THE ROLE OF HUMAN-COMPUTER INTERACTION IN MEDICAL INFORMATION SYSTEMS: PRINCIPLES AND IMPLEMENTATION OF MEDIGATE

BY

JOSEPH WILLIAM YODER

B.S., University of Iowa, 1989

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1992

Urbana, Illinois

Abstract

The solution to many of the problems of the computer-based recording of the medical record has been elusive, largely due to difficulties in efficient capture of those data elements that comprise the records of the Present Illness and of the Physical Findings. Reliable input of data has proven to be more complex than originally envisioned by early work in the field; this has led to more effort into the development of good interfaces.

In early systems, the focus was primarily on the storage and processing of the data rather than on the problems associated with the collection and display of the data and the associated issues of interface design. The characteristics of the user are very important to the development of a good interface system. The capacity of the physician to interact directly with elegant computer-based clinical aids can only be fully realized when the physician also interacts directly with the system for capture of the primary clinical data that s/he generates, which is then electronically available on-line for analysis and decision support.

The MEDIGATE System was developed to study some of the problems in interface design. The design employs an object-oriented approach through the direct manipulation of graphical objects, along with hypertext approaches and semantic networking to build a system that is more natural to the user. The primary design objectives of the MEDIGATE System are to develop and evaluate different interface designs for recording observations from the physical examination in an attempt to overcome some of the deficiencies in this major component of the individual record of health and illness.

Acknowledgments

Most of the early ideas for the MEDIGATE System were developed from early work implemented on the PLATO System. Many people read this thesis or provided input during its development or the development of the MEDIGATE System. Of these people, the following require special mention for their invaluable input: Professor Allan H. Levy, Professor C. L. Liu, Dr. Donald F. Schultz, and Dr. Benjamin T. Williams. I would also like to thank: Professor Steven Bruell, Professor Douglas Jones, and Professor Frank Kosier for their support and encouragement during my undergraduate work at The University of Iowa which motivated me to continue my education; Carl Foote and Rob Schaeffer for their early input on interface design; Stephen Schaeffer for his early design of the graphic caricatures; and Erik Littell for his help with programming and documentation of the *Multiple Selection Popup Menus for the MEDIGATE System*. Finally, I would like to thank my family for never losing faith in me and for providing me with constant support.

This research would not have been possible without the generous donation of equipment, software, and support from University Park Pathology Associates and LifeSpan Research Institute.

Table of Contents

Figures

I. Introduction

Since the inception of medical computing three decades ago there has been extensive discussion of the value of developing an interactive computer-based clinical record system for the practitioner, not only to provide routine decision support for patient care but for the capture of both contemporaneous and longitudinal data important to clinical epidemiology, quality assurance, risk management, and the development of increasing varieties of experiential based reasoning. Though there have been successes in limited areas, the development of complete and comprehensive computer-based record systems has been elusive, largely due to difficulties in efficient capture of those data elements that comprise the records of the Present Illness and of the Physical Findings.

Because of new technology and growing physician facility with and interest in computer-aided medical systems [Dick & Steen, 1991; Ball & Collen, 1992] substantial resources have been dedicated to exploring the feasibility of paperless medical record systems. With the development of sophisticated decision making capabilities, computerized medical data may now be automatically checked for completeness, analyzed for diagnostic support, and assembled into databases important to clinical research. New methods of human-computer interaction - touch screens, voice input, handwritten input, multiple active windows, interactive graphics - now allow the physician to maintain a direct dialog with the computer, moving it beyond the programmer's hands and integrating it as a tool into the physician's environment. But the capacity of the physician to interact directly with elegant computer-based clinical aids can only be fully realized when the physician also interacts directly with the system for capture of the primary clinical data that s/he generates, which is then electronically available on-line for analysis and decision support.

Development of systems for direct physician entry of patient medical data has proven very difficult, as demonstrated by the paucity of commercially available systems that do not rely on the intervention of a data entry clerk; the physician is thus deprived of the opportunity to interact directly with the data s/he has generated and to verify its appropriateness in the context of the record. The major problems lie in the need for physicians to communicate at a professional level with the computer. This suggests a user interface that organizes findings and anticipates the observations that physician intends to record.

Clearly, a medical computer system must embody some notions of clinical context, concepts, and methods of clinical inference and data acquisition. This obvious need had often posed a trap in earlier efforts. Previous attempts to design and build systems in this area have emphasized the diagnostic results of the system and the ability to encourage the physician to be complete and comprehensive. These goals may run counter to the myriad of information processing approaches that the physician has evolved to allow him to efficiently acquire and record information about the patient. Thus, a major goal of the research described here is to develop techniques for construction of a physical examination recording system that will debrief the individual physician as rapidly as possible and in a way that is cognitively satisfying.

Wolfe and Fries [1990] pointed out that the ultimate value of a chronic disease databank depends upon the quality of the data in it. Difficulties in obtaining reliable data are particularly exemplified in the inadequacies of the interface for the input of findings from the physical examination [Williams et al., 1989; Levy & Lawrance, 1991].

The fundamental goal of the MEDIGATE System is to investigate and make progress towards building an interface that is rapid and easy to use, that reduces error, and that

facilitates interactive processing and immediate user feedback, thus minimizing the incompleteness and/or inaccuracy of the data. This should provide the necessary impetus for physicians to use the system by providing useful tools and which will result in the formulation of logical principles for the organization of input. This report will be concerned with the characterization of a usable interface for the input of physical examination findings and will describe its implementation in the MEDIGATE System.

The MEDIGATE System (**M**edical **E**xamination **D**irect **I**conic and **G**raphic **A**ugmented **T**ext **E**ntry System) is a computer enhanced interactive graphic and textual record of the findings from physical examinations designed to provide ease of user input and to support organization and processing of the data characterizing these findings [Williams, Yoder, Schultz, 1990]. The primary design objective of the MEDIGATE System is to develop and evaluate different interface designs for recording observations from the physical examination in an attempt to overcome some of the deficiencies in this major component of the individual record of health and illness.

II. Evolution Of Computer-Based Medical Record Systems

The potential benefits of achieving computer manipulatable databases, even if not complete, substantially outweigh the largely putative present advantages of a rigid and idealized record. Though there has been widespread discontent over traditional paper based medical records, which do not support flexible organization, linkage, variety of manipulation, and ready retrieval of data, there has not been widespread acceptance of the computer alternatives for the acquisition and processing of medical data as originally envisioned. This state of affairs is associated with a number of unresolved problems. Physician interaction with a computer has often been too awkward and time consuming to be feasible. Furthermore, the interface has not provided the freedom to capture the richness of inherently graphic and pictorial findings, particularly germane to essentially topographic domains such as the physical examination, and there is consequent lack of confidence in the results of the processing owing to potential incompleteness and ambiguity of the data.

In these situations, physicians are denied an important opportunity for interaction with the data. Thus, the power of the computer is not realized in areas such as organization, processing, and retrieval of data for the individual patient over time, much less for the exploration of data banks of clinical findings to derive new knowledge, as in the context of clinical epidemiology. Current computer clinical record systems usually rely on paper-based encounter forms for the off-line capture of data from the physician, followed by secondary clerical input to the system by a data entry clerk, depriving the data gatherer the opportunity to assure data quality at the time of capture and the opportunity to retrieve data elsewhere in the record when such data would be helpful to the ongoing observational process.

Furthermore, traditional paper records also have their share of ambiguity. Such problems of ambiguity were less pronounced, or at least less obvious, when an individual practitioner

was responsible for the care of an individual patient over time; the practitioner was familiar with his own habits and customs, and generally recognized that when a finding was not recorded it was that either the observation was not part of his routine, or, conversely, that it was part of his routine and the absence of the finding in the record indicated that it was searched for but was not present. However, with the growth of specialization and group practice, and of medical care review, the lack of specification of such issues emerged as a paramount issue. No viable solution has yet been offered to deal with problems of acquiring unambiguous data and of direct user affirmation of the quality of the data for their reliable transformation into a form for computer processing inference and retrieval. Hence, the goal of current record system design is not merely the duplication of the traditional paper record, but the development of one with greater flexibility for review and cross reference with other parts of the record, and one in which ambiguities of the data are minimized. This latter is vital, of course, to any meaningful analysis of bodies of data on multiple patients for study of the natural history of disease and its therapeutic modification.

There have been substantial resources dedicated over the years to the development and implementation of medical information systems. Early computerized medical information systems were developed with the primary purposes of: qualitative improvement of the services to the patients; obtaining meaningful data to aid in the prognosis and treatment of diseases; and the development of ways for more efficient utilization of the limited resources available [Bakker, 1976; Davis & Simacek, 1976; Evans & Price, 1976; Griesser, 1976; Searle, 1976; van Egmond et al., 1976]. These early systems, although state-of-the-art for their time, still lacked the technology necessary to satisfy the endless demands for comprehensive medical information systems [Atsumi, 1976]. The thrust of many of these early medical computer systems has been to designate a set of findings that specifically

relate to a diagnostic category (inference methods) rather than to design acceptable methods for input of the findings (interface issues).

The value of even a small desktop system was seen in the early seventies with the use of the Leeds system for computer-aided diagnosis of acute abdominal pain. The Leeds system was implemented in UK at the Airedale District General Hospital in 1974. In this system, the diagnostic accuracy of junior staff rose by 10 to 15% in the initial two years of the study. This higher performance level was maintained for ten more years (1976-86) despite changes in staff [McAdam et al., 1990]. There has been continued development of computerized medical systems such as INTERNIST-1 [Pople, 1977; Miller et al., 1982], CADUCEUS [Fischer and Smith, 1990], QMR™ [Miller et al., 1986], Problem-Knowledge Coupler (PKC) [Weed, 1986], Iliad derived from the sequential Bayes HELP™ system [Warner et al., 1972; Hukill et al., 1987; Haug et al., 1987; Kuperman et al., 1991] and DXplain [Barnett and Cimino, 1987; Packer et al., 1988] to assist the physician in collection, diagnoses, and interpretation of medical findings. Most of these systems are largely limited to textual processing of the findings usually ignoring pictorial processing of the findings.

As early as the mid 60's, the processing of pictorial information for use in the medical field offered promise of dramatic and significant contributions to the health sciences [Yoder et al., 1967]. Concepts of the interactive medical record system with graphics were investigated and implemented on the PLATO (**P**rogrammed **L**ogic for **A**utomated **T**eaching **O**perations) system at the University of Illinois in 1974 [Williams et al., 1974, 1975, 1976], using an approach for the input of data from physical findings that has been used in paper form in several contexts and that has since been termed "direct manipulation" in the vocabulary of interactive design [Shneiderman, 1982, 1983, 1987]. By the mid eighties, a shift was seen within the medical community towards a greater use of images and graphics in primary

health care, along with the increased awareness of a need for alternative design of computer systems in terms of mixed initiative dialogs and the human interface [Engelbrecht, 1985; Ikeda et al., 1985; Mandil, 1985; Reichertz, 1985; Solheim & Hansen, 1985; Verplank, 1985]. Over the past few years, techniques such as PLATO's direct manipulation have been deemed useful but with evident need to be enhanced as in the current work being done in the MEDIGATE System [Williams, Yoder, Schultz, 1990].

III. Requirements of Medical Record Systems

The need for medical data in terms of quantity and quality to be used in computerized medical information systems has become even more evident because of such factors as: 1) the interaction among physicians in the now dominant group practice mode, 2) the increasing complexity of quality patient care and its assessment, 3) the increased demands for data by third party payers, 4) the clinical data needs of ancillary medical specialists such as pathologists, radiologists, and anesthesiologists, 5) the data demands of clinical outcome studies, and 6) the embryonic use of decision support systems for assistance to physicians in diagnosis, management, and therapy. Meeting these criteria is contingent on obtaining adequate routine access to reliable data of sufficient scope.

Discrete components of computerized medical record systems for several types of environments of clinical care have been successfully developed over the years. However, several critical domains of potentially worthwhile physician-computer interaction have not been addressed to the satisfaction of the practicing medical community; these include facile input of an account of the present illness, recording of observations from the physical examination, and contextual displays of patterns of laboratory data [Williams et al., 1989].

The importance of accurate acquisition and recording of the history and physical examination has been apparent since the origins of modern medicine in the 19th century. In the development of the Patient Computer Record, no problem has been more challenging than that of computer acquisition of data that physicians record after conversation with and direct observation and examination of the patient. An incentive for better methods of capturing and organizing the history and the physical examination has been provided by the development of electronic data approaches. However, the appearance of electronic methods

of storing and retrieving information has largely served only to highlight the unresolved problems of how to organize basic medical information. [Levy and Lawrance, 1991]. Therefore, one important requirement for computerized medical records is to organize the data in a manner that will be supportive of data acquisition sequences and protocols during the examination of the patient.

Reliable processing of data has made significant strides in the past decade, but the results are directly dependent not only upon the accuracy but also upon the clarity and lack of ambiguity of the input. Problems in achieving accurate input have not been adequately studied. Accurate input is dependent upon minimizing distortion during the transformation of data into a form which can be utilized by a computer. Although a system may be accurate in transformation and processing of data and processing, if the original data record is ambiguous, then for all practical purposes it is useless. Thus, consideration of methodologies that will minimize this ambiguity is critical.

Clinical data are often incomplete, highly subjective, qualitative, non-standardized, and sometimes ambiguous and difficult to verbalize. Contemporary interface designs have often not afforded appropriate tools to encapsulate and organize data of these types [Williams, Yoder, Schultz, 1990]. Thus, it is crucial during the development of any computerized medical record system to design ways to make clinical data more complete, such as having the physician check for additional findings, designating items that were specifically examined versus those not examined, and creating a better representation of the clinical data set by using standardize vocabularies.

Historically, health care databases have been developed for third party claims processing rather than as an aid to clinical care. This lead to the creation of ambiguous databases with

different terms and characteristics for the findings. This problem has been discussed for well over a decade with a number of standardized medical vocabularies considered for use in physicians workstations: the National Library of Medicine's (NLM) Medical Subject Headings (used to index medical literature) [National Library of Medicine, 1984], the National Center for Health Statistics' International Classification of Diseases with Clinical Modifications [United States Health Care Financing Administration, 1991], and the College of American Pathologists' (CAP) Systematized Nomenclature of Pathology (SNOP) [College of American Pathologists, 1965].

While all of these vocabularies have been very successful in terms of their intended purposes, none of them have constructed a good vocabulary for the representation of physical exam findings. CAP attempted to address this problem by expanding SNOP to form the Systematized Nomenclature of Medicine (SNOMED) [Cote, 1984], but it remains inadequate for the representation of physical exam findings. Recently, coding schemes are being developed towards resolving this problem such as work being done by Unified Medical Language System (UMLS) [Chute et al., 1988; Lindberg & Humphreys, 1990; Lindberg & Humphreys, 1992; Huff & Warner, 1990; Sperzel et al., 1990; Tuttle et al., 1988, 1989, 1990], Arden Syntax [Hripcsak et al., 1990; Clayton et al., 1990], and the composite clinical data set reflected in the Wisconsin Ambulatory Review Project (WARP) [WARP, 1991]. WARP has generated an initial listing of terms at the level of physical exam findings which shows promise. It is important to take into account this ongoing work during the development of any computerized medical system. The MEDIGATE System has been sensitive to these coding scheme endeavors, allowing for integration of these databases as they become available as clinical data dictionaries.

Experience in the Nephrology Unit of the Durham Veterans Administration Hospital [Stead and Hammond, 1987] revealed that problem-oriented [Weed 1968; Tufo et al., 1977], time-oriented [Fries, 1972], encounter-oriented, and graphical displays of subjective and physical findings, test results, and therapies must be mixed and matched upon demand. As noted earlier, paper-based records are still primarily used for capturing these data but are confined to representing the data in the manner in which it was recorded. By directly acquiring the medical record in a computer-based format, the data may be entered into the computer once and then individualized and displayed in many different formats upon request.

The initial clinical record has been regarded as composed of the records of the medical history and the physical examination, followed by a statement of diagnostic or differential diagnostic impressions, on the basis of which additional clarifying and/or confirming tests may then be ordered. The medical history itself is further subdivided into the "Chief Complaint," the "Present Illness" (an elaboration of the circumstances and qualifiers of the chief complaint), the "Review of Systems" (a systematic exploration of other complaints that may be associated), the "Past History" of illness, and a "Family History" that includes possible familial or hereditary disorders, and, to varying degrees, a psycho-social history and the like [Williams, 1982]. These components of the clinical record each present specific problems in the design of an interface for the collection of these types of data.

In the early consideration of the use of the computer in medical record systems, it was hoped that the processing power of the computer could be used to suggest diagnostic categories and strategies for sets of medical data. Even so, the utility of diagnostic support has probably been overstated. Much of diagnosis involves pattern recognition followed by the formation of a patient management plan [Williams, 1982]. Specifically, diagnosis helps to bring order to complicated data sets and avoid combinatorial explosion of permutations and

combinations of findings. Once a physician decides on a pattern such as appendicitis, there is rapid progression to a management plan for appendicitis. Either in the physician or in the informatics system, such approaches can be hazardous if they lead the physician to prematurely terminate the acquisition of findings, possibly missing coexisting problems, some of which may be just as important if not more important, than the original diagnosis. Such issues give rise to the consideration of a minimal set of findings to be acquired.

Because of the importance in obtaining a minimal set of findings, the substance and structure of the medical information should be organized around core defaults, and the reasoning structure developed with similar goals.

There has not always been a clear distinction between objective description of physical findings versus the attributes of findings and diagnosis. It is important to distinguish different levels of abstraction in entering data; often data entered are quasi-diagnostic terms and patho-physiologic states rather than raw observations. This contaminates the database and may invalidate inferencing.

There is a need to look at the ability to manage the problems associated with the findings rather than the expression of the findings. Often diseases are hypostatized, creating a disease entity, leading to an invariance in the characteristics of the disease rather than viewing disease as a process of interaction of casual factors and the patient as the host. An alternative approach to the use of computer in the medical field is to: 1) look at the collection of findings, basically ignoring diagnosis; and let the computer look at patterns and see if the computer comes up with what we know or maybe some new patterns, and 2) look at patterns and different types of managements and see what managements work best for a given pattern.

A reliable database along with elegant processing and retrieval provides for long term records of a patient that are consistent and understandable; especially of value when the patient visits many sites of care. Greater detail may be stored and retrieved when needed in following patients. The detail may be varied according to needs, as in epidemiology.

Physicians often acquire much of the medical history during the physical exam. This type of dual data entry is just as important as the recording of findings from the physical exam, but since input of the physical exam alone is so daunting, the focus of this paper will be on recording the findings of the physical exam.

IV. Interface Characteristics

Computer-based systems have been developed over the last two decades in order to help replace pen and paper in the examining room. Examples of such systems can be seen in the works of ARAMIS [Wolfe, Fries, 1990; Fries, McShane, 1986], CompuHX [O'Hagan, 1990], The Medical Gopher [McDonald & Tierney, 1986], A Computer Workstation for Clinical Medicine [Lenhard et al., 1990] and VoiceEM [Kurzweil 1991]. The goals of these systems have been to help with the recording of patient records, assist in diagnosis, and provide for fully legible summaries of findings. One of the major problems with these computerized medical systems is that the physician/user must often adapt significantly to the system rather than the system adapt to the user. The issues that deal with how the human user needs to interact with the computer is commonly referred to as Human-Computer Interaction (HCI) [Baecker & Buxton, 1987].

An important concept of Human-Computer Interaction is to better understand how people work and to examine how to support them before the development of an interface begins. As pointed out by Pressman [1987], in order to provide comprehensive methods for all phases of software design, software engineering paradigms must be developed. This will allow for better tools for automating these methods, more powerful building blocks for software implementation, better techniques for software quality assurance, and an overiding philosophy for coordination, control, and management of software development. These building blocks have been developed over time as software engineering techniques that prompt for the high level design through the development of requirement analysis. They focus more on what the user wants and how the user operates rather than on how the system will be developed. This latter results in constraining the user to the developed system

[Pressman, 1987; Gane & Sarson, 1977]. Systems that focus more on the operational characteristics of the user and how to support user needs are deemed user modeling systems.

Many ideas have been developed over time that focus on techniques for designing the user model [Brajnik et al., 1990; Baecker & Buxton, 1987; Hoeppner et al., 1986; Morik & Rollinger, 1985; Sleeman et al., 1985; and Spark-Jones, 1987]. This research has explored and proposed some possible solutions for expert interfaces that are capable of modeling and remodeling dynamically the users with which it is interacting in an unobtrusive way. The basic idea behind these expert interfaces allows for the development of an "Orthogonal System" that will minimize constraints to the user. In a sense, the user interface is such that the user does not have to operate in parallel with the methodology behind the storage and manipulation algorithms in the system. An "Orthogonal System" is a flexible tool that is sympathetic to the cognitive and manipulative style of the user. An example of a current highly "Orthogonal System" would be an interactive windowing environment that uses menus, color graphics and icons, and flexible input devices as with those provided by X-Windows, the Macintosh® OS, or by Microsoft® Windows 3.0 for use with IBM PCs. These clearly yield less restrictive, highly interactive user-friendly systems in contrast to a more restrictive (less orthogonal) system such as a command line operating system provided by UNIX™ or DOS. From a user's point of view, UNIX™ and DOS consist mainly of commands that are non-mnemonic in nature and unrelated to the function of the command or other commands that leave out characters in order to decrease the number of characters required to execute the command. For example, "cat" lists the contents of a file while "rm" removes or deletes a file. This makes it difficult for many users to master the non-intuitive commands to make even a minimal use of a UNIX™ system.

People are generally adept at visual thinking. Text should be replaced by graphics as much as possible. Good graphic design is an important aspect of user interfaces that allows for a direct manipulation style in computer user interfaces. For direct manipulation, the first design challenge is in inventing a set of objects appropriate for the users' needs and then representing them in a meaningful way that will facilitate intuitive use and minimize learning time.

The computing environment should go to any lengths to provide immediate feedback. People work with problem-domain concepts, while hardware works with different (operator/operand) concepts. Some of the conceptual burden in translating from problem-domain to computer-domain should be carried out by the machine, by making the machine work in concepts closer to those of the user's everyday world. Interfaces developed with these ideas in mind are generally regarded as easier to use and more adaptable to the users' needs [Verplank 1985].

Baecker and Buxton [1987] have examined and outlined many ideas and essential features for the development of a good user interface. They start by enumerating Hansen's [1971] early and insightful principles for the design of interactive graphic systems:

- "Know the user."
- "Minimize memorization," by allowing selection of items rather than entry of data, by using name instead of numbers, by ensuring predictable behavior, and by providing ready access to useful system information.
- "Optimize operations," by providing rapid execution of the most common operations, by changing the display as little as possible in satisfying a request, by exploiting "muscle memory," and by organizing and reorganizing command parameters based on observed system usage.
- "Engineer for errors," through the provision of good error messages, by designing so that common errors are not made, by allowing actions to be reversible, by providing redundancy, and by guaranteeing data structure integrity in the face of hardware or software failure.

They further examine those principles outlined by Foley and Wallace [1974] which look at the methods noted as *language principles* or *psychological principles*, in which interface design can enable effective conversation between human and machine:

> "The language in computer graphics communication is not one of spoken or even written words, but rather one of pictures, and of actions such as button pushes, lightpen indications, and joystick movements, which serve as words."
>
> "The language and context of the conversation must be the language of the man and must be natural to him... However, it is essential also that the language be efficient, complete, and have a natural grammar..."
>
> "The second guiding principle is that the system should avoid psychological blocks that often prevent full user involvement in an interaction. The most typical of these blocks are panic, frustration, confusion, and discomfort..."
>
> "The goals of language design, both action and picture, are to provide a language format which is natural and which does not add to the boredom, panic, frustration, and confusion of the user. Assuming that the required response times can be achieved, that the apparatus and environment have been properly constructed, that the semantics of the intended conversation are understood, and that the suitable tutorial material can be developed, there remains a need for well-designed sequences of input actions coordinated with output pictures to permit symbiotic communication."

Another important example of interface design principles are those enumerated and discussed by Baecker, Buxton and Reeves [Baecker et al., 1979; Baecker, 1980; Baecker & Buxton 1987]:

- The nomenclature used is oriented towards and appropriate for the application.
- The techniques are refined through careful observation of their use by real users, that is, the intended users of the ultimate system.
- Screen layouts are very carefully designed and refined.
- A small but effective set of input transducers is used. Too many can lead to wasted actions; too few can lead to cumbersome interactions.
- The techniques are natural, easy to learn, not cumbersome.
- The feedback given in response to user input is iconic and is appropriate for the task at hand.
- The feedback occurs rapidly. One would use different techniques if the system could not respond instantaneously to user input.
- The feedback occurs predictably. Unpredictable response is even worse than predictably slow response, leading to frustration, tension and anxiety.
- The techniques implemented are powerful, giving the user many degrees of freedom and control.
- The techniques allows the user to focus his attention, avoiding wasted hand and eye movements.
- The proper visual ground is presented.
- It is easy to escape from or abort the action.

- It is difficult to make mistakes, and the system is robust enough to minimize the danger from mistakes that are made.
- As few demands as possible are made on the user's memory.
- The various techniques embedded in the system share a unity of protocol - a common syntax, set of visual conventions, and interactive style. This, along with some of the other characteristics listed above, allows the user to focus on the application, not the communication.
- The nomenclature used is oriented towards and appropriate for the application.
- Finally, the techniques are very device dependent.

Many other lists of these principles can be found in Carroll [1987], Gould and Lewis [1985], Norman [1983], Rubinstein and Hersh [1984], and Shneiderman [1987].

As noted by Williams [1982], the major tasks of the physician are to acquire relevant clinical data, appropriate to the circumstance of care and resources available, to use this data in the light of the physician's medical knowledge to establish diagnoses or a list of problems, to establish a plan for management of these problems, and to monitor the progress of the patient through iterative data acquisition. From this monitoring the physician may alter the management plan and/or change the monitoring process. Thus it is important to construct a clinical set in which one can find important but subtle distinctions. This suggests a tool to drive the technology of obtaining meaningful data.

In the development of the Physical Exam interface, there are a number of important questions to consider:

- Can we enumerate these findings?
- What is the nature of the physical findings that comprises the data?
- What is the method of acquisition by the observer/physician?
- What are the procedures used to acquire the data?
- What is the nature of the observer as an intermediate interface?
- What are the user's needs?
- How do we know what the examiner needs?
- Is the system easy to learn?
- What are the users' idiosyncrasy?
- Can the system be used under different conditions?

Levy and Lawrance [1991] point out that in order to encourage the use of computers in health care settings, there must be incentives and training programs for using these systems. Positive reinforcement and on-line training can be strengthened through a well developed interface that allows for quick data entry and that is simple to use. Another important aspect in the use of the system by the health care professional will be the elimination of the middle man or data entry clerk, giving the physician/user immediate feedback following data input.

Even if 95-99% of medical findings and their related attributes are successfully classified, there would still be a need for free-text entry. This leads to the problem of capturing this free-text in a meaningful format in such a manner that will foster integrity of the database. One alternative is a machine learning approach where queries are done with a thesaurus, thus mapping terms that the user is familiar with into the common language provided by the database.

When the physician has many patients to see, the examination itself must come first while record keeping is secondary. Hence, a methodology for quick recording of the basic findings is very important for the initial input of a physical exam.

The recording of the physical exam exemplifies an area where recent evolutionary changes in computer interfaces are of particular value. Only recently have computer hardware/software systems evolved to the extent that computer processing can be utilized with significant acceptance within the medical field. A crucial aspect of acceptance is a dynamic interactive user interface. Elements in this evolution include multiple re-sizeable windows, improved graphics, menus, color graphics and icons, flexible input devices including voice and handwriting recognition, higher resolution, and object-oriented tools.

These features allow for rapid prototyping of systems through the use of valuable development tools such as those provided by User Interface Management Systems (UIMS) [Myers, 1988] and the like. They also allow for highly interactive direct manipulation (menus, graphics, icons, and text areas) along with a high-level object-oriented development system providing classes and frames which allow for semantic type processing that promote for an interface that is more intuitive to the user. These easy-to-use direct manipulation interfaces are among the most difficult to implement due to: 1) elaborate graphics, 2) multiple ways for giving the same command, 3) a "mode free" interface; where the user can give any command at virtually any time, and 4) semantic feedback inside inner loops of interaction techniques [Myers, 1988].

In current medical-sociological settings with concern for quality assurance and risk management, the medical record has taken on new dimensions; it is no longer a "simple" task of entering existing findings to achieve a "working diagnosis." Rather, significant diagnosis must often be excluded with appropriate documentation, usually with an explicit statement that certain specific findings are not found (the "negative" findings). Current forms of data acquisition within the medical setting often confuse those findings that were not checked with those findings that were checked for but not present. There is often a problem with use of terms such as normal and negative. Saying that the abdominal exam is "negative" or "normal" can create confusion or ambiguity unless default meaning of these terms for that examiner is explicit. Dealing with these types of problems can add significantly to the difficulty and time involved, particularly in the more classic forms of medical record keeping (i.e., pen and paper, dictation), along with cause for data loss and/or distortion. Thus, more initial time spent in obtaining and recording the data may prove beneficial - especially if more initial time spent allows for less overall time to be spent by the physician and/or helps minimize ambiguity. Often important is the provision of prompts

and reminders for the physician. These considerations may be crucial in offsetting the physician's reluctance to abandon the ostensibly "easier" methods of data input (eg. dictation and written notes).

At the core of developing an interface for the recording of findings from the physical exam arise a couple of important questions: Can a practical means be developed to arrive at a meaningful database of physical exam findings which requires no more time and is no more difficult than current techniques while harnessing the computer to transform graphic representations (inherently concise and qualitatively descriptive) into organized and cross referenced sets of comprehensive results of physical exams? This is not likely unless the physician becomes aware of the additional advantages. Can we amplify the utility of the physical exam by providing the physician with current and emerging technologies adapted for use in the medical settings such as voice recognition, rapid response diagnostic tools, and pre-programmed examination criteria, procedures, and protocols?

V. Design Principles of MEDIGATE

The MEDIGATE System[1] was developed to address some of the interface problems noted during the PLATO based research, with current database needs in mind. The primary function is to provide an intuitive user interface that is natural for the physician to use and that may also be employed as a front-end for existing medical record and consultation systems.

Looking to the needs of maximizing the degrees of freedom and allowing for an orthogonal system it can be seen that good graphical design is important with means for a dynamic interactive user interface such as those provided by direct manipulation techniques.

The interface requirements posed challenging problems which led to the decision to incorporate into the MEDIGATE System the following characteristics which satisfied most of the needs for a dynamic interactive user interface: 1) an object-oriented approach, 2) menus that allow multiple ways of giving the same command, 3) an interface where the user can give most commands at virtually any time, 4) multiple windows allowing the user to move around the system with ease, and 5) an elaborate graphical interface that is adaptable to the users' needs.

As noted earlier, there are presently a number of state-of-the-art tools available for direct computer acquisition of some parts of the medical record: portions of the medical history through direct protocol-based and/or branched program interaction with patients, and more or less comprehensive computer generated reports of laboratory data and imaging information. In surveying state-of-the-art clinical information systems, it is apparent that the

[1] The MEDIGATE System was developed under support from University Park Pathology Associates and LifeSpan Research Institute.

issues addressed in the PLATO based research have not been resolved in commercially available systems. In some cases, expert system techniques have been applied to the problem of presenting clinical data once acquired [Parker, 1987]; in most cases, the emphasis has been placed on clinical data capture only as incidental to mandatory billing and reporting activities, and in a few cases there has been work using the direct manipulation methods of the PLATO physical examination input for recording operative procedures [Wheeler and McConnell, 1987] and endoscopy results [Kahane, et al., 1987].

The initial patient record is broadly divided into 1) the record of the medical history, 2) the record of the physical findings, and 3) the record of laboratory results, the latter from tests ordered more selectively after the physician forms an initial "impression" following the initially less selective and more stylized systematic history and physical examination. Following initial basic general data such as vital signs (temperature, pulse rate, respiratory rate, blood pressure) and general behavior impression (alertness, distress, pain and the like), the examiner seeks a medical history according to a "chief complaint" (in a few words - why are you here?) and then elaboration of these current problem(s) in the narrative account of the "Present Illness." Next, the examiner proceeds to a more or less structured "Review of Systems," where minimal key information is both sought and recorded according to the major physiological and functional systems, each of which may involve several regions of the body. Thus, the history involves data structured according either/both to body region (where does it hurt?) and to physiologic systems (cardiovascular, nervous system and the like).

On the other hand, the requisition and recording of observations from the physical examination proceeds more accordingly to regions of the body, each of which may involve findings arising from the several physiologic systems represented - the abdomen, for

example, contains organs of the gastrointestinal, the genitourinary, the hematopoietic (blood forming), and the endocrine systems, and extensions of the cardiovascular, the nervous, and the musculoskeletal systems. The ordering of regions examined is highly variable depending upon individual patient's problems, physicians proclivities, and/or external standardizations such as routine/normal examination form. An invariant sequential approach will not accommodate these needs. This leads to the need for a system design that allows for maximum degrees of freedom.

The approach of the MEDIGATE System involves a departure from earlier efforts in this area. Rather than attempt to use the system as a great equalizer to blindly prompt the physician in a rigid, predefined unalterable sequence for additional information and to provide diagnostic advice, an explicit attempt is made to parallel the physician's current decision making and information gathering style, providing prompts only when relevant to the current context. The basic system for input of the physical findings is similar to graphic pen to paper approaches with which users are more familiar, thus allowing higher degrees of freedom for the capture of relevant data.

The MEDIGATE System presently focuses on aiding the practitioner in developing and maintaining the record of findings from the physical exam. The system utilizes a pre-defined object-oriented approach that allows the physician to graphically and textually describe the findings during an examination by the drawing of appropriate object-oriented graphics or icons representing the observations on a pictorial representation or caricature of an area of the body. The system also allows for free-text entry. Once an instance of a finding has been specified, the user can then indicate attributes of that finding by selecting the appropriate attributes and modifiers from a menu presented conjointly with the graphic.

With different iconic designs, a composite of multiple findings in a given location may be created, thus depicting graphically a state of related findings.

One issue of design and development involves the nature of the background figure in the active window on which the examiner-user "places" the icon indicating a finding. The important issue is the ease and clarity with which the examiner is able to record the findings through the relational landmarks on a topographic sketch. Furthermore, the choice of the optimal figure or caricature may depend on the input device to be used. In the PLATO implementation [Williams et al., 1989], input was by touch using an infrared grid over the interactive graphic display panel - though convenient, resolution was limited and immediate proximity to the examiner, and hence, often, to the subject, was required. Current mouse and trackball input devices along with the advent of pen-based computing not only provide much higher resolution for input but also support greater flexibility in the placement of the input monitor, leading to less intrusion on the attention and curiosity of the patient.

An important tactical design is to minimize the number of explicit frame selections required of the examiner by embedding such selection in pre-determined sequences and/or sets of frames sensitive to the environment of care, the patient, and the practice and custom of the examiner. These factors should determine not only the sequence of examination and input but also the level of detail involved at each stage.

In order to maintain the integrity of the database it is vital that the examiner not be prompted to respond to queries at a level of detail beyond that of his observations; it is only in this way that the system can "know" the level of detail implied by the examiner when s/he uses broad categories such as "within the normal range" or "normal to percussion and auscultation," as well as to be assured that the database thus secured does not contain surplus and/or spurious findings [Williams, Yoder, Schultz, 1990]. Similarly, the examiner must be

able to defer or leave out portions of the examination without generating operational complexity or other user penalty, and must be able to express uncertainty regarding any observation for which input is provided. Finally, the vast majority of observations (somewhat over 95%) [Williams et al., 1976] can be anticipated and properly characterized through appropriate system design. Provision must, of course, be made for input by free text of the other 5%.

Since the majority of observations from portions of the physical examination are normal most of the time, the custom of physicians has been, on paper forms, to simply and rapidly check off "normal" when appropriate for each of the regional or physiologic systems into which the physical examination is divided. The difficulty here is that there is no way for an external observer to know - according to the individual training and custom of the examiner, the environment of care, the problem presented during that clinical encounter, and the like - what elements of the physical examination the particular user has actually explored, and hence, what s/he intends to imply by "normal" or other term of art. In the MEDIGATE System, an essential component of an initial customization is that the user specify at the outset those variables that the user-examiner customarily/routinely means to imply when using default indicators such as "normal." This routine set of findings generates data input and associated text that is considered “normal” for a particular region. Hence, the resulting database (and narrative report) has greater reliability and validity than when assumptions as to the meaning of default indicators must be made by external users of the database.

There are a number of advantages achieved by the use of direct manipulation techniques, by the use of core defaults of routine sets of findings, and by the use of automatic text generation: 1) Entry is easy and expedient, 2) By not relying on user memory omission of findings is minimized, 3) Quality assurance is improved, 4) Risk management is enhanced, and 5) Record keeping for later use is facilitated.

VI. Implementation of MEDIGATE as a Medical Information System

One early difficult implementation detail of the MEDIGATE System was in the choice of the development tool. The development system for MEDIGATE had to allow for quick prototyping and be powerful enough to build a semantic network to represent MEDIGATE's knowledge base. MEDIGATE has two basic modes of operation: 1) a run time mode that allows for the physician to enter findings about a specific patient through direct manipulation techniques, and 2) an editing mode that allows an administrator to customize the system. The development system chosen is an object-oriented tool that provides for quick development of graphical user interfaces (GUI's).

The medical information system described here allows for quick entry of findings by permitting a physician to create a pattern of related findings (Retrograde Maps[2]) so that s/he may quickly map to this specific pattern of findings for a given patient, and then allow for modifications of this pattern to represent the precise findings for the patient of concern. Many other powerful editing features as noted below have been incorporated into the program to allow for customization of the interface to any user's specific preferences (for example what are the routine findings checked for and what sub-divisions does the physician use in a topographic area) along with the ability to update the semantic network.

Development System

In choosing a development tool, it became clear that a language was needed that allowed for quick prototyping, yet was powerful enough to meet the demands of database integrity and that was able to graphically represent physical exam findings and any of their related

[2] To the best of the author's knowledge, the term "Retrograde Mapping" was first coined by Ben T. Williams, M.D.

characteristics. Visual and object-oriented languages along with graphical prototyping tools showed the most promise and several were examined for possible use.

A visual programming language uses graphic representations such as circles, lines, rectangles to specify a computation. Visual programming has proven to be quite useful in prototyping. Recent research has investigated the usefulness of visual programming languages in object-oriented programming [Cox et al., 1989]. In a textual language, a program is arranged as a one-dimensional string, while in a visual language a program is arranged as a multidimensional picture (typically two- or 2.5-dimensional) [Golin et al., 1991].

The object-oriented approach of programming has provided for organization and reusability along with the ability for rapid prototyping. Cox [1986] gives the following outline of objects and their message passing:

> "An object is some private data and a set of operations that can access that data. An object is requested to perform one of its operations by sending it a message telling the object what to do. The receiver responds to the message by first choosing the operation that implements the message name, executing this operation, and then returning control to the caller..."
>
> "Functions and messages differ in several ways. For one thing, functions can have zero or more arguments, while messages always have at least one that identifies the object that is to receive the message. The part of a message that tells the object what to do is called the message selector. It corresponds to the function name determining what will happen, but with one crucial difference. A function name always identifies a single piece of executable code, but a message selector does not, because the code that will be selected also depends on which object was sent the message..."
>
> "The only substantive difference between conventional programming and object-oriented programming is the selection mechanism. This is not to minimize its importance, but to demystify what is basically a very simple notion. Its significance is that it moves a single responsibility across the boundary between the consumer of a service and the supplier of that service. In conventional programming, the consumer is responsible for choosing functions that operate properly on the data managed by that service. The selection mechanism moves that single responsibility off the consumer and onto the supplier. In this small changes lies all the power of the technique."

It makes more sense to examine how a class of objects should work rather than to describe how an individual object should work. This allows for the creation of classes of objects that can use the same code over and over again, rather than redevelop everything from the

ground up every time an individual object is created. Class hierarchy is a simple technique for organizing facts about the universe [Cox, 1986].

Cox [1986] goes on to point out that each instance of objects has two parts: a private part and a shared part. The private part holds the instance's private data fields, and the shared part holds information such as code that this instance shares with other instances of its class. The two parts are linked in a standard way understood by the messenger which uses the link to find tables describing the things the receiver (and all other objets of the receiver's class) knows how to do. Instantiation of objects is created at run-time by using an instance of a generic factory object. Thus, conventional programming tools emphasize the relationship between a programmer and his code, while object-oriented programming emphasizes the relationship between suppliers and consumers of code.

This higher emphasis on the user and the interface as provided by object-oriented approaches and visual prototyping languages led to the design decision of choosing SuperCard™ [Himes and Ragland, 1990] as the primary developing tool. SuperCard™ is a popular hypermedia program for the Macintosh® platform which offers a pre-defined limited object-oriented development system that has a graphical interface and provides the necessary tools for quick prototyping. It allows for the high level design of objects and their relations including hypertext links. It also allows for "Programming by Example" [Myers, 1988, Douglas et. al, 1990]; a method of programming where the developer demonstrates to the system what is to be done and then code is automatically generated to perform the task. The primary tools supplied by SuperCard™ for the object-oriented development of user interfaces are: graphic representation, functionality, spatial orientation - an object's cartesian position in the workspace, and attribute components of an object defining its static characteristics.

An object-oriented graphical interface such as the one provided by SuperCard™ provides for a methodology that supports an incremental, prototyping approach to application development. These systems typically provide the higher-level "classes" that handle the default behavior and the designer provides specifications for these classes that deal with the detailed behavior desired in the user interface [Myers, 1988]. Several graphical objects at a high level of abstraction may be specified along with their interrelationships and then focus can be turned to the internal development of the objects. This developing technique allows for the development of the attributes and methods for an individual object, ignoring the others until later. Thus graphical, highly-interactive direct manipulative interfaces that can provide for semantic network processing can quickly be prototyped.

SuperCard™ capitalizes on its multiple resizable windows, improved graphics, menus, color, flexible input devices, higher resolution, and its object-oriented interface that provide a good environment for prototype development in the domain discussed here. Components of the SuperCard™ interface (windows, buttons, fields of text, graphics, menus) are complete objects characterized by message passing, internal values, and a well-defined hierarchy. Programming scripts, written in SuperCard's own programming language (SuperTalk™), can be associated with any of the aforementioned entities to specify particular actions (for instance, what to do when clicking the mouse button). Following are examples of the primitive mouse gesture recognitions provided by SuperTalk™: 1) on mouse down do action, 2) on mouse up do action, 3) on mouse enter do action, 4) on mouse leave do action, 5) on mouse click do action, 6) on mouse double-click do action, and 7) on mouse still down do action. SuperTalk™ is also Turing-complete, providing all the necessary control structures, such as iterative and branching instructions.

SuperCard™ provides object-oriented programming without the class hierarchy through the use of a prototyped-based object-oriented approach [Douglas et al., 1990]. In a prototyped-based object-oriented approach, interface objects are prototypical objects that can be programmed. A prototypical object-oriented approach provides a single, fully general interface entity, allowing the user to define its attributes and behavior. There is no notion of inheritance or class-based specialization; however, support for the class-based approach is supported for abstraction and reuse by aggregating groups of objects and allowing for easy duplication by copy and paste functions. This basic framework allows the developer to build a functioning interface component quickly and easily.

Normally, in developing algorithms with conventional programming languages, the first considerations are usually done with the program exceptions in mind. This constrains the development of the program to those exceptions. In the object-oriented approach provided by SuperCard™, these exceptions can be ignored until later, supporting the prototyping of a high level interface quickly. Once the high level system design has been agreed upon (what the program should do rather than how), the developers can then turn their efforts to the design of the exceptions. This type of development effort has helped the MEDIGATE System to be designed as more of an "orthogonal" system.

Interface Specifications

The interface for the physical examination in the MEDIGATE System consists of a series of frames depicting the whole body and regions of the body; e.g., head, neck, chest. Each region has an associated set of physical examination findings. These are generated in relation to a primary category - usually in reference to a body system; e.g., abdomen region: gastrointestinal system and posterior chest: pulmonary (lung) system. Significant portions of other systems may be included; e.g., major blood vessels in the abdomen region. On

commencing the examination, the user either selects a given patient from the database of current patients, or creates a record for a new patient. This is followed by the opening of a window that contains the patient's profile (see Figure VI.1). The physician/user will then make any notes and select the physical exam part of the record through the selection of an appropriate icon.

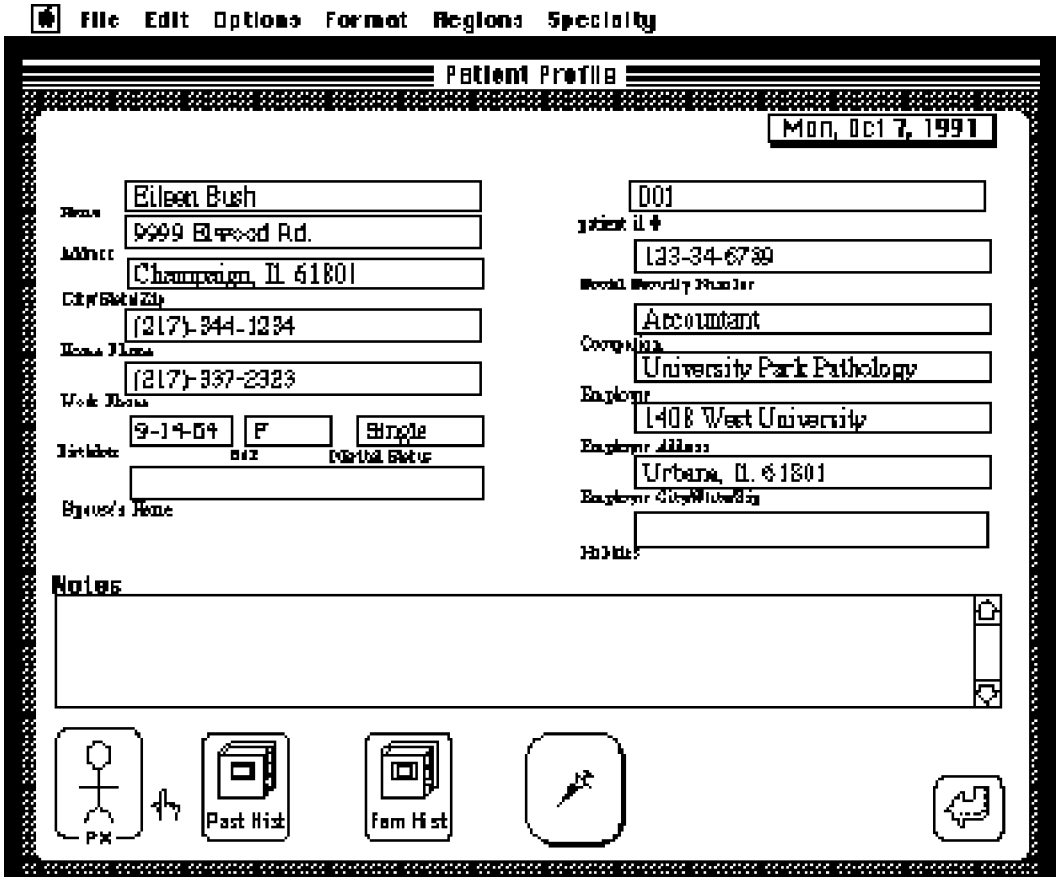


Figure VI.1 -- Patient Profile Window

This will open the starting physical exam window that contains a table of contents of that to which the user has immediate access (see Figure VI.2). As seen in Figure VI.2, all of the body regions except two have an "N" in the box to the immediate left. This designates what the physician routinely checks for and defaults to "N" for "Normal". Further frames or the

regions of immediate concern may be selected at any time by clicking on the appropriate region of the whole body caricature or through a menu.

Figure VI.2 -- Physical Exam Window

The order of frames may follow a sequence tailored to the user's custom, or a sequence that s/he has altered for a specific patient, or a sequence determined by external but pre-defined programs such as those dependent on complaints, patient response forms, questionnaires, site and/or circumstances of the patient encounter, or follow-up protocols, but always with the

option of overriding such pre-defined sequences to call up any desired input frame on demand.

The examiner-user interacts with a graphic caricature, in the active window, of the region of the body that is the subject of attention. The physician may quickly enter the current findings by: 1) duplicating, placing and copying pre-programmed graphics and/or icons on the topographic caricature, 2) drawing in the findings freehand style as they are noted, or by 3) mapping to a pre-defined grouping of findings and modifying the findings to match the specific findings for the patient in question (retrograde maps) [Williams, Yoder, Schultz, 1990].

Thus, the examiner places a given instance of a finding (object) in the desired location. These objects are selected from a hierarchical listing of classes of objects. The specific instance of the finding (object) encapsulates the given frame of knowledge about itself - i.e., each finding "maintains" the attributes that are important to its characterization. Hence, once a given instance of the object has been placed in the desired location, the examiner may select attributes about the object through the knowledge-base of information contained within the object specified. A listing of attributes to be selected is displayed by hierarchical menus and is also used for generation of the narrative notes of the physical findings and then for passage of the finding to the database. Some attributes are also computer generated through graphical placement of the object as in a specific sub-division of a region; e.g., Abdomen: Right Upper Quadrant. This saves user time and reliance on human memory. Upon completion of the exam, the physician will have immediate feedback as to which regions have been checked along with a note of any problem areas denoted by “P” (see Figure VI.3). In this patient “N” is replaced by “P” in the abdomen region.

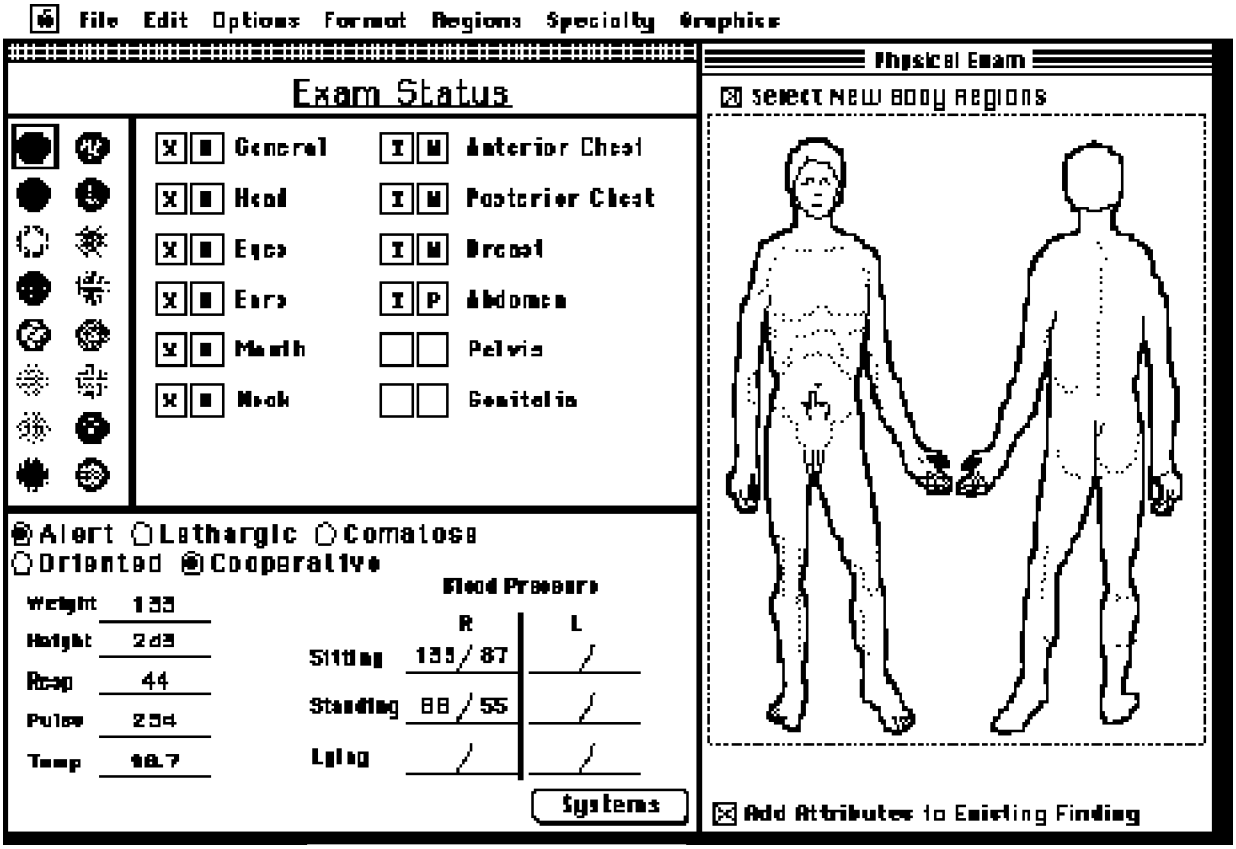


Figure VI.3 -- Physical Exam Window after Exam

As noted earlier, MEDIGATE has the ability to graphically represent the observations/findings through pictorial representations placed upon caricatures of the body. Bitmapped pictures take up a lot of disk storage; therefore, these objects are stored as numerical representations of structures in highly coded forms rather than as bitmapped pictures. These objects also contain basic knowledge about themselves such as their related attributes.

Knowledge Base

Basically, knowledge and its description for use in knowledge-based systems are categorized in one of the three following ways [Jelovsek et al., 1990]:

I. Declarative knowledge: knowledge about objects and their attributes
   A. Object: identification, definition, background, synonyms, abbreviation, relations between objects
   B. Discriminator: attribute, association, manifestation
   C. Category: set or sets to which the object belongs; the set's description--e.g., a group of closely related diseases requiring differential diagnosis
   D. Hierarchy: subset organization among objects or sets
II. Procedural knowledge: condition-action rules along with certainty measures
   A. Singular: isolated condition-action rule
   B. Sequential: a prescriptive recipe--e.g., a linear set of steps for performing a procedure
   C. Conditional: choice plus chance events--e.g., a decision tree for workup of a chief complaint
III. Causal knowledge: knowledge of the cause of an event
   A. Singular: isolated cause-and-effect rule
   B. Sequential
      1. Open loop: chain of causal events
      2. Closed loop: feedback loop (positive, negative)
      3. Model: a collection of sequential rule loops linked to each other in a network

The MEDIGATE database structure was generated by Donald F. Schultz, M.D., based upon early work on PLATO along with the categorization of medical findings as found in clinical texts [Bailey, 1949; Bauer, 1967; Buckingham, 1971; DeGowin, 1987; Berkow & Fletcher, 1987; Judge & Zuidema, 1968; Kampmeier, 1950; MacBryde & Blacklow, 1970; Seidel et al., 1987; Taylor, 1973].

All representations must provide some way to denote objects and to describe the relations between them. MEDIGATE's knowledge-base primarily uses declarative knowledge which has been developed through the use of a semantic network [Nilsson, 1980; Winston, 1984]

since this provides a powerful mechanism to denote objects and their respective relationships.

Semantic Network

As noted by Winston [1984], a good representation is often the key to turning hard problems into simple ones. In the simplest of terms, a semantic network provides a consistent categorization of all meanings represented in a concept through the use of semantic types and the relations between them. The semantic types are represented as nodes in the network, while the relations between them are represented by the links. The following are the definitions of some terms pointed out by Winston [1984]:

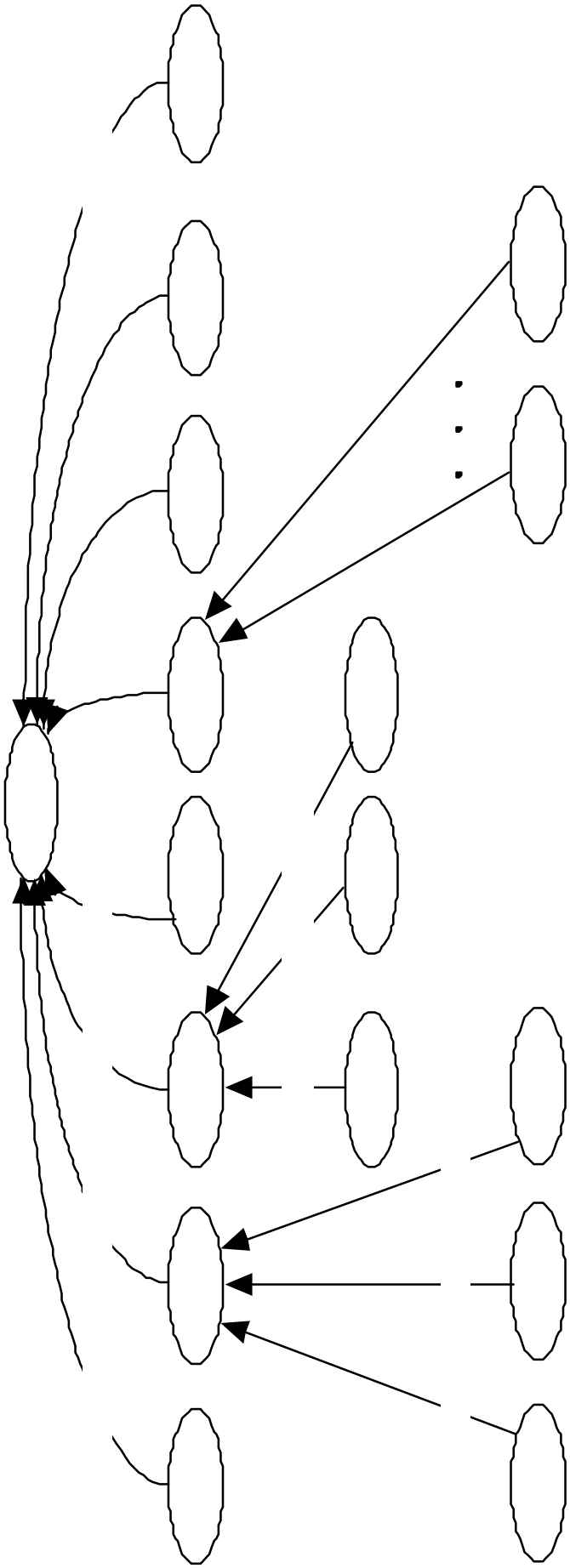
- A *representation* is a set of syntactic and semantic conventions that make it possible to describe things.
- The *syntax* of a representation specifies the symbols that may be used and the ways those symbols may be arranged.
- The *semantics* of a representation specifies how meaning is embodied in the symbols and the symbol arrangements allowed by the syntax.

Winston [1984] goes on to note that links make class-based inheritance opportunities explicit, thus facilitating the work of class-oriented inference procedures. By making the right things explicit, these links satisfy an important criterion for good representation. The following is a summary of links used in this program and their functions, as described by Winston [1984]:

- IS-A and AKO (a kind of) links make class membership and subclass-class relations explicit, facilitating the movement of knowledge from one level to another.
- VALUE facets make values explicit.
- IF-NEEDED facets make procedures purposes explicit, and they relate procedures to the classes those procedures are relevant to.
- Default facets make likely values explicit, without implying certainty.
- Perspectives make context sensitivity explicit, preventing confusion and ambiguity.

The MEDIGATE System maintains knowledge about objects/findings such as *Mass* and its related attributes through the use of a semantic network (see Figure VI.4). The purpose of the semantic network in the MEDIGATE System is to provide a categorization of the set of findings within the medical database and to provide a set of relationships between these concepts. The semantic types are the nodes in the network and the relations between them are the links. The primary links used within the MEDIGATE Semantic Network are the "IS-A" links, "AKO" links, and the "Property Of" links. The "Property Of" links are a form of value facets noted above. These links establish the hierarchy of types within the network. The relations stated between types may not always apply as some specific instantiation of a set type of an object may have some attributes that are <u>not</u> applicable.

An object such as *Mass* can have many attributes associated with it that give meaning to any specific instances of a *Mass* that may be created. The relationships between objects and their attributes are maintained by the semantic network through the noted links. As noted in Figure VI.4, *Mass* has properties such as *Mobility*, *Tenderness*, *Size*, and *Location* associated with it, while *Organs*, *Abcesses*, and *Neoplasms* are a kind of *Mass* that can inherit any attributes associated with *Masses* in general. Each property can also have specific attributes linked to them so as to better define them such as *Diameter* for *Size*.



One example of the system in action is the creation of a specific instance of a finding which is a mass interpreted by the physician as an enlarged organ. Up to then, *Organ*, which is a kind of (AKO) *Mass*, is just an abstract finding that has possible attributes associated with it. In other words, *Organ* is a sub-class of *Mass*, which in turn is a class of objects that can be instantiated at run-time. After describing the general shape and location of a specific instance of the mass, the user can then associate other attributes with it such as consistency, tenderness, depth. Figure VI.5 shows a semantic net for a specific instance of an *Organ;* MEDIGATE internally names the organ as "Finding12". The net in Figure VI.5 ascribes meaning to "Finding12" through *IS-A*, *Type of Organ*, *Shape, Tenderness*, and *Location* links.

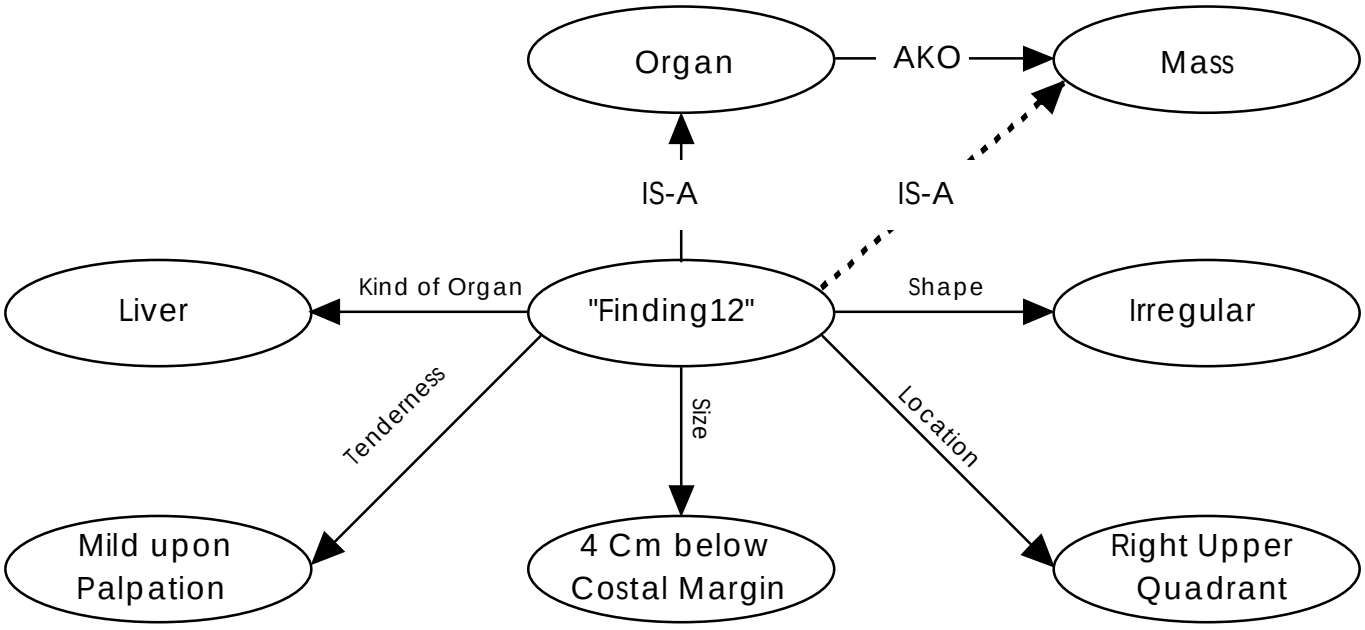


Figure VI.5 -- Specific Instance of "Finding 12"

Basically, *Organ* is an abstract entity with attributes associated with it and can inherit specific attributes associated with *Mass* such as Location, Shape, and Tenderness that

describes the specific makeup of any instantiation of a *Mass*. Therefore, when any specific instance of a *Organ* is created, it inherits all of the possible attributes that are associated with *Masses* in general. Any default attributes associated with a class of findings are automatically inherited upon instantiation of that finding.

Note that in Figure VI.5 there is a dotted line going from "Finding12" to *Mass* with an *IS-A* link. If the physician did not want to commit to the mass being an *Organ*, then the link from "Finding12" to *Organ* would be replaced by the dotted line link from "Finding12" to *Mass*, thus eliminating the *Organ* node completely.

Upon examination of the abdomen the physician will select the abdominal region (see Figure VI.6). This defaults to what the physician routinely checks for and finds as "normal" findings. As seen in figure VI.7 the physician routinely checks for scars, tenderness, point pain, masses, and guarding. If none of these are present, then they are designated normal (i.e., the "N" in the box directly left to the finding). If the patient has "Normal" findings within the abdomen, the physician simply clicks on the box to the left of the "N", generating an "X" to designate by an intentional affirmation that this patient has had all of these findings checked and they were normal. While most systems allow for the description of findings not checked to be listed as normal without any positive affirmation, thus possibly allowing for ambiguities, MEDIGATE forces the examiner to designate what has and what has not been checked. This constraint should minimize ambiguities and provide for a more complete database.

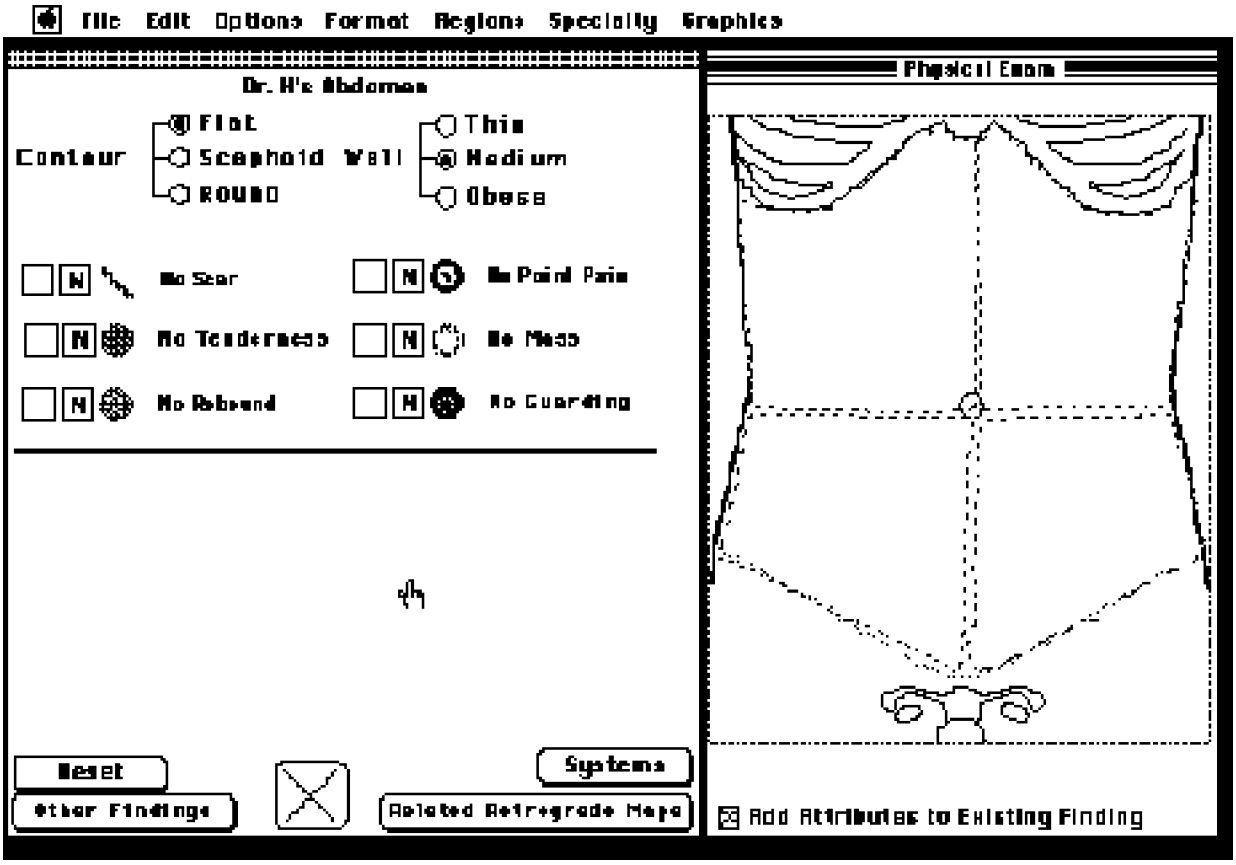


Figure VI.6 -- Abdominal Frame of Physical Exam

If there are abnormalities during the examination, a different procedure would follow. For example, during the examination, the physician notes the following findings: 1) Flat Contour, 2) Medium Wall, 3) Point Pain-Located Within The Right Lower Quadrant and Right Upper Quadrant, 4) Moderate Guarding-Located Within The Right Lower Quadrant, 5) Marked Tenderness-Located Within The Right Lower Quadrant and Left Lower Quadrant, 6) No Scar, 7) No Mass, 8) Bruit-Located in the midline area that is possibly an aneurysm and 9) Rash-Located Within the Right Upper Quadrant. The findings that are immediately visible through the finding window can be entered through the selection of the appropriate findings and then modifying each finding from a menu of possible attributes. These findings can either be drawn in free-hand or by selecting an icon from a palette of pre-defined shapes and reproducing that shape on the topographic caricature. Access to

other findings can be achieved through a couple of different means; either through the selection of “Other Findings” or through the selection of findings based upon “Systems”. Once an abnormality is noted, other abdomen findings are immediately displayed as noted in figure VI.7. Here the physician is able to select Bruit and free-hand draw a diagram showing its location and shape. However, rash is not available as an immediate finding within the abdomen frame. To enter this finding, the examiner can simply open the skin or dermatological findings window by selecting it from the “Systems” menu and enter the rash finding along with its attributes as noted in Figure VI.8.

After entering the attributes for a specific finding, text designating the appropriate attributes of that finding such as “Tenderness” is generated and displayed in the *Patient Report* Window (see Figure VI.7). Note that in this example, “Tenderness” has the following attributes associated with it: Location Within the Right Lower and Left Lower Quadrants and Marked. The locations are automatically generated by the previously defined sub-divisions (noted by dotted lines) while the “Marked” attribute was selected from a pop-up menu similarly to the pop-up menu noted in Figure VI.11. The text for these attributes is generated by an expert algorithm designed specifically for the MEDIGATE System (see Appendix 1 -- MEDIGATE’S Selection Popup Menus). The text generation algorithm can be customized to accommodate an area of practice or the examiner’s preferences. A crucial concern is the ongoing feedback to the physician provided by the developing textual report.

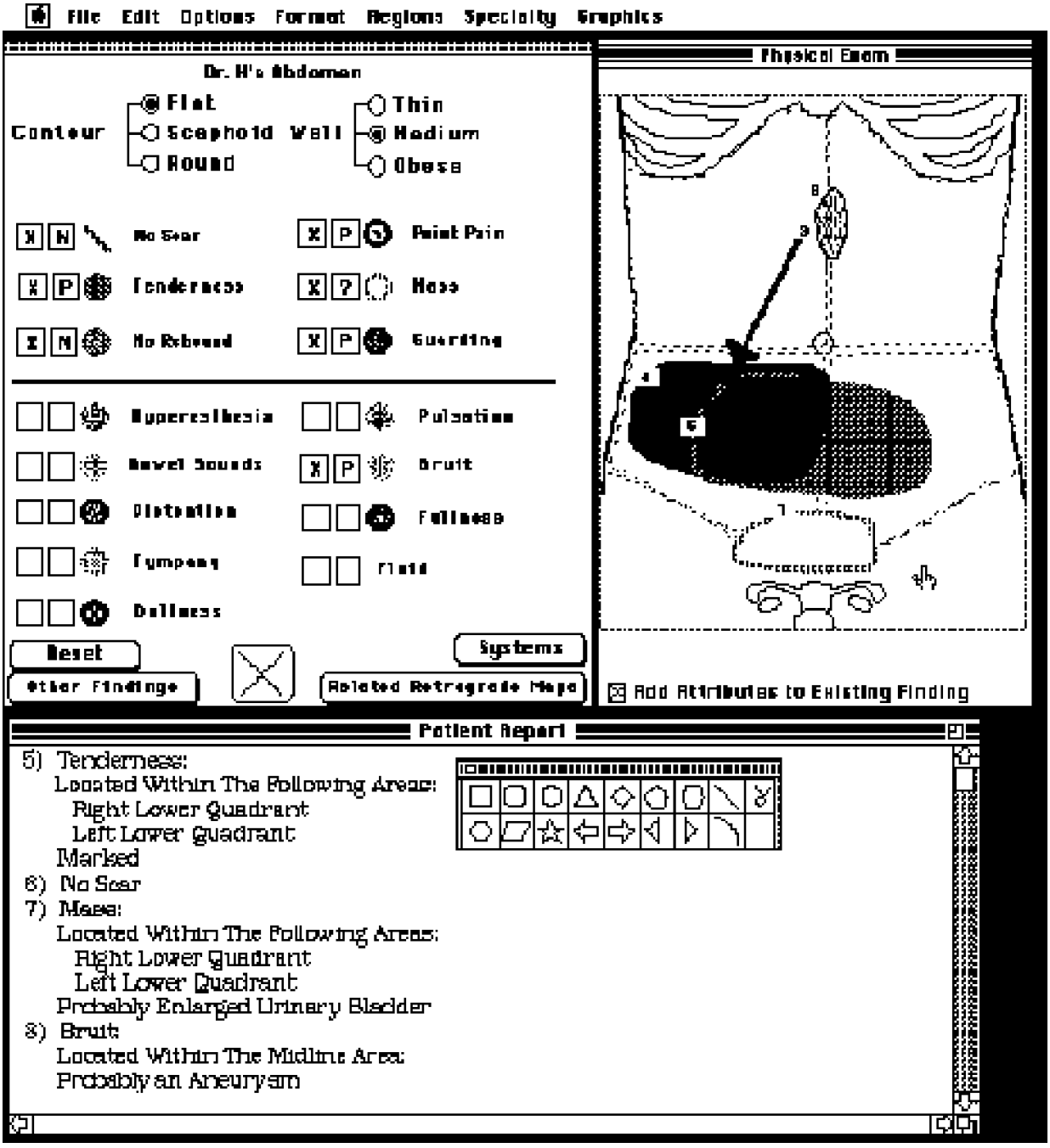


Figure VI.7 -- Other Findings of Abdominal Frame

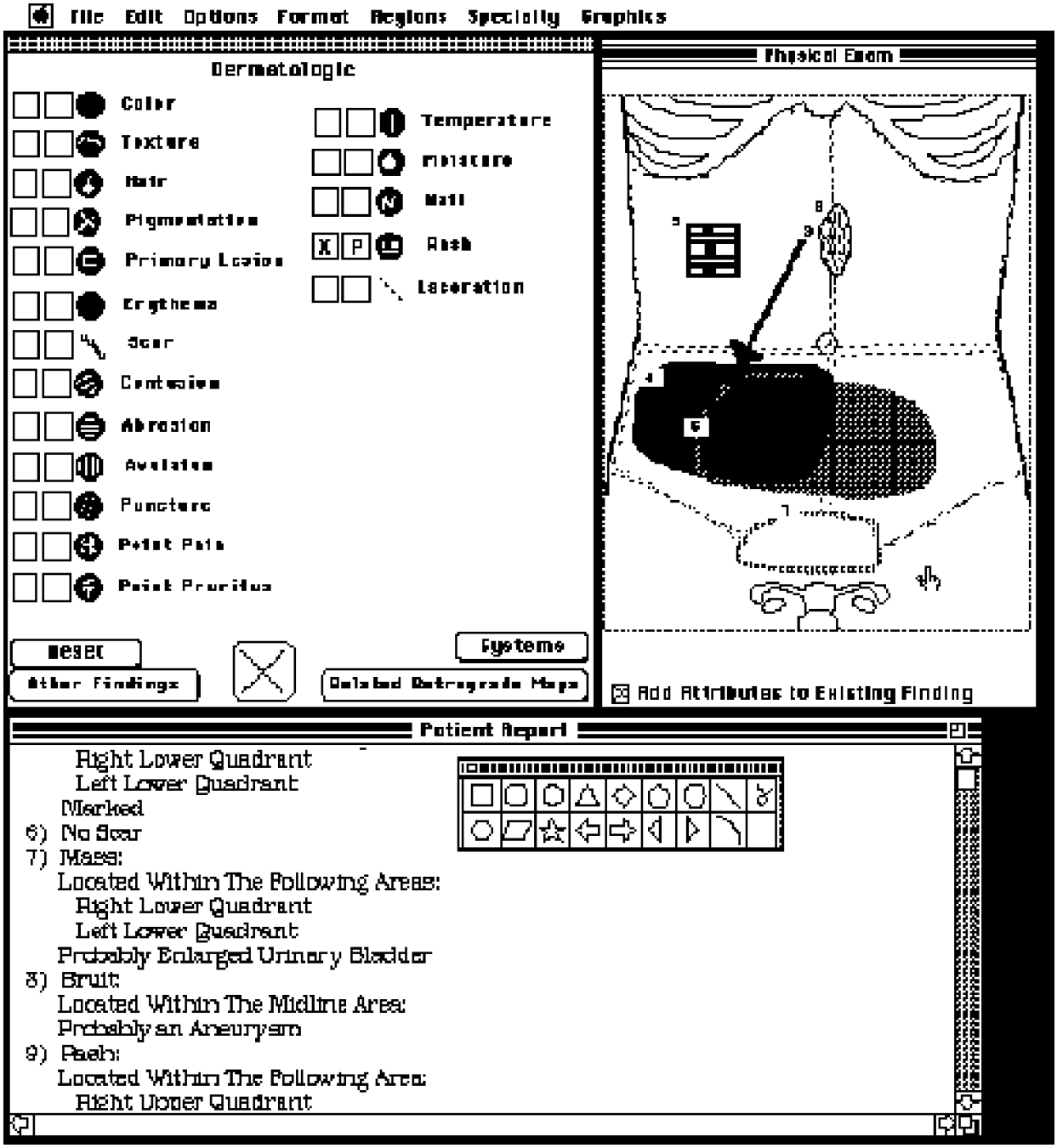


Figure VI.8 -- Dermatologic System Window

Retrograde Maps

Retrograde Maps (RMaps) were suggested by Ben Williams [1982] in the mid 70's as a method to allow the physician to select a pre-defined grouping of findings, thus enabling for the quick entry of the findings. As stated by Williams [1982], "The notion of retrograde mapping suggests that the choice of management plans may be mapped directly back to the primary data elements without an intermediate diagnostic sub-goal, or that within diagnostic classes no subclassification other than the associated data elements specific to the individual case need be attempted."

The idea of RMaps can be further expanded to that of a frame. A frame is a collection of semantic net nodes and slots that together describe a stereotyped object, act, or event [Winston, 1984]. MEDIGATE allows for the development of a frame of a collection of findings. These frames are the primary data elements that can be mapped directly back to depending upon the choice of the management plan. The RMaps within the MEDIGATE System are frames that consist of collections of findings that the physician can quickly map to and modify in order to match the specific findings for the patient in question.

An example of a RMap frame for appendicitis that could be created is noted in Figure VI.9. Any report about appendicitis will supply information about *Tenderness*, *Guarding*, *Point Pain*, and some other possible default attributes such as *Contour*, *Wall*, *Mass*, and *Scar*. In Figure VI.9 there are default slots to fill in that have slot-filling procedures attached.

Retrograde mapping could have been used for the appendicitis diagnosis noted before. As seen in figure VI.10, a retrograde map for appendicitis is selected from pre-defined RMaps that were created with the editing capabilities provided by MEDIGATE (see below). This pre-defined frame is very close to the grouping of findings that were selected before. With a

few minor modifications (add a mass, bruit, and rash along with their specific attributes, resize and modify the attributes of tenderness, and delete rebound) the desired frame of findings can be developed for the patient (see Figure VI.11). Notice that attributes related to a particular finding are displayed conjointly in a hierarchical menu (see Figure VI.11). In this instance, the attribute “Marked” is selected to modify the finding “Tenderness”.

This retrograde approach is feasible when the circumstances are such that in the majority of cases, the physician rapidly reaches a clear categorical impression that may then be used to generate the prototypic description that is implied in reaching that conclusion [Pauker et al., 1976]. This may be a tentative diagnostic impression and the retrograde map serves as a reminder of findings usually associated with a disease process, as well as a reminder of key findings that may differentiate other disorders with a similar presentation.

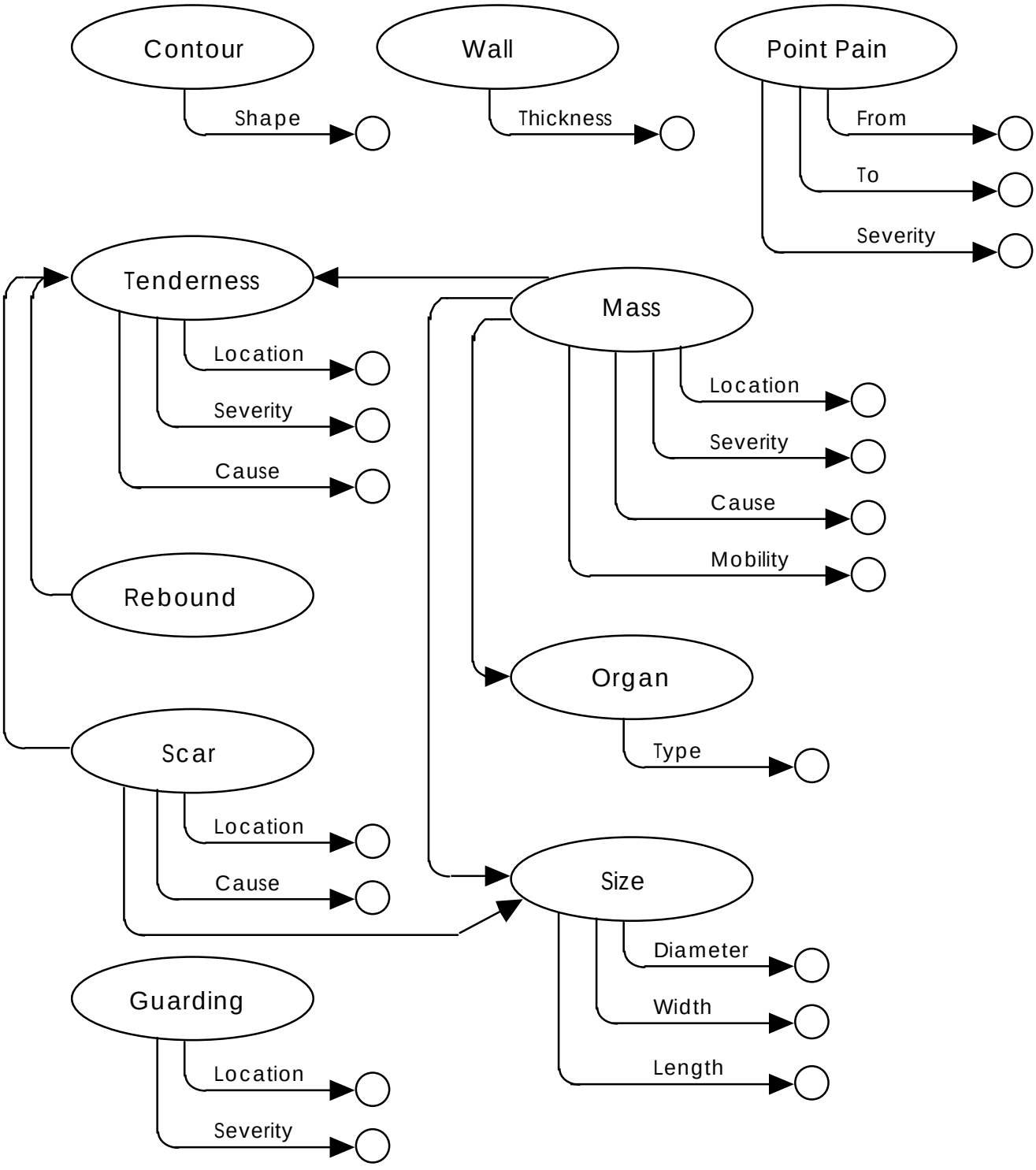


Figure VI.9 -- Semantic Frame for Appendicitis

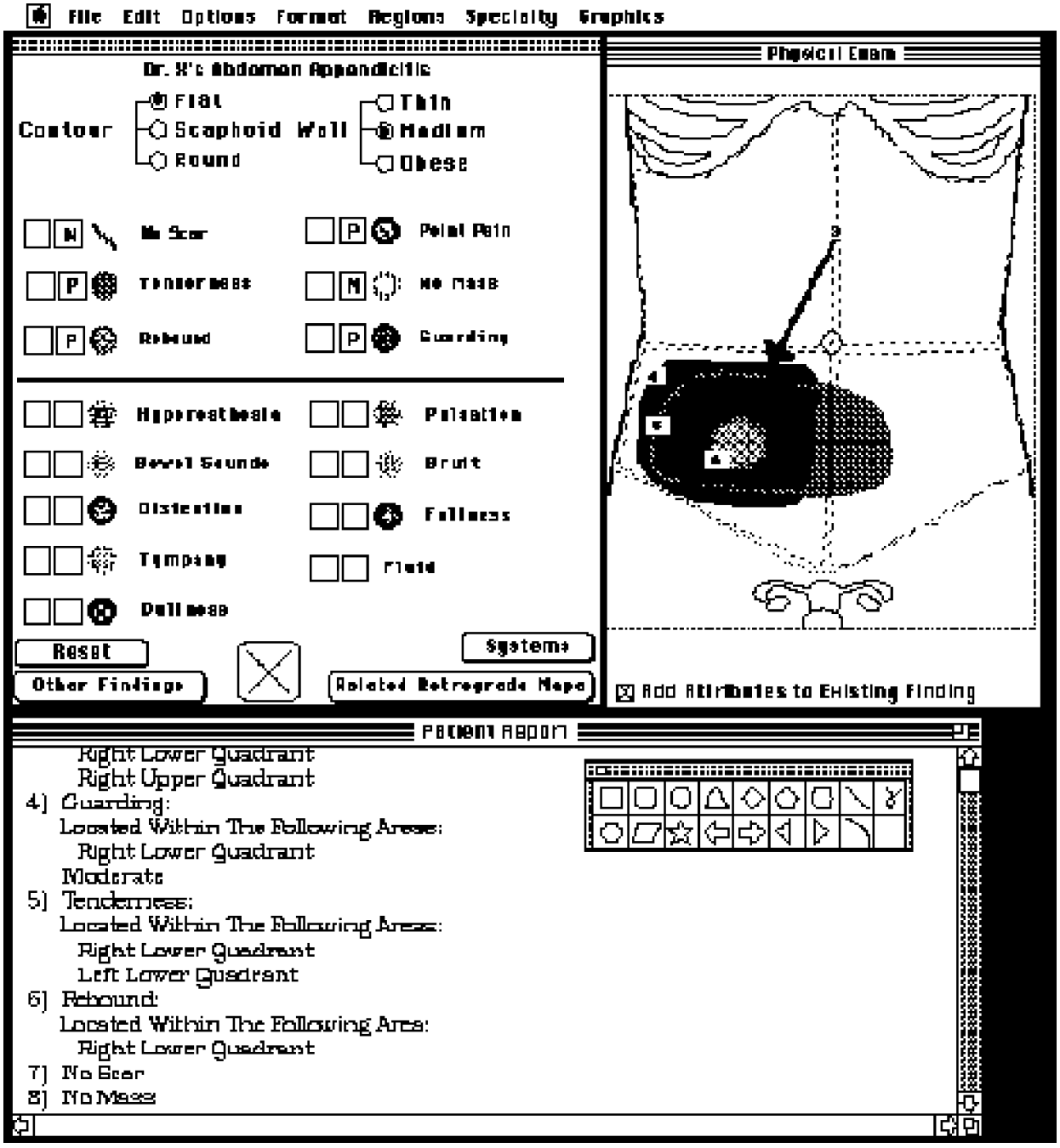


Figure VI.10 -- Retrograde Map of Appendicitis

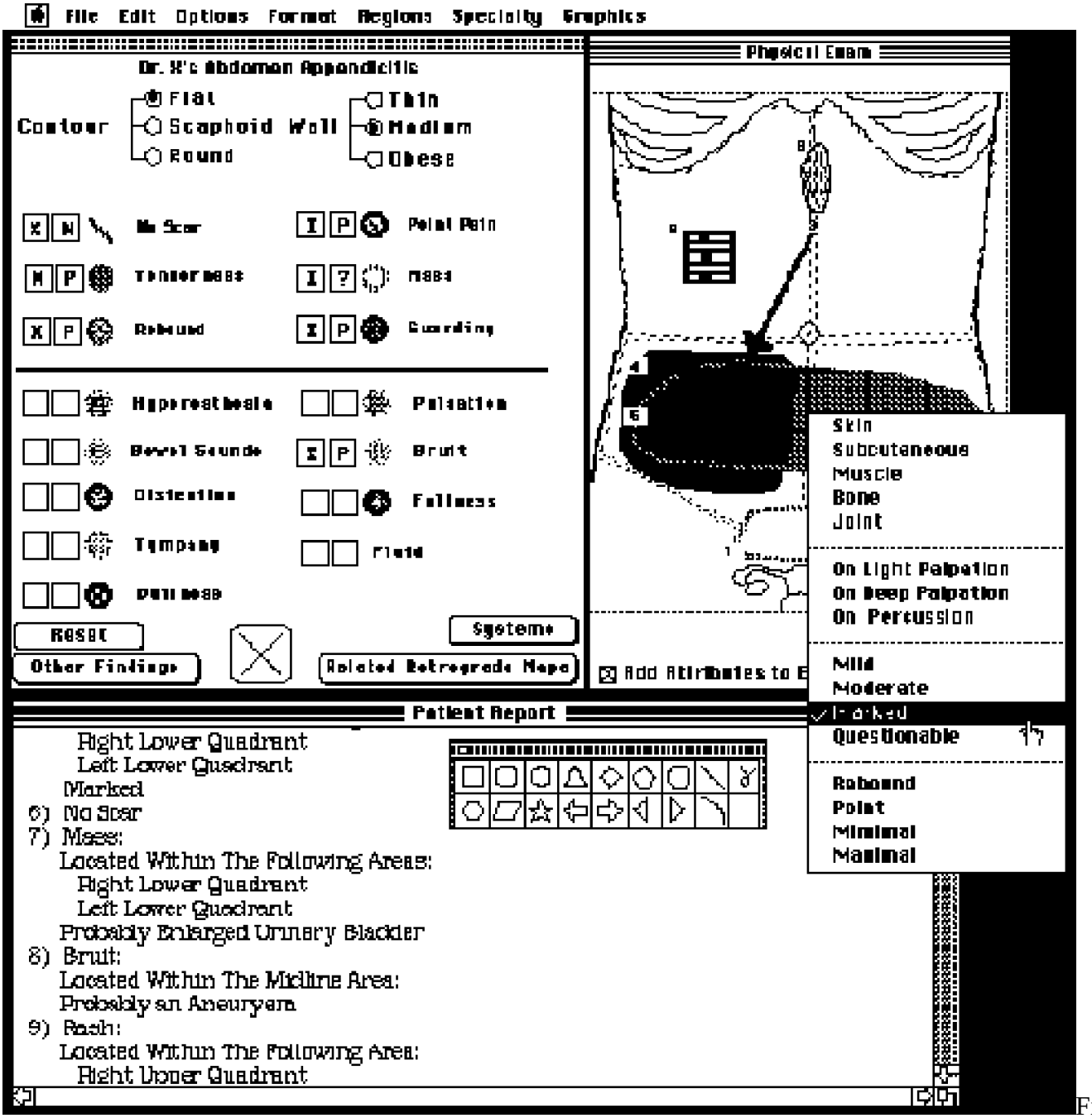


Figure VI.11 -- Abdominal Frame of Physical Exam

Editing Capabilities

For the interface to adapt to the needs and to the characteristics of the user, an easy and versatile editing feature is required. This is one of MEDIGATE's areas of emphasis and a great deal of development effort has been put into this area. Currently the MEDIGATE System editing capabilities allow for: 1) Custom Design of the listing of findings within a specific regional frame of findings, 2) Modification of the attributes associated with a finding and the related links in the semantic network, 3) Definition of sub-divisions, 4) Definition of RMaps, and 5) Design of the icon for a particular finding.

Custom Design of the Listing of Findings:

Figure VI.12 shows an example of how another physician could customize the abdomen frame. Here, the physician shows all available findings and has organized the findings that are routinely checked for on the left side of the finding window. The physician is allowed to custom design any frame of finding by adding new findings, by deleting current findings, or by renaming any findings. During this phase, the user also defines the report that will be generated for the "Normal Exams"

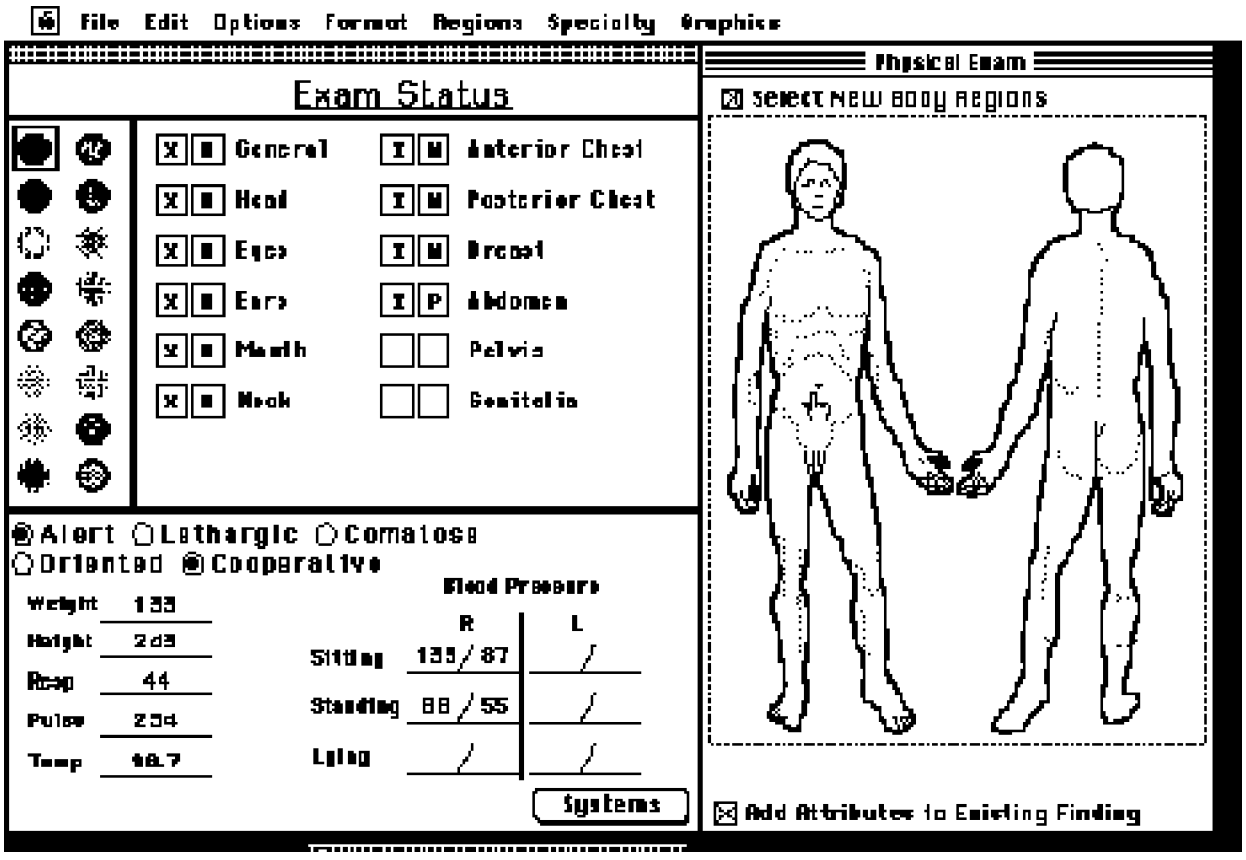


Figure VI.12 -- Alternative Abdominal Frame of Physical Exam

Modification of the Attributes and Semantic Links:

Figure VI.13 shows an example of the the "Editing Window" where the attributes can be modified and the semantic links set up. In this example, the attributes of the finding *Mass* are being modified; specifically the attributes of Liver are being modified. *Organ*, which is a kind of (AKO) *Mass*, has been previously linked as one of the menu items within *Mass*. Once this has been done, all the related attributes of *Organ* can be accessed and modified along with any of the sub-attributes. The "Editing Window" also allows for the user to define mutually exclusive sets and/or groupings of attributes along with the selection of normal and problem attributes (see Appendix 1 -- MEDIGATE'S Selection Popup Menus).

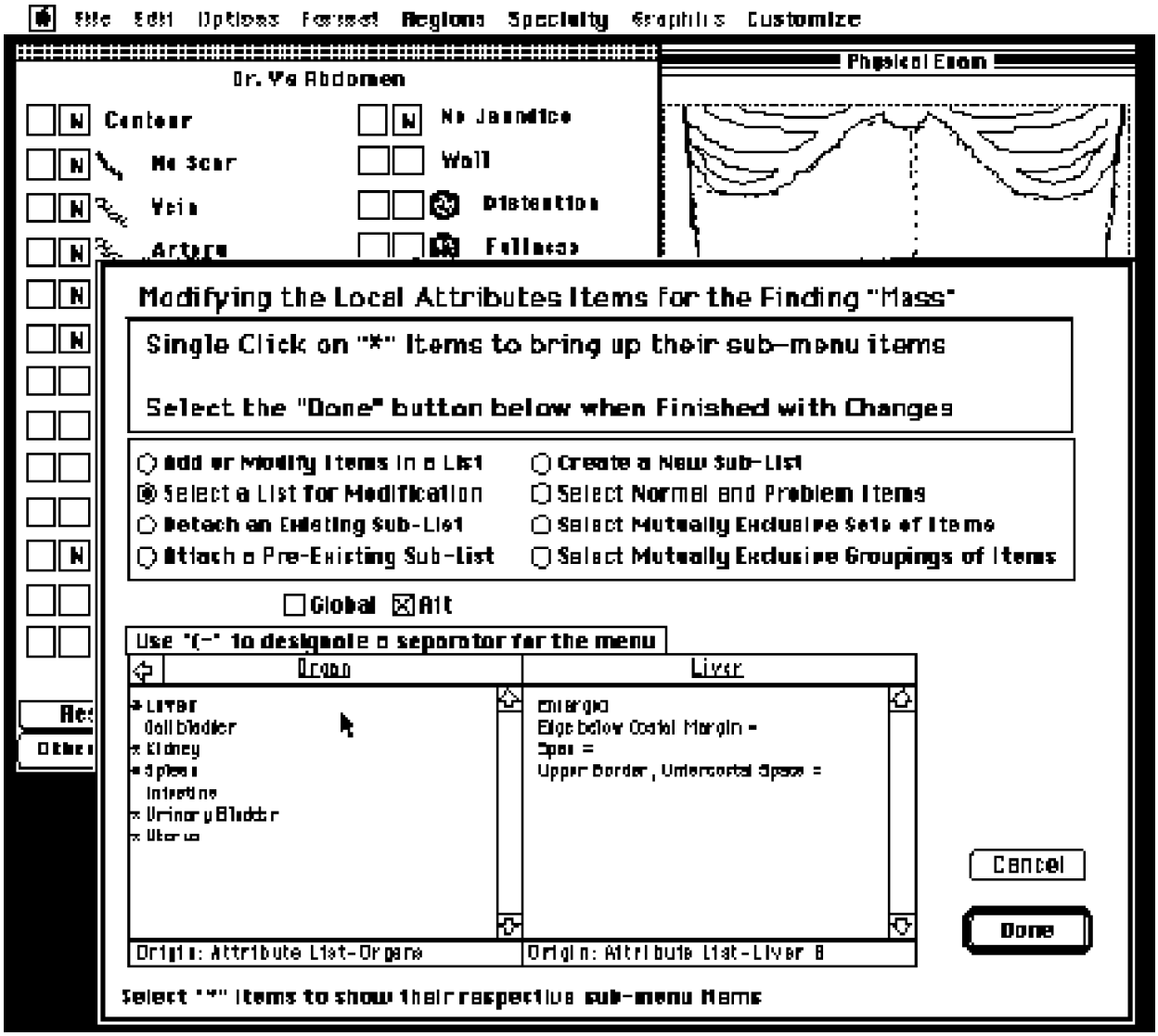


Figure VI.13 -- Editing Window

Definition of Sub-divisions:

Physicians can customize regions to include specific sub-divisions to meet their specific needs. Figure VI.14 shows an example of this. Initially the user is prompted to free hand draw in the boundaries of the desired sub-divisions. Note that in Figure VI.14, the user has drawn as indicated by dotted lines a sub-division within the *Abdominal Region*. After the user has drawn in a sub-division , s/he is prompted for the name of the new sub-division, which is indicated as "Right Upper Quadrant" in this example. The user can draw in as many sub-divisions as s/he wants and then select the "OK" button on the dialog window "Defining a Sub-Division" when s/he is ready to quit. This newly created sub-division will allow for MEDIGATE to automatically generate the location of any finding that is designated within its boundaries.

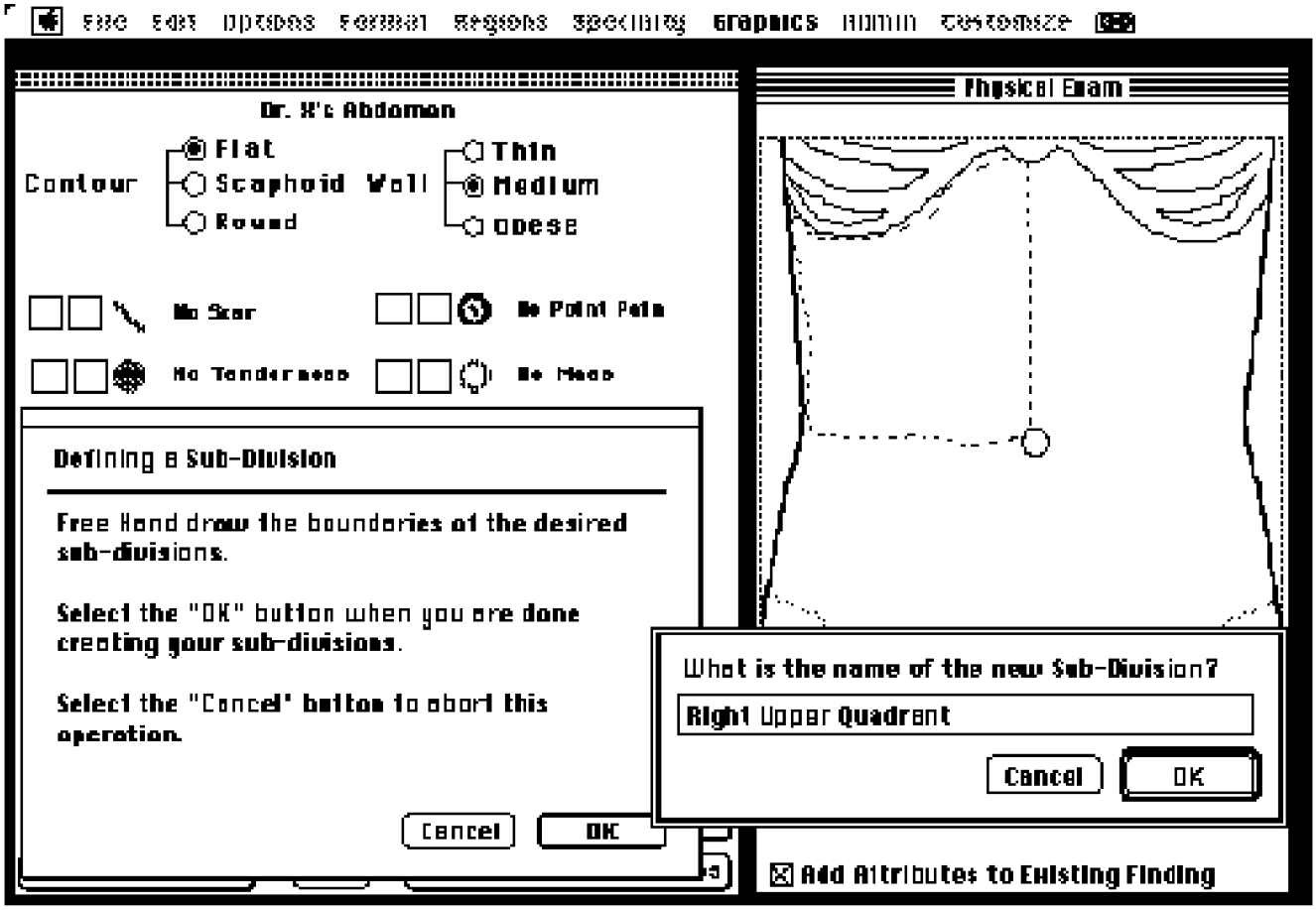


Figure VI.14 -- Defining Sub-Divisions

Developing Retrograde Maps:

Figure VI.15 shows how users can develop Retrograde Maps that fit into their area of practice. These RMaps are created by going to the region of interest and entering the related findings in reference to the desired RMap. At this point, the user is prompted for the name of the RMap; here the user is developing an *Appendicitis RMap*. As noted in Figure VI.15, in this example, the user has entered the following findings: Point Pain, Tenderness, Guarding, and Rebound Tenderness. Once the user has entered the desired findings, s/he selects the "OK" button to designate that the RMap is finished.

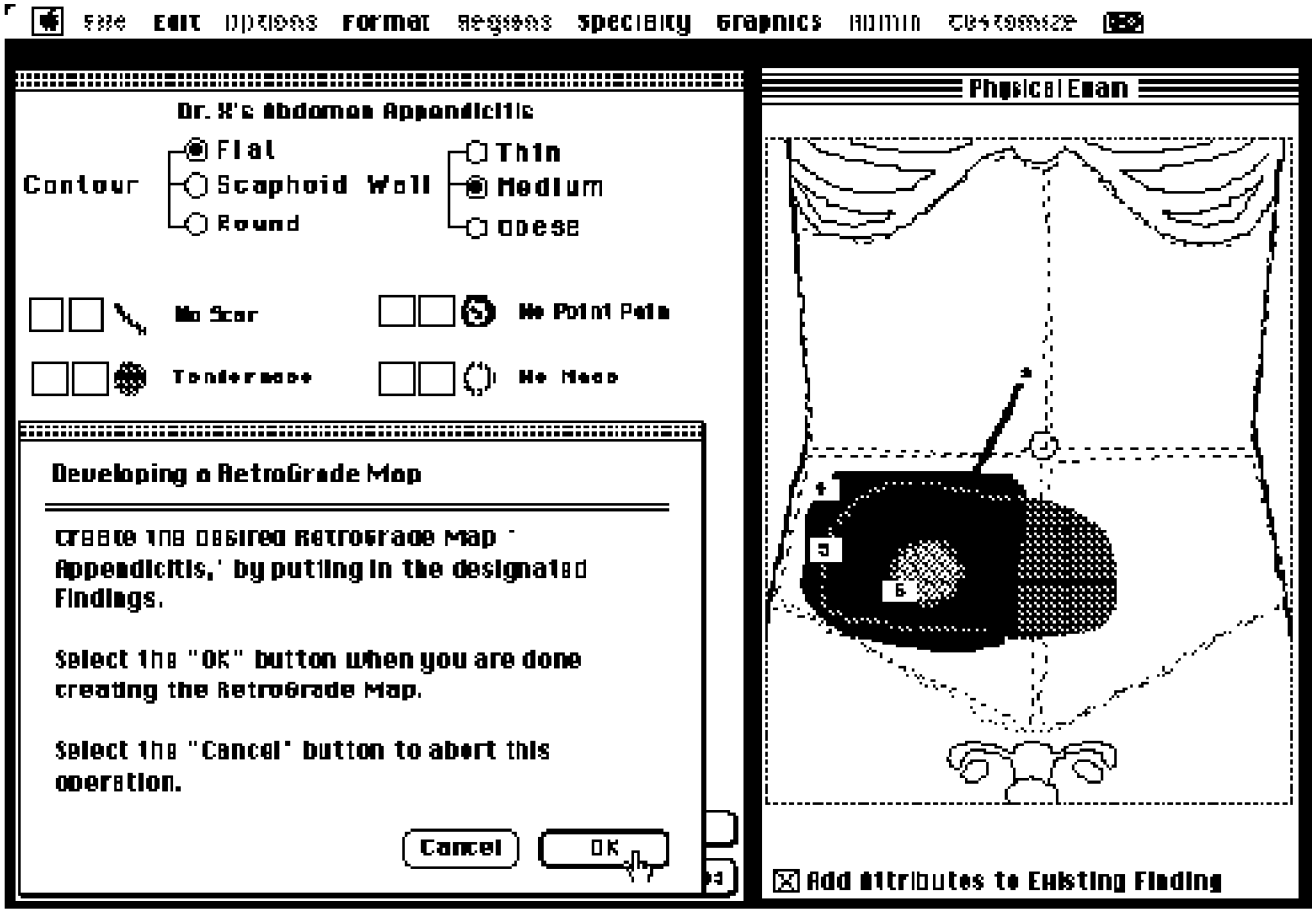


Figure VI.15 -- Developing a RMap

Designing the Icon for a Particular Finding:

The user can also design the attributes of the icon associated with any finding. This can be done either globally (i.e., all masses can have the same iconic shape and colors irrespective of body regions) or region specific; the demands of consistency favor the global approach in most cases. Figure VI.16 shows the window for setting these iconic attributes in action. The foreground and background colors of the fill pattern or border pattern can be set by choosing the appropriate colors from the color palette. Similarly, the foreground and background patterns of the fill and border can be set. This system also allows for the shape of the icon and the width of the border to be varied.

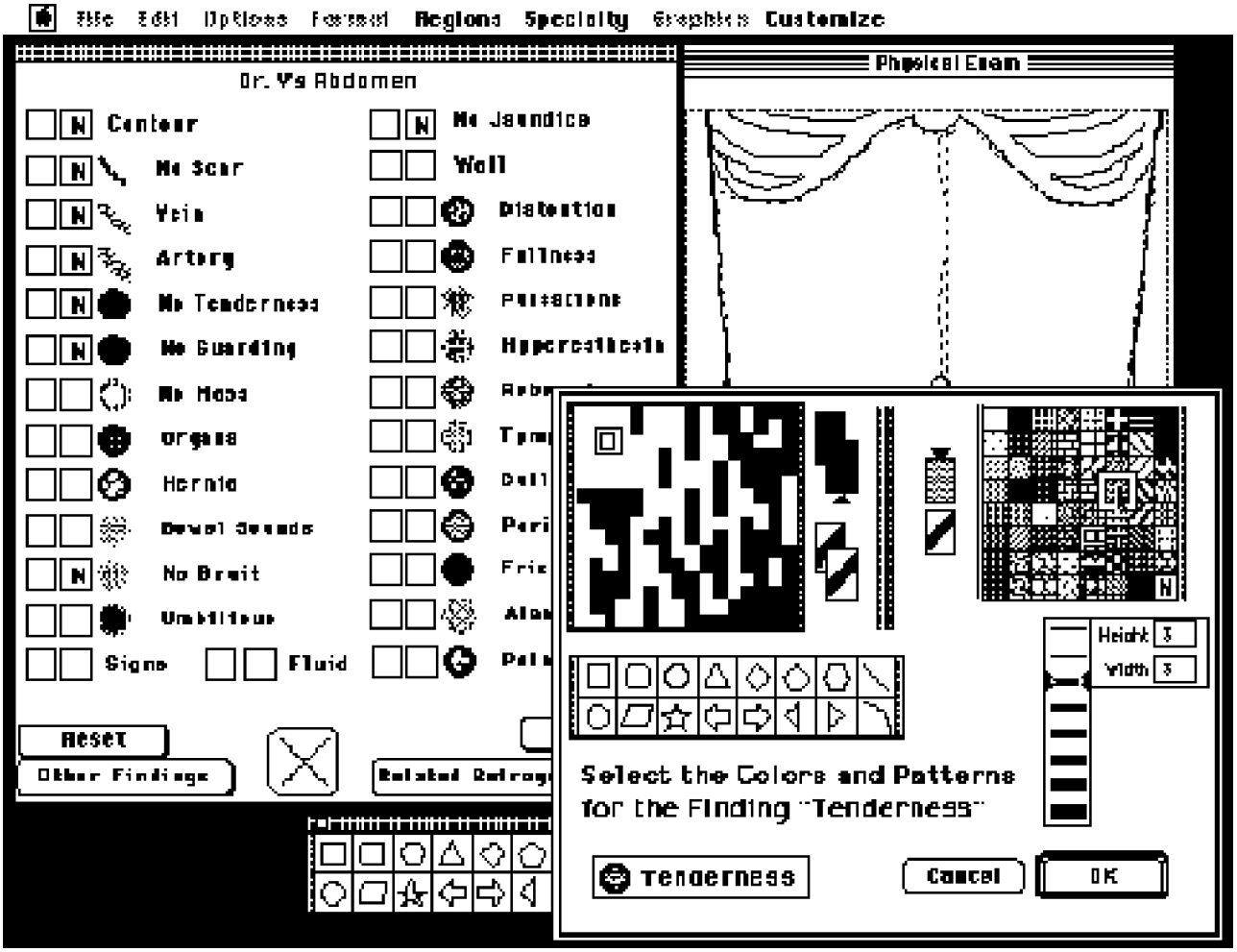


Figure VI.16 -- Setting the Iconic Attributes

Constraints

A constraint within a computerized system is a relationship among objects and data that is maintained when any of the objects change. These constraints should hold no matter what. Constraints are a natural way to express common relationships in graphical user interfaces [Myers, 1988; Myers et al., 1990]. The MEDIGATE System has incorporated graphical and data constraints as a means to insure that a graphical object maintains a particular relationship to a data value through the semantic network. For example, each noted finding has a set of attributes that characterize that specific finding. If the database of attributes is modified, the relationship of these attributes to their respective findings needs to change. An example of this is when the user changes the name of the attribute *Severe* of the finding *Tenderness* to *Marked*. Clearly, any textual and/or menu references to *Severe Tenderness* should be updated to reflect *Marked Tenderness*.

MEDIGATE solves these constraints in a lazy fashion [Zanden, 1990] by changing the relationships between the object and its changed attributes only when needed. For example, when the above change is made, no internal processing is done until the physician accesses a specific instance of the finding *Tenderness*. At that time, the system updates the references to *Severe* to reflect the changed attribute name of *Marked*.

VII. Proposed Evaluation of MEDIGATE

The evaluation of a computer interface can be a difficult task; specifically in terms of trying to define the parameters along with the trade-offs to compare to other systems. There are many basic parameters that could be evaluated in a system such as: ease of use, ergonomics, response time, completeness, ability to extract information from it, and errors of omission and accuracy. Once the crucial role of formulating significant questions has been achieved, there remains the problem of designing meaningful methodologies to obtain useful results. Chapanis [1959] has done much research in the generalized context of human factors while Ray and Ravizza [1986] along with Baecker and Buxton [1987] have characterized some specific research approaches that can be used in evaluating user interfaces. Baecker and Buxton [1987] outline research approaches in terms of the following four dimensions:

- Naturalistic observation versus true experiments. The former is the attempt to describe an ongoing process as it evolves over time; the latter is the process of studying relationships by manipulation of one or more independent variables directly and observing the effect on one or more dependent variables.
- Field research versus laboratory research. The latter gives us more control over the experimental factors. With the former, the subject is more likely to behave in a natural, realistic, ecologically valid manner.
- Scientist as participant versus scientist as observer. This approach deals with the extent to which the scientist is a part of the study or is a "passive" recorder of data.
- Few or many subjects. Possibilities range from the detailed study of one special individual to experiments on surveys of hundreds of individuals.

Certain considerations must be taken into account when comparisons are made between different input systems. For example, a pen and a blank piece of paper provide for a very "orthogonal" system through unlimited free text entry. However because of its convenience and minimal constraints, it allows for ambiguity, inconsistency, redundancy, and incompleteness. This leads to difficulties in database development, difficult retrieval and poor comprehension, not to mention problems of obscure handwriting, poor organization, and the idiosyncrasy of the observer. A more restrictive system such as a history/physical

exam form provides for orthogonality with some free-form entry of findings while adding organization and reminders.

One comparison done by Norman [1983] reveals the tradeoffs between menu-based systems and command language systems (see Table VII.1). This comparison demonstrates that principles from both menu-based systems and command language systems are important when considering the development of a user interface.

In evaluation, particularly as to the integrity of the database, it is important to consider negative findings as well as those findings that are not checked. Without this consideration, as noted before, an ambiguous database can evolve from the collection of findings. Regarding comparisons, the recording and remembering of these findings should prove difficult and time consuming with the use of dictation and paper interfaces. Thus, any evaluation should also compare dictation and paper interfaces for speed and ambiguity, especially when trying to designate finding that are not checked and negative findings.

Evaluation parameters of the MEDIGATE System are planned to be assessed by experimentation based upon physician's entry of findings obtained through examination of patients involving abdominal related disorders. Default data profiles relevant to diagnosis, treatment, or monitoring of abdomen disorders will be created. These schemata will be compiled not only from the QMR™ and PKC sources already noted but also from the empirically supported studies of [Pass, 1982; deDombal, 1984, 1985; Sox, 1983, 1986; Clarke, 1985; and Weed, 1986], case records of the Massachusetts General Hospital [Scully et al., 1991], and others. Initially, pen and paper input along with dictation will be performed with doctors along with their secretaries for transcribing. Data from this analysis will allow to give a clearer picture about MEDIGATE'S 1) speed of use, 2) prior knowledge

required, 3) ease of learning, 4) errors, 5) how useful is it, and 6) what is gained from what is lost. This analysis should also provide a comparison of the MEDIGATE System to other medical systems and supply the necessary information needed for future development.

TRADEOFFS BETWEEN MENU-BASED SYSTEMS[3] AND COMMAND LANGUAGE SYSTEMS		
ATTRIBUTE	MENU-BASED	COMMAND LANGUAGE
Speed of use:	Slow, especially if large or if has hierarchical structure	Fast, for experts; operation can be specified exactly, regardless of system state.
Prior knowledge required:	Very little - can be self-explanatory.	Considerable - user is expected to have learned a set of alternative actions and command language that specifies them.
Ease of learning:	High. Uses recognition memory: easier and more accurate than recall memory. Ease to explore system and discover options.	Low. Users must learn names and syntax of language. If alternatives are numerous, learning may take considerable time. No simple way to explore system and discover options not already known.
Errors:	Specification error leads to inappropriate action: difficult to determine what happened and to correct.	Specification error usually leads to illegal command: easy to detect, easy to correct.
Most useful for:	Beginner or infrequent user.	Expert or frequent user.

Table VII.1 -- Tradeoffs Between Menu-Based Systems and Command Language Systems

[3] Reprinted from Norman [1983]

VIII. Future Work and Ideas

Future work can be categorized into the following areas: 1) Extensions into other topographic regions; 2) Enhancement of the initial general start-up interface; 3) Expansion of the system for utilization in specific clinical settings; 4) Incorporation of standardized knowledge bases into the semantic network such as those provided by UMLS, QMR™, and Iliad; and 5) Extension of the core technology of MEDIGATE to include other types of medical procedures.

Currently the MEDIGATE System has only been fully incorporated into the abdomen and chest regions. Although work has been done towards completing the other topographic regions, this is still incomplete. Thus, the database needs to be enhanced to incorporate the principles developed throughout and to include complete listings of findings within each body region. The MEDIGATE program should also include hierarchical listings of findings in the different body systems, such as the orthopedic system.

Although it is very easy for the user to quickly move to a body region such as the abdomen and enter findings for that region, it is still difficult for the physician to quickly enter normal findings that s/he routinely checks for. As discussed earlier, work has been done towards quick entry of routine findings through the Exam Status window, but the front end interface still needs enhancement so that these routine checks can be done just as quickly as the physician does now, still retaining the ability to go deeper into the system upon demand.

Another useful future extension would be to incorporate the MEDIGATE System into specific settings and/or for specific purposes such as workman's compensation, family practice settings, speciality settings (orthopedics for example), and routine screening.

Progressive evaluation and iterative improvements will depend on different settings and their specific implementations.

Another problem is how to deal with patients seen in emergency departments in hospitals and urgent care centers. The advent of voice recognition technology [Kurzweil, 1991] and natural language recognition may prove very useful in addressing these needs. These systems still need to build upon a technology such as provided in MEDIGATE to promote some of the constraints that are not provided within these systems and to fully utilize the graphic interface for feedback. Such systems would use trigger phrases and a report framework tailored specifically for emergency medicine and built around the core technology within MEDIGATE. Thus, a voice recognition system would allow for a single spoken word or phrase to trigger a predefined sentence or paragraph along with possible graphics icon representations. Fill-ins allow standard phrases to be modified to describe a particular patient's status. Instead of dictating each word, trigger phrases are verbal shorthand to generate complete reports, in less time, and with fewer spoken words.

Work is now progressing to use the work of UMLS and the Arden Syntax into systems such as QMR™, Iliad, HELP™ and other consultation systems. When that work is completed there will be natural feedback into the semantic network, thus providing a more standard vocabulary. When recognized standardized vocabularies become more available, MEDIGATE should incorporate any such vocabularies into its database. This would also allow for MEDIGATE to act as a front end for some of these existing and developing systems.

The semantic network could be expanded to include procedural and causal knowledge as stated above. Event-describing frames would be created that fire a set of rules prompting for

further information or possible follow-up. An example of this would be: The physician found a Mass in the RLQ along with Tenderness and Point Pain. After this frame is created the system would then fire off related rules that relates (maps) these findings as possibly appendicitis. These rules could then quantify the likelihood of appendicitis and suggest or prompt for more related information.

It should also be feasible to use knowledge as incorporated in consultation systems such as QMR™ or Iliad for the development of the retrograde mapping approach to description generation, coupled with some explanatory reasoning on request. These expert systems could help to automatically map into a RMap previously developed. While most diagnostic expert systems operate "forward," from findings to intermediate and final conclusions, retrograde descriptive systems operate in the opposite direction, from conclusions at different levels of abstraction to templates of classical findings.

The ideas of RMaps can further be developed for automatic machine learning [Pionkowski et al., 1989]. New RMaps could automatically be developed or old RMaps could automatically be updated by allowing the system to do pattern matching and by noting trends. This could build upon the semantic network and hopefully be tied into a more standardized vocabulary such as work being done in UMLS.

Medical procedures extend beyond the physical examination. The principles of the initial PLATO and the current MEDIGATE System could be expanded to include many types of medical procedure, as has been suggested in the somewhat similar recent approaches described for recording the findings of gastrointestinal endoscopy [Kahane et al., 1987] and of cardiac surgical procedures [Wheeler and McConnell, 1987]. In addition, there are several computer-based diagnostic and management support systems, often using expert

systems approaches, as well as teaching tools for medical students and practitioners such as Iliad; these often lack a user interface capable of facile data input, especially for the collection of physical findings. These design principles could be employed in MEDIGATE to help develop more effective interfaces for use within many other areas in the medical field.

Recently, much development has been done with pen-based computers. Pen-based system are not only more natural for the physician to use but also brings the view closer to the action, thus providing more direct continuous control which will provide for higher precision. As previously noted, written records can not be as precise as with MEDIGATE without much effort. This is because MEDIGATE emphasizes completeness and precision, and suppresses ambiguity by its use of the current technology which promotes: 1) accuracy (graphical feedback), 2) verification with immediate feedback, and 3) the teasing out of subtleties from clinical data. By the blind implementation of a medical record with a pen-based system, there will still be the problems of ambiguity, inaccuracy, and incompleteness that are associated with hand-written records.

Pen-based systems built on a good interface that forces some constraints could allow for a more “orthogonal” system, thus giving the user the freedom of using a tool that they are familiar with (the pen) and also providing them with a way to organize their data better, along with the other benefits that are provided by the processing power of a computer. This would provide the physician with continuous control methods such as scribing, drawing, and gesturing for obtaining more qualitative and quantitative data. There are still unresolved problems associated with handwriting recognition techniques that some of the methods being studied by fuzzy logic may help answer and make pen-based systems available as a valuable interface tool.

Multimedia technology is also progressing at a promising rate and the integration of audio and video links that provide for immediate reference may also be a powerful addition to any medical information system. This could allow for the examiner to call up a medical reference and/or a video picture of a procedure of concern, thus getting immediate feedback and reminders.

IX. Summary and Conclusions

The solution to many of the problems attendant upon the computer-based recording of the medical record has long awaited the advanced technological development of computer hardware and software systems. Reliable input of data has proven to be more complex than originally envisioned by early work in the field. This has led to more effort into the development of good interfaces. The characteristics of the user are very important to the development of a good interface system.

In the past, most medical information systems have not utilized many of the tools of software engineering, especially in the area of interface design. Recently technological advances in computer science have provided some means to meet the complex needs of the medical community.

Components of the medical record system are examined as repositories of medical data, each with characteristic problems of input and retrieval. In early systems, the focus was primarily on the storage and processing of the data rather than on the problems associated with the collection and display of the data and the associated issues of interface design.

An adequate interface system must be concerned with the cognitive and manipulative style of the user together with the characteristics of the data. Additional software engineering techniques can be developed to utilize this principle to provide for an easily useable interface.

The MEDIGATE System has been developed with current technology to study some of the problems in interface design The design employs an object-oriented approach through the

direct manipulation of graphical objects, along with hypertext approaches and semantic networking to build an “orthogonal” system that is more natural to the user.

Principles of the MEDIGATE System can be applied to newer technologies such as pen-based computers and voice recognition systems, thus allowing for greater flexibility and versatility in the user interface. MEDIGATE could be used as a core component for these emerging technologies, thus enforcing completeness, disambiguation, and precision.

It is envisioned that the principles embodied in the MEDIGATE System will play a significant role in future computer-based medical information systems by allowing for the capture of meaningful data and providing for display and utilization of the data, thus enhancing the quality of patient care, risk management, and clinical research.

References

Atsumi K.; "Technological and Social Assessment of Medical Information Systems"; Proceedings of the IFIP Working Conference on Information Systems For Patient Care: Review, Analysis and Evaluation, Amsterdam, The Netherlands, van Egmond J., de Vries Robbe' P.F., & Levy A.H. ed, pp. 57-74, 1976.

Baecker R.M.; "Human-Computer Interactive Systems: A State-of-the-Art Review"; in Processing of Visible Language 2, Kilers P.A., Wrolstad M.E., and Bouma H. eds, Plenum Press, New York, pp. 423-443, 1980.

Baecker R.M., Buxton W.A.S., & Reeves W.; "Towards Facilitation Graphical Interaction: Some Examples from Computer-Aided Musical Composition"; Proceedings of the 6th Canadian Man-Computer Communications Conference, pp. 197-207, May, 1979.

Baecker R.M. & Buxton W.A.S.; Readings In Human-Computer Interaction; Morgan Kaufmann Publishers, Inc., Los Altos, California 1987.

Bailey H.; Demonstration of Physical Signs in Clinical Surgery; Eleventh Edition by The Williams & Wilkins Co., Baltimore 1949.

Bakker A.R.; "The Information System of Leyden University Hospital"; Proceedings of the IFIP Working Conference on Information Systems For Patient Care: Review, Analysis and Evaluation, Amsterdam, The Netherlands, van Egmond J., de Vries Robbe' P.F., & Levy A.H. eds, pp. 221-248, 1976.

Ball M.J. & Collen M.F. (Editors); Aspects of the Computer-based Patient Record; Springer-Verlag, New York 1992.

Barnett G.O, Cimino J.J., Hupp J.A. & Hoffer E.P.; "DXplain: Experience with Knowledge Acquisition and Program Evaluation"; Proc. 11th Ann. Symp. Comp. Appl. Med. Care, 150-154, IEEE 1987.

Bauer J.; Differential Diagnosis of Internal Diseases; Third Revised and Enlarged Edition by Grune & Stratton, New York 1967.

Berkow R., & Fletcher A.J.; The Merck Manual of Diagnosis and Therapy; Fifteenth Edition by Merck Sharp & Dohme Research Laboratories, Rahway, N.J. 1987.

Brajnik G., Guida G. & Tasso C.; "An Expert Interface for Effective Man-Machine Interaction"; in Cooperative Interfaces to Information Systems, L. Bolc and M. Jarke, Eds., Berlin, FRG: Springer-Verlag, pp. 259-308, 1986.

Brajnik G., Guida G. & Tasso C.; "User Modeling in Expert Man--Machine Interfaces: A Case Study in Intelligent Information Retrieval"; IEEE, **20**(1)166-185, 1990.

Buckingham W.B., Sparberg M. & Martin B.; A Primer of Clinical Diagnosis; Harper & Row, Publishers, New York 1971.

Carroll J.M., ed.; Interfacing Thought *Cognitive Aspects of Human-Computer Interaction*; MIT Press, Cambridge, MA 1987.

Chute C.G., Yang Y., Tuttle M.S., Sherertz D.D., Olson N.E., & Erlbaum M.S.; "A Preliminary Evaluation of the UMLS Metathesaurus for Patient Record Classification"; Proceedings of The Fourteenth Annual Symposium on Computer Applications in Medical Care, Washington, DC USA, Miller, R.A. ed: Institute of Electrical and Electronics Engineers, pp. 161-165, 1990.

Clarke J.R.; "A Comparison of Decision Analysis and Second Opinions for Surgical Decisions"; Arch. Surg., **120**:844-847, 1985.

Clayton P.D., Hripcsak G., & Pryor T.A.; "Emerging Standards for Medical Logic"; Proceedings of The Fourteenth Annual Symposium on Computer Applications in Medical Care, Washington, DC USA, Miller, R.A. ed: Institute of Electrical and Electronics Engineers, pp. 27-31, 1990.

College of American Pathologists, Committee on Nomenclature and Classification of Disease; Systematized Nomenclature of Pathology; Chicago, IL 1965.

Cote R.A., ed.; Systematized Nomenclature of Medicine; Second Edition by the College of American Pathologists, Skokie, IL 1984.

Cox B.J.; Object-Oriented Programming An Evolutionary Approach; Addison-Wesley Publishing Company, Reading, MA 1986.

Cox P.T., Giles F.R., & Pietrzykowski T "Prograph: A Step Towards Liberating Programming From Textual Conditioning"; In 1989 IEEE Workshop on Visual Languages, pp. 150-156, October, 1989.

Davis L.S. & Simacek T.A.; "Distributed Data Base Medical Information Systems"; Proceedings of the IFIP Working Conference on Information Systems For Patient Care: Review, Analysis and Evaluation, Amsterdam, The Netherlands, van Egmond J., de Vries Robbe' P.F., & Levy A.H. ed, pp. 163-190, 1976.

de Dombal F.T.; "The O.M.G.E. Acute Abdominal Pain Survey - Progress Report, 1982"; Scand. J. Gastroenterol. **19**(suppl.95):28-40, 1984.

de Dombal F.T.; "Analysis of Symptoms in the Acute Abdomen"; Clinics in Gastroenterol. **14**(3):531-543, 1985.

DeGowin R.L.; DeGowin & DeGowin's Bedside Diagnostic Examination; Fifth Edition by Macmillan Publishing Company, New York 1987.

Dick R.S. & Steen E.B. (Editors); The Computer-Based Patient Record: An Essential Technology for Health Care; National Academy Press, Washington, D.C. 1992.

Douglas S., Doerry E., & Novick D.; "QUICK: A User-Interface Design Kit for Non-Programmers"; Proceedings of the ACM Siggraph Third Annual Symposium on User Interface Software and Technology, Snowbird, Utah, ACM Press, pp. 47-56, 1990.

Engelbrecht R.; "Experiences in Designing Human-Computer Interfaces for Doctors' Office Computers"; Proceedings of the IFIP-IMIA Second Stockholm Conference on Communication in Health Care, Stockholm, Sweden, Peterson H.E. & Schneider W. ed, pp. 219-233, 1985.

Evans C. & Price H.C.; "The Doctor Patient Interaction Or Consultation: Can the Computer Assist in This Dialogue?"; Proceedings of the IFIP Working Conference on Information Systems For Patient Care: Review, Analysis and Evaluation, Amsterdam, The Netherlands, van Egmond J., de Vries Robbe' P.F., & Levy A.H. ed, pp. 117-132, 1976.

Fischer O. & Smith J.W.; "Contextual Pathognomony: A Computationally Useful Extension Of Pathognomony"; Proceedings of The Fourteenth Annual Symposium on Computer Applications in Medical Care, Washington, DC USA, Miller, R.A. ed: Institute of Electrical and Electronics Engineers, pp. 44-50, 1990.

Foley J.D. & Wallace V.L.; "The Art of Natural Graphic Man-Machine Conversation"; Proceedings of the IEEE, **62**(4):462-471, 1974.

Fries J.F.; "Time-oriented Patient Records: A Computer Data Bank"; JAMA, **222**:1536-1542, 1972.

Fries J.F. & McShane D.J.; "ARAMIS (The American Rheumatism Association Medical Information System)"; The Western Journal of Medicine, **145**:798-804, Dec. 1986.

Gane C., & Sarson T.; Structured Systems Analysis: tools & techniques; McDonnel Douglas Professional Services Company, Saint Louis, Missouri 1987.

Gould J.D. & Lewis C.; "Designing for Usability: Key Principles and What Designers Think"; Communications of the ACM **28**(3):300-311, 1985.

Griesser G.; "Presentation and Use of Medical Basic Information in Kiel KIS"; Proceedings of the IFIP Working Conference on Information Systems For Patient Care: Review, Analysis and Evaluation, Amsterdam, The Netherlands, van Egmond J., de Vries Robbe' P.F., & Levy A.H. ed, pp. 205-220, 1976.

Hansen W.J.; "User Engineering Principles for Interactive Systems"; AFIPS Conference Proceedings 39 Montvale, N.J., pp. 523-532, AFIPS Press, 1971.

Haug P.J., Warner H.R., Clayton P.D., Schmidt C.D., Pearl J.E., Farney R.J., Crapo R.O., Tocino I., Morrison W.J., & Frederick P.R.; "A Decision-Driven System to Collect Patient History"; Comp.Biomed.Res., **20**:193-207, 1987.

Himes A., & Ragland C.; Inside SuperCard™ The Complete Guide for Macintosh® Developers and Advanced Users with technical assistance from Bill Appleton, Microsoft Press, Redmond, Washington 1990.

Hoeppner W., Morik K., & Marburgber H.; "Talking it over: The Natural language dialog system HAM-ANS"; in Cooperative Interfaces to Information Systems, L. Bolc and M. Jarke, Eds., Berlin, FRG: Springer-Verlag, pp.189-258, 1986.

Hripcsak G., Clayton P.D., Pryor T.A., Haug P., Wigertz O.B., & Van der lei J.; "Emerging Standards for Medical Logic"; Proceedings of The Fourteenth Annual Symposium on Computer Applications in Medical Care, Washington, DC USA, Miller, R.A. ed: Institute of Electrical and Electronics Engineers, pp. 200-205, 1990.

Huff S.M., & Warner H.R.; "A Comparison of Meta-1 and HELP terms: Implications for Clinical Data"; Proceedings of The Fourteenth Annual Symposium on Computer Applications in Medical Care, Washington, DC USA, Miller, R.A. ed: Institute of Electrical and Electronics Engineers, pp. 166-169, 1990.

Hukill Michael J., Ward K.M., Haug P.J., & Warner H.R; "HELP Decision Support on the Macintosh®"; Proc. 11th Ann. Symp. Comp. Appl. Med. Care, pp.155-157, IEEE 1987.

Ikeda H., Ohtomo M., & Kambe M.; "Intelligent Access of Health Care Databases"; Proceedings of the IFIP-IMIA Second Stockholm Conference on Communication in Health Care, Stockholm, Sweden, Peterson H.E. & Schneider W. ed, pp. 257-260, 1985.

Jelovsek F.R., Catanzarite V.A., Price R.D., & Stull R.E; "Learning Theory and Knowledge Structures in Computer-Aided Instruction"; M.D. Computing **7**(2):98-102, 1990.

Judge R.D. & Zuidema G.D.; Physical Examination: A Physiologic Approach To The Clinical Examination; Second Edition by Little, Brown and Company, Boston 1968.

Kahane S.N., Johannes R.S., & Lenhard R.E.; "Collecting Data After Medical Procedures: Designing Workstation Methods and Creating Incentives"; 3rd Ann. Intl. Conf. Comput. Med. Rec., Chicago, Institute for Medical Record Economics, Inc., 1987.

Kampmeier R.H.; Physical Examination in Health and Disease; F. A. Davis Company, Publishers, Philadelphia 1950.

Kuperman G.J., Gardner R.M., & Pryor T.A.; HELP: A Dynamic Hospital Information System; Springer-Verlag, New York 1991.

Kurzweil; "VoiceEM"; Internal documentation by Kurzweil AI Systems 1991.

Lenhard R.E., Kahane S.N., Richmond D.W., Phipps K.J., Ardolino M.K., Kearney L.A., & Lifshitz K.L.; "A Computer Workstation for Clinical Medicine"; Journal of Medical Systems, **14**(5):227-243, 1990.

Levy A.H. & Lawrance D.P.; "Data Acquisition and the Computer-based Patient Record"; in Aspects of the Computer-based Patient Record, M. J. Ball and M. F. Collen, Eds., Springer-Verlag, New York, pp. 125-139, 1992.

Levy A.H. & Lawrance D.P.; "Information Retrieval"; in Aspects of the Computer-based Patient Record, M. J. Ball and M. F. Collen, Eds., Springer-Verlag, New York, pp. 146-152, 1992.

Lindberg D.A.B. & Humphreys B.L.; "UMLS Knowledge Sources: Tools for Building Better User Interfaces"; Proceedings of the Fourteenth Annual Symposium on Computer Applications in Medical Care, Washington D.C., USA, Miller, R.A. ed, pp. 121-125, 1990.

Lindberg D.A.B. & Humphreys B.L.; "The Unified Medical Language System (UMLS) and Computer-based Patient Records"; in Aspects of the Computer-based Patient Record, M. J. Ball and M. F. Collen, Eds., Springer-Verlag, New York, pp. 125-139, 1992.

MacBryde C.M., & Blacklow R.S.; Signs and Symptoms; Fifth Edition by J. B. Lippincott Company, Philadelphia 1970.

Mandil S.H.; "The Relevance of Images and Graphics to Primary Health Care"; Proceedings of the IFIP-IMIA Second Stockholm Conference on Communication in Health Care, Stockholm, Sweden, Peterson H.E. & Schneider W. ed, pp. 45-66, 1985.

McAdam W.A., Brock B.M., Armitage T., Davenport P. Chan M. Dombal F.T.; "Twelve years' experience of computer-aided diagnosis in a district general hospital"; Annals of the Royal College of Surgeons of England, **72**:140-146, 1989.

McDonald, C.J. & Tierney, W.M.; "The Medical Gopher-A Microcomputer System to Help Find, Organize And Decide About Patient Data"; The Western Journal of Medicine **145**:823-829, Dec. 1986.

Miller R.A., Masarie F.E. & Myers J.D.; "Quick Medical Reference (QMR™) for Diagnostic Assistance"; MD Computing, **3**:34-48, 1986.

Miller R.A., Pople H.E. Jr., & Myers J.D.; "Internist-1, An Experimental Computer-Based Diagnostic Consultant for General Internal Medicine"; The New England Journal of Medicine, **307**:468-476, 1982.

Morik K. & Rollinger C.R.; "The real estate agent -- Modeling Users by Uncertain Reasoning"; AI Mag., **6**(2):44-52, 1985.

Mutalik Pradeep, Rose J.S., Fisher P., Swett H.A., & Miller P.L.; "Internist-Like Diagnostic Strategies Structure the Form and Content of Explanation in an Expert Critiquing System"; in Proceedings 1988 Spring Symposium Series, AI in Medicine, Stanford University, 3/22-4/88, pp. 65-66.

Myers B.A.; Creating User Interfaces by Demonstration; Academic Press, Inc. San Diego, CA 1988.

Myers B.A., Giuse D.A., Dannenberg R.B., Zanden B.V., Kosbie D.S., Pervin E., Mickish A., & Marchal P.; "Garnet: Comprehensive Support for Graphical, Highly Interactive User Interfaces"; Computer pp.71-85, November 1990.

National Library of Medicine, Library Operations; Medical Subject Headings; Bethesda, Maryland 1984.

Nilsson, N.J.; Principles of Artificial Intelligence; Morgan Kaufmann Publishers, Inc., 1980.

Norman D.A.; "Design Principles for Human-Computer Interfaces"; Proceedings of CHI '83, pp. 1-10, 1983.

O'Hagan N.K.; "CompuHx: An Interactive Health Appraisal System"; Proceedings of The International Health Evaluation Association Annual Symposium on The Art and Science of Preventive Medicine, La Jolla, California USA, Felitti, V.J. ed, pp. 106-114, 1990.

Packer M.S., Cimino J.J., Hoffer E.P., Barnett G.O, Kim R., & Zatz S.; "Updating the DXplain Database"; Proceedings of the Twelfth Annual Symposium on Computer Applications in Medical Care, New York USA, Greenes, R.A. ed: Institute of Electrical and Electronics Engineers, pp. 96-100, 1988.

Parker D.; "Cerner's PowerChart: Just What the Doctor Ordered"; IntelliNews, IntelliCorp, Inc., Mountain View, Cal. 1987.

Pass T.M., Komaroff A.L. & Ervin C.T.; "Categorical Approaches to Clinical Decision Support"; in Computer Aids to Clinical Decisions, Vol 1; CRC Press, Boca Raton, FL. 1982.

Pauker S.G., Gorry G.A. & Kassirer J.P.; "Towards the simulation of clinical cognition: taking a present illness by computer"; Am.J.Med., **60**:981, 1976.

Pionkowski R., Baskin A., Williams B.T.; "A Prototype for Adaptable Physician-Directed Data Entry"; Medinfo 89: Proceedings of the 6th Conference on Medical Informatics, Beijing-Singapore, Barber B, Cao D, Qui D, Wagner G ed, North-Holland, Amsterdam, pp. 1156-1159, 1989.

Pressman R.S.; Software Engineering A Practitioner's Approach; Second Edition by McGraw-Hill Book Company, New York 1987.

Punch Wm.F., Smith J.W. & Chandrasekeran B.; "Indexing and Use of Deep Knowledge during Compiled Diagnosis"; in Proceedings 1988 Spring Symposium Series, AI in Medicine, Stanford University, 3/22-4/88, 71-72.

Reichertz P.L.; "The Need For Alternative Design Of Computer Systems For Improved Human-Computer Communication"; Proceedings of the IFIP-IMIA Second Stockholm Conference on Communication in Health Care, Stockholm, Sweden, Peterson H.E. & Schneider W. ed, pp. 85-97, 1985.

Rubinstein R. & Hersh H.M; "Design Philosophy"; in The Human Factor: Designing Computer Systems for People, Digital Press, Burlington, MA Chpt 2, 1984.

Scully R.E., Mark E.J., McNeely W.F., McNeely B.U.; "Case Records of the Massachusetts General Hospital: Weekly Clinicopathological Exercises"; The New England Journal of Medicine, **325**(9):1156-1159, 1991.

Scully R.E., Mark E.J., McNeely W.F., McNeely B.U.; "Case Records of the Massachusetts General Hospital: Weekly Clinicopathological Exercises"; The New England Journal of Medicine, **325**(18):1295-1302, 1991.

Searle R.H.; "Medical Record Access and Linkage and the Question of Informed Patient Consent"; Proceedings of the IFIP Working Conference on Information Systems For Patient Care: Review, Analysis and Evaluation, Amsterdam, The Netherlands, van Egmond J., de Vries Robbe' P.F., & Levy A.H. ed, pp. 117-132, 1976.

Seidel H.M., Ball J.W., Daines J.E., Benedict G.W.; Mosby's Guide to Physical Examination; The C.V. Mosby Company, St. Louis 1987.

Shneiderman B.; "The Future of Interactive Systems and the Emergence of Direct Manipulation"; Behavior and Information Technology, **1**(3):237-256, 1982.

Shneiderman B.; "Direct Manipulation: A Step Beyond Programming Languages"; IEEE Computer, **16**(8):57-69, Aug. 1983.

Shneiderman B.; Designing the User Interface: Strategies for Effective Human-Computer Interaction; Addison-Wesley, Reading, Mass 1987.

Sleeman D., Appelt D., Konolige K., Rich E., Sridharan S., & Swartout B.; "User Modeling Panel"; Proceedings 9th International Joint Conference of Artificial Intelligence, Los Angeles, CA., pp. 1298-1302, 1985.

Solheim B.G. & Hansen R.; "Actual Problem For Human-Computer Communication In Hospital Laboratory Systems"; Proceedings of the IFIP-IMIA Second Stockholm Conference on Communication in Health Care, Stockholm, Sweden, Peterson H.E. & Schneider W. ed, pp. 81-84, 1985.

Sox H.C.; "Noninvasive Testing in Coronary Artery Disease"; Postgrad.Med., **74**(3):319-336, 1983.

Sox H.C.; "Clinical Prediction Rules in Practice"; Hosp.Pract., pp.100-103, 11/30/86.

Spark-Jones K.; "Issues in User Modelling for Expert Systems"; in Artificial Intelligence and Its Applications, A. G. Cohn and J. R. Thomas, Eds., John Wiley, Publishers, New York pp. 183-195, 1986.

Sperzel W.D., Earlbaum M., Fuller L., Sherertz D., Olson N., Schuyler P., Hole W., Savage A., Passarelli P., & Tuttle M.; "Editing the UMLS Metathesaurus: Review and Enhancement of a Computed Knowledge Source"; Proceedings of the Fourteenth Annual Symposium on Computer Applications in Medical Care, Washington D.C., USA, Miller, R.A. ed, pp. 136-140, 1990.

Stead W.W., & Hammond W.E.; "Demand-Oriented Medical Records: Toward a Physician Work Station"; Proceedings of the Eleventh Annual Symposium on Computer Applications in Medical Care, Los Angeles CA, Stead W.W. ed, pp. 275-279, 1987.

Taylor R.B.; A Primer of Clinical Symptoms; Harper & Row, Publishers, New York 1973.

Tufo H.M., Bouchard R.E., Rubin A.S., Twitchell J.C., VanBuren H.C., Weed L.B., & Rothwell M; "Problem-oriented approach to practice"; JAMA, **238**:414-417 1977.

Tuttle M.S., Blois M.S., Erlbaum M.S., Nelson S.J., & Sherertz D.D.; "Toward a Biomedical Thesaurus: Building the Foundation of the UMLS"; Proceedings of The Twelfth Annual Symposium on Computer Applications in Medical Care, New York USA, Greenes, R.A. ed: Institute of Electrical and Electronics Engineers, pp. 191-195, 1988.

Tuttle M.S., Sherertz D.D., Erlbaum M.S., Olson N.E., & Nelson S.J.; "Implementing Meta-1: The First Version of the UMLS Metathesaurus"; Proceedings of The Thirteenth Annual Symposium on Computer Applications in Medical Care, Washington, DC USA, Kingsland, L.C. III ed: Institute of Electrical and Electronics Engineers, pp. 483-487, 1989.

Tuttle M.S., Sherertz D.D., Olson N.E., Erlbaum M.S., Sperzel W.D., Fuller L.F., & Nelson S.J.; "Using Meta-1--The First Version of the UMLS Metathesaurus"; Proceedings of The Fourteenth Annual Symposium on Computer Applications in Medical Care, Los Alamitos, California USA, Miller, R.A. ed: Institute of Electrical and Electronics Engineers, pp. 161-165, 1990.

United States Health Care Financing Administration; International Classification of Diseases, 9th Revision, Clinical Modification; Fourth Edition, Washington, D.C. 1991.

van Egmond J., Decoussemaker L., Wieme R.J., & Bossaert W.; "Implementation of an Information System For Patient Care in the University Hospital of Gent"; Proceedings of the IFIP Working Conference on Information Systems For Patient Care: Review, Analysis and

Evaluation, Amsterdam, The Netherlands, van Egmond J., de Vries Robbe' P.F., & Levy A.H. ed, pp. 191-203, 1976.

Verplank W.L.; "Graphics in Human-Computer Communication: Principles of Graphical User-Interface Design"; Proceedings of the IFIP-IMIA Second Stockholm Conference on Communication in Health Care, Stockholm, Sweden, Peterson H.E. & Schneider W. ed, pp. 113-130, 1985.

Warner H.A.; "The Brief Encounter Review: An Investigation in Graphic Computer-Based Medical Record Reporting Systems"; Report No. UIUCDCS-R-74-648, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL May 1974.

Warner H.R., Olmsted C.M., Rutherford B.D.; "HELP -- A program for medical decision making"; Comput Biomed Res, **5**:65-74, 1972.

WARP; "Wisconsin Ambulatory Review Project"; WARP Draft Directory, University of Wisconsin, August 1990.

Weed L.L.; "Medical records that guide and teach"; New England Journal of Medicine, **278**:593-657 1968.

Weed L.L. & Hertzberg R.Y.; Dr. Weed's Coupler™; PKC Corporation, South Burlington, VT 1986.

Wheeler R. & McConnell D.; "A Revolutionary Approach to Medicine: Anatomy-Based Computer Systems Provide Easy Physician/ Patient Input"; 3rd Ann. Int. Conf. Comput. Med. Rec., Chicago, Institute for Medical Record Economics, Inc., 1987.

Williams B.T.; Computer Aids to Clinical Decisions; CRC Press, Inc., Boca Raton, Florida, Vol 1 1982.

Williams B.T., Johnson R. & Chen T.T.; "PLATO-Based Medical Information Systems-Overview"; Proceedings of the 1st Conference on Medical Info. Systems, Urbana, Illinois, pp. 145-149, October 17-18, 1974.

Williams B.T., Chen T.T., Schultz D.F., & Johnson R.; "A Terminal-Oriented Clinical Record System"; Computer Graphics, **9**(1):115-135, 1975.

Williams B.T., Chen T.T., Schultz D.F., Moll J.D., Flood J.R. & Elston J.; "PLATO-Based Medical Information System--Variable Keyboards"; Proceedings of the 2nd Conference on Medical Info. Systems, Urbana, Illinois, pp. 56-61, September 24-25, 1975.

Williams B.T., Chen T.T., Johnson R., & Schultz D.F.; "A Terminal-Orientated Clinical Record System" in Perkins W.J., (ed), Biomedical Computing; University Park Press, Chpt 36, pp. 311-321, 1976.

Williams B.T., Foote C.F., Galassie C., Schaeffer R.C.; "Augmented Physician Interactive Medical Record"; Medinfo 89: Proceedings of the 6th Conference on Medical Informatics, Beijing-Singapore, Barber B, Cao D, Qin D, Wagner G ed, North-Holland, Amsterdam, pp. 779-783, 1989.

Williams B.T., Yoder J.W. & Schultz D.F.; "The MEDIGATE System for Direct Entry of Physical Findings by the Examiner: User Interface Issues"; Proceedings of The International Health Evaluation Association Annual Symposium on The Art and Science of Preventive Medicine, La Jolla, California USA, Felitti, V.J. ed, pp. 106-114, 1990.

Winston P. H.; Artificial Intelligence; Second Edition by Addison-Wesley Publishing Company, Inc., MA 1984.

Wolfe F., & Fries J.F.; "ARAMIS today: Moving toward internationally distributed databank systems for follow-up studies"; Clinical Rheumatology, **6**(Suppl. No. 2):93-102, 1987.

Yoder R.D., Evans M.R. & Sweeney J.W.; "Processing Pictures With Computers"; JAMA, **200**(13):119-123, June 26, 1967.

Zanden B.V. & Myers B.A.; "A Constraints Primer"; Computer pp.74-75, November 1990.

Appendix

Multiple Selection Popup Menus for the MEDIGATE System

(V 2.03)

"Requirement Analysis"

University Park Pathology Associates

Prepared by Erik P. Littell & Joseph W. Yoder

Table of Contents

Part 1: What is the Multiple Selection Popup Menu?

The Multiple Selection Popup Menu is a pop-up menu interface developed for the MEDIGATE System. This Popup Menu was developed primarily because the standard pop-up menu systems currently available do not provide the wide variety of functionality which are required of the MEDIGATE System. The original purpose was to allow the user the ability to open a pop-up menu and not have it disappear whenever the mouse button was let up. Now that it is complete, the user can call up sub-menus at the appropriate times which will allow him/her to select several items at once without having to open the pop-up menu each time. This will save the user a lot of time and frustration.

The current pull-down Menu XCMD provided for the Macintosh® limited the Semantic Network capabilities of the MEDIGATE System by allowing for only two levels of menus and limiting the terminology and text that can be used within the database. Also the database had many other limiting factors based upon how findings within the same geographical region can have sub-menus. In other words if Mass and Tenderness had size as an attribute, and we wanted size to have a sub-menu, the old version of MEDIGATE database forced the sub-menu for size to be the same. Sometimes this is desirable, but often it led to problems when the sub-menu list was based on the finding that it was modifying.

Another limitation on the old pull-down Menu XCMD used for MEDIGATE was that the user may have to possibly select a finding many times if they wanted to modify something about that finding. For example, if there is a Mass that is Mildly Tender, Mobile and has an Irregular Shape, the user would have to carefully navigate the mouse many times to select the Mass ICON and use a pull-down menu that disappeared every time. This created many mouse clicks and much movement of the mouse to enter any meaningful attributes about a particular finding.

This new pop-up menu system does more then just stay on the screen until the user tells it that s/he is done. Since a new pop-up menu system was being created from the ground up anyway, several new features were added for both the user and the developers ease. Some of these features are: 1) mutually exclusive items and groups which will automatically deselect incompatible items when new ones are selected; 2) a simple “normalcy” check which returns a N,P, or ? based on the given information about the normalcy of the selected items; and 3) automatically generate text. The primary goal of this new pop-up system is to make the look and feel of the MEDIGATE System better by making the pop-up menus more intuitive, and easier to use.

Part 2: User Interface

The best feature of the pop-up system is that it makes attribute selection as easy as possible for the user. The MEDIGATE System's Knowledge Base has been developed around a Semantic Network. First, a specific instance of an object is created, such as a specific instance of a Mass say Mass_1. Up until this specific instance is created, Mass is an abstract finding which may have several possible attributes associated with it. Once it is a specific mass, attributes may be assigned to it such as Location, Shape, Consistency and Tenderness. These attributes are assigned to an instantiation of an object through the pop-up menus.

There are a couple of ways to pop-up the static menus. The user could click on a specific instance of an object for which the attributes need to be set or changed. Or, the menu can be opened by clicking and holding over the general name of an object that has pop-up menus available to it (see Figure 2.1 for an example of a pop-up menu and sub-menu). Once the pop-up menus are opened, they look similar to normal menus, but are different in several important ways. Perhaps the most important thing you'll notice is that the menu will stay open whether or not the mouse button is held down. This feature is one of the main reasons for creating this new interface. The user is free to click on and select or deselect as many items as s/he chooses. To check or un-check an item, the user simply has to put the mouse over the item, and click. Due to the fact that this is only a prototype version, and is completely written in SuperCard™ at this time, the reaction time is slow, and the user may have to click and hold down the button for a little while for his action to be registered; a quicker version with near instantaneous reaction time can be created with "C" or "Pascal" if deemed necessary. If the mouse button is being held down, no selection will be made until the mouse button is released, then the item which the mouse is over will be selected.

When using the menus, there are three special features that the user should be aware of. The first one of these is the only one which the user can see directly. Any item which ends with a '=' symbol when selected will call up a dialog box asking you to input data for that item (see Text generation in the Returned Values Section for more details on how '=' works). After an appropriate value has been entered, that value becomes a part of that item, and the menu will expand to accommodate it. At any later time, the information which you inserted can be changed or deleted by simple re-selecting that item. The next two features are not as visible, but are very similar. These cases have to do with items which cannot logically be selected at the same time. A simple example of this might be location: Left, Right, Center. If these three values were a sub-menu then it might not make sense for all three of these items to be selected at the same time. Therefore, the pop-up menu knows if one of these items is selected. Should

the user selects another item, then the currently selected one will be deselected. This is done to save the user the time and effort of selecting a new value, and deselecting the old one. There are two different degrees of items which are mutually exclusive. The highest degree is that there may be whole blocks of items which cannot have any items selected in any two of them at the same time. However, items within one of these blocks may or may not be mutually exclusive with each other. This brings us to the next level of mutually exclusive items, a block of items in which no two items may be selected at the same time within that block. The user need not worry about these last two special cases other than to be aware that items will be automatically deselected at times.

Some items may have a '>' to the right of them. This indicates a sub-menu is available for this item. When one of these items is selected a sub-menu is opened. Sub-menus like the main menu will stay open until the user is finished, the user closes them, or the user opens up another sub-menu at a level higher than it. Any item selected in a sub-menu will select all of the items above it that opened the menus down to it. By clicking to the left of the name of a sub-menu item quickly, you will select or deselect that sub-menu item without opening its sub-menu. If you select that item, you will select that item only, and none of its sub-items. If you deselect it, you will deselect it and all of it's sub-items that were selected.

The last couple of items on every menu are special items which are automatically added. One of these items is the "Done" item. The last item on the main level, and every sub-menu is "Done". By selecting this item, you tell the pop-up menu system that you are finished inputting data. All of the menus will be closed, and some information processing will be done to generate the output text and the N/P/? information (see the Returned Values Section for more technical details). The system you are using will take care of handling this information. It is only mentioned here so that the user will know what the computer is doing during the long delay after clicking on "Done".

On the main menu the second to last command is "Cancel". If this item is selected, all the menus are closed, and any changes that have been made are thrown away. This command is included in case the user make a mistake, or changes his/her mind about making any changes. The second to last command on sub-menus may or may not be included and is the command "Go back 1 sub-menu". This command allows the user to close a sub-menu without opening another at a higher level. Any sub-menus below the closed menu will also be closed. The reason for this is that sub-menus can be arbitrarily deep, and may start to clutter up the screen and cover each other. A convenient way of removing menus you no longer need is by selecting this item.

If the mouse is clicked outside of the menus, then a dialog box will open, asking the user if s/he wishes to Cancel, Throw away changes, or if s/he is Done. If the user clicks Cancel, then the *Pop-up Menu Program* will continue as if nothing had happened. If the user clicks on Throw away changes, it is just like clicking on the Cancel item on the main menu and the Pop-up Menu will disappear and return operation to the calling program. If the user clicks on Done, it is just like clicking on the Done item on the menus. Figure 2.1 on the next page is a sample diagram of a pop-up menu and sub-menu with several of the things which were mentioned in this section pointed out for your convenience.

This is all the information that the user needs to know to run the pop-up menus. The way in which text is generated, and the values of N/P/? are calculated are not necessary in order to use the menus. The following section provides technical information about such things for developers and the curious.

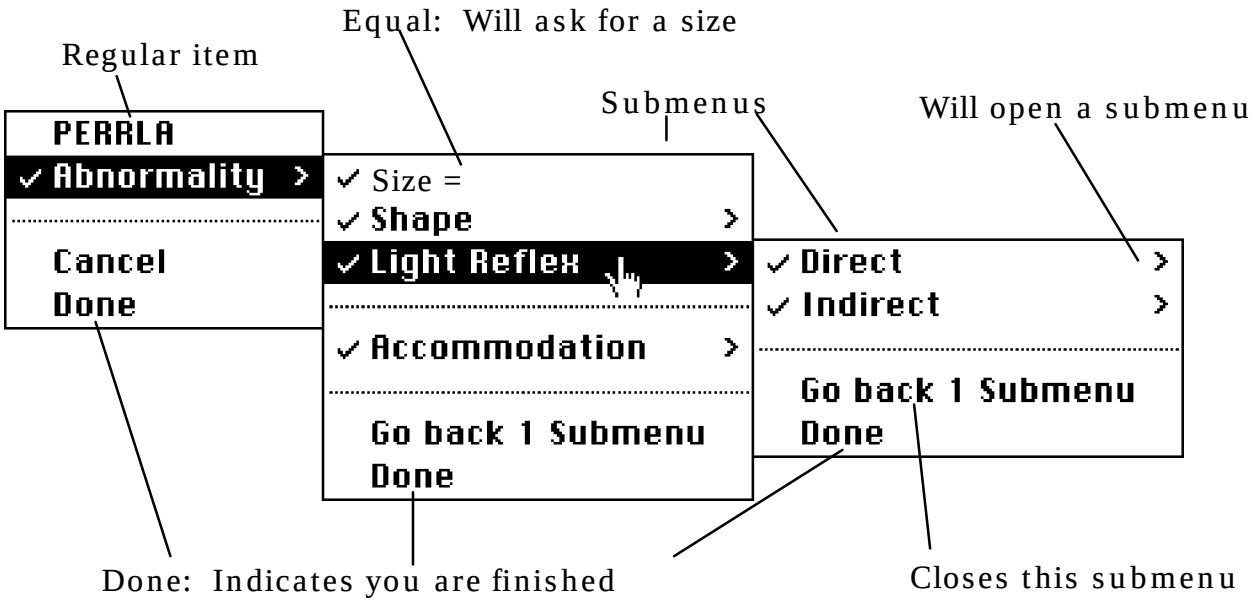


Figure 2.1: A sample pop-up menu and sub-menus

Part 3: Technical Information

The code which actually implements the pop-up menu system is currently a hybrid of normal SuperCard™ code and code which has been compiled using CompileIt!. This section is split into 4 parts, *System Requirements*, *Database Organization*, *Calling Parameters*, and *Returned Values*. This section is meant for the use of the developer who is trying to write an interface around this pop-up menu system. This section describes all of the details required to open and run the pop-up menus. The *System Requirements Subsection* gives an overview of the system required to run the pop-up menu system, and what parts of it need to be transferred into the program in which it is being used. The *Database Organization Subsection* describes how the pop-up menu program expects the information to be organized in the fields in the database from which it creates the menus. The *Calling Parameters Subsection* describes the parameters which must be passed to the calling function that opens the pop-up menus. And finally, the *Returned Values Subsection* describes what is returned, and all of the conditions that the pop-up system uses to calculate the return.

I. System Requirements

The Popup menu system is designed to be as simple, and compact as possible. The program runs in the SuperCard™ environment, so the basic system requirements are a Macintosh® system which SuperCard™ 1.5 will run on. This includes a Macintosh® with 2 megabytes of RAM, and System 6.0 or greater. To run the pop-up system, there are 2 icons, 7 X functions and commands, and 1 window which need to be copied into your project. The other items described below need to be supplied by you, and you will have to specify their location. The window which needs to be copied is the window named “Popup 1”, the two icons are “Up Arrow”, and “Down Arrow”, and the X functions and commands that need to be copied are “Put_Item”, “Get_Item”, “List_Less_Than”, “Its_In”, “Prepare_Sort_Menus”, Verify_Sort”, and “Sort_Line”. All of these items are needed, and if any of them are not present then the program may not work properly. The program will create a window named “Popup #” for every sub-menu which is created beyond the ones which already exist.

II. Database Organization

There is a lot of information which is needed in order to properly open the menus, and make them work. This section describes the format of the different databases which the pop-up function uses. There are three different sources of

information which are used, Menu fields, Special case fields, and the object script which holds the information on the previously selected items. Of the three, only the first two are described here. The reason for this is that the third is dealt with entirely by the program. All that the user or the developer needs to know is that the program keeps track of this data, and deletes anything which is wrong, or that it does not need. When the pop-up menu is called, the specific object/finding name is passed as a parameter. This object's script stores the information about selected items from the menu so **DO NOT** pass the name of an object which has any needed information in it's script already. Any invalid data in the script will be removed and any updated data about selected items will be returned and stored in the object's script.

Menu fields

When the pop-up menu system is called, three (3) different card and window names are passed to the system. The fields which contain the information for creating the menus must all be on these 3 cards. The 3 cards and windows are, in order of presentation on the menu: Global attributes, Specific attributes, and Second set of Specific attributes. On all menus and sub-menus, if there are any Global attributes, these will appear at the top of the menu. Any Specific attributes will appear next, and finally Secondary attributes will be placed at the end of the menu. A divider will automatically be placed between the groups if one is not already there.

Each group of items for menus and sub-menus is a field on the appropriate card. For the main menu, a field name will also be passed for each of the types. Information will be read from these fields, and the menu will be created from that information. Each item of the menu is kept as information in the fields which is separated by lines. The line contains information such as: Mutually exclusive groupings (both kinds), the item name, sub-menu fields if any, and if the item is N, P, or ?. Any blank lines in the field will be removed.

Each line is separated into 6 items separated by a '• ' (control-p). This was selected as the item delimiter because it is unlikely to be used in a name. The six items from left to right are:

Grouping indicators
Item name
Name of field for global items for Sub-menu
Name of field for specific items for Sub-menu
Name of field for secondary items for Sub-menu
N/P/?

If there is any information in the seventh item, it will be ignored, however it is not suggested that anything be put in the database other than what is suggested because future upgrades may use this space.

The Grouping indicator item consists of two numbers separated by a space. The first digit is used to indicate blocks of mutually exclusive items within groups. The acceptable values for this number are 0, 1, 2, 3. A zero indicates that the item is independent of other items in the group. A 1 indicates that the item is the first item of a block of mutually exclusive items. A 2 indicates that the item is in the middle of a block of mutually exclusive items. And finally a 3 indicates that the item is the last of a block of mutually exclusive items. Please note that a block of mutually exclusive items must be bracketed by a 1 and a 3 with all values in-between having a 2. If this is not done, unpredictable results will happen. The second number can be any value greater than or equal to zero. This number indicates that it is a member of group n<umber>, and that anything which is selected in this group is mutually exclusive with any group above or below it with number <> n. The search will search up and down, ignoring separators, until it finds a group with a number of zero. Zero indicates independent groups. Any data input in this section for a separator is ignored. Note that although it is possible to have a block of mutually exclusive items which spans across two blocks of mutually exclusive items, this is not recommended because unpredictable results may occur.

The Item name field may consist of any combinations of characters and letters except control-p, control-t, return, enter, delete, escape, and any other non printable characters. The display font is Chicago, so the acceptable characters are limited by the Chicago character set. If the last character of the name is an '=' then the menu system will take care of calling a dialog box to get the information that is missing. Items which have an equal sign as the last character cannot have sub-menus, so do not put anything into the sub-menu fields of the same line. A separator is indicated by a name of "(-" so do not use this as an item name unless you wish to make a separator.

In the next three fields are the names of fields on the appropriate cards which contain sub-menu items. If the indicated fields are empty or do not exist, then the item will be treated just like a normal item, and no sub-menu indicator will appear on the menu. Again all information in these fields are ignored if the item is a separator.

The last field is for use in determining if the selected item is N<ormal>, P<roblem> or ?<questionable>. If there is nothing in this spot, or if it is not "N", "P"

or "?" then it is automatically assumed to be "?". For more information on this field, see N/P/? under the Returned Values Section.

There is one last special feature. This feature allows the developer to specify items which execute special commands when they are clicked on. If this option is used, there are three things that need be noted. One is that the menu system does not quit, after you're program is finished running, execution will return to the menus, and they will continue. Secondly, the command that is specified must be a one line command. The command is executed as is using the do command. Thirdly, an item that has this option cannot be checked or unchecked by the menu system. If you decide to use this option, then you simply put the command that you want executed in the third item preceded by two '•'s (option-shift-v). Please note that this is not a supported option and is still under development. If you choose to use it, please be careful of possible interactions or other problems.

Figure 3.1 shows a menu and a sub-menu, and the fields that make up the items in the menus. The various sections of data that are described above can be seen, and hopefully the relationships better understood.

Special cases fields

Text generation allows for 3 special cases. One where if a certain word is the last word of an item, then it is replaced by the sub-items below it. These are called the "Replace_Words". If the first word of an item is in another list, then all the text generated from it's sub-items will be indented and forced onto the next line. These are called the "Force_Lines". And finally, some items don't look right when they appear in their normal order in the text generation, so these items can be specified in the final list. This list is a listing of all the lines which are to be printed in the reverse order of the normal text generation scheme. These are the called "Reverse_Lines".

All three of these lists are stored in fields, whose names and locations are passed to the calling function. All three fields must be on the same card. Replace_Words and Force_Lines must be a sorted lists of words, one word per line. The compare which is used is not case sensitive, but it is better if you match cases as much as possible. Reverse_Lines is different only in that an entire item is specified per line, and therefore may consist of more than one word. Again the compare is not case sensitive, but it is better to match the cases if possible. See the Text generation subsection of the Returned Values section for further details.

Figure 3.1: A sample pop-up menu and sub-menu with the appropriate data fields

III. Calling Parameters

The function which is called for a pop-up menu is “Popup_Menu”. The following are the 12 parameters that need to be passed to the “Popup_Menu” function, and are listed in order:

Finding_Params
Global_Params
Local_Params
Second_Params
Special_Params
Add_Function
Auto_Popup
Auto_Light
Go_Back_Show
Do_Action
Do_Dashes
Mouse_Position

Unfortunately, the calling function “Popup_Menu” has not been compiled yet, so the function cannot simply be called, but must be sent to card field “MenuItems” of window “Popup 1”.

Some of the parameters have several parts. When a parameter has several parts, these parts are not separated by the normal SuperCard™ item separator of “,”, but by the item separator of control-p.

Finding_Params specifies a card level object whose script contains the attributes which have already been selected for that object. The assumption is that an objects script will hold the list of attributes that have been assigned it. If the specified object does not exist, or if this parameter is left blank, then no findings will be read, or saved. Also no returned values will be generated. There are 4 items in Finding_Params. The first one specifies what type of object it is, for example cd, fld, graphic, etc. The second specifies the name of the object. The third is the name of the card that the object is on. And the fourth is the name of the window that the object is in.

Global_Params, Local_Params, and Second_Params specify the field, card, and window which the respective menu sections use. They each have three items which in order are Main menu field name, Card name, and Window name. If any of these are blank, or do not exist, they are just simply ignored.

Special_Params list information about the special cases fields. Text generation involves several possible special cases. To make things as general as possible, 3 cases were found which satisfied most conditions which were wanted. Therefor the first three parameters are the names of fields which hold the information for these cases. The next parameter is the name of the card that these three fields are one, and the last parameter is the name of the window that the fields are one. The three field names in order are Force_Lines, Replace_Words, and Reverse_Lines. For more information on these fields, see Database Organization, or Text generation under the topic of Returned Values. Note that unlike most other items if the window that the fields are supposed to exist in does not exist, no error is generated, and the program continues, but if the card or specified field does not exist an error may be generated.

Add_Function specifies the name of a function to call when the pop-up menu program is finished. The purpose of the Add_Function is to send a message of what to do (such as add the text to a specific field) when the Pop-up Menu program finishes. The Add_Function consists of two items the name of the function it self and the name of where the function is. If this parameter is empty, then it is assumed that there is no function to call when quitting, and it is ignored. If it is non-empty, then when the pop-up menu exits the function is called, no matter what conditions it exits on. The function is passed 3 parameters, which are the name of the object where the information was stored, a string indicating if the user canceled or clicked done, and N/P/?.

Auto_Popup is a flag which indicates if sub-menus should automatically be popped up when an item is highlighted, or if it should only be popped up if the item is selected. True indicates automatic pop-up, and is the default value if the passed value is empty or not true or false.

Auto_Light is a flag which indicates if lines should be highlighted below the mouse whether or not the mouse button is held down. False indicates that items should not be highlighted under the mouse unless the button is being held down, and is the default value if the passed value is empty or not true or false.

Go_Back_Show is a flag which indicates if the menu item “Go Back 1 Sub-menu” should be shown on sub-menus or not. True indicates that the item should be shown, and is the default value.

Do_Action is a flag which indicates whether or not to allow the selection of items. True indicates that the selection and deselection of items is allowed, and is the default value.

Do_Dashes is a flag which indicates if dashes should be appended to lines in the generated text. False indicates that they should not be, and is the default value. For more information about this feature, read Text generation under Returned Values.

Mouse_Position is the position where the top left corner of the pop-up menu to be. This parameter is not optional, and does not have a default value.

A sample call to the “Popup_Menu” function is provided below. This is meant as a sample of how you might call the function, and to show how the parameters all fit together. Note that "• " is control-p.

```
On mouseDown
    put globalLoc (clickLoc ()) into Mouse_Position
    put the number of wd "MediGate PopUp" into Window_Name
    put word 2 of Finding_Name & "• " & Menu_Name & "• " & the short name of this cd & "• " & the
        short name of this wd into Finding_Param
    put Menu_Name & "• " & Menu_Name & "• " & Window_Name into Local_Param
    put Menu_Name & "• " & "Global Menus" & "• " & Window_Name into Global_Param
    put "Force down first words" & "• " & "Replace last words" & "• " & "Reverse Lines" & "• " &
        "Special Fields" & "• " & Window_Name into Special_Param
    put "This_Add_Function• wd 3" into Addition_Func
    put true into Auto_Popup
    put false into Auto_Light
    put quote & Finding_Param & quote & "," & quote & Global_Param & quote & "," & quote &
        Local_Param & quote & "," & quote & "1• 1• 2" & quote & "," & quote & Special_Param & quote
        into Grouped_Parameters
    open inv wd "Popup 1"
    send "Popup_Menu" & Grouped_Parameters & "," & quote & Addition_Func & quote & "," & quote &
        Auto_Popup & quote & "," & quote & Auto_Light & quote & "," & quote & "true" & quote &
        ",true,true," & quote & Mouse_Position & quote to cd fld "MenuItems" of wd "Popup 1"
End mouseDown
```

IV. Returned Values

As explained earlier, return out of the pop-up menu is done by calling the passed function “Add_Function”. The first parameter passed to Add_Function is the name of the item in which the attribute data was stored in it’s script (usually the object which we are modifying the attributes for). The second string is either “Cancel” or “Done”. Each string indicates which of the respective ways the menu system quit operation. The last parameter is “P”, “N”, or “?”, indicating which of the three has precedence in the selected attributes. The final bit of returned data cannot be passed as a parameter and so is returned in the global variable “Return_Field”. The last two values need a bit more explaining on how they are generated, and are explained in the next two sections.

N/P/? generation

The primary purpose for the development of N/P/? generation was to help aid the MEDIGATE System in automatically generating if the physician/user found a problem while examining a patient. Each item in a menu or sub-menu has an associated value of N<ormal>, P<roblem>, or ? <questionable>. If this information is missing, then the item is assumed to be ?. Sub-menu items have the value of any items selected in it’s sub-menu. If there are no items selected in the sub-menu, then the sub-menu item will have the value it is assigned. If a sub-menu item has no sub items checked, and there is no value assigned to it, then it will be given a default value of “?”. The original data fields will NOT be changed, this is just what value will be returned for that item. If all of the selected items have a final value of “N” then “N” will be returned. If all of the selected items have a final value of “N” or “?” then “?” will be returned. That is “?” has precedence over “N”. “P” has the highest precedence of all three. If even one of the selected items has a value of “P”, then “P” will be returned.

Text generation

One of the most time consuming jobs of this pop-up menu system is the text generation. When the user quits normally through the “Done” command, text is generated from the selected commands which is meant to provide a quick look for the user at what he has selected in an outlined block of text. The called function “Add_Function” is expected to either ignore this text, or to place it into a field where the user can look upon it later and see the attributes of an object without opening up the menu again. It is important to note that this pop-up menu system was developed

specifically with MEDIGATE in mind, so that the text generation may or may not be useful outside of that application.

The basic form that the generated output text takes, without any special conditions, is all of the sub-menu items which lead down to a particular item are written backwards with the lowest item written first, and the highest level item written last at the end of the line. If a sub-menu item has more than one sub-item checked, then that item is treated as the bottom, and the text from it and above is generated and put on one line. Then, below that line and indented, the sub-items text are generated and placed in order. A couple of examples may help to understand this procedure. In figure 3.2, we can see that the item selected is a simple path of selected items going down 3 sub-menus.

Figure 3.2 - Text Generation Example 1

The text which would be generated from the above selection would be:

Blue Color Eyes

As another example, let us change the selection slightly, and instead select the two following items as noted in Figure 3.3:

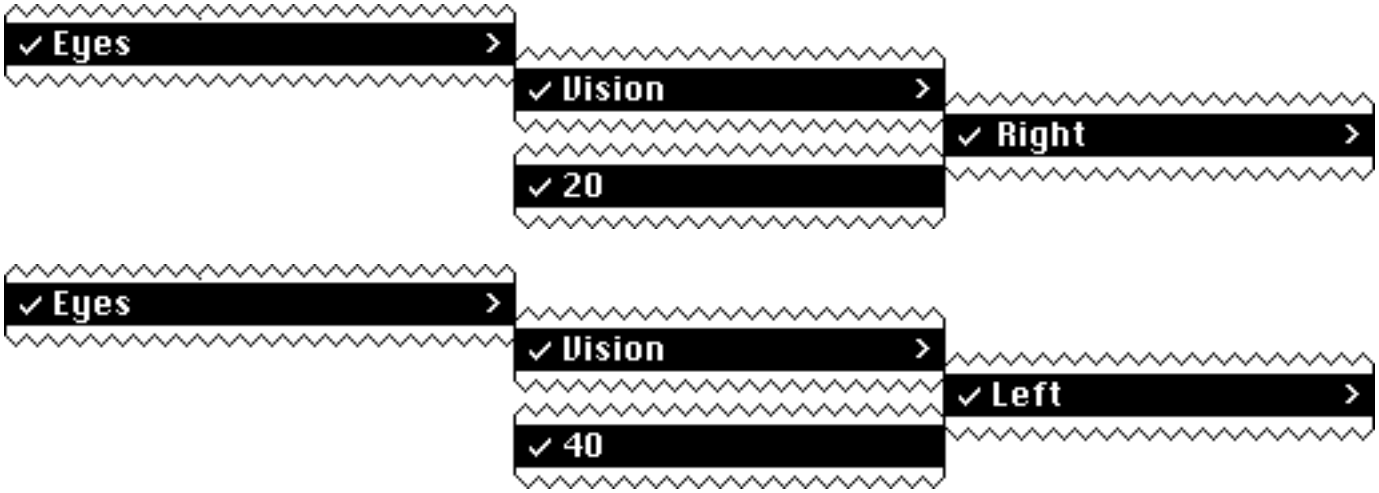


Figure 3.3 - Text Generation Example 2

The text which would be generated from these selections would be:

Vision Eyes

20 Right
40 Left

As a final example, consider if the all of the attributes from Text Generation Examples 1 and 2 were selected, then the following text would have been generated:

The text generated would be:

Eyes
Blue Color
Vision
20 Right
40 Left

As stated earlier, this is the basic form and there are 5 special conditions which change this basic configuration.

The first, and most basic condition is specified by the passed flag of "Do_Dashes". This command causes 2 things to be different from normal text generation. The first difference is that when an item is indented on the next line, a dash is appended to the end of the listing so far. The second difference is that items from the main menu that have exactly one sub item, will be listed first, with a dash after them, and then the text for the sub item after it on the same level.

For example, this is how the three previous text generation examples would look with this condition set to true:

Text Generation Example 1

Eyes - Blue Color

Text Generation Example 2

Eyes - Vision
20 Right
40 Left

Text Generation Example 3

Eyes -

Blue Color
Vision
20 Right
40 Left

The next special condition is the set of replace words. If the last word of an item is in the list of replace words, then if there is a sub-menu with only one item selected, that item will replace the specified word. The word is not replaced if it is on a sub-menu that has more than one item selected, in which case the item stays as it was, and text is generated as normal.

An example of this is shown in Figure 3.4 menu below:

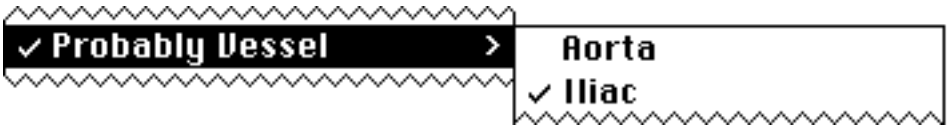


Figure 3.4 - Replace Example

Vessel is a replace word as described above, therefor the text which would be generated for this selection would be:

Probably Iliac

If the last character of a line is "=" then the text above that item is generated as normal, and that item is inserted and indented on the next line. The only operation which has precedence over this is the replace words. If the item directly above an item with the last character of "=" has as it's last word, a word that is to be replaced, then the word will be replaced on the same line by the item with the "=".

Figure 3.5 - Equal sub-item Example

Since Height is an item which has an "=" after it and Description is not a replace word it will be forced down to the next line, and indented. Therefore the text which will be generated will be:

Physical Description
Height = 55mm

Of course, with Do_Dashes set to true, this would generate this text:

Physical Description -

Height = 55mm

If “Description” had been a replace word then the text generated would have been:

Physical Height = 55mm

The next special condition is the set of words which if they appear as the first word of an item cause any sub items to be forced down to the next line. This condition is just like adding an extra “invisible” item to a sub-menu. This extra item makes this special condition redundant to a menu with several sub-items selected, or an equal item selected. Again, if the last word of an item is also a replace word, the replace word has precedence, and this condition is ignored in favor of that one.

Figure 3.6 - Force Words Example

Physical is a force down word, which will cause Unknown to appear on the next line indented. If Unknown is the only item selected, the generated text will be:

Physical Description
Unknown

Of course, with Do_Dashes set to true, this would generate this text:

Physical Description -
Unknown

Note that the text normally generated for this selection without any of these special conditions would have been:

Unknown Physical Description

The last special condition with the lowest precedence is the set of items whose order is reversed when generating the text. If an item is on the list of items whose order is to be reversed, and it is the last item, then it’s position will not be any different. If however, it has a sub-item, and none of the above conditions are true, then this item will appear before, rather than after it’s sub item. This is perhaps the most difficult item to understand, and therefore an example is in order. Consider the following selections as noted in figure 3.7.

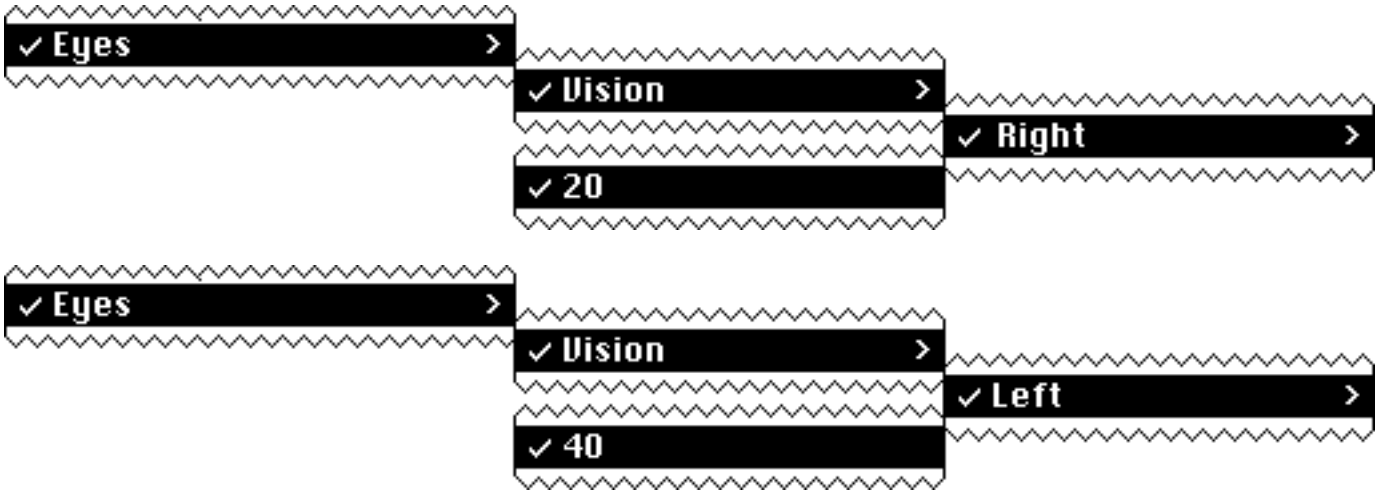


Figure 3.7 - Text Generation Example 2

If “Left” and “Right” were on the list of items to be reversed, then the text generation for this selection would be:

Vision Eyes
Right 20
Left 40

With Do-Dashes set to true, the text generated would be:

Eyes - Vision
Right 20
Left 40

This concludes a general description of the text generation technique. It may be difficult at first to understand all of the relationships between all of the special cases, or to understand the special cases themselves. The best way to see how text is generated is to play with it on some test data and observe how the text is actually generated.