-Nov-85 12:06')
  (PROG (PWINDOW TEXT)
    [SETQ PWINDOW (GETPROMPTWINDOW WINDOW 1 (QUOTE
      (HELVETICA 12 BOLD]
      (RESETLST (RESETSAVE (TTYDISPLAYSTREAM PWINDOW))
        (SETQ TEXT (TTYIN PROMPT)))
      (REMOVEPROMPTWINDOW WINDOW)
      (RETURN TEXT])
```

□ From Matt Lindenfelser, Oakside Laboratory

If you've ever used the FREEMENU Library package in one of your applications, you probably realize what a convenient and powerful user interface it provides. The following pieces of advice make it a little more versatile by enabling the Skip/Next, Cleat, and Undo keys on the 1186 keyboard for use with EDITITEMS.

All three keys become operative when an item in a FreeMenu is being edited. The Skip/Next key will cause the editing caret to jump to the next EDITITEM in the window (or to the first item if the current item is the last), the Clear key will cause the item to be set to the null string (""), and the Undo key will restore the item to its value before the last Clear, Undo, or edit operation performed on it.

```
(ADVISE (QUOTE FM.FORMATMENU)
  (QUOTE AROUND)
  (QUOTE (LET (W)
    (SETQ W *)
    (WINDOWPROP W (QUOTE FM.EDITITEMS)
      (for X in (WINDOWPROP W (QUOTE FM.ITEMS))
        when (EQ (FM.ITEMPROP X (QUOTE TYPE))
          (QUOTE EDIT))
        collect X))
    W)))

(ADVISE
  (QUOTE (\GETKEY IN \FM.EDIT-ITEM))
  (QUOTE AROUND)
  (QUOTE
    (LET
      (CCC)
      (SELECTQ (SETQ CCC *)
        530 (COND
          ((WINDOWPROP WINDOW (QUOTE FM.EDITITEMS))
            (WINDOWPROP WINDOW (QUOTE FM.NEXTITEM)
              (OR (CADR (MEMBER EDITITEM
                (WINDOWPROP WINDOW
                  (QUOTE FM.EDITITEMS))))
                (CAR (WINDOWPROP WINDOW
                  (QUOTE FM.EDITITEMS))))))
          523 (FM.ITEMPROP EDITITEM (QUOTE SAVED)
            (FM.ITEMPROP EDITITEM (QUOTE LABEL)))
          (FM.CHANGELABEL EDITITEM WINDOW ""))
          516 (FM.CHANGELABEL EDITITEM WINDOW
            (OR (FM.ITEMPROP EDITITEM (QUOTE SAVED)
              (FM.ITEMPROP EDITITEM (QUOTE LABEL)))
              ""))
            (RESTART.PROCESS (THIS.PROCESS)))
        CCC])
```

THE UNIVERSITY SETTING

```
'((SETQ BEFORELOGOUTFORMS NIL)(LOGOUT T))
```

- Also we run in the (VMEMSTATE PURE) mode so that everybody starts with a clean version of the system.
- We also remove the "Hardcopy" option from all window menus (even the background one). People tend to click "Hardcopy" unnecessarily and waste time and paper on the laser printer.

Fun things to put in the Sysout and/or init.lisp

- File Browser
- Dinfo (which implies Tedit and a bunch of others)
- Virtual keyboard (which we use mainly in the French mode, we have even added options to the backgroundmenu to switch easily between standard and French keyboard)
- I never understood why the prompt window is in black; the important messages seem hard to read. So we inverted the prompt window and use a bigfont.
- ?? now gives only the commands (no values, no editcommands, no other messages). ?? is now more like the history command on Unix as far as the output of the command of course.

I hope this information will be useful.

I also take the occasion to express my concern about the licensing of software products on a "per machine" basis. We bought Xerox Quintus Prolog and were told that the price was for one copy. We thought that we had a copy on the file server and could use it anywhere. That is not the case; we can use it on only one machine which means that we can not use it for courses as we do not have money to buy six copies. We returned to using C-Prolog on a Vax, and I find it a shame.

We also used Smalltalk on the 1108 but since Smalltalk and Lisp work in different environments, we have to bring down the machines and reboot to get Smalltalk instead of Lisp. That is very time consuming so we do not have Smalltalk projects going on (but knowing your background, I think you could not care less....).

I sincerely appreciate the opportunity to express these "teaching concerns" about the use of 1108s in a university setting. The basic assumption is that the machines are not supposed to be shared, but we have easily found "modus vivendi" which works well except for occasionally waiting to get the appropriate sysout from the file server when the user before you used a different environment.

NEWS AND ANNOUNCEMENTS FROM XEROX AIS

EDUCATIONAL SERVICES

The 3-1/2 day Conversion Seminar, which is designed to help our users convert to Xerox Common Lisp (XCL), has been very well received. The Seminar covers XCL and the new aspects of the programming environment available in Lyric. Current capabilities of the system regarding code portability i.e., pure

Common Lisp to XCL, pure XCL to the outside world and Koto Interlisp to Lyric Xerox Lisp will be covered. In general the Seminar directs you in the transition process to Xerox Common Lisp.

We also offering a 2-day lecture only version of the Conversion Seminar. The lecture-only Conversion Seminar is given in cities throughout the country or as an on-site class. The workbook for this special class includes a floppy disk with source code for the class exercises and answers. Contact your Xerox AIS Sales Representative or the Xerox Educational Services Coordinator at extension 2676 at the toll free numbers for exact dates and locations of the Conversion Seminar and Conversion Lectures.

Schedule of Xerox AIS Educational Services classes

DATE (week of)	CLASS	LOCATION
21 July	Conversion Seminar	Pasadena, CA
3 August	LISP I: Introduction to Xerox Lisp	Pasadena, CA
10 August	LISP IB: Intermediate Xerox Lisp Programming	Pasadena, CA
17 August	LOOPS I: Programming in Xerox LOOPS	Pasadena, CA
31 August	Conversion Seminar	Pasadena, CA
28 Sept	LISP I: Introduction to Xerox Lisp	Pasadena, CA
5 October	LISP IB: Intermediate Xerox Lisp Programming	Pasadena, CA
2 Nov	LISP I: Introduction to Xerox Lisp	Pasadena, CA
9 Nov	LISP IB: Intermediate Xerox Lisp Programming	Pasadena, CA
16 Nov	LOOPS I: Programming in Xerox LOOPS	Pasadena, CA
7 Dec	Class To Be Determined	

SUBMISSION TO NEWSLETTER

Masterscope is published quarterly by Xerox Artificial Intelligence Systems, 250 North Halstead Street, Pasadena, CA 91109. Opinions expressed herein are those of the authors and do not necessarily reflect Xerox Company Policy.

Masterscope is intended as a communications forum, providing interaction between customers/users of Xerox AI products and between the users and Xerox AIS Management. Please send comments and submissions for future *Masterscope* issue to:

Via network:

Welch@Ohio-State.ARPA
Welch@Ohio-State.CSNet
{ihnp4,cbosgd}!osu-eddie!welch

or:

AINewsletter ↑ .pasa@Xerox

Via US Mail:

Arun Welch
Laboratory for AI Research
Computer Information and Science Department
Ohio State University
2036 Neil Avenue Mall
Columbus, Ohio 43210-1277

MAILING LIST UP-DATE

It's time to up-date the *Masterscope* mailing list. If the name and/or address on your copy are incorrect or if there are others in your company that would like to receive *Masterscope*, please complete the following form and return to:

Paula Burke
Masterscope
Xerox AIS
250 North Halstead Street
P.O. Box 7018
Pasadena, CA. 91109
or
Burke.Pasa@Xerox

Name _____
Company _____
Address _____

Phone # _____
Electronic Address _____
(please indicate full net address)

Xerox ®, Interlisp-D, Xerox Lisp, Xerox LOOPS, and Xerox Quintus Prolog are trademarks of Xerox Corporation
July 1987

Printed in U.S.A.

Masterscope
Xerox Artificial Intelligence Systems
250 North Halstead Street
P.O. Box 7018
Pasadena, CA 91109-7018