1989

Interactive Simulations for Knowledge Acquisition

Sorel Bosan
College of William & Mary - Arts & Sciences

INTERACTIVE SIMULATIONS FOR KNOWLEDGE ACQUISITION

A Thesis

Presented To

The Faculty of the Department of Computer Science

The College of William and Mary in Virginia

In Partial Fulfillment

Of the Requirements for the Degree of

Master of Science

---

by

Sorel Bosan

1989

APPROVAL SHEET

This thesis is submitted in partial fulfillment of the requirements for the degree of

Master of Science

Sorel Bosan

Author

Approved, June 1989

Richard H. Prosl

R. H. Prosl, Chairman

Richard M. Bloch

R. M. Bloch

Stefan Feyock

S. Feyock

Keith Miller

K. W. Miller

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

First of all, I would like to extend my thanks to my advisor Dr. Richard Bloch. His advise, support, and patience has provided me with motivation throughout my thesis and will continue to do so for many more research projects to come. And for that I am forever grateful.

I would also like to thank Dr. Feyock, Dr. Miller, Dr. Park, and Dr. Prosl for their invaluable advice throughout the project. In addition, I thank my friend Audrey for providing me with enthusiasm, even through the hardest of times. Last but not least, I would like to thank my parents whose tireless support has been my inspiration for this thesis, as it also was for my entire education.

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

In constructing knowledge based systems which utilize perceptual expertise the major problem is that the knowledge acquisition techniques available are generally verbal and are inappropriate for communicating perceptual knowledge. This thesis tests a methodology for the acquisition of perceptual knowledge utilizing an interactive computer simulation. Issues for the construction of an appropriate simulation, the elicitation of knowledge with the use of the simulation and the construction of a knowledge base from the simulation data are discussed. The methods are presented in general and their implementation is demonstrated with the use of a simulation of the second hand of a clock. Results from 4 experts showed that the knowledge acquired by the interactive simulation and incorporated into an expert system produced judgments that were highly correlated with similar judgments made by each expert. The feasibility of utilizing interactive simulations to acquire perceptual knowledge from one or more experts and of translating that information into an effective and verifiable expert system is demonstrated.

INTERACTIVE SIMULATIONS FOR KNOWLEDGE ACQUISITION

CHAPTER I

INTRODUCTION

Perceptual knowledge, is an essential part of most forms of expertise. It is primarily required of real world tasks where physical properties must often be observed in order to make some judgement. For example, in the field of medicine, perceptual knowledge is used in the diagnosis of movement disorders, in rehabilitation, in dermatology, and in most other specialties ranging from general practice (listening to the heart beat) to radiology (analysis of medical images). In robotics, perceptual knowledge is needed for movement related decision making and planning as well as the recognition of objects. Finally, perceptual knowledge is needed for expert systems, and automated control systems, where decision making involves observing physical objects (such as engine repair, automated car driving etc.).

Although the importance of perceptual knowledge is clear, no widely accepted general technique exists for its acquisition and incorporation into expert systems. Knowledge acquisition is the process of acquiring knowledge from any source, including books, films, and observation of the task. The most commonly used sources, however, are experts in the

field of interest. Acquiring knowledge from experts is thus a subset of knowledge acquisition, and is referred to as knowledge elicitation.

Methods for eliciting knowledge have been studied intensively for the past two decades. Early methods almost exclusively relied on verbal interviews with experts [1]. Verbalizing expertise, regardless of the domain of knowledge, requires conscious awareness of information and procedures used in decision making. Often, this awareness is not present [2]. Eliciting knowledge using verbal interview techniques requires experts to translate their knowledge from the internal representation, to a new verbal representation which could be expressed clearly in an interview [3]. Such a translation is difficult and slow, even for verbal knowledge representations, and it causes knowledge acquisition to become the bottleneck in the construction of knowledge based systems [1,2].

The difficulty of translating expertise into verbal representations is especially problematic for perceptual knowledge. Verbal representations would be principally required for visual, auditory and somatosensory knowledge. Translating other sensory information, such as olfaction, would be even more problematic.

Current research in knowledge elicitation aims to speed up the transformation process by making it possible for experts to transfer their knowledge in a representation as close as possible to the internal representation they utilize in their expertise [3]. Research for automating the knowledge elicitation process by transferring some of the responsibilities and tasks of a knowledge engineer to a computer is also underway [4]. However, the majority of the knowledge domains explored have been related to the cognitive aspects of knowledge, with techniques which render them difficult and inconvenient for use in perceptual knowledge acquisition [1,3,5].

For the elicitation of perceptual knowledge, the use of interactive computer simulation provides a promising and straightforward technique with current technology. Interactive computer simulations are computer controlled sensory models of real life stimulus conditions. These can be modified by its user through the use of interactive controls. Until there are advances in computer controlled stimulators, however, graphical and auditory modalities are the easiest to simulate.

Interactive controls allow the user to provide digital or analog input to modify any or all of the features of a simulation. As the values associated with the interactive

controls are modified, the corresponding features of the simulation change as well. In this way, the form or the behavior of the simulation can be systematically changed to match some standard form or behavior. The values associated with these adjustments can be stored. An expert can be asked to modify the features of a simulation until it matches an internal standard the expert uses in making judgments about a particular form or behavior of a stimulus. The values of the expert's adjustments are representative of the expert's judgement of what a particular form or behavior of an object should be.

The experiment described in this thesis tests the use of an interactive computer simulation for perceptual knowledge elicitation. The experiment also tests a procedure for constructing optimized rule bases from the simulation data and integrating expertise from multiple experts.

For the tested interactive simulation and knowledge base development methodology to be useful, it must produce a knowledge base capable of decisions or judgments similar to those of the expert whose knowledge was elicited. Thus, the verification of the knowledge base by showing a strong correspondence between its decisions and the expert's would indicate that interactive simulation methodology is a suitable knowledge acquisition method, for developing expert systems.

Failure to verify the knowledge base and its decision behavior suggests that either the tested methodology or the particular application of the methodology is flawed.

CHAPTER II

METHODOLOGY

Several different approaches can be used in the development of computer simulations for knowledge acquisition. The method presented here consists of four stages (see fig. 1) including :

A) Construction of the interactive simulation.

B) Knowledge elicitation.

C) Construction of a knowledge base from the data obtained by the simulations.

D) Verification of the resulting expert system.

The following four sections present these stages both generally and in the specific context of a simple stimulus, the second hand of a clock and its associated tick sound.

A) The Simulation

The first step in the construction of the simulation is the identification of the domain of interest. This study uses the correct functioning of the second hand of a clock and its associated tick. There were several reasons for choosing this example. Expertise in second hand behavior was relatively available. In addition, it offered simple perceptual

judgments which had both visual and auditory components. The example was complicated enough, however, to include most of the issues that would be encountered with more complicated domains, such as the use of multiple dimensions and modalities. Finally, the simple audio-graphical simulation could be programmed with available hardware.

The next step requires identification of the features of the selected stimulus which will be simulated. A simulation need not contain all the features of the physical phenomena being modelled. Indeed, a simulation model by definition abstracts physical events. The features which vary and influence the behavior of a simulation are, for this study, called dimensions. The selection of dimensions determines the aspects of the stimulus situation which will be altered or judged. The features which do not change are selected to give context to the chosen dimensions. The choice of static features is not crucial, however, and any features, as long as they place the dimensions chosen in context, are appropriate.

A physical stimulus may have more variability associated with its form or behavior than is perceived or used in judgments. This is because only a limited number of dimensions can be observed and evaluated by a person at any given time [6]. As a result, the dimensions to be used in the

simulation may be a subset of all of the possible physical dimensions. Furthermore the choices must include the dimensions used in judgement behavior, although including additional dimensions would not impair the performance of the methodology.

The choice of dimensions also depends on the particular evaluation task. For example, the stimulus dimensions used to distinguish between birds as a function of flight patterns are obviously different from the dimensions used to differentiate birds as a function of song patterns.

For the clock example, in addition to wanting two perceptual modalities, a desire to limit the number of dimensions yielded four dimensions. They did not, however, cover the range of variability associated with a clock exhaustively. Instead, these dimensions were chosen to allow for the proper modification and control of the clock simulation, in the domain of interest.

Two of the four dimensions dealt exclusively with visual aspects of the movement of the second hand. One dimension used both visual and auditory aspects of the movement. The fourth dealt with the auditory modality exclusively.

The four dimensions used were :

i) Arc Length

Arc length visually determined, in degrees, the extent of the rotation of the second hand associated with each one of its discrete movements, or ticks.

ii) Tick interval

Tick interval was the interval between each movement or tick sound of the second hand, measured in milliseconds.

iii) Starting Position

Starting location allowed for the visual adjustment of the position of the second hand, enabling the second hand to be aligned with the numeral markings.

iv) Tick Delay

Tick delay determined the delay between the auditory tick sound and the arm movement in milliseconds. The tick sound was allowed to sound before or after a movement of the second hand.

The time related dimensions tick_interval and tick_delay were generated using the internal clock of the microcomputers. The rotational movement of the second hand of the clock, associated with length and starting position, was generated with the use of algorithms adapted from Stevens [7].

To allow expert adjustment of a simulation's characteristics the simulation can be made modifiable by associating an interactive control with each of its dimensions. Interactive controls allow experts to modify each dimension, and as a result, alter the overall form or behavior of the simulation. Expertise transfer occurs when the experts adjust the simulation dimensions to make them match their concept of a correctly working second hand and the values of the adjusted dimensions are recorded.

The interactive controls used in the clock example were designed to ensure a uniform modification interface for all dimensions. Each dimension was assigned a number. The dimension names and their corresponding numbers were displayed on a status line at the bottom of the screen (see fig 2). Each dimension could be modified, once selected by pressing its corresponding number on the keyboard and by using the '+' and '-' keys. Pressing the '+' key increased the value of the dimension at that moment by the value of the grain size. Pressing the '-' key had the opposite effect. The dimensions could be modified in any order, and any number of times. When all dimensions were adjusted by the expert to match his/her internal standard, pressing 'q' would signify the end of the modification. The values for the parameters at that point were written out to a database. Instructions describing this

operation were read to each expert (see appendix D).
The resulting simulation constructed for this investigation consisted of a circular clock face displayed on the screen with the locations of the twelve numeral positions clearly marked (see fig 2). A second hand of a clock was drawn as a line extending from the center of the clock to its inner perimeter. The second hand moved clockwise around the clock face in discrete steps, with each step accompanied by a tick sound of 1 msec. duration. The sound was generated by the internal speaker of the computer under program control.

The smallest possible distinguishable change, or the grain size associated with each interactive control should be chosen so as to allow adequate knowledge transfer. This grain size is a function of the decision task at hand. For example, in judging the differences between the weights of two objects, differences of a few grams can be perceived if the two objects weigh in the order of a few grams. However, the perceivable difference can only be in the order of kilograms if the object weights are in the order of kilograms. Therefore, the simulation should include the capability to change the simulation characteristics in adequately sized steps to reflect an expert's knowledge.

Furthermore, the range of modification for each interactive control should also be identified. The range must

capture the different forms or behavior of interest associated with the stimulus. As a result, a sufficiently large range should be provided which allows an expert to differentiate, with his/her expertise, all of the desirable forms or behavior associated with the stimulus. Thus, in the example of distinguishing bird song patterns, an auditory simulation must have sufficient adjustment range to allow experts to distinguish crows and hummingbirds from chickens.

Initially, for the clock example, the values for the modification range and the grain size were estimated. During the testing of the simulation, user feedback was used to change these initial values to make the modification of the simulation easier. In addition, the data obtained from these tests were also used to fine tune the grain size and modification range choices.

The audio-visual simulation was programmed in Turbo Pascal Version 5.0, on IBM-PC/AT class machines. A variety of graphic board/ video monitor combinations were used. These included a Hercules graphics card with monochrome display; EGA graphics card with multisynch achromatic display; and EGA graphics card with multisynch color display. The sound was generated using the 'sound' command built into Turbo Pascal.

As a final point in the construction of the simulation, it must be remembered that the performance of the expert

system constructed from the simulation data is the final test of whether the simulation was constructed appropriately. Problems in any of these areas can undermine the performance of a resulting expert system so that it fails to perform as the expert would.

B) Knowledge Elicitation

In this methodology, knowledge elicitation takes place entirely through the use of the interactive simulation. The experts are asked to adjust the dimensions of a simulation to make them match their internal concept of a particular form or behavior. This is repeated a number of times, each time storing the final adjustment values. This process is then carried out for other forms or behavior of interest.

For the clock example, the experts were instructed on the use of the simulation, and all the controls were explained (See Appendix E). Next, the experts were asked to modify the clock simulation until, in his or her judgement it was a correct representation of a second hand of a correctly working clock. This was the only behavior of interest. The experts were asked to repeat this process 10 times. Built into the program were 10 different starting positions for the dimensions. They were chosen to be equally split between the

two extreme ends of the ranges of individual dimensions so as to avoid any bias in the final results. The ten simulations were presented to the expert in sequence without intervention. The expert was given the option of rest between simulations. At the completion of each modification task, the final choice for each dimension was stored in a file. Thus a value base of ten data points for each dimension per expert was obtained.

A total of 10 experts were tested. The clock example was chosen to ensure that people would have needed no special education to become experts, making it simple to find experts for testing the methodology. As a result no special selection process was followed in selecting experts. Of these ten experts four never completed the testing process. The data collected from two were lost due to disk failure. The results of the remaining four are presented in the results section.

C) Construction of the Knowledge Base

Having obtained the values bases, a method is needed to translate them into a knowledge base. If the knowledge base is to consist of rules, then the expert's values for the dimensions must be translated into rules. If the knowledge base is to consist of frames, the translation would be to

frames (for a further discussion of what particular form of knowledge representation to use see appendices B and C). In this study, expert's settings on the four clock dimensions were translated into Prolog clauses, or rules.

Perceptual knowledge used in making judgments usually takes the form of pattern classifications. Each pattern being classified, or distinguished by the expert system, represents a possible outcome for the knowledge base. The process of converting an expert's dimension data into a knowledge base must provide both the appropriate decision paths as well as the necessary decision outcomes.

There are at least two basic ways to develop rules which provide the appropriate decision paths to the final decision outcomes needed. One follows the traditional knowledge engineering approach. This approach emulates the experts verbal description of how they would combine the various features, represented by dimension values, to derive the final classification behavior. This method is flawed. It expects an expert to use values derived from his simulation behavior to determine final classifications when, ordinarily, his expertise in determining classifications does not employ such values.

Another way to develop the rules relies on the knowledge

engineer's appropriate choice of dimensions and the naturally resulting method for the combination of dimensional information. If, for example, an interactive simulation was designed to acquire data for an expert system distinguishing different birds' songs, an expert could be requested to adjust the simulation values for several dimensions to match the songs of different birds. The expert would be asked to adjust the simulation to match an albatross call, a canary song etc. Since data is collected for each dimensions, for each bird, a rule structure naturally follows with a rule for each dimension, of the form:

```
Dimension_Name(Value, Result) :-
        Value <= upper_bound_for_Bird_1,
        Value >= lower_bound for_Bird_1,
        Result is bird1;
        Value <= upper_bound_for_Bird_2,
                        ...
        Value >= upper_bound_for_Bird_N,
        Value >= lower_bound_for_Bird_N,
        Result is birdn.
```

Value is the value associated with each dimension of the simulation. All rules have the same structure.

The combination of rules to make decisions can follow naturally as well. If it is assumed that the expert adjusts appropriate dimensions, then combining dimensions using the logical AND, so that :

```
Birdname(V1, V2, ... VN, Result) :-
        Dimension_Name1(V1, Result1),
                    ...
        Dimension_NameN(VN, ResultN),
        Result1 = Result2,
        Result2 = Result3,
              ...
        ResultN-1 = ResultN.
        Result is ResultN.
```

The use of one knowledge structure for all dimensions simplifies the construction of the knowledge base and aid work towards its automation (see Appendices B and C).

Each of the dimensional rules is used to classify a value associated with a particular dimension. As a result the rules serve as a form of pattern recognition method (see figure 3). The decision ranges are used as the means of pattern classification.

Each decision range could conceivably be constructed from a single data point obtained from the simulation. This could

be done by selecting the boundaries of the decision range to maximize the performance of the resulting knowledge base. Applying an arbitrary multiplicative constant could also provide a decision range from a single value. Use of a sufficient number of data points, however, ensures that the mean of these points will lie near the middle of the decision range. As a result, the boundary values are of nearly equal distance away from the mean in opposite direction. Furthermore, the standard deviation obtained from these data yields additional clues about the boundaries, simplifying the search for the appropriate decision range.

For the clock example, there was one behavior of interest, a correctly working clock. The decision range for a correctly working clock for each dimension was obtained by first calculating the mean of the ten data points associated with each dimension, and then calculating the standard deviation. Finally, the decision range was obtained using the calculations :

upper_bound = mean + (2 * sd)

lower_bound = mean - (2 * sd)

The choice of 2 * sd comes from the fact that if the distribution is Gaussian, 2 * sd would include 95% of the correctly working clock judgement.

One clause, or rule per dimension was constructed of the form :

```
correct_Dimension_Name( Value ) :-
        Value <= upper_bound_for_Dimension_Name,
        Value >= lower_bound_for_Dimension_Name.
```

where upper_bound and lower_bound were computed for each of the dimensions as described above. No 'result' argument was used to return the classification. Instead, the success of the rule signified the 'correctly working clock' decision for each dimension.

As a last step in the construction of the knowledge base, a knowledge structure (or structures) to combine the separate dimensional classification judgments to classify the overall stimulus must be constructed. For the clock example, the binary decisions of correctly working or not correctly working, for each dimension, were combined conjunctively to yield the final decision. This final decision was expressed with the rule :

```
correctly_working_second_hand( V1, V2, V3, V4) :-
        correct_tick_delay(V1),
        correct_tick_interval(V2),
```

correct_arc_length(V3),

correct_starting_position(V4).

where V1 .. V4 the values associated with the dimensions.

The goal behind the construction of this knowledge base is to match the judgement behavior of the expert as closely as possible. One way to improve this match is by adjusting the appropriate decision ranges. Although the choice of + or - 2 * standard deviation for calculating the decision range boundaries would probably yield acceptable results, they are not necessarily the best. An optimization was done to maximize the average correlation of the knowledge base decisions with those of the experts they were obtained from. The optimization was carried out by comparing the average correlations of the knowledge bases constructed using :

upper_bound = mean + (x * sd)

lower_bound = mean + (x * sd)

where

0 < x =< 2.5

It should be noted however, that this is by no means the only way to approach optimization.

The knowledge elicitation time with experts is relatively short for this methodology. The combination of multiple

experts to maximize system expertise is, thus, made possible by this method.

To test the use of multiple experts, all data points obtained from the experts were used in calculating the upper and lower boundaries for the same decision ranges. The performance of the combined expert system was compared to that of expert systems constructed from single experts. Furthermore, an optimal range, as described above, was found for the combined rule base.

D) Verification

The verification of the expert system constructed from the knowledge base obtained can be carried out by applying similar methods used for other expert systems. An expert, and the knowledge base constructed can be presented with a number of different stimuli. The correlation between the knowledge base judgments, and the expert's judgments must be high in order for the knowledge base to be considered valid.

Another way to approach verification would be to use the simulation as a means of supplying the test cases, rather than an actual physical stimulus. In order to be able to use the simulation in place of the object however, the simulation must be verified as a correct representation of the physical object.

The verification methodologies used for this study had a three fold purpose :

i) verifying judgments or decisions derived from each knowledge base as strongly related to that expert's own judgments.

ii) finding if the relationship between judgments derived from one expert's knowledge and other experts' judgments was significant.

iii) finding if knowledge bases developed by using multiple experts was more highly related to individual and group judgments than knowledge bases developed by any one expert.

For each point of interest, the verification was done by providing each expert used in knowledge elicitation with 20 pre determined simulations of the second hand. The expert was instructed to study each one carefully to determine whether all features of the second hand of the clock were representative of a correctly working clock (See appendix F for instructions). If so, they were instructed to respond 'yes', signifying that the simulation represented a correctly working second hand of a clock. Otherwise, they were asked to respond 'no'. Then, the twenty sets of dimensional values presented to the experts with the simulations were fed into the rule bases under study. The response of these rule bases

for each simulation were noted. Finally the phi correlation of the answers for the expert and the knowledge base judgments were computed.

The first of the above mentioned goals was obtained by computing the correlation between the judgement of an expert and that of the knowledge base constructed from his simulation results. For achieving the second goal, the correlation between each expert's judgments and the responses of knowledge bases, constructed from the other experts simulation results, were computed. Finally the phi correlations for each expert's answers and that of the combined rule base was calculated and compared with the correlations obtained for the first and second goals, as a means of obtaining the third goal.

CHAPTER III

RESULTS

For each expert the correlation between the judgments of the expert system based on his/her simulation data and his/her own judgments was calculated (Tables 2 and 3). The average correlation for the 95% Gaussian decision range was 0.708 and ranged for individuals from 0.577 to 0.811. When the decision range was optimized the average correlation increased to 0.821 and ranged for individuals from 0.655 to 1.000. In all cases, each expert's judgments of the second hand of a clock, and the judgments derived from the knowledge bases were highly and significantly correlated at or above the 95% confidence level. The validity of the interactive simulation methodology in eliciting individual expert's knowledge in an accurate and useful fashion is demonstrated by these consistently high correlations. Indeed, all experts' decision ranges for the arc length and starting position dimensions matched that of a real clock, and the decision range for the tick interval dimension included the true 1 second interval.

The correlation of the expert system's decisions based on the simulation data of one expert, with the judgments of

other experts were compared (Tables 3 and 4). For individual knowledge bases based on the 95% Gaussian decision range the average correlation between one knowledge base and the other experts' judgement ranged from 0.544 to 0.774. The average correlation between one expert's knowledge base judgments and other experts' judgments was 0.646. These correlations are somewhat lower than the correlations between an expert's judgments and the judgments of the knowledge base derived from that same expert. They are, however, still significant and show that the knowledge acquired is general knowledge usable by others, rather than idiosyncratic knowledge of little use to others.

A few of the correlations between one particular expert's knowledge base judgments and other experts' judgments were not significant. This does not however reflect on the quality of the knowledge acquisition methodology but suggests that the level of expertise for that expert was not the same level as the other experts.

Interactive simulation facilitates the knowledge acquisition process to the extent that it becomes quite possible to acquire knowledge from more than one expert with very little additional effort. This, in turn, makes possible the development of 'smarter' knowledge bases than would be generated by a single expert.

When the knowledge in the knowledge bases of the 4 experts were combined and correlated with the individual judgments of the 4 experts, the average correlation was 0.702. This is an increase from the 0.646 level obtained comparing the knowledge base decisions from one expert with other's behaviors. Thus it is possible, by combining knowledge base information, to improve performance of an expert system developed using interactive simulations.

Summarizing the results it can be seen that the knowledge base decisions correlated significantly with the experts' decisions they were acquired from. The correlations of these knowledge bases were slightly lower for other experts although they remained mostly significant. Using a combined knowledge base, on average, improved the predictive ability of the expert system for a range of experts in comparison to individual expert's knowledge bases. Finally, the high significance of the correlations, despite the small number of experts, trials and tests used suggests that the methodology is robust.

CHAPTER IV

DISCUSSION

A) Issues

In order for the simulation methodology presented in this paper to be useful for knowledge elicitation, a number of issues must be addressed. These issues will be discussed in the following two sections.

i) Issues Related to the Construction of Simulations

One issue is how to identify the dimensions of a stimulus to be used in the simulation. In the current study the rudimentary nature of the expertise assisted in making the choice of dimensions easier. For real world future applications such as diagnosis of movement disorders, the identification of dimensions would be a far more demanding task.

A variety of methods can be used for identifying the separate dimensions of a stimulus. For stimuli with a small number of state variables (varying features of a stimulus, which when combined describe the real world behavior of a stimulus completely and accurately) modeling tools can be used to identify the variables. This would involve constructing

mathematical models of the stimulus and require considerable familiarity with its functionality. Due to the small number of state variables, modelling should be a relatively straight forward task, however. Furthermore, it is quite conceivable that since the number of state variables is small all of them would be used by an expert in decision making. As a result, the state variables can be used as the dimensions.

For stimuli which possess a large number of state variables, however, this methodology would present a problem. Since only a few of these state variables can be used by an expert, using all the state variables as dimensions would introduce considerable redundancy. In addition, differentiating the state variables which are used to make expert judgments from those which are not can be quite difficult. For such stimulus situations, psychophysical scaling techniques may be appropriate. Of these techniques the application of multidimensional scaling, cluster analysis and Pathfinder networks to the elicitation of knowledge about levels of abstraction for a domain has been studied by Cooke and Mcdonald [8].

While modelling, psychophysical scaling, and other statistical techniques can be used to identify dimensions of a complex stimulus situation, even knowledgeable trial and error could be used successfully. Verification of the expert

system would disclose if expert behavior was adequately predicted by the knowledge base decisions. If dimensions were omitted which contributed significantly to expert judgments, verification would show poor correlations between expert and knowledge base judgments. If dimensions were included in the simulation which were not used by the expert in adjusting the simulation then they would not contribute to the correlation between knowledge base and expert. In either case, dimensions which produce significant correlations between knowledge base judgments and expert judgments are empirically valid no matter how they were identified.

Another issue is the identification of the grain size associated with each dimension. The choice of dimensions with well known properties eased the process of choosing the appropriate grain size for the time related dimensions for the current study. Furthermore, the resolution of the graphics boards used dictated the grain size for the visual dimensions. For future applications, however, the properties of the dimensions chosen will not necessarily be obvious. As a result, other approaches must be considered.

Psychophysical thresholding techniques present a promising approach [9]. These techniques cover a variety of simple and complicated stimuli, and should easily be convertible for use with the stimulus of interest. Knowledge

engineers may also choose to familiarize themselves with the stimulus and produce an estimate. One approach that might be considered, using the smallest possible step permitted by the simulation, may be appropriate when such a choice does not introduce considerable problems for designing a responsive and accurate interactive controls.

Yet another issue, identification of the range of modifiability for each dimension, is once again important. An improper choice can result with the inability of the expert to adjust the stimulus properly. For the clock example, the largest range permissible by the simulation implementation was used.

For future applications, unfortunately, no general use methods exist to aid with this task. As a result, the knowledge engineer may choose to familiarize himself/herself with the stimulus in order to make a reasonable estimate. However, in the absence of a reliable estimate, allowing as large a range as permitted by the computer simulation is a good idea. In any case, the correctness of the chosen ranges can be determined by the performance of the knowledge base constructed.

ii) Issues Related to the Knowledge Base Construction

One issue is the determination of the decision range for use in each dimension's knowledge structure. As mentioned previously, the distribution of the judgments may not necessarily be Gaussian. As a result, determining this distribution would identify the appropriate range. However, if the distribution cannot be identified, the proper range could be constructed by the use of other methods which would provide clues as to what the distribution should be.The combinations of the dimensional decisions would be done using their certainty factors.

The presence of separate knowledge structures for a number of dimensions raises an important issue, how to combine them in decision making. The important point to consider is that the result of the method used in combining these dimensions must predict the behavior of an expert accurately. It is quite conceivable that for different situations and stimuli, different methods should be used. For example, for simple objects with relatively independent dimensions taking the conjunction of the decisions made independently on each separate dimension may be enough as it was done for this study. For more complicated objects Bayesian statistics [10] or multi-dimensional psychophysical techniques [11] can be used. Both of these methods would require information about

the certainty of each dimension and would yield a certainty factor for the final decision as well.

Another issue relevant to the construction of knowledge bases, is whether the knowledge bases' reasoning can be explained. It should be emphasized that the need for explanations that can be used by people is entirely dependent on the application. Such explanations will be pointless if the experts are not aware of their own reasoning process. This lack of awareness may be especially true for large numbers of perceptual expertise as signified by the difficulty in using traditional knowledge acquisition methodologies for perceptual knowledge.

There are two important points to consider when looking at explanations. One is whether the knowledge base obtained can be explained at all. The second is whether these explanations match the reasoning used by human experts. Since a number of explanation schemes already exist for different types of expert system shells, they could easily be utilized in obtaining explanations. This solves the problem associated with the first point.

Coming up with an answer for the second point is a far more complicated task. In this case, choosing the dimensions and the combination method for the dimensions based on the final judgement performance is not enough. It requires the

reasoning of the knowledge base to be meaningful to people. Since the reasoning process used by the knowledge base will be completely determined by the dimensions and the method for combining them, if explanations are necessary, extra emphasis must be placed in choosing the dimensions and the combination methodology. This would be needed to ensure the construction of a knowledge base whose reasoning can be understood by its users.

B) Disadvantages of the Methodology

One disadvantage of this methodology is the difficulty associated with the construction of a simulation, in particular for complex little studied stimulus. This problem would be overcome if the issues addressed above are resolved. Until then, however, the construction of a simulation will remain as the major cost behind this methodology.

Another disadvantage is related to the data type requirement the methodology places on the input. As mentioned previously, for verification of the expert system data may be fed into the expert system directly from the stimulus or from the simulation. For obtaining data directly from the stimulus, appropriate instruments must exist for quantifying the dimensions used in the simulation. On the other hand, if the simulation is to be for the input data it must be

verified. For the simulation to be verified, once again, data corresponding to the dimensions of the simulation must be obtained directly from a stimulus. This data would be used to simulate that particular stimulus for comparison with the stimulus. So, as a result, dimensions used in the simulation must be quantifiable readily with the use of existing instrumentation.

C) Advantages of the Methodology

An interactive simulation methodology for knowledge elicitation, as described above, eliminates the need for any form of intervention from the knowledge engineer during knowledge elicitation. This, in turn can increase the efficiency and the reliability of the simulation technique relative to other methods.

Since verbalizations are reduced to a minimum, knowledge elicitation time is significantly reduced. Construction of simulations, especially for complex objects, however require considerable amount of time. The overall time for knowledge acquisition, therefore, depending on the object will improve relatively less. Since the time spent with the expert is the most costly portion of knowledge acquisition, however, the cost of knowledge acquisition should be reduced considerably. In addition, reduced knowledge elicitation time facilitates

the elicitation of knowledge from multiple experts, enabling the incorporation of multiple expertise into a single knowledge base which in turn can improve the performance of the expert system constructed.

Finally, the knowledge base/simulation pair could be used as a training tool. If a link is formed for passing values from the simulation to the expert system any adjustments made by a trainee could be passed to the expert system for evaluation. A reverse link would enable training through the playback of appropriate simulations as determined by the expert system.

CHAPTER V

CONCLUSION

Interactive computer simulations provide an effective methodology for perceptual knowledge acquisition. Interactive computer simulations, with the use of proper human computer interaction methods provide a novel approach for knowledge elicitation allowing it to be a more appealing and an easier process for the experts than repeated verbal interrogation.

Even the simple clock example used, however, presented difficulties. Many of these difficulties relate to the construction of real time interactive simulations. Construction of real time simulations can be a demanding task, especially for complex objects. Furthermore, research efforts are needed to solve difficulties associated with the determination of the variable features for use in the simulation and the construction of a knowledge base from the simulation data. These would include, in addition to the issues mentioned in the previous chapter, determination of the ideal number of experts to use for elicitation, and the appropriate number of data points to obtain from them.

The results of this paper clearly establish that the use interactive simulations is certainly feasible and it works.

Future use of computer simulations for complicated situations should provide solutions to many of the problems mentioned here.

The most desirable long term goal would be to develop formalisms and appropriate theories which could be used generally for all perceptual domains and stimuli. The result of this could be a general automated system with tools enabling the construction of a simulation of the object under study for use in knowledge elicitation, and constructing a knowledge base from the interactive simulations.

TABLES

TABLE 1

DIMENSIONS FOR THE CLOCK SIMULATION

Dimension	Range of Modifiability	Grain Size
Tick Interval	400 - 2000 ms.	25 ms.
Tick Delay	-400 - 400 ms.	5 ms.
Starting Position		0.5 deg.
Arc Length	0.5 - 45 deg.	0.5 deg.

TABLE 2

CORRELATION OF KNOWLEDGE BASE JUDGMENTS CONSTRUCTED USING 2*SD AS BOUNDARIES AND EXPERT JUDGMENTS

	Knowledge Bases Constructed from			
Judgement of	Subject1	Subject2	Subject3	Subject4
Subject1	0.811	1.000	0.638	0.704
Subject2	0.599	0.739	0.471	0.816
Subject3	0.734	0.503	0.577	0.302
Subject4	0.644	0.818	0.522	0.704

TABLE 3

CORRELATION OF OPTIMIZED KNOWLEDGE BASE JUDGMENTS AND EXPERT JUDGMENTS

	Knowledge Base Constructed from			
Judgement of	Subject1	Subject2	Subject3	Subject4
Subject1	1.000	0.818	0.724	0.707
Subject2	0.739	0.903	0.535	0.579
Subject3	0.503	0.302	0.655	0.000
Subject4	0.818	0.798	0.592	0.724

TABLE 4

CORRELATION OF COMBINED KNOWLEDGE BASE JUDGMENTS AND EXPERT JUDGMENTS

	Combined Knowledges Base with Boundaries	
Judgement of	2 * sd	Optimized
Subject1	0.811	1.000
Subject2	0.599	0.739
Subject3	0.734	0.503
Subject4	0.664	0.818

FIGURES

FIGURE1

FLOWCHART OF THE METHODOLOGY

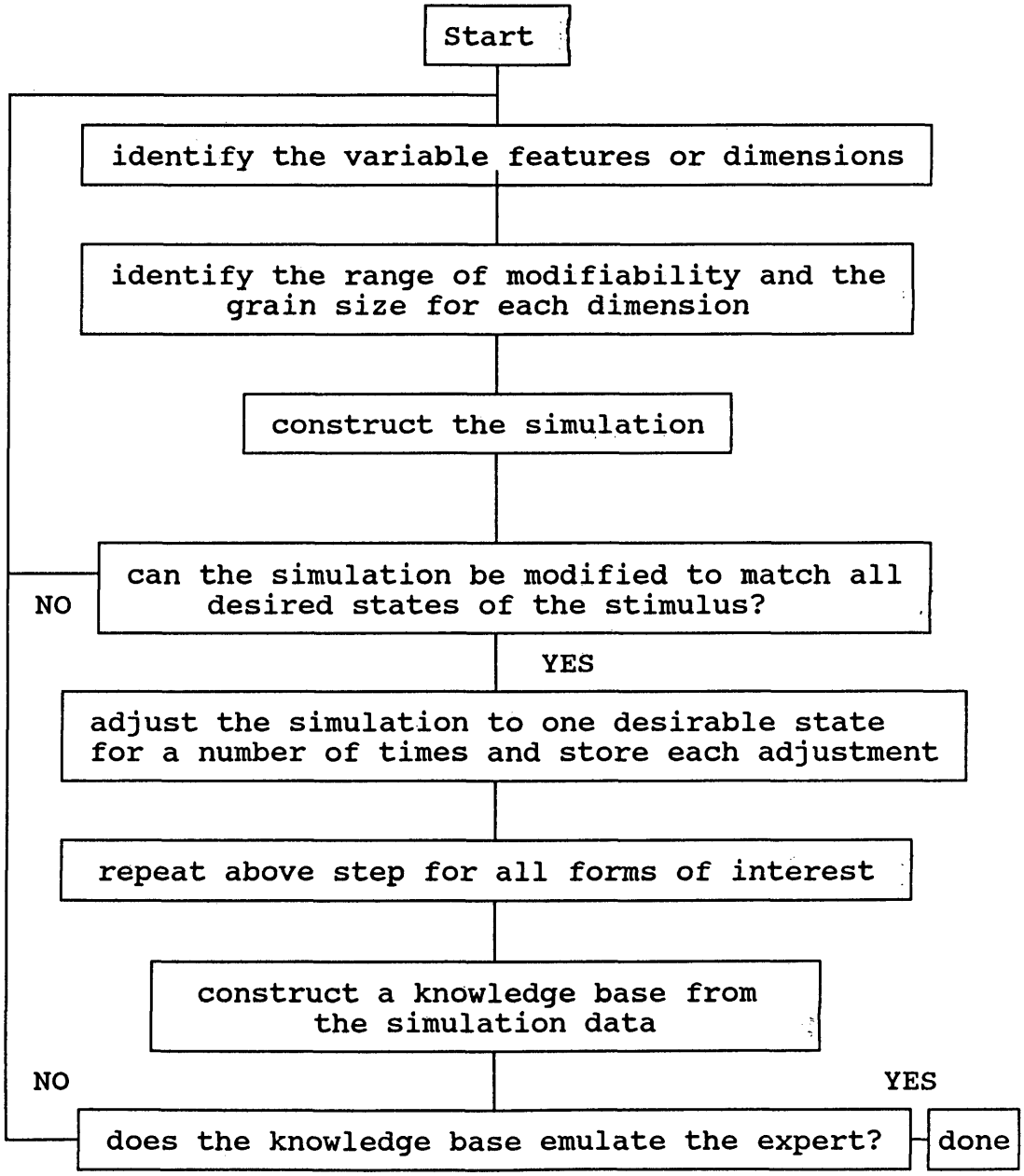


FIGURE 2

THE CLOCK SIMULATION

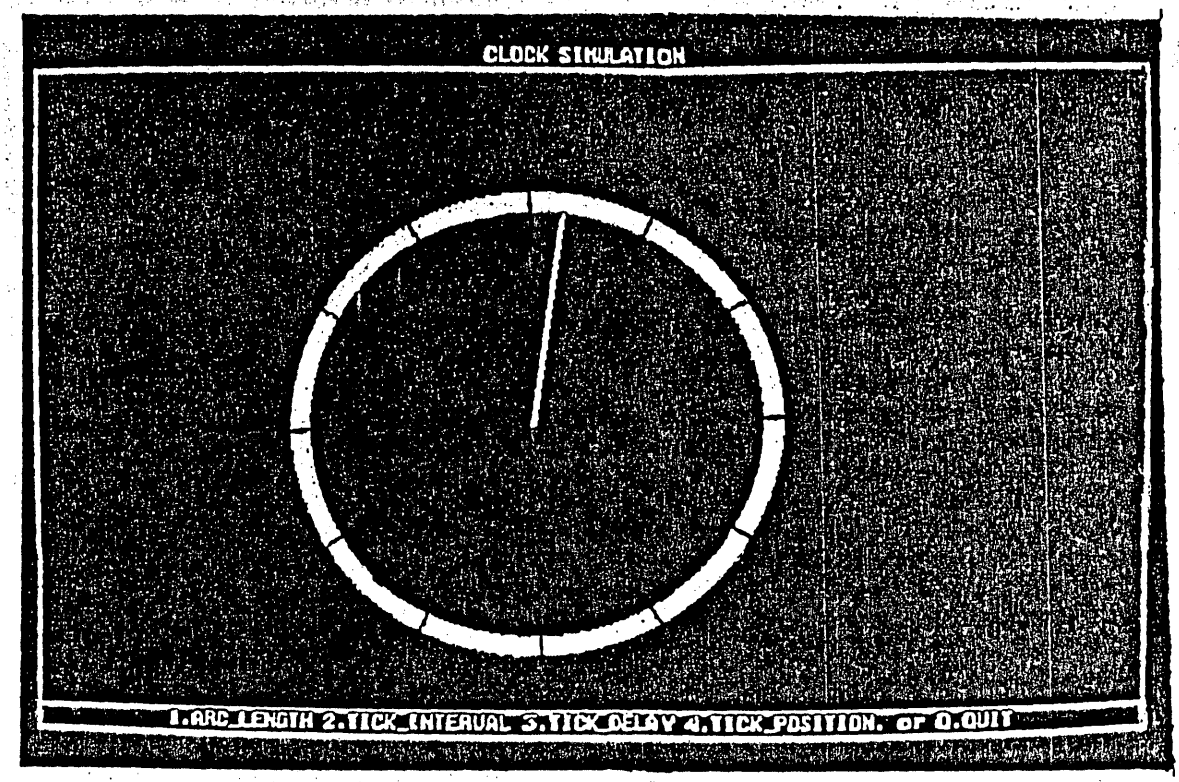
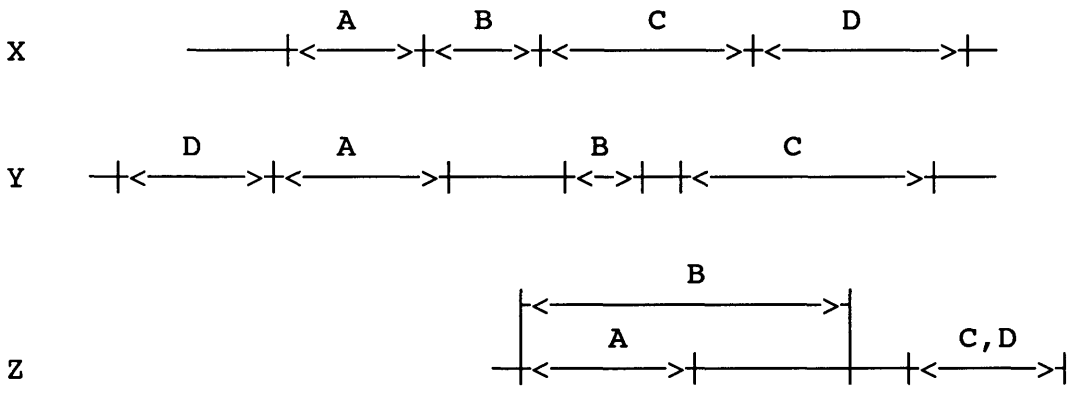


FIGURE 3

DECISION RANGES



——— : Numerical range of modifiability for a dimension

L
├<———>┤ : Numerical decision range for behavior or form L

APPENDIX A
A BRIEF OVERVIEW OF KNOWLEDGE ACQUISITION METHODS

Early methods for knowledge acquisition relied on verbal elicitation of knowledge from experts. This verbal elicitation took a variety of forms. The more popular ones included, verbal interviews where the knowledge engineer carries out a free flowing conversation with an expert [1], structured interviews [12] where the form and the flow of the interview questions are pre determined, and protocol analysis [12], where knowledge is collected and analyzed by having experts "think aloud" or introspect and verbalize.

The verbal methods, although fairly straight forward, were inefficient at extracting knowledge from experts. In particular, the lack of awareness of expertise, or its intuitive nature as well as the difficulty of verbalizing it caused knowledge acquisition to become the bottleneck in the construction of knowledge based systems [13].

In order to facilitate the transfer of intuitive, or subconscious knowledge from experts a variety of psychological methods have been utilized. Psychophysical scaling methods such as multidimensional scaling and cluster analysis [8], as well as Kelly's Personal Constructs theory [14] have been used

in organizing knowledge, investigating its underlying structure.

These methods rely on identifying grids of descriptive knowledge primitives and their connection patterns used by experts. Although successful in eliciting fairly detailed information, and readily adaptable for interactive knowledge acquisition, they are time consuming, and can be difficult to use by the experts.

Another approach towards facilitating the transfer of intuitive knowledge involves designing the knowledge base system to facilitate knowledge acquisition [3]. This is accomplished by enabling the knowledge representation primitives to match as closely as possible the task level primitives employed by the experts. This approach can be fairly difficult to apply, however, since the task level representation primitives may not be readily obtainable.

Another approach towards overcoming the problems associated with verbal knowledge acquisition methods has been tailoring the knowledge acquisition strategies to a particular task [4]. This involves classifying the task at hand as one of the many application tasks which include design, diagnosis, control etc. and then applying the appropriate knowledge acquisition tools for that application task. This approach,

however, assumes that well defined knowledge acquisition tools are available for application tasks which may not be the case. Furthermore, it assumes that a particular task may be classified as one or another form of application task whereas many tasks do not clearly fall within the boundaries of any one application task.

One recent knowledge acquisition strategy employs machine learning algorithms [15]. These algorithms include learning from examples[16, 17], model based learning [18] and inductive learning algorithms [19] amongst others. The learning algorithms are superior to other knowledge acquisition methodologies in that they facilitate the maintenance and upgrade of the knowledge based systems after they are constructed. However, they are very limited in scope, and can rarely be used as the only knowledge acquisition strategy (a notable exception to this may be Michie,s ID3 algorithm [16])

Knowledge acquisition is, currently, one of the most active research areas related to knowledge based systems. Most of the research, however, is done as a part of developing a knowledge based system for a particular task, and the knowledge acquisition tools developed for that task are later presented as alternatives to the existing tools. These tools reflect this approach in that they are usually restricted to

a particular domain of knowledge and if they are applied to other domains, they are slow and inefficient. As a result, verbal knowledge elicitation methods, although inaccurate and slow, are still the predominant knowledge acquisition methodologies used.

APPENDIX B

AUTOMATED RULE BASE GENERATION

Although it is not particularly difficult to manually construct rules from the data obtained from this simulation, in general, automating this procedure is a cost reducing step. To achieve automation, a link has to be formed between the expert system and the simulation. The link would be used for passing the boundary values associated with each dimension from the simulation to the expert system. This link need be no more complicated than the 'escape' clause developed by Feyock [20].

The escape clause provides a means of calling a pascal procedure from prolog itself. The pascal procedure in fact would be the simulation. It must be noted at this point that the direction of call (i.e. prolog to pascal or pascal to prolog) is not really important although the ability of pascal routines to call prolog to build the rule base would be more convenient.

Once the appropriate values are passed to the prolog procedure a combination of func, arg, univ, clause and assert statements could be used for constructing rules with these

values. The names of the dimensions can be supplied by the simulation program or prompted for by prolog. It is important to note that the constructed rules must be prolog clauses if prolog itself is to act as the inference engine. This would complicate the construction of rules since not all prolog implementations allow the use of reverse meanings of such clauses as univ which would be necessary for building the rule.

The presence of an automated rule base construction is important for the future work related to the automation of the interactive simulation methodology. It eliminates the need for the knowledge engineer completely from the knowledge acquisition stage to knowledge base construction. For the interactive simulation methodology to be completely automated however, the construction of the simulation must be automated as well.

APPENDIX C

KNOWLEDGE STRUCTURE CHOICE

One of the important points to consider in using the interactive simulation methodology is what type of knowledge representation to use. Since no special inferencing techniques are required from the final knowledge base constructed, practically any knowledge representation scheme desired could be used. If the knowledge acquired using the interactive simulation method is to be combined with knowledge obtained from other sources the integration can be achieved with relative ease. This is true even if the knowledge bases obtained using other methodologies place certain requirements on the representation scheme used. Such flexibility can be achieved since the knowledge acquired using the interactive simulation methodology is highly portable, requiring no special knowledge representation schemes.

It is conceivable, however, that the knowledge obtained using the interactive simulation method is to be the primary, or even the only source of knowledge in an expert system. For such a system it may be worthwhile to consider 3 issues before choosing the final form of knowledge representation.

First, the necessary inference engine for processing the knowledge structure of choice must be available.

Next, the reasoning format of the knowledge representation should be considered. This point is especially important for large knowledge bases. The closer the reasoning format of the knowledge representation is to that of humans, the easier it is for people to read and understand the knowledge base. For example, instead of the rule base that was used for the clock, an equivalent frame base could have been constructed :

```
Frame : Object
        ISA     :
  return  True

Frame : Clock
        ISA         : Object
        Dimension1  : Dimension1_Name = True
                  ..
        DimensionN  : DimensionN_Name = True

Frame : Dimension
        ISA     :
    prompt Dimension_Value
    return  Test_Correct(Dimension_Value)
```

```
Frame : Dimension_Name
        ISA         : Dimension
        Upper_Bound : dimension_name_upper_bound
        Lower_Bound : dimension_name_lower_bound

Test_Correct(Dimension_Value) = True
      if Dimensional_Value >= Lower_Bound
       & Dimensional_Value <= Upper_Bound
```

It should be observed, however, that for humans rules are a more natural way of expressing this knowledge. This, in turn, makes the reading and understanding of the rule base easier. This would be especially appreciated for larger knowledge bases.

As a final point, the suitability of the knowledge representation for automatic rule formation (as discussed in appendix B) may be considered. Again, using the frame example, it can be seen that a general frame structure can easily be constructed. Next, instances of this general frame can be used for constructing each dimension. As discussed in appendix B, however, construction of automated rule bases using prolog clauses can be considerably more complicated.

The final choice of the knowledge structure, as a result, will depend on the particular situation. Number of issues will be considered (such as how important is the automatic knowledge base construction and many others depending on the situation) and the final decision will be reached after carefully weighing the pros and cons of each knowledge representation. The availability of prolog was the main reason behind choosing prolog clauses as the form of knowledge representation for this thesis.

APPENDIX D

INSTRUCTIONS FOR CLOCK SIMULATION

We are studying how people judge the accuracy of the second hand of a clock. On the screen you can see a simulated clock face with a second hand which ticks as it moves. There are several features of this "clock" which need to be adjusted to make the clock appear to operate correctly. You select which feature to adjust by entering the number that corresponds to the feature identified under the clock. Feature 1 is the distance the second hand travels with each tick. If you enter the number '1' you can increase the distance travelled by pressing the '+' key, and decrease the distance travelled by pressing the '-' key.

DEMONSTRATE 1

Feature 2 is the time between ticks. To adjust the interval between ticks to equal one second, press the 2 key and use the '+' key to increase the interval or the '-' key to decrease the interval. Feel free to use the keys in any order and any number of times.

DEMONSTRATE 2

Feature 3 controls the relationship between the "tick" sound and the movement of the second hand. Enter a 3 to

adjust the feature. Press the '+' key to move the sound forward , toward happening before the movement, and press the '-' key to move the sound backward in relation to the movement.

DEMONSTRATE 3

Finally, feature 4 involves adjusting the location of the arm at the end of each second so that the second hand behaves as it would on a clock by pointing toward the numbers. Adjusting this feature does not affect any of the other features. Press the '+' key to move the location of the hand clockwise and the '-' key to move the location counter-clockwise.

DEMONSTRATE 4

Do you have any questions? We will do this procedure a total of 10 times. We want you to model a true second hand as closely as possible so take as much time as you need. We can take a break at any time you wish.

APPENDIX E

INSTRUCTIONS FOR CLOCK JUDGMENTS

We are going to show you 20 different simulations of a clock with a second hand. We want you to look at the distance travelled with each tick, the time each "second" takes, whether the "tick" sound corresponds correctly to the hand movement, and whether the second hand points accurately at the number locations to judge if, overall, each clock presented is correct or incorrect. If any of the features of the clock are not correct in your judgement, please judge the clock to be incorrect. If all the features of the clock are correct, please judge the clock as correct. Take as much time as you need, we are interested in accuracy, not speed. Do you have any questions? We can take a break any time you wish.

APPENDIX F

PROLOG RULEBASE FOR THE CLOCK EXAMPLE

```
correcly_working_second_hand( V1, V2, V3, V4) :-
          correct_tick_delay(V1),
          correct_tick_interval(V2),
          correct_arc_length(V3),
          correct_starting_position(V4).
correct_tick_delay( Value) :-
          Value <= upper_bound_for_tick_delay,
          Value >= lower_bound_for_tick_delay.
correct_tick_interval( Value ) :-
          Value <= upper_bound_for_tick_interval,
          Value >= lower_bound_for_tick_interval.
correct_arc_length( Value ) :-
          Value <= upper_bound_for_arc_length,
          Value >= lower_bound_for_arc_length.
correct_starting_position( Value ) :-
          Value <= upper_bound_for_starting_position,
          Value >= lower_bound_for_starting_position.
```

APPENDIX G

CODE FOR THE CLOCK SIMULATION

```
program Clock_simulation;

uses
  Crt, Dos, Graph, Printer, GraphSet;

{*****************************************************************************}
{--------------------------- GLOBAL DATA STRUCTS ---------------------------}
{*****************************************************************************}


const
  r      = 150;
  r_1    = 110;
  StartX = 320;
  StartY = 175;
  Min    = 0;
  Max    = 719;
  Sp_Max = 720;

Type
  Pos_Array = array [Min..Max] of integer;

Var
  CurPort    : ViewPortType;
  Pos_X      : Pos_Array;
  Pos_Y      : Pos_Array;
  Pos_XT     : Pos_Array;
  Pos_YT     : Pos_Array;
  Sp_Grain   : integer;
  Ps_Grain   : integer;
  Sd_Grain   : integer;
  Space_Step : integer;                    { Arc Length in 0.5 degrees }
  SoundDelay : integer;                    { Tick delay                }
  PauseTime  : integer;                    { Tick interval             }
```

```
  Ps_Tm_Ar   : array [0..4] of integer;        { Starting Positions       }
  Sp_St_AR   : array [0..4] of integer;
  Sd_Dl_Ar   : array [0..4] of integer;
  FileName   : string[12];                     { Output File }
  F          : Text;
  Toggle     : boolean;


{*****************************************************************************}
{---------------------- INITIALIZE DATA STRUCTURES -------------------------}
{*****************************************************************************}


Procedure Init_Structs;

Var
  i    : integer;
  X, Y : real;

  Procedure Rotate(deg : integer; var X, Y : real);

    Const
      radian =  0.00872664626;               { radian equiv. of 0.5 deg }

    Var
      X1, Y1 : real;
      angle  : real;

    Begin
      angle := deg * radian;
      X1 := x - StartX;
      Y1 := y - StartY;
      x := X1 * cos(angle) + y1 * sin(angle) + StartX;
      y := y1 * cos(angle) - x1 * sin(angle) + StartY;
    End;


Begin
  Space_step := 1;
  Pos_X[min] := StartX;
  Pos_Y[min] := StartY - r_1 + 4;
```

```
  X := Pos_X[min];
  Y := Pos_Y[min];
  For i := min + 1 to max do                    { Calculate  the inner   }
    begin                                       { perimiter of the clock }
      Rotate(Space_Step, X, Y);
      Pos_X[i] := StartX + round((StartX - x) * 1.29);
      Pos_Y[i] := round(Y);
     end;
   Pos_XT[min] := StartX;
   Pos_YT[min] := StartY - r_1 - 11;
   X := Pos_XT[min];
   Y := Pos_YT[min];
   For i := min + 1 to max do                    { Calculate the outer }
     begin                                       { perimiter           }
       Rotate(Space_Step, X, Y);
       Pos_XT[i] := StartX + round((StartX - x) * 1.29);
       Pos_YT[i] := round(Y);
     end;
   Sp_grain := 1;
   Ps_Grain := 25;
   Sd_Grain := 5;
   Ps_Tm_Ar[1] := 1500;
   Sd_Dl_Ar[1] := 100;
   Sp_St_Ar[1] := 20;
   Ps_Tm_Ar[2] := 1700;
   Sd_Dl_Ar[2] := -90;
   Sp_St_Ar[2] := 8;
   Ps_Tm_Ar[3] := 450;
   Sd_Dl_Ar[3] := -100;
   Sp_St_Ar[3] := 20;
   Ps_Tm_Ar[4] := 1400;
   Sd_Dl_Ar[4] := 70;
   Sp_St_Ar[4] := 2;
   Ps_Tm_Ar[0] := 550;
   Sd_Dl_Ar[0] := 150;
   Sp_St_Ar[0] := 18;
   Toggle := true;
   FileName := 'Test.dat';
   Assign(f, filename);
   Rewrite(f);
End;
```

```
{*************************************************************************}
{--------------------------- WELCOME SCREEN -----------------------------}
{*************************************************************************}


Procedure Initial_screen;

begin
  RestoreCrtMode;
  Writeln;
  Writeln;
  Writeln;
  Writeln('                          Welcome to ExperClock ');
  Writeln;
  Writeln('                               Version 3.2       ');
  Writeln('                                24/02/89');
  Writeln;
  writeln('                                   by            ');
  Writeln('                               Sorel Bosan       ');
  Writeln;
  Writeln('                         Dept. of Computer Science');
  Writeln('                         College of William & Mary');
  Writeln;
  Writeln;
  Writeln('                                 advisor ');
  Writeln('                            Dr. Richard Bloch');
  Writeln;
  Writeln('                         Dept. of Research and MIS');
  Writeln('                          Eastern State Hospital');
  Writeln;
end;  {initial Screen}


{*************************************************************************}
{--------------------------- WRITE RESULTS -----------------------------}
{*************************************************************************}
```

```
Procedure Results(Count, Yindex : integer);

Begin
  RestoreCrtMode;
  Writeln(f, ' For experiment ', Count : 3, ' the results are :');
  Writeln;
  writeln(f, ' Spacing     ==> ', space_step / 2 : 3:3, ' degrees');
  writeln(f, ' Timing      ==> ', Pausetime / 1000 : 3:3, ' seconds');
  writeln(f, ' Placement   ==> ', Yindex mod (space_step div 2) : 5, ' ticks off');
  writeln(f, ' Sound Sync  ==> ', SoundDelay/1000 : 3:3, ' seconds');
  writeln(f);
  Writeln(f, '            WITH');
  writeln(f);
  writeln(f, ' Space Grain ==> ', Sp_Grain * 0.25 : 3:3, ' degrees');
  writeln(f, ' Time Grain  ==> ', Ps_Grain : 5, ' miliseconds');
  writeln(f, ' Sound Grain ==> ', Sd_grain : 5, ' miliseconds');
  writeln(f);
  if toggle then
    begin
      Writeln(' For experiment ', Count : 3, ' the results are :');
      Writeln;
      writeln(' Spacing     ==> ', space_step / 2 : 3:3, ' degrees');
      writeln(' Timing      ==> ', (pausetime) / 1000 : 3:3, ' seconds');
      writeln(' Placement   ==> ',yindex mod (space_step div 2) : 5, ' ticks off');
      writeln(' Sound Sync  ==> ', SoundDelay/1000 : 3:3, ' seconds');
      writeln;
      Writeln('            WITH');
      writeln;
      writeln(' Space Grain ==> ', Sp_Grain * 0.25 : 3:3, ' degrees');
      writeln(' Time Grain  ==> ', Ps_Grain : 5, ' miliseconds');
      writeln(' Sound Grain ==> ', Sd_grain : 5, ' miliseconds');
      writeln;
      writeln;
    end;
End;


{*****************************************************************************}
{---------------------------- SET UP THE CLOCK ----------------------------}
{*****************************************************************************}
```

```
Procedure Set_Up_Clock;

  Var
    i      : integer;

begin
  SetGraphMode(GraphMode);
  ClearDevice;
  FullPort;


  { PaintScreen }
  MainWindow('CLOCK SIMULATION');
  StatusLine('1.ARC_LENGTH 2.TICK_INTERVAL 3.TICK_DELAY 4.TICK_POSITION. or Q.QUIT');
  GetViewSettings(CurPort);

  { Initialise Clock }

  SetColor(15);
  Circle(StartX, StartY, r + 5);                { Draw  Clock }
  Circle(StartX, StartY, r - 5);
  SetFillStyle(SolidFill, 15);
  FloodFill(StartX, StartY +  r_1 + 5  , 15);
  SetColor(0);
  i := min;
  repeat                                        { Mark numeral positions }
      setColor(0);
      Line(Pos_X[i], Pos_Y[i], Pos_XT[i], Pos_YT[i]);
      i := i + 60;
  until i = Sp_Max;

end;  {Set Up Clock}


{*****************************************************************************}
{------------------------------ SET UP MENU --------------------------------}
{*****************************************************************************}
```

```
Procedure Set_Up;
Var
  Quit : boolean;
  ch   : char;
  no   : integer;

Begin
  Quit := false;
  While not quit do
    begin
      clrscr;
      GotoXY(30, 4);
      writeln('1.  Adjust  Arc Length Grain');
      GotoXY(30, 6);
      writeln('2.  Adjust  Tick Interval Grain');
      GotoXY(30, 8);
      writeln('3.  Adjust  Sound Delay Grain');
      GotoXY(30, 10);
      writeln('4.  Adjust  Starting Tick Interval');
      GotoXY(30, 12);
      writeln('5.  Adjust  Starting Sound Delay ');
      GotoXY(30, 14);
      writeln('6.  Adjust  Starting Arc_length');
      GotoXY(30, 16);
      writeln('7.  Change Output File Name');
      GotoXY(30, 18);
      Writeln('8.  Result Display   Toggle = "' , toggle, '"');
      GotoXY(30, 20);
      Writeln('9.  Reset Clock to Correct Values ');
      GotoXY(30, 22);
      writeln('Q.  Quit');
      GotoXY(0, 28);
      write('Please Make A Choice ==> ');
      readln(ch);
      clrscr;
      Case Ch of
        '1' : begin
                writeln('Current Arc Length Grain is ', Sp_Grain  :6);
                write('Enter New Arc Length Grain ==> ');
                Readln(no);
```

```
        Sp_Grain := no;
      end;
'2' : begin
        writeln('Current Tick Interval Grain is ', Ps_Grain  :6);
        write('Enter New Tick Interval Grain ==> ');
        Readln(no);
        Ps_Grain := no;
      end;
'3' : begin
        writeln('Current sound delay Grain is ', Sd_Grain  :6);
        write('Enter New Sound Delay Grain ==> ');
        Readln(no);
        Sd_Grain := no;
      end;
'4' : begin
        write('Enter Starting Tick Interval ==> ');
        Readln(no);
        PauseTime := no;
      end;
'5' : begin
        write('Enter Starting Sound Delay ==> ');
        Readln(no);
        SoundDelay := no;
      end;
'6' : begin
        write('Enter Starting Arc Length ==> ');
        Readln(no);
        Space_Step := no;
      end;
'7' : begin
        close(f);
        writeln('Current Data File Name is "', Filename, '"');
        write('Please Enter Data File Name ==> ');
        ReadLn(FileName);
        Assign(f, FileName);
        Rewrite(f);
      end;
'8' : Toggle := not Toggle;
'9' : begin
        PauseTime  := 1000;
        SoundDelay := 0;
```

```
              Space_Step := 12;
             end;
       'Q', 'q' : quit := True;
      else begin
             writeln(' Incorrect Choice');
             WaitToGo;
           end;
     end;  {case}
   end { while}
end;



{*****************************************************************************}
{----------------------- ANIMATE USING LINE DRAWING ----------------------}
{*****************************************************************************}


Procedure linedraw;
{ Demonstrate Line Animation }


var
  Ch         : Char;
  key        : integer;
  Msg        : String[11];
  XIndex     : integer;
  X_Next     : integer;
  Yindex     : integer;
  Y_Next     : integer;
  I          : integer;
  quit       : boolean;
  X, Y       : real;
  Finish     : boolean;
  Time       : longint;
  TrialCount : integer;
  count      : integer;

Begin

  TrialCount := 1;
```

```
Count := 0;
finish := false;
while not finish do
  begin

    PauseTime  := Ps_Tm_Ar[count];              { Set Starting Values }
    SoundDelay := Sd_Dl_Ar[count];
    Space_Step := Sp_St_Ar[count];
    count := (count + 1) mod 5;

    TextBackground(0);
    clrscr;
    write('Would You Like to Use the Set Up Menu ==> ');
    readln(ch);
    if (ch = 'y') or (ch = 'Y') then Set_up;
    Set_up_Clock;
    XIndex := min;
    Yindex := min;
    Y_Next := min + Space_Step;
    X_Next := min + Space_Step;
    setColor(15);
    Line(StartX, StartY, Pos_X[Xindex], Pos_Y[Yindex]);
    Delay(pauseTime);
    quit := false;
    Time := PauseTime;
    key := 2;


{ Move the arm around }

    repeat

       If SoundDelay >= 0 then            { Tick delay }
         begin
           Sound(440);
           Delay(1);
           NoSound;
           Delay(SoundDelay);
           SetColor(0);
           Line(StartX, StartY, Pos_X[Xindex], Pos_Y[Yindex]);
           SetColor(15);
```

```
    Line(StartX, StartY, Pos_X[X_Next], Pos_Y[Y_Next]);
     end
 else
  begin
    SetColor(0);
    Line(StartX, StartY, Pos_X[Xindex], Pos_Y[Yindex]);
    SetColor(15);
    Line(StartX, StartY, Pos_X[X_Next], Pos_Y[Y_Next]);
    Delay(Abs(SoundDelay));
    Sound(440);
    Delay(1);
    NoSound;
  end;
Yindex := Y_Next;
Y_Next := (Yindex + Space_step) mod Sp_Max;
Xindex := X_Next;
X_Next := (Xindex + Space_step) mod Sp_Max;


Time := Time - 7 - ABS(SoundDelay);       { Tick interval }

repeat
  If KeyPressed  Then
    begin
      Ch := ReadKey;                      { Update response }
      case ch of
       '1' : key := 1;
       '2' : key := 2;
       '3' : key := 3;
       '4' : key := 4;
       '+' : case key of
               1 : if (space_Step + Sp_Grain) <= 180
                     then space_Step := space_Step + Sp_Grain;
               2 : if (PauseTime + Ps_Grain) <= 2000
                     then PauseTime := PauseTime + Ps_Grain;
               3 : if (SoundDelay + Sd_Grain) <= 200
                     then SoundDelay := SoundDelay + Sd_Grain;
               4 : begin
                     SetColor(0);
                     Line(StartX, StartY, Pos_X[Xindex], Pos_Y[Yindex]);
                     yindex := (yIndex + 1) mod Sp_Max;
```

```
                        xindex := (xindex + 1) mod Sp_Max;
                        y_next := (y_next + 1) mod Sp_Max;
                        x_next := (x_next + 1) mod Sp_Max;
                        SetColor(15);
                        Line(StartX, StartY, Pos_X[Xindex], Pos_Y[Yindex]);
                      end;
                end;
          '-' : case key of
                  1 : if (space_Step - Sp_Grain) >= 1
                        then space_Step := space_Step - Sp_Grain;
                  2 : if (pauseTime - Ps_Grain) >= 400
                        then PauseTime := PauseTime - Ps_Grain;
                  3 : if (SoundDelay - Sd_Grain) >= - 200
                        then SoundDelay := SoundDelay - Sd_Grain;
                  4 : begin
                        SetColor(0);
                        Line(StartX, StartY, Pos_X[Xindex], Pos_Y[Yindex]);
                        yindex := (yIndex - 1) mod Sp_Max;
                        xindex := (xindex - 1) mod Sp_Max;
                        y_next := (y_next - 1) mod Sp_Max;
                        x_next := (x_next - 1) mod Sp_Max;
                        SetColor(15);
                        Line(StartX, StartY, Pos_X[Xindex], Pos_Y[Yindex]);
                      end;
                end;

            'q' : Quit := True;
            else ;
          end;
        end;  {if keypressed}

        Delay(100);
        Time := Time - 100;
    until Time < 100;

    Delay(Time - 1);
    Time := PauseTime;

until quit;
RestoreCrtMode;
Results(TrialCount, Yindex);
```

```
    Write('Would you like to quit (Y) ==>  ');
    Readln(ch);
    if (ch = 'y') or (ch = 'Y') then finish := true;
    TrialCount := TrialCount + 1;
   end;
  close(f);
  {Prepare For Exit}
end; { PutImagePlay }


{ program body }

begin
  Initial_Screen;
  Init_Structs;
  Linedraw;
end.
```

APPENDIX H
CODE FOR GRAPHICAL SETUP

```
Unit GraphSet;

{*************************************************************************}

Interface

uses
  Crt, Dos, Graph;


{*************************************************************************}
{----------------------  GLOBAL  CONSTANTS -------------------------------}
{*************************************************************************}

const
  { The names of the various device drivers supported }
  DriverNames : array[0..10] of string[8] =
  ('Detect', 'CGA', 'MCGA', 'EGA', 'EGA64', 'EGAMono',
   'RESERVED', 'HercMono', 'ATT400', 'VGA', 'PC3270');

  { The five fonts available }
  Fonts : array[0..4] of string[13] =
  ('DefaultFont', 'TriplexFont', 'SmallFont', 'SansSerifFont', 'GothicFont');

  { The five predefined line styles supported }
  LineStyles : array[0..4] of string[9] =
  ('SolidLn', 'DottedLn', 'CenterLn', 'DashedLn', 'UserBitLn');

  { The twelve predefined fill styles supported }
  FillStyles : array[0..11] of string[14] =
  ('EmptyFill', 'SolidFill', 'LineFill', 'LtSlashFill', 'SlashFill',
   'BkSlashFill', 'LtBkSlashFill', 'HatchFill', 'XHatchFill',
   'InterleaveFill', 'WideDotFill', 'CloseDotFill');

  { The two text directions available }
  TextDirect : array[0..1] of string[8] = ('HorizDir', 'VertDir');
```

```
  { The Horizontal text justifications available }
  HorizJust  : array[0..2] of string[10] = ('LeftText', 'CenterText', 'RightText');

  { The vertical text justifications available }
  VertJust   : array[0..2] of string[10] = ('BottomText', 'CenterText', 'TopText');

{*************************************************************************}
{----------------------- GLOBAL VARIABLES ------------------------------}
{*************************************************************************}

Var
  GraphDriver : integer;  { The Graphics device driver }
  GraphMode   : integer;  { The Graphics mode value }
  MaxX, MaxY  : word;     { The maximum resolution of the screen }
  ErrorCode   : integer;  { Reports any graphics errors }
  MaxColor    : word;     { The maximum color value available }

{*************************************************************************}
{------------------------------ PROCEDURES -----------------------------}
{*************************************************************************}


procedure Initialize;

function Int2Str(L : LongInt) : string;

procedure DefaultColors;

procedure DrawBorder;

procedure FullPort;

procedure MainWindow(Header : string);

procedure StatusLine(Msg : string);

procedure WaitToGo;


{*************************************************************************}
```

Implementation

```
{*****************************************************************************}
{----------------------- INITIALIZATION PROCEDURE -------------------------}
{*****************************************************************************}

procedure Initialize;
{ Initialize graphics and report any errors that may occur }
begin
  { when using Crt and graphics, turn off Crt's memory-mapped writes }
  DirectVideo := False;
  GraphDriver := Detect;                    { use autodetection }
  InitGraph(GraphDriver, GraphMode, '');    { activate graphics }
  setGraphMode(GraphMode);
  ErrorCode := GraphResult;                 { error? }
  if ErrorCode <> grOk then
  begin
    Writeln('Graphics error: ', GraphErrorMsg(ErrorCode));
    Halt(1);
  end;
  MaxColor := GetMaxColor;  { Get the maximum allowable drawing color }
  MaxX := GetMaxX;          { Get screen resolution values }
  MaxY := GetMaxY;
end; { Initialize }

{*****************************************************************************}
{---------------------- INT_to_STRING PROCEDURE ----------------------------}
{*****************************************************************************}

function Int2Str(L : LongInt) : string;
{ Converts an integer to a string for use with OutText, OutTextXY }
var
  S : string;
begin
  Str(L, S);
```

```
  Int2Str := S;
end; { Int2Str }

{*****************************************************************************}
{------------------------ SET MAX DEFAULT COLOURS --------------------------}
{*****************************************************************************}

procedure DefaultColors;
{ Select the maximum color in the Palette for the drawing color }
begin
  SetColor(MaxColor);
end; { DefaultColors }


{*****************************************************************************}
{------------------------ DRAWBORDER PROCEDURE -----------------------------}
{*****************************************************************************}
procedure DrawBorder;
{ Draw a border around the current view port }
var
  ViewPort : ViewPortType;
begin
  DefaultColors;
  SetLineStyle(SolidLn, 0, NormWidth);
  GetViewSettings(ViewPort);
  with ViewPort do
    Rectangle(0, 0, x2-x1, y2-y1);
end; { DrawBorder }

{*****************************************************************************}
{------------------------ SET VIEWPORT to WHOLESCREEN --------------------}
{*****************************************************************************}

procedure FullPort;
{ Set the view port to the entire screen }
begin
  SetViewPort(0, 0, MaxX, MaxY, ClipOn);
end; { FullPort }

{*****************************************************************************}
{------------------------ SETUP SCREEN -------------------------------------}
```

```
{*************************************************************************}

procedure MainWindow(Header : string);
{ Make a default window and view port for demos }
begin
  DefaultColors;                          { Reset the colors }
  ClearDevice;                            { Clear the screen }
  SetBkColor(0);
  SetTextStyle(DefaultFont, HorizDir, 1); { Default text font }
  SetTextJustify(CenterText, TopText);    { Left justify text }
  FullPort;                               { Full screen view port }
  OutTextXY(MaxX div 2, 2, Header);       { Draw the header }
  { Draw main window }
  SetViewPort(0, TextHeight('M')+4, MaxX, MaxY-(TextHeight('M')+4), ClipOn);
  DrawBorder;                             { Put a border around it }
  { Move the edges in 1 pixel on all sides so border isn't in the view port }
  SetViewPort(1, TextHeight('M')+5, MaxX-1, MaxY-(TextHeight('M')+5), ClipOn);
end; { MainWindow }

{*************************************************************************}
{---------------------------- DISPLAY STATUS ----------------------------}
{*************************************************************************}

procedure StatusLine(Msg : string);
{ Display a status line at the bottom of the screen }
begin
  FullPort;
  DefaultColors;
  SetTextStyle(DefaultFont, HorizDir, 1);
  SetTextJustify(CenterText, TopText);
  SetLineStyle(SolidLn, 0, NormWidth);
  SetFillStyle(EmptyFill, 0);
  Bar(0, MaxY-(TextHeight('M')+4), MaxX, MaxY);      { Erase old status line }
  Rectangle(0, MaxY-(TextHeight('M')+4), MaxX, MaxY);
  OutTextXY(MaxX div 2, MaxY-(TextHeight('M')+2), Msg);
  { Go back to the main window }
  SetViewPort(1, TextHeight('M')+5, MaxX-1, MaxY-(TextHeight('M')+5), ClipOn);
end; { StatusLine }

{*************************************************************************}
{-------------------------- WAIT to ABORT --------------------------------}
```

```
{*****************************************************************************}

procedure WaitToGo;
{ Wait for the user to abort the program or continue }
const
  Esc = #27;
var
  Ch : char;
begin
  GotoXY(1, 25);
  write(' Hit a key to Continue ==>');
  repeat until KeyPressed;
  Ch := ReadKey;
end; { WaitToGo }



begin
  Initialize;
end.
```

APPENDIX I
THE OPTIMIZATION CODE

```
Program optimise;

Const
     TestSize = 18;
     DataSize = 4;

Type
    Range = Record
              up_bound : real;
              l_bound  : real;
             end;
    Data = array[1..DataSize] of real;

Var
   R       : array[1..DataSize] of Range;
   Mean    : Data;
   Sd      : Data;
   Opt_A   : Data;
   Opt_B   : Data;
   Opt_C   : Data;
   Opt_D   : Data;
   A, B    : Data;
   C, D    : Data;
   Cor     : Data;
   Opt_Cor : Data;
   Ans     : array[1..Datasize, 1..TestSize] of char;
   Cor_Sum : real;
   Max_Cor : real;
   Opt_X   : real;
   Test    : array[1..TestSize] of real;
   i, j, x : integer;


Function Phi(a, b, c, d : real) : real;
```

```
Begin
  If (a+b=0) or (b+d =0) or (c+a = 0) or  (d+c = 0) then phi := -2
   else  Phi := (a*d - b*c)/sqrt((b+a)*(d+c)*(b+d)*(a+c))
End;


Procedure Initialize_Data;

Begin
  Ans[1,  1] := 'n'; Ans[2,  1] := 'n'; Ans[3,  1] := 'y'; Ans[4,  1] := 'n';
  Ans[1,  2] := 'y'; Ans[2,  2] := 'y'; Ans[3,  2] := 'y'; Ans[4,  2] := 'n';
  Ans[1,  3] := 'y'; Ans[2,  3] := 'y'; Ans[3,  3] := 'n'; Ans[4,  3] := 'y';
  Ans[1,  4] := 'n'; Ans[2,  4] := 'n'; Ans[3,  4] := 'n'; Ans[4,  4] := 'n';
  Ans[1,  5] := 'n'; Ans[2,  5] := 'n'; Ans[3,  5] := 'n'; Ans[4,  5] := 'n';
  Ans[1,  6] := 'n'; Ans[2,  6] := 'n'; Ans[3,  6] := 'y'; Ans[4,  6] := 'n';
  Ans[1,  7] := 'y'; Ans[2,  7] := 'y'; Ans[3,  7] := 'y'; Ans[4,  7] := 'y';
  Ans[1,  8] := 'y'; Ans[2,  8] := 'y'; Ans[3,  8] := 'y'; Ans[4,  8] := 'y';
  Ans[1,  9] := 'n'; Ans[2,  9] := 'n'; Ans[3,  9] := 'n'; Ans[4,  9] := 'n';
  Ans[1, 10] := 'y'; Ans[2, 10] := 'n'; Ans[3, 10] := 'y'; Ans[4, 10] := 'n';
  Ans[1, 11] := 'y'; Ans[2, 11] := 'y'; Ans[3, 11] := 'y'; Ans[4, 11] := 'y';
  Ans[1, 12] := 'y'; Ans[2, 12] := 'y'; Ans[3, 12] := 'n'; Ans[4, 12] := 'y';
  Ans[1, 13] := 'n'; Ans[2, 13] := 'n'; Ans[3, 13] := 'n'; Ans[4, 13] := 'n';
  Ans[1, 14] := 'y'; Ans[2, 14] := 'n'; Ans[3, 14] := 'y'; Ans[4, 14] := 'y';
  Ans[1, 15] := 'y'; Ans[2, 15] := 'n'; Ans[3, 15] := 'y'; Ans[4, 15] := 'y';
  Ans[1, 16] := 'n'; Ans[2, 16] := 'n'; Ans[3, 16] := 'n'; Ans[4, 16] := 'n';
  Ans[1, 17] := 'y'; Ans[2, 17] := 'y'; Ans[3, 17] := 'y'; Ans[4, 17] := 'y';
  Ans[1, 18] := 'y'; Ans[2, 18] := 'y'; Ans[3, 18] := 'n'; Ans[4, 18] := 'y';
  Test[1]  := 0.85;  Test[2]  := 1.0;   Test[3]  := 1.15;  Test[4]  := 1.5;
  Test[5]  := 1.7;   Test[6]  := 0.85;  Test[7]  := 1.1;   Test[8]  := 1.2;
  Test[9]  := 0.7;   Test[10] := 0.95;  Test[11] := 1.0;   Test[12] := 1.05;
  Test[13] := 0.75;  Test[14] := 0.95;  Test[15] := 1.125; Test[16] := 2.0;
  Test[17] := 1.0;   Test[18] := 1.15;
  Mean[1]  := 1.158; Mean[2]  := 1.110; Mean[3] := 1.045;  Mean[4]  := 1.263;
  Sd[1]    := 0.155; Sd[2]    := 0.102; Sd[3]   := 0.214;  Sd[4]    := 0.144;
  Max_Cor := -5;
End;
```

```
Procedure Initialize_Structs;
  Begin
    For i := 1 to DataSize do
      Begin
        R[i].up_bound := Mean[i] + (x/10) * sd[i];
        R[i].l_bound  := Mean[i] - (x/10) * sd[i];
        A[i] := 0;
        B[i] := 0;
        C[i] := 0;
        D[i] := 2;
      End;
    Cor_Sum := 0;
  End;


Begin
  Initialize_Data;
  For X := 1 to 100 do
    begin
      Initialize_Structs;
      For j := 1 to TestSize do
        For i := 1 to DataSize do
          If (Test[j] <= R[i].up_bound) AND (Test[j] >= R[i].l_bound)
            then  If Ans[i, j] = 'y' then A[i] := A[i] + 1
                                     else C[i] := C[i] + 1
            else  If Ans[i, j] = 'y' then B[i] := B[i] + 1
                                     else D[i] := D[i] + 1;
      For i := 1 to DataSize do
        begin
          Cor[i]  := phi(A[i], B[i], C[i], D[i]);
          Cor_Sum := Cor_Sum + Cor[i]
        end;
      If (X = 15) or (X = 20) then
        For i := 1 to DataSize do
          begin
            writeln('Corrolation for subject ', i:1, ' is ', Cor[i] :1:3);
            writeln('A for subject ', i:1, ' is ', A[i] :1:3);
            writeln('B for subject ', i:1, ' is ', B[i] :1:3);
```

```
            writeln('C for subject ', i:1, ' is ', C[i] :1:3);
            writeln('D for subject ', i:1, ' is ', D[i] :1:3);
          end;
      If Max_Cor < cor_sum then
        begin
          Max_Cor := Cor_Sum;
          Opt_X   := X/10;
          For i := 1 to DataSize Do
            begin
              Opt_A[i] := A[i];
              Opt_B[i] := B[i];
              Opt_C[i] := C[i];
              Opt_D[i] := D[i];
              Opt_Cor[i] := Cor[i];
            end;
        end;
    End;
  For i := 1 to DataSize do
    Begin
      writeln('Opt corrolation for subject ', i:1, ' is ', Opt_Cor[i] :1:3);
      writeln('Opt A for subject ', i:1, ' is ', Opt_A[i] :1:3);
      writeln('Opt B for subject ', i:1, ' is ', Opt_B[i] :1:3);
      writeln('Opt C for subject ', i:1, ' is ', Opt_C[i] :1:3);
      writeln('Opt D for subject ', i:1, ' is ', Opt_D[i] :1:3);
    End;
  Writeln('The X is : ', Opt_X :3:3);
End.
```

BIBLIOGRAPHY

1. Hayes-Roth, F., Waterman, A. D., Lenat, D. B. (1983). Building Expert Systems. Addison Wesley, Massachusetts, p. 127-167.

2. Dixon, N. (1981). Preconscious Processing. Wiley, Chichester.

3. Gruber, T. R., Cohen, P. R. (1987). "Design for acquisition: principles of knowledge-system design to facilitate knowledge acquisition". Int. J. Man-Machine Studies, 26:143-159.

4. Kitto, C. M., Boose, J. H. (1987). "Choosing Knowledge Acquisition Strategies For Application Tasks". IEEE Proc. 8:96-103.

5. Garg-Janaradan, C., Salvendy, G. (1987). "A conceptual framework for knowledge elicitation". Int. J. Man-Machine Studies, 26:521-531.

6. Gregor, J. A., (1972). Experimental Psychology. John Wiley, New York.

7. Stevens T. R. (1989). Graphics Programming in C. M&T publishing, California.

8. Cooke, N. M., McDonald, J. E. (1978). "The application of psychological scaling tecniques to knowledge elicitation for knowledge based systems". Int. J. Man-Machine Studies, 26:533-550.

9. Woodworth, R. S., Schlosberg, H. (1961). Experimental Psychology. Holt Rinehart & Winston, New York.

10. Sheridan, T. B., Ferrel, W. F. (1974). Man-Machine systems: information, control, and decision models of human performance. The MIT Press, Massachussetts.

11. Falmagne, J. (1985). Elements of psychophysical theory. Oxford University Press, New York. p. 258-281.

12.Hoffman, R., (1989). "A brief survey of methods for extracting the knowledge of experts". SIGART Newsletter, 108:19-27.

13.Waldron, V. R., (1989). "Investigating the communication problems encountered in knowledge acquisition". SIGART Newsletter, 108:143-144.

14.Kelley, G. A., (1955). The Psychology of personal constructs. Norton, New York.

15.Lavrac, N., (1989). "Methods for knowledge acquyisition and refinement in second generation expert systems". SIGART Newsletter, 108:63-69.

16.Michie, D. (1986). "Machine learning and knowledge acquisition". In: Expert Systems: Automating knowledge acquisition. Addison-Wesley, New York.

17.Bareiss, E. R., Porter, B. W., Wier, C. C., (1988). "Protos: an exemplar-based learning apprentice". Int. J. Man-Machine Studies, 29:549-561.

18.Morik, K., (1987). "Knowledge acquisition and machine learning - the issue of modelling". IEE. Colloquium on 'Knowledge acquisition for Knowledge based systems', London, p. 4/1-4

19.Messier Jr., W. F., Hansen, J. V., (1988). "Inducing rules for expert system development: an example using default and bankruptcy data". Mangement Science 34:1403-1415.

VITA

Sorel Bosan

Born in Larnaca, Cyprus, the author came to the U.S.A. in Aug 1983 as a CASP (Cyprus America Scholarship Program) Scholar to study Computer Engineering at Case Western Reserve University. He received his B.Sc. in Engineering, with honors, in May 1987.

He entered the College of William and Mary, Department of Computer Science in Aug 1987. During the course of his studies there, he served as a research assistant at the Eastern State Hospital of Virginia.