TOWARDS THE ON-LINE DEVELOPMENT OF VISUAL INTERACTIVE SIMULATION MODELS

Stephen John Withers

Submitted in accordance with the regulations concerning admission to the degree of Doctor of Philosophy

University of Warwick

School of Industrial and Business Studies

September 1981

CONTENTS

ILLUSTRATIONS

Photographs

Figures

ACKNOWLEDGEMENTS

I wish to acknowledge the help and encouragement given by the following individuals and organisations.

Dr. Robert Hurrion, my supervisor.

My fellow research students: Bill Fisher, Jeremy Brown, Giles Rubens, and Carlos Moreia da Silva.

Other members of the School of Industrial and Business Studies, and staff in the University Library, Computer Unit, and Computer Science Department.

John Hough and Brian Carrier of Rolls-Royce Ltd, Aero Division, Derby, also Robert Caldwell and John Gibson of British Steel Corporation, Tubes Division, Corby for providing the opportunities for industrially based research.

The Science Research Council (now Science and Engineering Research Council) for their financial support.

I.C.I. Corporate Laboratory for the use of their colour terminal, and constructive comments on an early version of the development system.

Tandy Corporation (Branch UK) for the loan of the printer used to produce this document.

Catherine Withers for her proofreading and patience.

SUMMARY

Reviews of previous work on visual interactive simulation, and on the interface between humans and computers were undertaken, the latter considering the physical and psychological aspects of the subject.

Two simulation projects carried out in association with Rolls-Royce Aero Engines and the British Steel Corporation are described in detail.

As a result of these projects and the review of previous studies, a major weakness in the technology of visual interactive simulation was identified: while the visual representation aids validation, verification, and experimentation, no facilities are provided to assist the analyst in the task of model construction. Simulation program generators are of proven use for non-interactive models, but a visual model requires a graphically oriented approach.

The main section describes the design and implementation of a substantial extension to the simulation software developed at Warwick. This allows the design and development of displays to be carried out 'on-line', while preserving the one-to-one correspondence between simulation entities and their visual representation. It is suggested that this has the potential to significantly reduce the elapsed time taken to develop visual simulation models, while increasing the involvement of the user (or sponsor) in the modelling process, especially when 'pre-defined' entity types are used to minimise the amount of model-specific coding required.

Finally, potential routes for the further development of visual interactive simulation are discussed, including the implementation of a 'simulation language' interpreter within the existing software. This would result in a system which was fully interactive, easing model development as well as experimentation.

Chapter 1

VISUAL INTERACTIVE SIMULATION

The History of Visual Interactive Simulation at Warwick

In order to set the scene for the current research, a description of the earlier work at Warwick is necessary. Hurrion (1976) provided the foundations, describing the facilities needed in computer software to support discrete event digital simulation models which would include a dynamic visual display of the system, and facilities which allow interaction with the running model. He developed a system with these features in mind, and used it successfully in two industrial studies from which he drew two conclusions which have yet to be contradicted: that visual interactive simulation promotes the user's confidence in the model, and that this way of combining the specific skills of man and machine may produce better results (e.g. sequencing a number of items through a series of operations in a shorter time) than either can achieve in isolation.

Secker's thesis describes his modification of Hurrion's software in order to use a new computer and an alpha-numeric visual display unit (vdu) in place of the graphics terminal used by Hurrion. When this had been achieved, he started work on a major study of a dye-stuffs manufacturing plant owned by I.C.I. The problems were being experienced in scheduling batches of chemicals through the plant in order to meet supply and demand

constraints, and achieving adequate performance in terms of plant utilisation and other criteria.

Due to the scale of the project, Secker was unable to complete the task. He had made a significant contribution to the development of the field, as the nature of the plant and its processes was such that the existing method of representation (using queues of entities) was not applicable. Instead, what was effectively a map of the plant was produced, showing the state of each vessel. This provided a display which was familiar to those involved in the operation of the plant (Secker, 1977).

Responsibility for the completion of the dye-stuffs model was taken by Rubens (1979). Great care was taken to protect the integrity of the model and its data structures from accidental corruption by the user ('bullet-proofing'). This does not imply that users are seen to be stupid (the term 'idiot-proofing' is sometimes used as a synonym for bullet-proofing), instead it should be seen as an attempt to provide the computer with a 'mental set' matching that of the user. When two people are discussing something, much of the conversation is implicit, rather than explicit - "Watch the game last night?", "Sure, Dalgleish's second goal was amazing!". The second speaker has not only understood that the questioner was talking about football, but that the Liverpool match was the one in question. 'Bullet-proofing is also an appropriate term because of its defensive connotations, in this case against malicious input.

Bullet-proofing was achieved by providing model-specific interactions which check the validity of all values entered by the user. These interactions did not completely replace the low-level interactions which work on the data structures, as they are very powerful tools in the hands of a person familiar with the model structure, especially during the process of verification and validation. The other important feature of these model-specific interactions is that they are designed to speak the user's language. Rubens gives as an example the procedure employed when the user wants to add a campaign (a unit of production) to the head of the queue for a certain reaction area. Throughout the dialogue, the model refers to 'campaigns' and 'reaction areas', instead of abstractions like 'entities' and 'sets'. Another aspect of this problem is that different types of specialists use the same word with different meaning: "the word 'Batch' to plant staff means a part of a campaign whereas to a computer specialist it may imply an alternative to interactive computing" (page 49).

Considerable time and effort was spent on refining the displays originally designed by Secker. The first task was to allow the use of an ISC Intercolor terminal to permit the colour coding of certain display features, and to provide graphics facilities.

In order to fill the need for more detailed information about certain critical stages of production a number of additional displays were developed. This

information was presented in two different ways - time series graphs (to show the utilisation of certain pieces of plant), and bar charts (e.g. the display of the status of the whole plant).

Two of Rubens' conclusions are relevant to the current author's research. The first is the importance of the separation of display generation from the underlying model. This allows changes to be made to the displays without affecting the validity of the model. The other is that the usefulness of the model was largely due to the close fit between the displays and the manager's mental image of the plant.

By the time Rubens became involved with I.C.I., the company was sufficiently convinced of the potential benefits of visual interactive simulation to instigate a further study. Fisher's (1981) work describes an investigation of the feasibility of using the techniques to model a continuous process plant, using the 'Acids and Primaries' plant as a guinea pig. As in the dye-stuffs project, the goal was a model which could be used as a day-to-day on-line decision aid.

The existing simulation software could only be used for discrete models, so a major addition was required to handle the continuous nature of the new problem. This was achieved by implementing a new type of entity: the vessel. In fact, a vessel is represented within the system by two simulation entities, although the programmer need not be

aware of this fact. The first entity is the one defined in the program, and it may be freely used by the designer: events may be scheduled for it, and its attributes may take any reasonable value. The second entity is invisible to the programmer as the system uses it to record the vessel's contents and the rates of flow into and out of it.

Whenever a vessel attribute is changed, the system recalculates the time at which the contents will reach one of four levels: the maximum or minimum capacity, or the upper or lower warning level. The underlying assumption is that the relationship between time and flow is linear. Provision for differential equations would make the system more generally applicable, but the need has not arisen for this additional complexity.

Fisher discusses the practical problems of display design, stressing the importance of involving the potential user in the process, and of designing for a person, rather than people in general. He also finds these points relevant when developing the dialogues provided to allow users to control the model. In the broader context he sees a need for a change of "emphasis from interactive programming to the tools needed to develop interactive programs" (page 131).

Contemporary with Rubens' and the early stages of Fisher's work, Brown (1978) was continuing the development of the system software for use in the original area of discrete manufacturing applications.

The first of three projects with Rolls-Royce Aero Division was the construction of a visual interactive simulation model of a foundry. A batch simulation model had already been constructed by the company's O.R. group, permitting the objective of determining the usefulness of visual modelling to line managers, without the pressures imposed by a brand new study. At the first demonstration the Foundry Manager immediately recognised the layout of the main display as a representation of the foundry, even though line managers were not involved in the design of the displays.

Several important points were noted during the demonstration. Problems due to the remote access of the University computer via telephone lines caused some scepticism - the appearance of spurious characters (due to noise on the line) worked to reduce confidence in the model. The jerkiness of operation caused by the time-sharing system also contributed to this. The slow line speed (300 Baud, or 30 characters per second) meant that a complicated display could take 30 seconds or more to appear on the screen; this soon became tedious.

On the positive side, the dynamic nature of visual simulation allowed the managers to question certain points about the model of which they had not been aware in the batch model, and the ability to experiment with different operating strategies and see their effects very quickly was seen by managers as a useful and attractive "gimmick".

After this project a considerable effort was made to rationalise and develop the simulation software in order to ease the task of model development and to increase the general applicability of the system. This was achieved by providing higher-level procedures, and by adding scheduling and time-advance routines. Further extensions were made on an ad hoc basis.

The next study came about as O.R. staff had been unable to give a satisfactory explanation to a manager for the effects of different priority rules on the performance of a manufacturing shop. Although they understood the processes involved, they were unable to communicate this knowledge to the manager. A simple model was designed, with one man operating any one of six machines. Each batch of the anonymous product had to be processed on each machine in a fixed sequence. Two operating rules were used, "select the batch that has the fewest operations remaining", and "select the batch that has the most operations remaining". The results of an existing (but not precisely comparable) simulation suggested that the former would give superior performance. The simplicity of the model provided an ideal opportunity to test the new version of the system software.

Use of the model to compare the rules showed that (as expected by those who had experience of the batch simulation) selection of the batch with fewest operations remaining resulted in a significant reduction in work in progress, and the steady state was reached more quickly.

When the manager concerned saw the model in action "he was able to understand what had previously been beyond his comprehension" (page 83), showing the potential of visual interactive simulation as a teaching aid, particularly when the outcome of a course of action is counter-intuitive (see also Forrester, 1969, page 109).

A different set of problems were posed by the subject of the final study. Its objective was to investigate the workings of a scheduling mechanism proposed for use in the production of spare parts. Parts are produced in three stages: detail, sub-assembly, and assembly.

The distinctive nature of the project resulted from two sources: the 'users' were O.R. analysts, rather than managers, and the level of abstraction meant there was no obvious visual representation of the system.

Histograms and time-series graphs were also used to present summary information, with the dynamics of the model shown in tabular form, the figures being updated as time passes. Once the model had been validated, the dynamic display became less relevant. Since display forming was so time consuming, a 'fast forward' mode of operation was developed which switched off all displays for a user-specified period. This mode of working, combined with the dump and restart facility (users may save the state of a model, usually at a decision point, and later return to that situation to compare the effects of a different choice) made it possible to compare the effects of

different rules, investigating the cause of any surprises by reverting to the normal display mode. Thus the benefit to the users resulted from the interactive, rather than the visual nature of the simulation.

The study was successful in that the fact that the users were analysts instead of managers did not detract from the usefulness of visual interactive simulation, and that an abstract system had been successfully modelled using this technique.

Work at Other Instititions

Work which may be classified under the heading visual interactive simulation has been carried out at places apart from the University of Warwick. Palme (1977) describes a set of extensions to Simula which give that language the facilities needed to allow interaction with a running model, and to show a visual representation of the model. Recent releases of the simulation programming language ECSL (Extended Control and Simulation Language) are also equipped to dynamically display the model's state (Clementson, 1980).

Some specific models (as opposed to the programming systems described in the last paragraph) have appeared in the literature. Bazjanac (1976) describes a model used to investigate the evacuation of a building's occupants by elevator (conventional wisdom insists that lifts should never be used when a building is on fire, but the study

showed that evacuation is completed faster and without additional risk to life and limb if lifts are used).

A very interesting system has been reported by Whitefield, et al (1980). This is an interactive aid for air traffic controllers which includes a simulation model capable of predicting the future state of the airspace based on flight plans and aircraft performance, allowing controllers to see the effects of different plans before making a decision. The system has also been used to explore the possibility of controllers working in pairs, each handling a different part of the overall task, and communicating directly and through the shared database.

Dynamic Interactive System Simulation (DISS) is the name given by Spearman to a type of simulation modelling which combines aspects of systems simulation (i.e. what operational researchers generally mean by 'simulation') and of dynamic simulation (implying life-like models, such as flight simulators). The description given (Spearman, 1980) makes it clear that DISS is synonymous with visual interactive simulation. Indeed, many of the features of his implemented system parallel ideas developed at Warwick: "A rose by any other name..."

There has also been interest in the implementation of visual interactive models on microcomputers. Apart from Secker (1979) and Hurrion (1980), a number of models of this type have been successfully developed at Lancaster University using an Apple II microcomputer (Ellison, 1980),

and Spearman has implemented his DISPAC (Dynamic Interactive Simulation PACkage) on a TRS-80. A micro-computer implementation of ECSL running under the CP/M operating system is now available.

This chapter provides an overview of the work carried out at Warwick and other institutions to provide discrete event digital simulation with two complementary features. The first is dynamic displays which show the state of a model. Having given the user a window into what is otherwise a 'black box', the systems also provide the ability to control the model as it runs, possibly overriding programmed operating rules or exploring the effects of different algorithms.

The spares scheduling model developed by Brown was used as a basis for further investigation. This work was carried out by the present author, and is described in Chapter 2.

Chapter 2

THE ROLLS-ROYCE SCHEDULING DEMONSTRATOR

The O.R. staff at Rolls-Royce Aero Division had become interested in visual interactive simulation as a result of Hurrion's work. Contact was established and a research student from Warwick carried out three modelling projects with the company (summarised in Chapter 1 of this thesis). The major conclusions drawn by Brown from his work on a stock control problem were that visual interactive simulation was a viable technique for investigating abstract systems (i.e. those with no obvious visual representation), and that it was a suitable tool when the end-users are analysts, rather than managers (Brown, 1978).

The present author became involved as the Rolls-Royce analysts believed that a model of this type could be used as a training aid, as it could show the effects of different production and ordering schedules on stock levels and on the ability to meet demand for the product. The stability of these schedules is also important, as the disruption of existing plans may lead to considerable ill-feeling on the part of suppliers and production staff.

Such a model was constructed. Known as the scheduling demonstrator, or simply the demonstrator, it is a two stage system. Raw materials are purchased from outside, and made into a finished product. The demand for the product is generated randomly at the start of each accounting period (a.p.) of 20 working days. The statistical distribution

used may be rectangular, poisson, or gamma, and the type of distribution used, as well as its parameters may be altered while the model is in use.

As suggested above, the user's task is to enter schedules for orders and production. This is done each a.p., and there are three ways of doing so. The existing schedule may be carried over (i.e. no changes to the plan), a completely new schedule may be entered, or the program will produce a 'recommended' schedule based on the state of the system.

An example of the main display generated by the model is shown in Photograph 1. This provides the user with details of the current schedules, forecasts of demand, stocks, work in progress and other information. Histograms and timeseries graphs are used to indicate various measures of performance, including such things as unsatisfied demand for the product, and the number of changes to the schedules (broken down by the amount of notice given).

The amount of work needed to complete a batch of work in progress or stock on order is expressed as the leadtime. Each operation represents one day's work. The number of operations performed in an a.p. on each batch is variable, and depends on the batch's priority rating. This priority rating is in terms of expected lateness - the difference between expected time to the completion of the batch and the time the schedule calls for its completion. Thus a job which is behind schedule will have a high priority rating, which is intuitively correct. Having calculated this

Photograph 1

Main display of Rolls-Royce model

Photograph 1

Main display of Rolls-Royce model

rating, the model determines the amount of work to be carried out on the batch by applying a simple function. Between upper and lower limits there is a linear relationship between the rating and the number of operations completed. If the rating falls outside the range, it is treated as if it was equal to the appropriate limit (Photograph 2 illustrates the shape of the function). The values of the limits and the equivalent number of operations may be changed by the user.

'Forecasts of demand' were mentioned in an earlier paragraph. These are calculated from a weighted average of the previous forecast and the actual demand. As with other parameters the user has the option to explore the effects of varying the relative weights.

Finally, there is a minimum stock level for both stages. This is expressed as a multiple of the current forcast for the stage, and affects the size of the batches that will be started or ordered. The minimum stock level, the forecast demand, and the size of batches already in progress/on order and the leadtime together determine the batch size.

One model-specific interaction was added to those written by Brown. This was called OMANUA and when used at the start of a run allows the user to override the default settings of all the parameters of the model.

The user documentation for the model may be found in Appendix 1.

Photograph 2

Graph of priority function

Photograph 2

Graph of priority function

The model was demonstrated to the manager whose subordinates were responsible for scheduling. Despite the fact that telephone line noise repeatedly disconnected the terminal from the computer he agreed that it showed clearly the nature of the problem, and that it could provide a useful training aid.

One of the advantages of visual interactive simulation is that it increases the client's involvement with the model (Bowen, et al, 1979b). This may be promoted by making the the use of the model as straightforward and uncomplicated as possible. A particular problem is raised by the common practice of using typewriter style keyboards for communication with the model. In common with many others, the potential users of the model were not accustomed to keyboards, so it was decided to investigate the use of the spoken word to control the model.

A commercial speech recognition device was purchased, the Heuristics SpeechLab Model 20A. This is a low cost device (around £150), which plugs into an Apple II micro-computer, and is capable of matching sounds against a maximum of 32 previously defined patterns. In practice, more reliable results are obtained by using two or three samples of each word, although this reduces the number of available commands.

This problem may be eased by defining more than one set of patterns, and loading the pattern appropriate to the context from a disk file or from an otherwise unused section of memory. The latter would be noticeably faster

if a machine code program was used to effect the transfer, but the use of disk files is simpler to program, and would be required to save the users' 'voice patterns' to avoid the need to retrain the device each time it was used. Since the objective was to test the general suitability of speech input to visual interactive simulation models, it was decided that the disadvantages of using disk files were not sufficient to warrant machine coding. In any case, there would be no great loss of time if it was found to be necessary at a later stage.

A problem soon arose in that certain commands which sound quite different to the human ear could not be resolved by the SpeechLab. As a result the spoken word "SCREENS" was used in place of the command "REFO".

Two programs were written in order to use the Apple II as a voice-operated terminal. Listings are given in Appendix 2. The first task is to train the system to recognise the user's voice, and to store the 'voice print' on disk. The available commands are displayed on the screen one at a time, and the SpeechLab associates the noise it 'hears' through its microphone with the command word. There is no requirement for the spoken word and the command word to be the same, so the problem of discrimination mentioned above may be circumvented without the need to change the simulation program.

Before the second program (called VOICETERMINAL) can be used the Apple II must be connected to the host computer. Fortunately the Apple RS-232 communications

interface includes firmware which allows it to be used as a simple terminal. The simulation program is then run from an ordinary terminal, and the "SCRN" interaction used to attach the Apple as the subsidiary screen. Once this is done, it is necessary to re-enter BASIC and run VOICETERMINAL.

The following commands may then be spoken:

Spoken word	Equivalent typed command
RUN	RUN
KILL	KILL
RECOMMENDED	R
OLD	O
NEW	N
HIST	HIST
SCREENS	REFO

For most purposes these commands suffice, as few people have difficulty in entering numbers from a keyboard - probably due to the widespread use of calculators. Numbers could be spoken, but the hardware used is not sophisticated enough to permit recognition of "one-hundred-and-thirty-five" instead of "one" - "three" - "five" - "enter". This is clumsy and slower than keyboard entry, especially if a separate numeric keypad is available.

The results of the trials were inconclusive. On one hand the opportunity to actually tell the computer what to do was popular, on the other the effort involved in speaking clearly and consistently was not always rewarded by reliable recognition. This problem was encountered more

often by people from the south of England, probably due to their softer vowel sounds (e.g. pronouncing the word "path" as "par-th", rather than "pah-th").

The advantage of the novelty value of voice input should not be discounted. One of the strengths of visual modelling is that it helps to gain the user's interest and attention, but this must last long enough to reveal the possible contribution of the underlying model. The risk is that if the recognition device is not sufficiently accurate, the overall effect will be to discourage new or potential users.

The market for voice recognition devices has grown since the work described here was carried out, and in common with other manufacturers, Heuristics, Inc. have continued to develop its products. One of its recent offerings, the Model 7000 Voice Controller, may be used with virtually any terminal and is said to give accurate and reliable recognition of up to 128 commands. Its price is higher than the Apple II/Model 20A combination, but it is not as expensive as the high-performance devices designed for use in military aircraft. Whether or not its performance is adequate for use in visual interactive simulation can only be determined by field trials.

Conclusion

The results of the study supported the idea that a visual interactive simulation model could be a useful training aid, showing the mechanism within a non-trivial system which may otherwise be treated as a 'black box'.

As a completely separate exercise, a Systems Dynamics representation of the model was constructed by the O.R. department at Rolls-Royce. It was interesting to discover that the results obtained from similar schedules were in close agreement despite the differences between the two approaches.

There were two reasons for considering the question of input devices. The first is a moral issue: it is the author's belief that designers of all types of systems which involve people have a responsibility to make those systems fit for use by those people, and that human requirements should not be sacrificed for the sake of technical convenience. The second point is more hard-nosed. The considerable amount of time and effort which goes into the construction of a visual interactive simulation model will be wasted if the model is not used (the modeller may gain something from the process, but this is less likely to solve the perceived problem). One reason why such a model is never used extensively may be because it requires more effort on the part of its intended user than he is prepared to undertake, therefore consideration of human factors may make the difference between a

project's success or failure. It may happen that the modeller himself will act as an interface between the model and its user, in which case the dialogues need not be problem-oriented, although a healthy working environment is still important.

The use of speech to control simulation models has much to commend it, especially in situations where users are not familiar with keyboards (ease of use and novelty are particularly significant), but the limitations of affordable hardware (in terms of marginal cost and the funds available to the author) worked against the expected advantages. Given the nature of the dialogues used in this study, even minor improvements in the technology of voice recognition are likely to result in a usable speech entry system, as only isolated words need be processed: it is not necessary to cope with the complications of continuous speech.

Keyboards and voice recognition devices are not the only mechanisms which allow a human to interact with a computer. A number of alternatives are discussed in the next chapter.

Chapter 3

INPUT DEVICES

While evaluating the SpeechLab, the broader area of input devices for interactive computer systems was investigated, using Umbers' paper as a framework. Although he is concerned with process control applications, much of the work reported is appropriate to the interactive modelling environment (Umbers, 1976).

Keyboards

The results of earlier work at Warwick (e.g. Rubens) suggested that a normal keyboard was not really suitable as a main input device when a model is intended for use by managers, as that combination was found to be both slow and error prone. An exception was reported by Fisher who found that a Plant Manager preferred to use a keyboard to the other available devices. This was because keyed commands could be 'stacked' (i.e. several commands entered together), whereas he had to wait for the next set of options to be displayed before indicating his selection with the light pen. There is another reason for avoiding keyboards - a study reported by Dooley (1976) found that managers have a negative attitude towards typing, and will therefore resist systems which require keyboarding.

There are two main reasons for asserting the inferiority of the standard keyboard. The first is that the layout is intrinsically unsound, from the bio-mechanical point of view. There are two aspects to this

statement: the physical arrangement of the keys, and the allocation of symbols to keys. If the reader holds his or her hands palm down, with the elbows bent, the suggestion that keyboards should be made with a section for each hand will seem reasonable. Similarly it is not unimaginable that keys should be arranged in arcs, instead of straight lines, in order to match the curve of the fingertips.

The standard keyboard layout is known as 'QWERTY', after the first six letters on the top row of alphabetic characters. This was designed in the 19th century, with the objective of minimising type-bar clashing, a problem with the primitive mechanisms. The corollary is that it makes life more difficult for the operator. Attempts have been made to redesign the keyboard to suit its users. The most widely known examples are the Dvorak, and the more recent Maltron keyboards. The latter also uses an improved physical key layout. The failure of either to become established is largely due to the inertia behind QWERTY, rather than any shortcomings in their design.

Computer terminals are often equipped with a separate numeric keyboard. Fisher reported that users found this type of auxilliary keypad a useful adjunct to a light pen for numeric data entry. Apart from the problems associated with the use of light pens for this type of input, the similarity with the familiar pocket calculator probably played a part. However, some human factors studies suggest the commonly used calculator layout:

7 8 9

4 5 6

1 2 3

0

is inferior (in terms of accuracy) to a less popular alternative, the telephone keyboard

1 2 3

4 5 6

7 8 9

0

The work of Conrad and Hull (1968) suggests that the improvement in accuracy when using the latter is that it conforms to people's expectations of how numbers should be arranged.

While adding machines and calculators were commonplace and push-button telephones rarely encountered, the problem was less acute, but as the latter become more common, and the use of the public switched telephone network for data communications increases, difficulties due to the lack of a single standard for numeric keyboards may be expected to increase. However, a recent study (Nakatani and O'Connor, 1980) found that layout had no significant effect on performance. It is suggested that this is because calculators and push-button telephones have become commonplace (at least in the U.S.A).

Nakatani and O'Connor were primarily concerned with investigating the effects of speech feedback on the touch-keying (i.e. where the operator does not look at the

keyboard) of numeric data. An example of this type of task is keying a telephone number while reading it from a directory. They found that speech feedback led to greater accuracy without loss of speed. In particular they state that "casual users... stand to benefit substantially from speech feedback" (page 651). The applicability of these findings to other keying tasks is uncertain, as the subjects were shown randomly generated numbers which they immediately entered. Nakatani and O'Connor express concern that when numbers are being keyed from memory the feedback could have an adverse effect on performance by interfering with the process of recall.

Special purpose keyboards, including some sophisticated designs, have been used to ease the task of communicating with a computer. There are two main types - those whose keys have fixed functions, and those with dynamically redefinable functions. Umbers cites a paper which suggests that operators prefer to use special keyboards rather than a normal typewriter keyboard. An interesting enhancement is described by Wear and Dorf (1970). Their approach is to disable those keys which are invalid in the current context, while illuminating the valid keys. This arrangement prevents certain types of errors, and one user described the system as 'more relaxing'.

Problems arise when a large number of commands are needed, due to the sheer size of the keyboard. In such circumstances dynamic keyboards come into their own.

In general, a cathode ray tube (crt) is used to display the labels. The simplest arrangement places the buttons adjacent to the screen, an arrangement used in the Hewlett-Packard HP300 computer system. More sophisticated designs use reflections to make the labels appear on the keytops (e.g. Knowlton, 1975). The disadvantage of this sophistication is the expense involved, and so work at Warwick has followed the simpler arrangement (Hurrion and da Silva, 1981).

Other Devices

The alternatives to keyboards fall into two broad classes, albeit with a degree of overlap. The first group consists of devices used to point at items of interest on the screen, the other provides an analogue of the screen surface.

Light Pens

Light pens (and the less common light guns which are simply pens fitted with a pistol grip) are used to point directly at the screen (a light pen was used in the work described in Chapter 6). They work by detecting the bright spot on the screen as the electron beam passes beneath the pen. The time between this event and the start of the scan may be interpreted in terms of the position of the pen on the screen. Some kind of 'averaging' (by means of software or additional hardware) is generally needed to improve the stability of the co-ordinates. As might be expected, the use of a light pen or light gun allows much faster

positioning of a cursor than is possible with the keyboard (Goodwin, 1975). Another finding of the same study was that users strongly preferred the light gun to the pen, as it was more comfortable to hold and 'fire'. Despite this, light guns are rarely found.

Touch Displays

There are a variety of ways of detecting the presence of a finger or other pointer on a display screen. One of the early approaches used electrically conductive cross-wires. When a wire is touched, the change in capacitance is detected. Such a system is described by Johnson (1967). The disadvantage of touch-wires is that the number of sensors areas must be limited, because they obscure the display screen.

McEwing (1977) describes a 'live' interactive system which used touch-wires, finding them to be robust, easy to use, and acceptable to the operators. Another advantage was that operators made fewer errors than they did using ordinary keyboards.

A significant improvement in resolution may be obtained by replacing the wires with beams of infra-red light (Bird, 1977; Ciarcia, 1978; and the 'Touch Terminal' manufactured by Carroll Inc.). The device described by Bird has a resolution of 3mm across the whole screen, and by allowing vernier adjustments, a final resolution of less than 1mm may be achieved.

An alternative approach is used in a range of touch panels produced by TSD Display Products, Inc. The technique used is to generate a wave at the boundary between the panel and the air. Whenever contact is made with the screen the wave is reflected, and the position of the contact obtained in a manner similar to radar. The active area is 8" by 10" on A 15" diagonal screen, with a maximum resolution of 0.05".

Stammers and Bird (1980) report a study of the use of touch panels by air traffic controllers. All the subjects were in favour of touch input, but concern was noted about the positioning, size, and angle of the screen (the normal placing of a display screen is likely to cause fatigue when a touch panel is used). In addition, several controllers expressed a desire for more positive feedback, and the need for easy recovery from erroneous input (e.g. an 'ignore last entry' command).

Analogue Devices

Devices which provide an analogue of the screen surface include the joystick, tracker ball, 'mouse', and tablet.

The first three are similar in that they relate the position of the cursor to that of the control. Joysticks and tracker balls will be familiar to many people due to their use in video games, but the 'mouse' is less popular. It consists of a small block, fitted with a pair of wheels at right angles to each other. The 'mouse' is used by

moving it across a surface (e.g. the desk top) thus rotating the wheels which are connected to potentiometers. The settings of the potentiometers determine the x and y co-ordinates generated by the device.

An important factor in the use of these three devices is that the operator does not have to pick them up (as he does a light pen), and they stay in the position in which they are left (although some joysticks have self-centering mechanisms). English, et al (1967) compare the use of these and other devices, and conclude that "what is important to fast, efficient display selection is the particular feel to the user of the thing he grasps and moves".

The tablet has features in common with both categories of device, as it combines pointing with the separation of display and input areas. It is used by pressing a stylus (or similar instrument) against its surface, which is very similar to using a pen or pencil. The tablet has been found to be most useful when entering graphical information (i.e. sketching), but it seems less suitable for other types of communication.

Once a particular device has been selected, thought should be given to the way in which it will be used. The effect of layout on keyboarding performance has already been discussed, so it should not be surprising that the layout of displays used for data entry by pointing can affect performance. Long, et al (1977) describe an experiment which compared the performance of naive subjects

using three formats: linear

1 2 3 4 5 6 7 8 9 0

matrix (telephone format), and 'cash register'

1 1 1 1 1 1

2 2 2 2 2 2

3 3 3 3 3 3

4 4 4 4 4 4

5 5 5 5 5 5

6 6 6 6 6 6

7 7 7 7 7 7

8 8 8 8 8 8

9 9 9 9 9 9

The device used was a vertically mounted graphics tablet, so the task was very similar to using a light pen. They conclude that all three may be used successfully by naive users, finding no significant difference in accuracy. With numbers containing 50% of zeros no difference in speed was found either, but the cash register arrangement was significantly slower when 10% were zeros.

Although the conclusions of this study are not prescriptive, they are important because they show that the arrangement of items on a display used for data entry by pointing can have an effect on performance.

Conclusion

What lessons may be drawn from the investigation discussed above? First of all, whatever type of input device is employed it must be reliable and easy to use,

otherwise users will not be bothered with it. Second, the nature of the interaction has a bearing on the suitability of a particular device (e.g. keying is preferred to speech when entering numeric data). Finally, there is a growing body of human factors information which can be utilised to help users obtain the best performance from the equipment.

A man-machine dialogue involves communication in two directions: human to computer, and vice versa. As with input, a variety of devices may be used for computer output. At present the medium used most frequently is some kind of visual display, whether on a video screen or 'hard copy' (i.e. paper or microform). Where other forms of output are employed, they are generally used to reinforce some aspect of a display. Indeed, most computer terminals have a bell or bleeper which may be sounded to attract the user's attention.

In view of the primacy of visual displays an investigation was undertaken into their design and use.

Chapter 4

DISPLAYS

As with input devices, the published work on displays may be divided into two main categories: 'hard' ergonomics, concerned with factors affecting perception and physical well-being (such as character heights), and the 'softer' area dealing with the cognitive aspects.

The former is neatly encapsulated in 'The VDT Manual' (Cakir, Hart, and Stewart, 1979), which deals with both input and output aspects of video terminals. They describe the physiology of human vision and the factors by which it may be affected, in terms of the design of the terminal and the way it is used. The design considerations involve such factors as the size and design of the characters, the separation between lines, the luminance and contrast of the display, and freedom (or otherwise) from reflections in the screen.

Workplace design is also considered important. Apart from any legal requirement to provide a safe and healthy working environment, insufficient attention to working conditions could result in a potentially useful simulation model being ignored. It is not unknown for experimental sessions to last for hours, involving extensive display watching punctuated with spells of interactive dialogue. Under inappropriate conditions this may easily lead to backache, headache, and other types of discomfort. As a bare minimum, a variable-height seat and an adjustable

Chapter 4

DISPLAYS

As with input devices, the published work on displays may be divided into two main categories: 'hard' ergonomics, concerned with factors affecting perception and physical well-being (such as character heights), and the 'softer' area dealing with the cognitive aspects.

The former is neatly encapsulated in 'The VDT Manual' (Cakir, Hart, and Stewart, 1979), which deals with both input and output aspects of video terminals. They describe the physiology of human vision and the factors by which it may be affected, in terms of the design of the terminal and the way it is used. The design considerations involve such factors as the size and design of the characters, the separation between lines, the luminance and contrast of the display, and freedom (or otherwise) from reflections in the screen.

Workplace design is also considered important. Apart from any legal requirement to provide a safe and healthy working environment, insufficient attention to working conditions could result in a potentially useful simulation model being ignored. It is not unknown for experimental sessions to last for hours, involving extensive display watching punctuated with spells of interactive dialogue. Under inappropriate conditions this may easily lead to backache, headache, and other types of discomfort. As a bare minimum, a variable-height seat and an adjustable

footrest should be provided, with adequate desk space for working papers, calculators, and other paraphernalia. Lighting should be arranged to avoid reflections and glare.

Display formats are touched upon - it is suggested that simplicity and relevance are important in the design of an uncluttered display, and that displays should be consistent with each other and with other items (such as hard-copy documents). Some guidelines to the related area of coding are also offered: a maximum of two levels of brightness should be used (it is unlikely that more levels could be distinguished reliably), inverse video can be a very effective means of highlighting, blinking should only be used to draw the operator's attention to a single item, and colour coding can be a good way to allow the rapid discrimination between classes of object, although some eight percent of the male population are said to be colour blind.

Display Design

Relatively little published work has been found dealing with the principles of display design, and much of it is concerned with the fast and accurate identification of a target item (e.g Shontz, et al, 1971). However, some general principles are apparent in the literature, in particular the notion of compatibility. There are two principal aspects to this, compatibility between displays, and compatibility with the user's world-view and stereotypes (McCormick, 1964, chapter 9; Singleton 1969).

Singleton particularly stresses the need to consider the purpose for which the user needs the displayed information. The beneficial effects of displays which provide a familiar representation of the system under study have been noted in earlier work at Warwick (e.g. Brown, 1978, page 51).

In view of the availability of an ISC colour vdu, and the positive feedback obtained from existing users of the device (although problems had arisen when a thick layer of dust collected on the screen of a terminal used on a shop-floor, making a particular colour invisible), it was decided to investigate the literature on the use of colour in displays. The suggestion that colour displays were popular with users was echoed by Hopkin (1977), but again, much of the emphasis was found to be on search and identification tasks (e.g. Christ, 1975). Poulton and Edwards (1980) also report an experimental study of colour coding in a search context, but they cite studies showing that colour coding is so effective that it can 'overrule' other coding dimensions (or to quote from the paper, it can "reduce the effectiveness of independent orthogonal codes").

For the purposes of visual interactive simulation, the most important findings seem to be:

that the use of redundant colour coding can increase accuracy of identification (Christ, 1975), and may be used to reinforce another coding dimension (e.g. to highlight out-of-tolerance data items;

that colour stereotypes exist (e.g. red means danger) and may be exploited;

and that despite the incidence of colour-deficiency, the manipulation of controllable and/or selectable parameters allows the accomodation of about 98 percent of the male population (DeMars, 1975), with women being less prone to colour-blindness.

Conclusion

The material covered in this section suggests that increased user involvement in the design of displays for visual interactive simulation models may result in displays which are more satisfactory by virtue of a better fit between them and the user's mental model of the system, and because of their consistency with the user's stereotypes.

Attention should also be paid to the working environment in which the computer terminal is to be used, otherwise users may suffer physical discomfort, and coding dimensions may disappear.

Chapter 5

There are two parallel streams to this thesis: one is practical and problem-oriented; the other theoretical. This chapter describes a second modelling project.

THE BASIC OXYGEN STEEL-MAKING SIMULATION

The problem described in this chapter was of interest because it stemmed from the inability of two parties to see the other's point of view. It was hoped that a visual interactive simulation model of the system would help to improve communications across planning boundaries, showing the participants the effects their actions were having on the other operational area. The project also provided an opportunity to put into practice the lessons that had been learned about man-machine interaction.

Steelmaking at Corby

The Basic Oxygen Steel-making Process (BOSP) converts iron into steel. Described simply, the process involves charging the steelmaking vessel with molten iron and scrap metal, and then passing oxygen through the mixture in at least two phases (or blows). This removes impurities from the metal, and then it is analysed and any required adjustments made in order to obtain steel with the desired properties. A more detailed description may be found in Doyle (1969, pages 59-60).

At the British Steel Corporation (BSC) works at Corby, Northamptonshire, iron was produced in blast furnaces and

transported the short distance to the steel-making area by train. On arrival at the BOSP plant iron was put into storage vessels known as mixers. There were three mixers, but at any time only two were in use, while the third was undergoing maintenance. Once the mixers were full there was no other storage available for molten iron.

Oxygen was produced on site, by a plant owned by BSC and operated under contract by the British Oxygen Corporation (BOC). There were two distinct supplies - one side operating at high pressure (600 p.s.i.), the other at 350 p.s.i. Each supply had its own storage tanks, but were linked by an interconnecting valve (ICV). This valve was intended to ensure that both sets of storage tanks became full at the same time, as the production rates of the two supplies were independent. This would only work if the setting of the ICV corresponded to the ratio of the supply rates, which varied according to the operating requirements of BOC. To make matters worse, the ICV was both inaccessible and awkward to adjust, so it was rarely at the ideal setting.

The total oxygen storage capacity was 11.2 tonnes, which was approximately the amount required to produce a batch of steel. The small storage capacity meant that the rate at which steel could be produced was largely dependent on the oxygen supply rate. The working rules were that the tanks must be at least 70% full before a stage 1 blow could start; stage 2 blows required 40% of capacity. In

addition, a BOSP vessel could not be charged with iron unless the other was idle, or had reached the stage 2 blow. This was to prevent both vessels requiring oxygen simultaneously.

Another effect of the small storage capacity was that the tanks soon became full when little or no oxygen was being used. Since the contractual minimum supply rate was approximately 10 tonnes per hour, the tanks would fill in little more than an hour. When this happened, incoming oxygen was wasted as it was simply released into the atmosphere. The problems with the ICV meant that one side could start venting before the other set of tanks were completely full.

It was this wastage of oxygen that was the cause for concern. The objective of the project was to determine a way of reducing the losses without disrupting the production of steel. The line management were acutely aware of the problem, and plant personnel had agreed to adopt measures to reduce the wastage. The Fuel Controllers were responsible for wastage, but their decisions about the oxygen supply rates were based largely on the estimates of requirements supplied by the BOSP personnel.

These estimates were usually very optimistic (i.e. they were based on a higher level of production than that which was likely to occur), since the BOSP personnel preferred to have plenty of oxygen available to allow for the possibility of breakdowns and so they could schedule their work to suit themselves. The shift pattern (three

shifts per day, Monday to Saturday) had a bearing, as the 6 a.m. to 2 p.m. shift on Saturday was a non-blowing shift, although the production of iron continued. This resulted in an increase in iron stocks, which caused a higher rate of production at the beginning of the week.

The Fuel Controllers were aware of the production workers' ploy, and therefore tried to second-guess the estimates, but the feedback from the BOSP personnel when delays occurred due to a lack of oxygen was much more immediate than that from the process of accounting for oxygen losses. These behavioural factors added to the technical problems, and resulted in a high level of wastage.

The problem only appeared when steel production was reduced because when working nearer to capacity, production tends to be smoother, resulting in a stable demand for oxygen. At the prevailing production rates the demand was necessarily irregular. Since the storage capacity and the freedom to alter the supply rates were both limited, the result was greater losses. This fact was understood, and a new 'standard' level of losses set above the old one.

The Simulation Model

The model of the BOSP was simplified by combining the two active mixers into a single entity. This meant that on occasions an event could be delayed while the mixer was busy, when in practice the other could be used for the second operation. The oxygen tanks were combined in a

similar manner, which was equivalent to the assumption that the ICV was always correctly set. The justification for this aggregation was that plant personnel experienced the oxygen supply as a single stream, and the operating rules described above referred to the total storage capacity. In addition, the Fuel Controllers could only specify the total supply rate, while BOC decided the division between the high and low pressure supplies.

Data for the model were available from the operational records of the steelworks. In some cases (e.g. the duration of a stage 1 blow) it was possible to fit a theoretical distribution to the data, but empirical distributions had to be used for the remainder. This was open to criticism, but the data bore no resemblance to any of the standard statistical distributions. The practice of using empirical distributions in such cases is supported by Naylor, et al (1966, page 69) and by Gordon (1978, page 122).

The model logic is illustrated by the activity cycle diagram shown as Figure 1. The existing simulation software was adequate to develop the model.

Displays

No attempt was made to represent the physical layout of the plant. Instead, the objective was to show the entities and their state as clearly and simply as possible. When two entities are involved in the same activity they are connected by a dashed line, representing the flow

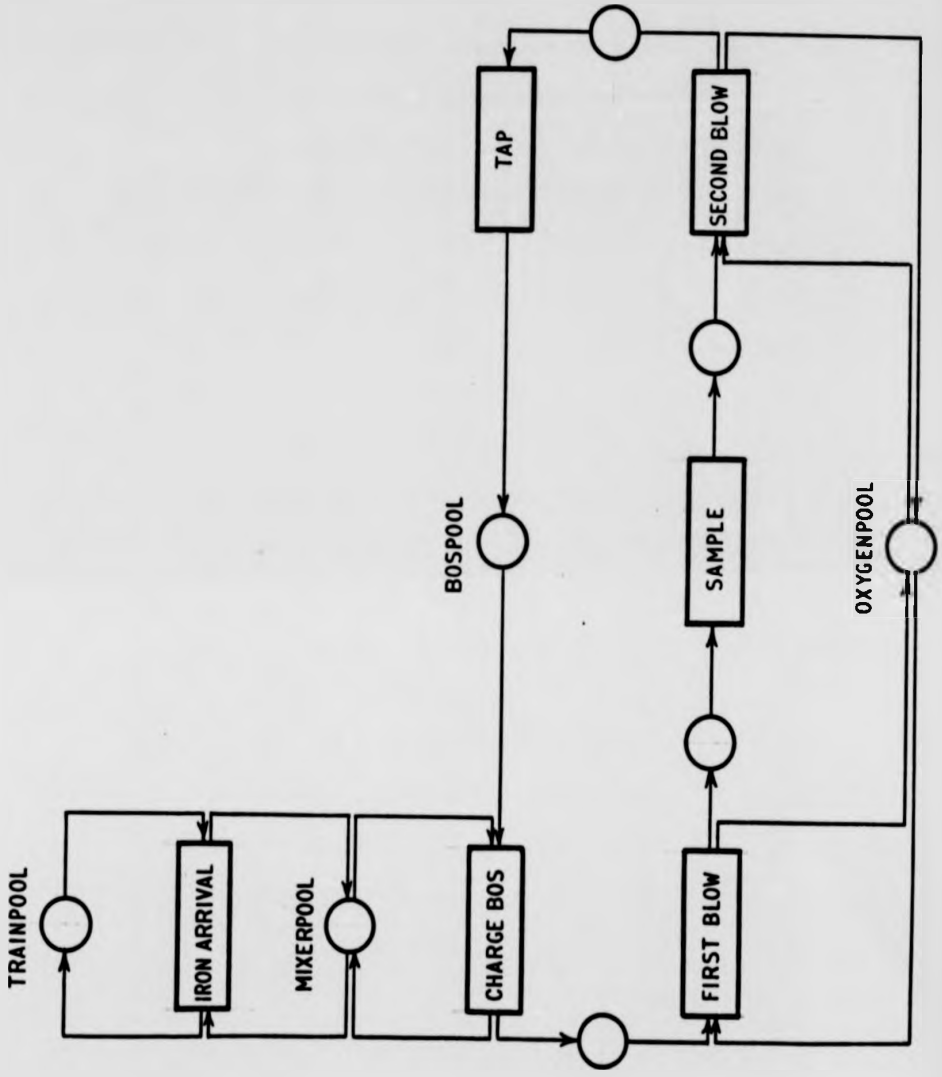


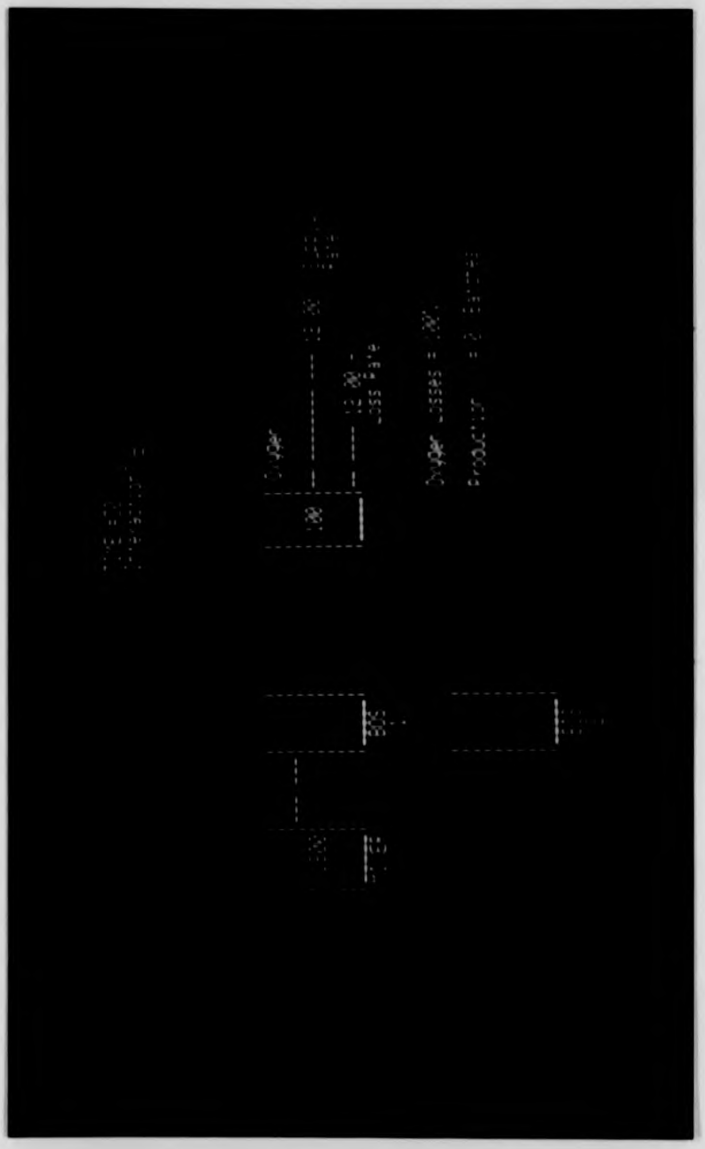
Figure 1

Entity cycle diagram of BOS process

between them. For example, when a BOSP vessel is charged, iron flows from the mixer to the vessel - see Photograph 3.

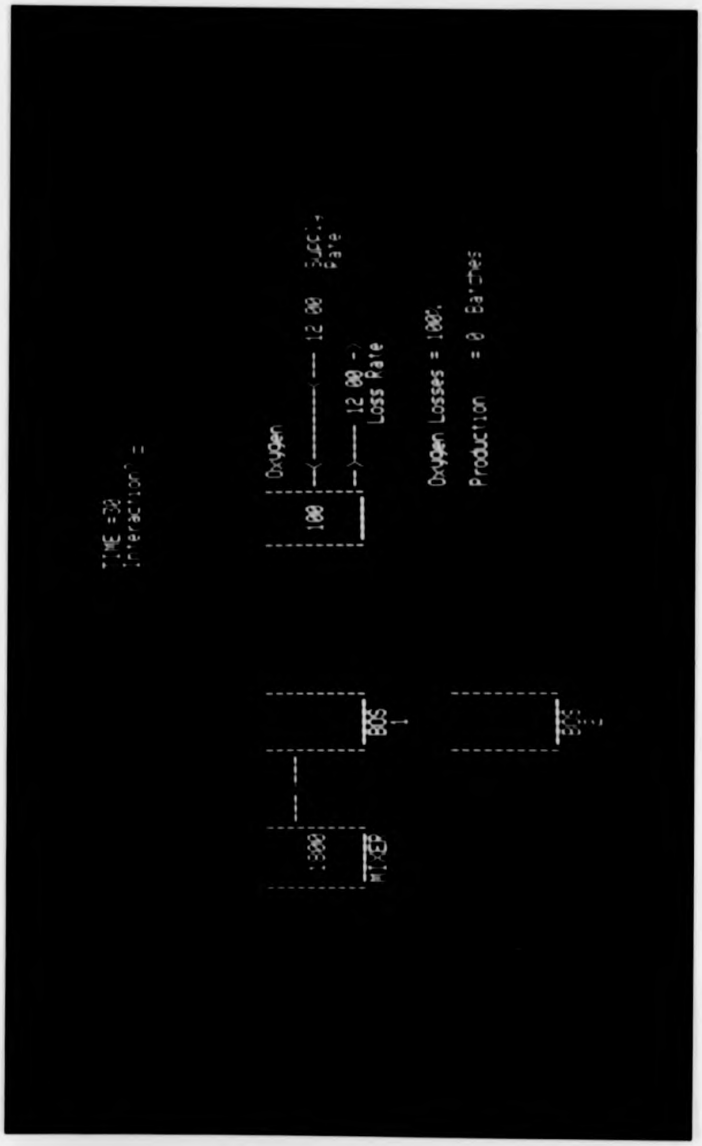
The three most important pieces of information provided by the model are the quantity of oxygen wasted, the amount of iron in the mixer, and the delays caused by a lack of oxygen. The first is shown on the main display as a percentage of the total quantity of oxygen supplied. The current level of iron in the mixer is shown on the main display, but it is also recorded in a histogram. Delays to stage 1 blows are shown in histogram form, but they may be observed on the main display due to the dynamic nature of visual simulation. These items are updated every 15 minutes (simulated time), or whenever an event forces the display of a current value.

The units of measurement used to express the values of different items may seem inconsistent, but an attempt was made to express quantities in whatever units were normally used for the purpose. For example, the oxygen supply is shown in tonnes per hour, but the amount in the storage tanks is expressed as a percentage of maximum capacity. Although this required special conversion routines within the program (as a single set of units must be used internally), it was felt that this would make the model easier to use, and increase the user's confidence in it - consider the importance of face validity in psychological testing (Anastasi, 1968, page 104). The major exception to this is that time is expressed in minutes since the start of the first shift on Monday, instead of days, hours, and



Photograph 3

The BOSP model in action



Photograph 3

The BOSP model in action

minutes. This was justified because the model was not intended as an on-line decision tool, but as an aid to understanding and investigating the problem. This meant that the primary user would be an O.R. analyst, familiar with this method of representing time in a simulation model. Another drawback of the natural expression of time is that it could make the entry of long production schedules even more laborious. Modification of the program to show the day and time would have been a simple matter.

The main fault with the displays is that it is not possible to determine the state of a BOSP vessel unless it is idle. There is no visible difference between the two blows, or between waiting for a blow to start, waiting between blows, and tapping (taking the newly made steel from the vessel). To this extent the design criteria were not met. There are at least two possibilities for coding the various stages. Differentiation could be achieved by the use of a selection of colours when displaying the vessel. This would be feasible due to the small number of states to be distingushed, but it would require the use of the ISC or a similar colour terminal. Such devices are generally larger, and therefore less portable than an ordinary vdu, and at the design stage it seemed likely that it would be necessary to run the model in Corby. An alternative is to represent the contents of the vessel with different symbols or textual labels (e.g. "BLOW 1") according to its state. This has the advantage of operating with any type of terminal. The need to use a

standard vdu also meant that a keyboard was the only input device available.

Interactions

It was necessary to develop a small number of problem-specific interaction facilities. These are described below.

Interaction	Description
SO	Schedule Oxygen. Under the terms of the contract, notice must be given of required changes in the oxygen supply rate. This interaction allows the user to specify one or more rate changes, requesting the time of the change and the new rate.
SC	Schedule Charge. Working in a similar manner to SO, this interaction allows the user to specify the times at which the model will attempt to start the production of a fresh batch of steel. The vessel to be used must also be specified.
OTHERS	There are a few processes apart from BOSP which use oxygen from the storage tanks. This interaction sets the total rate at which these processes consume oxygen.

TIGHT — This schedules enough batches of steel to allow the model to run for three weeks. Work on each batch starts as soon as the previous one is completed.

LOOSE — Similar to TIGHT, but batches are scheduled at 60 minute intervals.

OXRATE — Allows changes to the oxygen supply rate with immediate effect. Intended for use in conjunction with TIGHT or LOOSE when specifying initial conditions for a non-interactive run of the model.

AUTO — This was added at a late stage to allow the evaluation of a particular operating strategy. When selected, the oxygen supply rate is set to its contractual minimum during venting, and to maximum when a batch is waiting for oxygen. No other changes are made. This strategy could not be employed (as notice of changes must be given), but it aided understanding of the system.

Use Of The Model

The use of the visual interactive simulation system made the task of model verification relatively easy. At that stage a small number of minor corrections were made, and some new information was incorporated into the model. An example was the replacement of the 'standard time' for

the duration of the sampling activity with a stochastic process, based on the recorded durations.

Once confidence in the model had been established it was used mainly in the batch mode, in order to save time. Extensive use was made of the dump and restart facility to allow the rapid comparison of a number of alternative strategies. BSC O.R. staff were impressed by the speed and ease with which this could be done.

Use of the model showed that due to the time lag between requesting an oxygen rate change and the change occuring, it is better to forecast an overall requirement for the week and adhere to that estimate without making tactical changes, as these may make an already bad situation worse. In practice, the problem with the ICV makes such fire-fighting even more hazardous.

By following this recommendation it should have been possible to reduce oxygen wastage to a figure within the new standard of 20 percent. Unfortunately it is not possible to report that this statement was proved to be correct, as the announcement of the decision to cease steel production at Corby coincided with the conclusion of this study. Against the grim background of impending closure, no further changes in operating procedure were made.

Conclusion

The most important result of the study was the realisation that the existing system did nothing to assist in the process of model formulation and development. This

was supported by the author's experiences in the Rolls-Royce project, and (in retrospect) by the projects carried out by other workers at Warwick (summarised in Chapter 1). Although visual interactive simulation allows a client to look inside a model, the model development process is still opaque to him. This is far from ideal, since much can be learned during the construction of a model (Buxton, 1968, pages 53-55) and it is the client, rather than the analyst who is in a position to act upon the knowledge thus gained. Greater involvement in the earlier stages of a project is also likely to give him a tool which meets his needs, and one in which he has confidence. This confidence would stem from a better understanding of the model's nature and any inherent limitations. The remainder of this thesis describes the author's attempt to make good this shortcoming, and suggestions for further improvements.

The steel-making study also supported previous findings that visual interactive simulation eases model validation, fosters confidence and encourages experimentation.

Chapter 6

VISUAL INTERACTIVE MODEL BUILDING

Examination of the descriptions of earlier visual interactive simulation projects revealed a weakness which was also experienced in the studies reported in this thesis: although the task of model validation and verification is eased by the facilities provided by the simulation system, there is nothing to assist in the more difficult area of model development. The tribulations involved in the development of a visual interactive simulation model may be understood by considering the steps involved (of course, parallels occur in the development of almost any type of model or program).

Once the analyst has been invited to tackle the problem experienced by the project's sponsor, the first stage is normally a discussion of the manifest problem, followed by a systems analysis to ensure that it is the underlying problem that is attacked. If the nature of the problems is such that visual interactive simulation is an appropriate tool (perhaps because it is not amenable to analytical methods of solution, or because processual factors are important - 'why?', rather than 'how?') the analyst will try to discuss the system with the potential user of the model in order to attempt to comprehend the way he sees the system. As noted above, it is likely that communications difficulties will be experienced in view of the differing professional backgrounds of the participants.

Having gained an understanding of the system under study, and the way it is viewed by its controllers, the analyst may start to build the model. The logic is defined in much the same way as with any other discrete simulation language (perhaps using entity cycle diagrams), but the display aspects and any dialogues must also be considered. These aspects are usually considered in parallel, since the simulation system automatically provides a visual representation of queues.

Displays have been designed either on paper (usually squared to represent the constraints imposed by a vdu), or by using the cursor control keys of an off-line vdu in order to place characters on the screen. The latter has the major advantage of showing how the finished display will appear.

Whichever approach is used, the analyst is faced with the tedious task of writing routines to produce the desired display. Although this is not difficult, there is plenty of room for minor mistakes (such as transposing x and y co-ordinates), and so this can be very time consuming.

When the initial model (with its associated displays) is running, it is quite likely that changes will be requested by the user, since he can now 'see' the model which was until now inside the analyst's head. The changes may be related to the logic of the model (in the Rolls Royce project described in Chapter 2, a misunderstanding of the priority mechanism was exposed by the visual representation), they may be cosmetic (e.g. a request for

the colour-coding of certain critical pieces of information in order to highlight out-of-tolerance items), or the display of additional pieces of information may be requested.

Once the changes have been negotiated, the analyst must go away and implement them as it is not feasible to make even simple changes there and then. For example, the client may ask for a minor change to be made in a display, and in order to comply the programmer must rewrite the appropriate portion of the program and re-compile before the result may be seen. On Warwick University's time-sharing system there is a comparatively low limit on the amount of process time taken by a compilation in order to provide a reasonable service to the majority of users. As a result, these large simulation programs must be compiled under the batch system. This may take several hours (during particularly busy periods), and so the immediacy normally experienced by the client during experimentation is lost.

A partial solution would be to arrange the program into modules which are separately compiled and then bound (or linked) together. If any changes were required, only the affected module need be recompiled, and then a process known as 'replacement binding' used to insert the new object module into the existing bound code file. This process works well with Fortran programs on the B6700 computer, but the use of Algol causes complications, and therefore this technique is not used.

When the analyst has coded the agreed amendments, he is faced with the process of removing the 'bugs' from the altered code. Another common mistake is failing to erase a piece of information before displaying its new value. For example, if '100' is overwritten with '20', the result will be '200', assuming numbers are left justified as is the usual practice.

Another difficulty is identified by Rubens (page 93). He observes that if the display generating code is entwined with the simulation logic any changes to the displays must be followed by re-verification of the model logic in case of side-effects.

The problems do not end when the model is accepted as an adequate representation of reality. Previously unimagined suggestions may be put forward - "What would happen if we installed a buffer between units 23 and 27?" The model can only be used to investigate such questions if it is altered to include such a buffer, and this requires a modification of the program. Even if a similar buffer exists between another pair of units, it cannot simply be replicated at run time. Another possibility is that after some experimentation, it is decided to 'de-aggregate' some aspect of the model. An example of this could have occurred in the BOS model, where a single entity represented the pair of independent mixers. No provision is made for the creation of a second mixer, and its connection it to the rest of the system, although the

capacity of the original mixer could be altered appropriately.

Another type of problem can arise when examining the effects of different operating rules. In discrete event simulation, it is common practice to define a 'pool' of entities which cycle through a number of activities, even though in the real system new entitities are created (for example, a model of a car assembly plant may contain a number of 'cars' being repetitively rebuilt). It may happen that while the declared size of the pool is sufficient for most circumstances, certain rules (or combinations of rules) may require a larger pool. This could occur if the size of the buffers in a decoupled production system are increased. When this occurs the program will call for operator intervention, but the user does not have the option of increasing the size of the pool at run-time. Other systems (e.g. SIMSCRIPT) avoid this difficulty by creating and destroying entities as required.

Thus there are two aspects of model development to be considered: facilities which lighten the programming burden, and those which increase the involvement of the client in the modelling process.

Client - Analyst Communication

Boothroyd's (1978) description of the process of O.R. includes the observation that not only is there a difference between client and analyst in terms of the range of possible actions visualised, but that the visualisation

or articulation of the alternatives is in terms of different words and images. This helps to explain the success of visual interactive simulation, insofar as such a model acts as a bridge between two or more professional cultures. However, such an effect is only possible once an initial model has been constructed. Client and analyst are still faced with the difficult task of communication with few external references until that time.

The problems of communication and problem definition are being tackled by Eden and his colleagues at the University of Bath. They are mainly concerned with 'soft', organisational problems, rather than the 'hard', operational areas which are the traditional fields of operational research analysts, arguing that concentration on tactical issues carries a risk of relegation to organisational backwaters.

Their approach (Eden and Jones, 1980) is to use a computer system to ease the task of integrating the subjective realities inhabited by the various actors, producing a 'cognitive map' of the problem area. This map is then used to define the problem faced by the group, and to explore the effects of policy changes on the things with they are concerned. It should be stressed that they see the role of the consultant as that of a catalyst, helping his clients define and solve their own problems. This contrasts with the prescriptive approach implied by much of the O.R. literature.

These concepts are consistent with some of the findings of early visual interactive simulation projects. Bowen, et al, (1979b) report that their client's commitment to the project was enhanced by the interactive nature of the model, while the dynamic visual display provided a channel for the rapid dissemination of information to the other people involved in the real system. However, they also note: "[the client's] requirements, when using the model, depend partly on how he currently sees the problem" (page 841).

The Experimental Development of a Prototype System

The above findings indicate that the visual interactive simulation system would be greatly enhanced by the addition of facilities which allow the interactive construction and modification of models and their associated displays. The 'program generator' approach has proved successful for conventional batch simulation (Davies, 1979). Such generators often use an interactive questionnaire to obtain the necessary information from the user, as pioneered by Clementson (1980b). This approach was rejected for two reasons. Firstly, it is not easy to describe a display in words and numbers until it has been designed, and it was intended to produce a system which would aid the user in that design task. Secondly, in an experimental system it is desirable to allow the maximum flexibility in order to deal with new situations and entity types (such as Fisher's extensions to allow the modelling

of continuous processes). If a program generator was developed at this stage there would be a risk that the underlying simulation software would be 'frozen' prematurely.

Consideration of the factors discussed above suggests that what is required is the provision of facilities which:

allow the alteration of the display aspects of the model without changing the program;

allow entities and other model constructs to be created interactively;

and which allow the behaviour of entities to be changed at run-time, without restricting the choice to those behaviours which were specified when the model was initially constructed.

A decision was made to extend the existing simulation software rather than build afresh for pragmatic reasons. If the new system was upwardly compatible with the old one, it would be possible to use existing applications programs without modification. The question of compatibility caused several problems when changes were made in the past. It may well be necessary to completely re-write the system in the near future, but the existing structure was adequate for the purposes of the current research.

Display Design

The provision of display design facilities will be discussed first. The most common way of designing a

display is to use pencil and paper (normally graph paper or a coding form). This usually involves extensive use of an eraser, and when the layout is transferred to the screen of a computer terminal the results can be disappointing, due to the difference between the aspect ratio of hand written and 'electronic' characters (the reader is invited to compare Photograph 1 and Figure 2). It is possible to work at a terminal which is not connected to a computer in order to produce a meaningful and aesthetic display (the display shown in Bowen, et al (1979a, page 40) was developed with the aid of an 'intelligent' terminal which provided simple editing functions and the ability to save the an image of the display on diskette for later inspection), but this still leaves the tedious and error prone task of writing program code which will reproduce the image, and if any changes are requested the code must be altered.

If the terminal was on-line to a computer which was running a suitable program, it would be possible to record the shapes produced by the user, eliminating the coding stage. In this way the information about the display is taken from the program, and placed within the data structure. This simultaneously satisfies the requirements that the programmer be relieved of the task of writing program segments to generate the display, and that it should be possible to alter the display while the model is being used.

Figure 2

The original design for the main display of the Rolls-Royce Model

Graphics

Having chosen this route, it became necessary to identify an appropriate data structure for the graphical information, subject to the constraint of compatibility with the existing system. The area of computer graphics has received much attention over the years, having implications for Computer Aided Design, Computer Aided Instruction (or Learning), and other exploratory areas including the general area of simulation. Burchi (1980) provides a summary of the available technology and its applications.

An early and influential system called Sketchpad was developed by Sutherland at M.I.T. (Sutherland, 1963). For the present author's purposes, the most important concept was that copies (or 'instances') of simple elements, each instance having attributes of position, etc, may be combined to give a more complicated image, and that any changes made to an element (or 'subpicture') affect all uses of that subpicture. For example, a half-hexagon subpicture may be defined, and two instances displayed to form a hexagon. If the definition is changed to that of a semi-circle, the hexagon immediately becomes a circle. Moreover, if the hexagon had been used as part of another shape, the appearance of all occurrences of that shape will change accordingly.

There is a resemblance between Sutherland's instances of subpictures and the nature of an entity in the simulation software, as both are distinguished from others

of their type by attributes. An entity has attributes specifying things like its name and timecell, while instances have position, size, and rotation, plus a pointer to the subpicture itself. By combining the two approaches, simulation entities may be seen as instances of various prototypes from which they get their pattern, but with each having its own position in addition to other attributes.

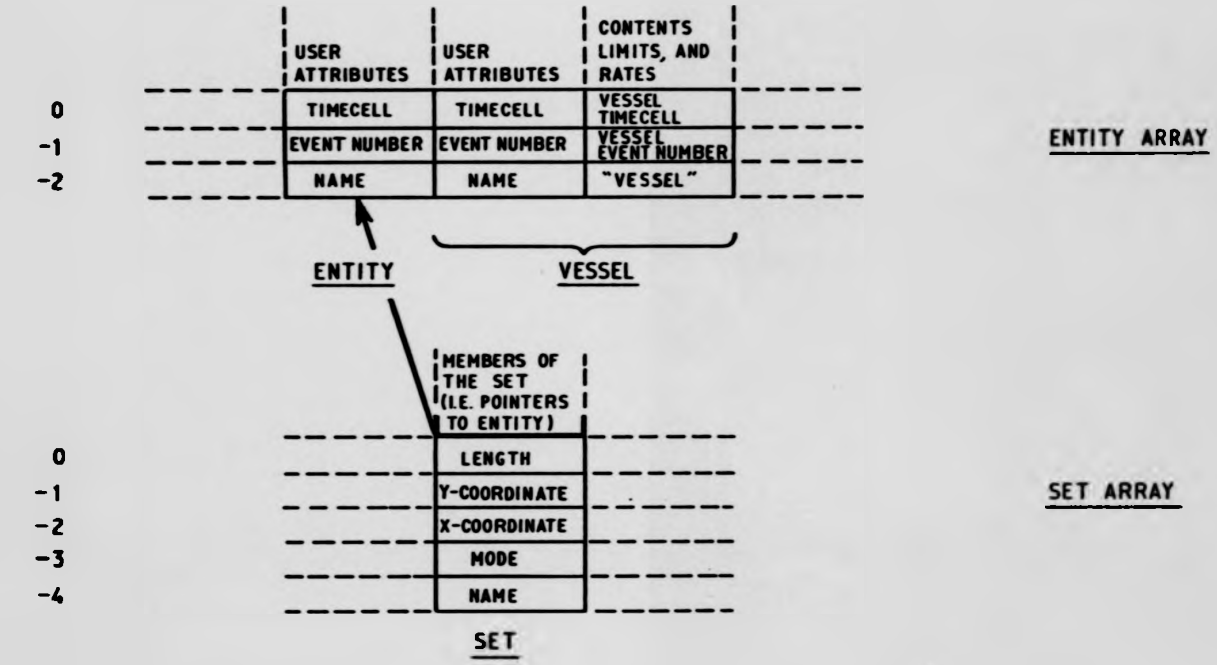
Implementation

In the existing simulation system each entity has a number of attributes. Some of these are used to record system information (such as the entity's name, and the time at which it will next take part in an event), and the remainder may be used for any purpose the modeller sees fit. In order to implement the new features a number of extra system attributes were defined. These specify an entity's type, screen coordinates, logical screen number, and colour. 'Logical screen number' refers to the method used for displaying information on a terminal. Each piece of output is directed to one of a number of logical displays, and according to the settings of a set of software 'switches' it is then sent to a combination of the attatched terminals. For example, if two terminals are in use in one place, it is likely that they will be used to display different pieces of information, perhaps a dynamic process display on one, and a histogram of some measure of performance on the other. If the terminals were in different locations (which may be seen as a specialised

form of teleconferencing) it is more likely that they will be used to show identical displays. These and other effects may be obtained by specifying the appropriate logical to physical screen mapping. No coding beyond that needed to generate the logical display is needed in the user program, as the system software provides an interaction to change the switch settings at run time. The data structure used in the existing system is shown graphically in Figure 3.

The value of an entity's type attribute is stored as a real variable. If an attempt to match it with the name of a special (pre-defined) type fails, it is treated as a pointer to a user-defined type. Links (discussed below) are an example of such a special type. Reference to Figure 4 may clarify the description of the extended system.

The graphical aspects of entity types are stored as 'display sets'. These are maintained in a similar way to regular simulation sets, and hold a number of 'display elements'. These display elements are special purpose entities and have three attributes. One attribute acts as a pointer to a string of characters which make up a section of the image of the entity. More specifically the string is a sequence of consecutive, horizontally adjacent symbols, with spaces included only when explicitly entered by the user. This condition is necessary to avoid unintentional erasure of part of another image. The two remaining attributes locate the string relative to the current position of the entity. This position is defined



NB The structure used for histograms and distributions is not shown.

Figure 3

Data structure of the existing system

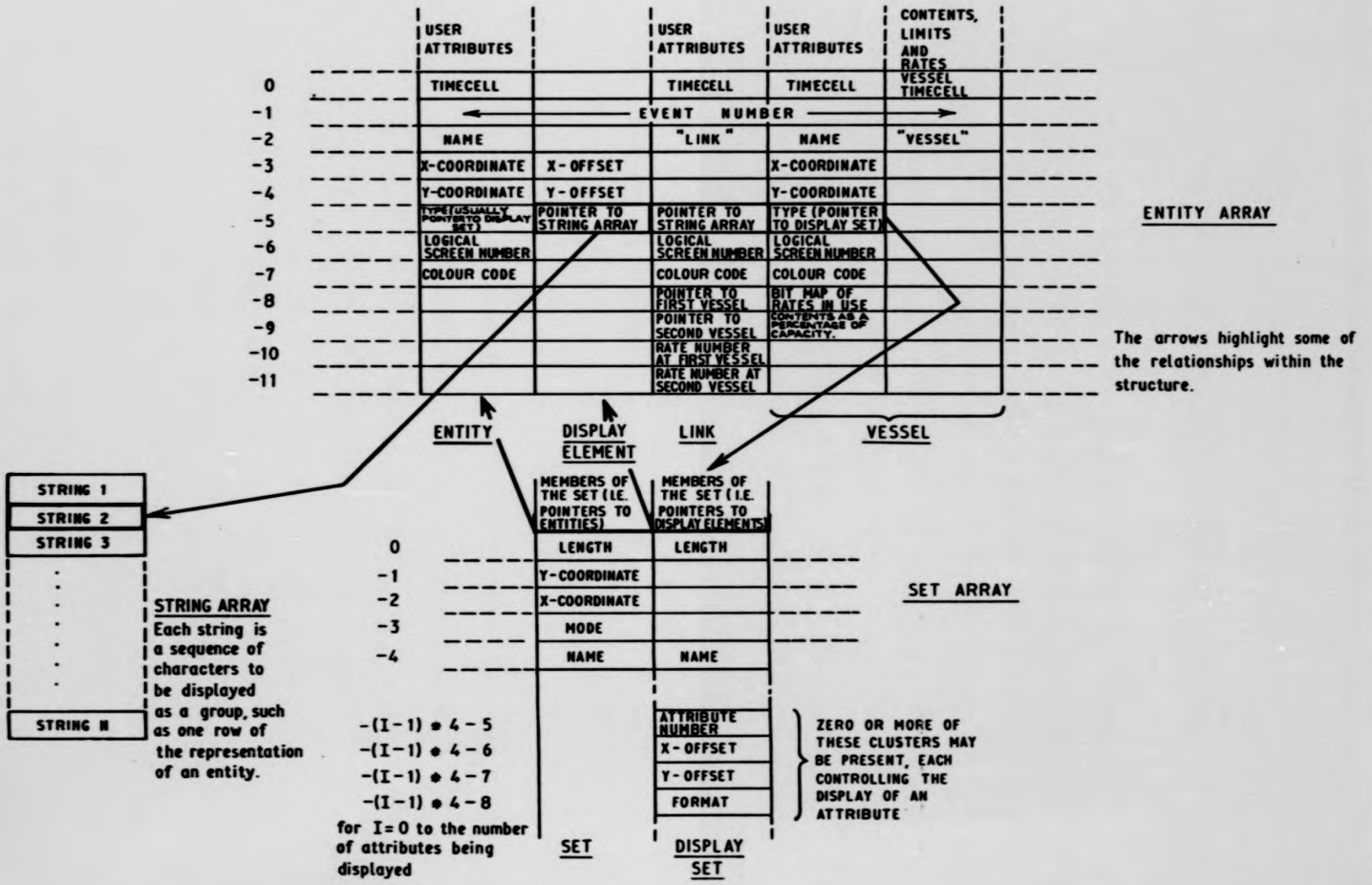


Figure 4

Data structure used in the extended system

to be the coordinates of the lower left corner of the image.

The display element strings are held in an array, and duplication is avoided by testing for the existence of a matching string before storing a new one. In practice, the amount of memory used to store these strings is such a small proportion of the total that the benefits are not significant. In the light of the inefficiencies elsewhere in the system, the avoidance of duplication should be seen as an acknowledgement of the need to consider such factors in a commercial environment.

The user may interactively design the visual representation of an entity type. Symbols are entered from the keyboard, using the cursor control keys (up, down, left, and right) to move about the screen. Within the program's data structure, a conceptual cursor (i.e. a pair of pointers) moves about an array of characters which acts as a map of the screen. Thus at all times a correspondence is maintained between the program and the display.

This correspondence is achieved in part by the use of a special input mode where each key depression is read as a complete record. This facility caused additional software complications by interfering with the special input mode used to detect a 'break' signal while a model is running. To avoid problems it was necessary to close the interactive files and re-open other output-only files before opening the single character input file. This difficulty appeared because the two special input modes were not intended to

work together, rather than any inherent weakness in the design.

The user signals the completion of this design phase by sending the Null character (control-@). On receipt of this code the program returns to its normal mode of input/output. The screen map is then analysed in order to identify the required display elements. These elements are created, and their attributes given values appropriate to their position and corresponding string of characters. Once created, the elements are placed in the new display set.

Shapes may be re-defined by editing their existing appearance. Operationally, this is very similar to defining a new shape, except that instead of being blank, the screen initially contains a copy of the existing image.

In general, it is not sufficient to place a static image of an entity on the screen, even if it is possible to move it around. In particular, it is necessary to have some means of identifying one entity from others of the same type. This is achieved by allowing the designer to specify that a number of attributes of the entity may be displayed at locations relative to its position on the screen. The value of an attribute may be shown in four different ways. The first three are similar, interpreting the value as a string, or as a real or integer quantity. The use of attributes to store strings is limited to a maximum of six characters, but this has been found to be sufficient for items like the name of an entity. The

fourth display method treats the value of the attribute as a code. The value is rounded and is used as an index into a table of string variables, each of which may contain up to $2^{16}-2$ characters. The string corresponding to each code is entered by the user as part of the type definition process (see Appendices 4 and 5 for further details).

Although it is intended that all 'pre-defined' entity types should use the structure described above, exceptions must be made when entities of a particular type do not share an identical representation. An example is the links which are used to connect vessels. While all links behave in a similar way their precise appearance depends on the twists and turns needed to join the two vessels. Additional complications arise when vessels are moved, as it then becomes necessary to change the shape of the link to maintain the visual connection. The use of the graphics facility on the ISC terminal also meant that the existing structure would not be suitable. Instead, each instance of a link has a pointer to a string which (when sent to the ISC) results in a line with the correct appearance being drawn. When the shape is to be manipulated the string is decoded into a list of co-ordinates specifying the end points and corners of the line.

Continuous Process Facilities

The behaviour of 'vessels' (Fisher, 1981), which are used to model continuous processes (such as chemical manufacture), is well defined. An event occurs whenever a

vessel becomes completely full or empty, or when its contents reach an upper or lower warning level. When such models are being interactively developed, these events simply tell the user what has happened and then enter the interaction routine to allow him to take appropriate action. To assist in the construction of networks of vessels, additional entity types have been developed. These are the link, the source, and the sink.

Links

Links were developed to simplify the construction of networks of vessels. Once the vessels have been created, a link may be drawn on the screen as described above, and the logical connection is achieved by recording as attributes of the link, the identity of the two vessels concerned and the number of the 'hole' being used on each vessel. Before the development of links, it was the responsibility of the programmer to relate the flow through a hole to that of another vessel. The system software aggregates the flows in and out of each vessel in order to obtain a net rate which is used to determine the occurence of events.

To produce the representation of a particular link the light pen is used to indicate successive key points on the screen (which still shows the current display). These points are the two ends and the corners. As they are entered, the lines are

drawn in order to allow the user to see what he has done so far.

Like any other type of entity, a link's representation may be changed. The technique used is to point the light pen at a corner or endpoint, and then at the desired location for that point. The program erases the existing part of the line, and then draws the new section. This has a major weakness in that new corners cannot be defined, and if a corner is straightened out, it is very difficult to find again! This has not been a particular problem when testing the system, but if it ever became necessary to make a major alteration to a display it seems likely that it would be necessary to define one or more new links to replace existing ones which would require radical changes.

The existence of links increases the potential simplicity of interaction with the model. For example (assuming certain improvements had been made to the graphics software) it would be possible to change the rate of flow between two vessels by pointing to the link with a light pen and then keying the new rate. The direction of flow could be specified by moving the light pen along part of the link in the appropriate direction, as part of the action of selecting the link. The current method is to type the names of the two vessels, and the program responds with the current rate and direction

of flow. A new rate may then be entered, prefixed by a minus sign if the direction is to be reversed.

Once links had been built into the system software, the task of providing lagged rate changes was simplified, again because the system keeps track of the holes used for a particular connection. When a lagged change is requested, the rate of flow from the source vessel is immediately altered. A dummy entity (i.e. one which has no equivalent in the real system) has its attributes set to record the new rate, destination vessel, and hole. An event is then scheduled to occur after the specified time-lag. This event makes the changes necessary to the rate of flow through the destination hole.

Sources and Sinks

The difference between sources and sinks is cosmetic. Both are actually vessels with maximum and minimum capacities which approximate plus and minus 'machine infinity' respectively. The difference between the two shows only when creating the entity: if it is to be a source, the system asks which vessel it feeds, and in the case of a sink, which vessel feeds it. As both start with a level of zero, it is easy to determine the quantities which have flowed in or out of the system. A refinement would be to automatically refill or empty the vessel when the limits are reached, keeping a

count of the number of times this was done as an attribute of the entity. This would avoid problems when very high rates of flow were used, and/or the simulation run for a long period.

Model Logic

Although the ability to interactively produce a visual representation of the model can help to bridge the gap between analyst and client, there is much to be said for the ability to manipulate the model even though it is incomplete. For this reason the creation of an entity or other item actually occurs within the data structure of the model. In this way the various system interactions are available to change the state of the model. For example, entities may be placed within, or moved between sets. Given access to the random number generators, it is possible to hand simulate the working of a queueing model, with the computer moving the entities around the system. Use of the model in this mode may (in some relatively simple cases) be sufficient, but even when it is inadequate it may still be beneficial. This is because the bridging effect referred to at the beginning of this paragraph can help the analyst learn about the system.

It will be recalled from the summary of previous research that one of the advantages of visual interactive simulation it that it eases the task of model validation, since the users are given a window into the model. By starting without programmed control rules and allowing the

controller of the real system to operate the model, the analyst is in a better position to learn about the system prior to coding the control rules. This co-operative use of the system should also give the client (especially if he has not previously encountered visual interactive modelling) an opportunity to consider how the finished model will be used, which will give the analyst a chance to develop the kinds of interaction facilities which will be needed, rather than simply relying on his own imagination and experience (which as we have seen, is likely to differ from that of the actual user of the model).

This early model manipulation is only possible because the behaviour of certain types of entity can be specified in advance. In queueing models, entities join a queue, get transferred to another, and so on. There is no great difficulty in manipulating the model to correspond with thoughts like "Initially load all machines by hand... expedite JOB 8... load JOB 4 on the CUTTER" (Hurrion, 1978; on the job-shop scheduling problem).

The user as a control entity seems to be a powerful aspect of interactive modelling. It has already proved useful in experimental projects (e.g. Hurrion, 1978), as well as in model development (Nelson, 1979). Its potential could be harnessed in better ways than it has been in the system described here. A possible approach would be to have a system event (which shall be referred to as CALLUSER). Associated with the user would be a set of 'things to do' (dummy entities), arranged in chronological

order. Each time CALLUSER occurred, the message corresponding to the head of this list would be displayed for the user to act upon, and then the item would be removed from the list. CALLUSER would then be re-scheduled as appropriate to the next item on the list. This list could be viewed as a diary or as an agenda (cf Greenberger and Jones, 1967). Part of the user's action might be to schedule another event, perhaps the start of the next occurence of the activity. Having determined the time at which it is to occur, a dummy entity would be placed at the appropriate point in the list, with attributes set to indicate the time of occurence and the message to be displayed. If the entity entered at the top of the list, CALLUSER would be rescheduled to the appropriate time. This arrangement would free the user from much of the book-keeping associated with hand simulation, while avoiding the need to write model-specific code at the early stages of the project, and it would work just as well with queue-oriented models as with networks of vessels.

The main weakness in this proposal concerns events which can only occur under certain conditions, in our example this is typified by the operating rule that a 'stage 1 blow' cannot start unless the oxygen storage tank is at least 80% full. An obvious way of handling such events is to provide a command to put the message back into the queue to resurface one time unit later (or after other event notices scheduled for the current time, if any exist). This is not really satisfactory, but the provision

of facilities allowing the entry of statements about the logical structure of the model would require radical changes to the structure of the simulation software. This point will be taken up again in Chapter 7. In the meantime, it is possible to pre-define libraries of entity types and their associated logic as has been done with vessels.

Examples of the Use of the On-Line Model Development System

The facilities provided for model development may be described most easily by an example of their use. A network of vessels will be described, since this entity type is well developed (i.e. a 'library' exists), and to make the example as concrete as possible, it will be taken from the steelmaking study reported in Chapter 2 (this model has been successfully built anew using the development system).

A skeletal program is still necessary, although the one attached as Appendix 3 will suffice for most applications of this type. It is unlikely that at this stage any special coding will be required.

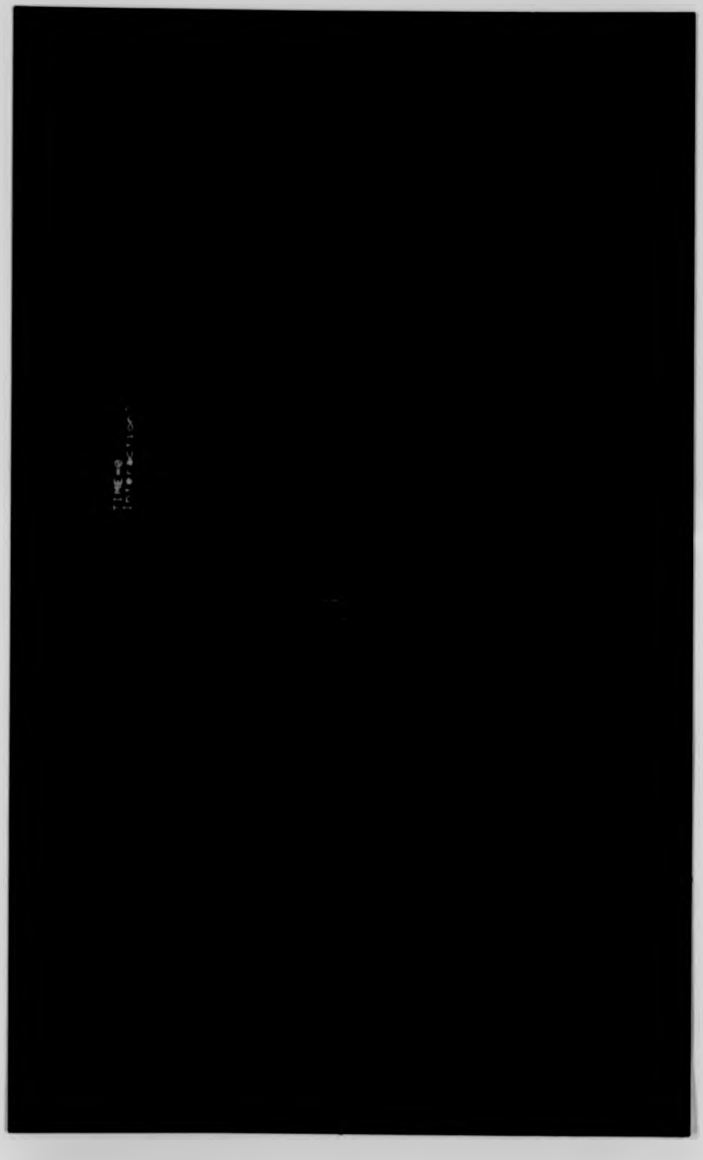
When the program is run, 'design' mode is entered automatically. The distinction between 'design' and 'run' modes is not strictly necessary, but it was felt that it might impose a certain discipline, discouraging changes in a model in the course of a run which could lead to meaningless results. A second possible advantage is that it restricts the number of commands which are valid in each

context, making the provision of dynamic function keys more feasible. Tesler (1981) argues against such modes, claiming that they make life much more difficult for users by restricting access to certain functions to particular circumstances, even though it would be perfectly sensible to use them in others. As an example, he describes a scene where a user of a text editor wishes to examine his file directory to identify the file from which he wishes to copy a portion of text. He can only do this by leaving the editor, entering the filing system, listing the directory, leaving the filing system, re-entering the editor, and returning to his place in the document. Surely this is not what is meant by a 'user-friendly' system?

The problem seems to lie in the determination of the extent to which users are protected from their errors. For example, if pressing control-D always deletes the 'current' file, then if the file was not visible (perhaps the output from an interactive program is occupying the whole screen), then a user could quite easily press control-D instead of shift-D when 'talking' to the program, not see any response and therefore assume that the keystroke did not register. He might then re-type the instruction (correctly), and press on. Eventually he tries to look at his file, but it has vanished. If the system can only recover to the state it was in before the last command (to cater for noticed errors), our user has a problem. This could possibly be extended to a larger number of commands, but where is the line drawn?

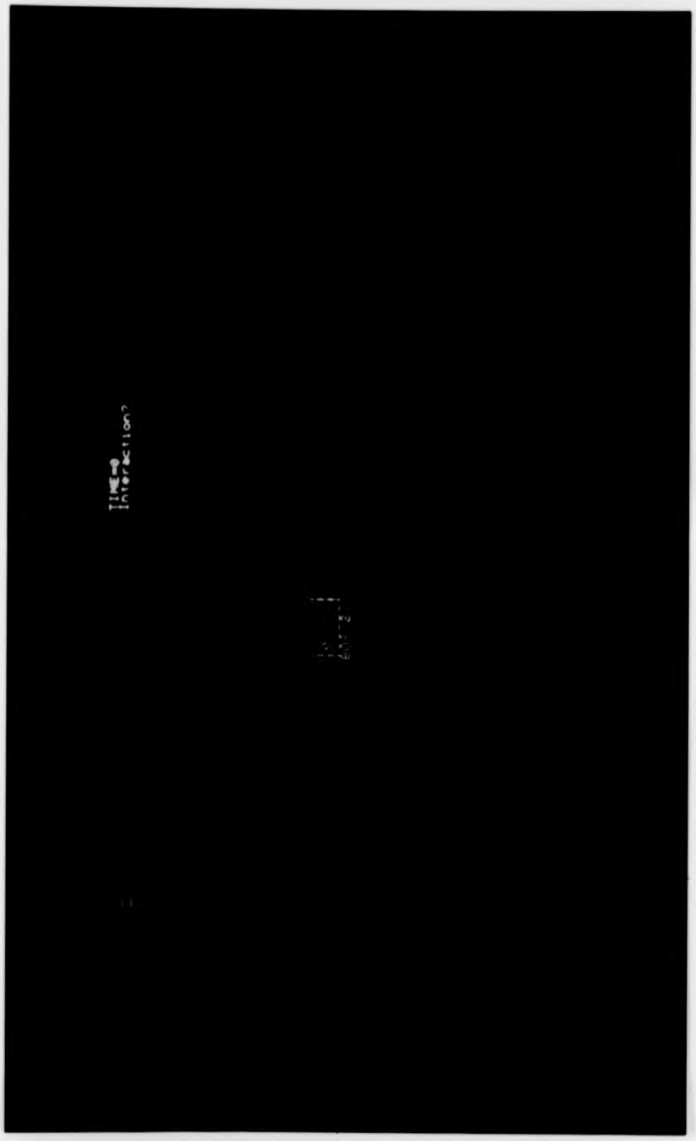
Our example will begin with the creation of the BOSP vessels. The command to create a new entity of an existing type (either user-defined, or from a library) is NEXT (as in 'create next entity'). The system asks for the type to be used, and checks that such a type exists. The name of the entity is entered (a maximum of six characters at present, but this has been found to be sufficient in practice), along with its position, foreground and background colours, and the logical screen number on which it is to be displayed. In the case of vessels, the maximum and minimum contents and warning levels are also specified. All these pieces of information are stored as attributes of the entity, and so may be altered either by interaction or by statements within a program. While it is possible to change an entity's type, this practice is inadvisable since certain types are treated as special cases when the original simulation system's data structure proved insufficiently flexible. Such a type is the vessel, which actually consists of two simulation entities. One may be used in the same way as any other entity, but the second provides the characteristics associated with a vessel (such as capacity and rates of flow).

This process is repeated for the second steelmaking vessel, resulting in the display shown in Photograph 4. Vessels representing the mixers and oxygen storage tanks are created in a similar way to the steelmaking vessels (see Photograph 5).



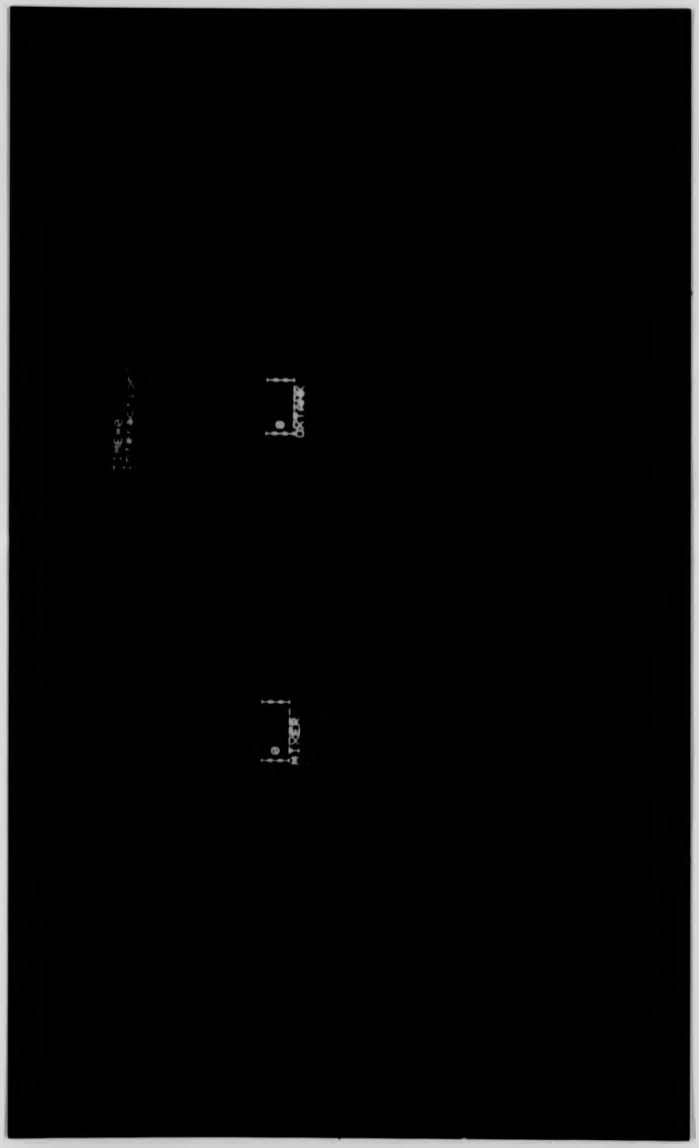
Photograph 4

The steelmaking vessels



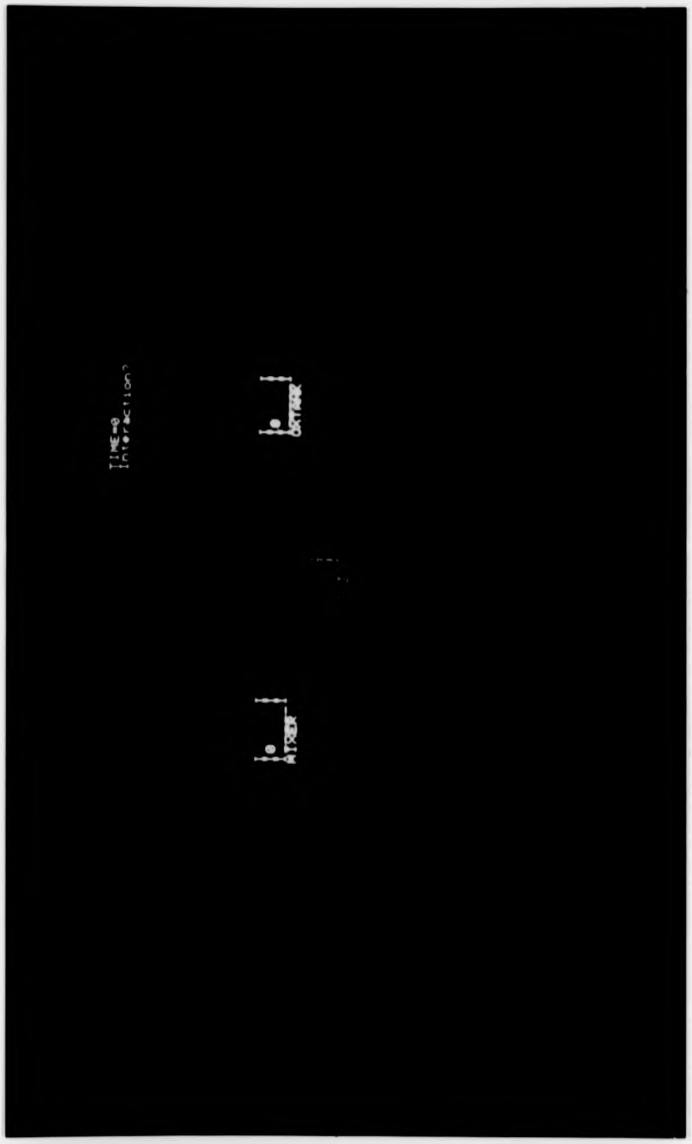
Photograph 4

The steelmaking vessels



Photograph 5

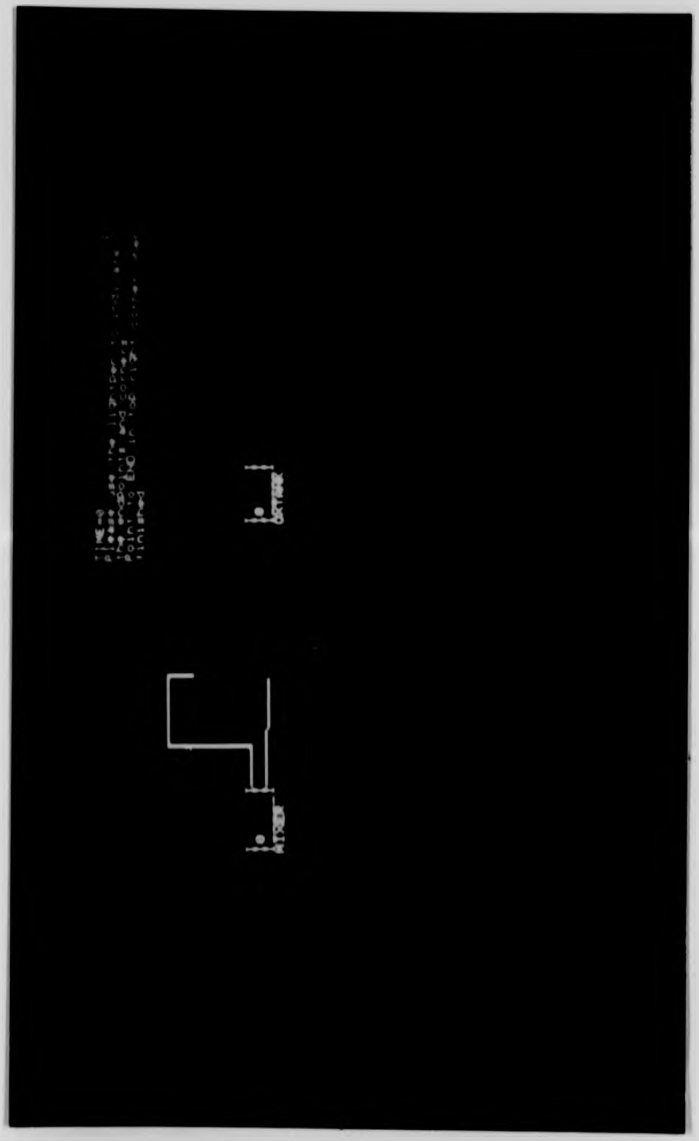
The steelmaking vessels, mixer, and oxygen storage tank



Photograph 5

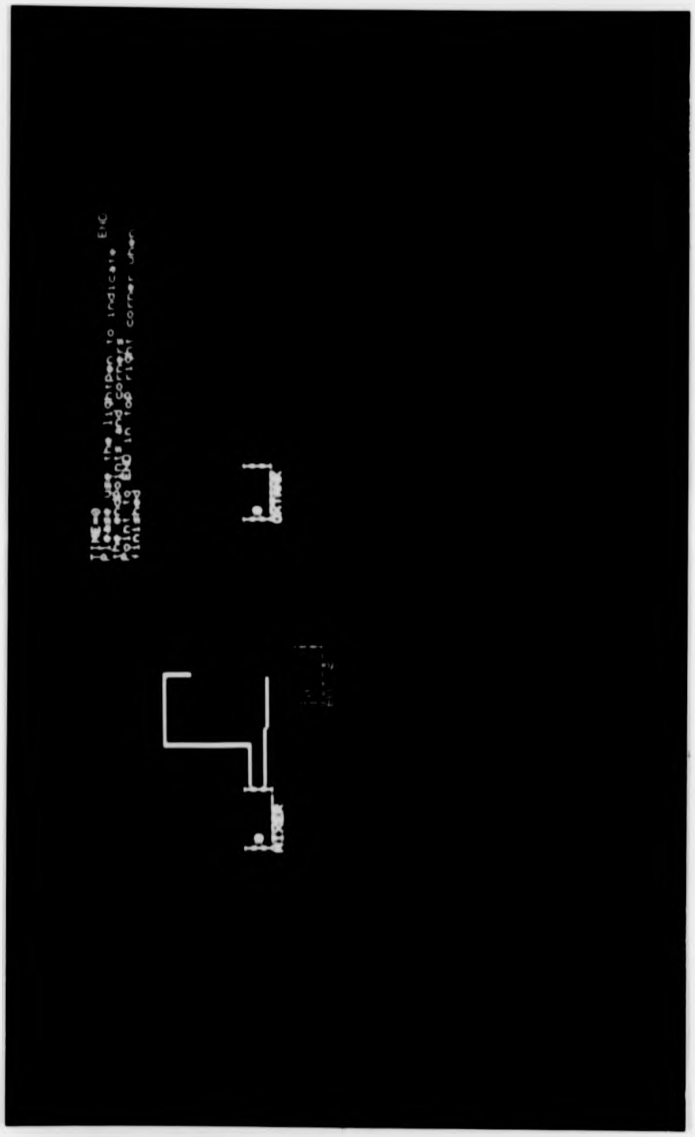
The steelmaking vessels, mixer, and oxygen storage tank

The next step is to connect the vessels together. This is achieved by creating a number of Link entities, which have attributes specifying which pair of vessels they connect, and by which 'holes' (each vessel may have a number of flows in or out, the maximum number being specified in the data file). Once again, the NEXT interaction is used to create a new entity. Entering 'LINK' as the entity type results in the user being asked which two vessels are to be connected. The system checks that the names entered are those of vessels, and that both have a spare hole. The light pen is then used to draw the representation of the link on the screen. Working from one vessel to the other, the pen is pointed at the corners of the required line (with care, it is possible to draw embellishments like arrowheads to enhance the display). A 'blob' acts as a cursor, helping to aim the pen, and when the desired position is reached the tip of the pen is pressed against the screen, which sends the coordinates of the 'blob' to the host computer. As successive points are entered the line is displayed on the screen (see Photographs 6a, 6b, and 6c). Since all the lines are often required to be either parallel or perpendicular, an option is provided which averages the endpoints of lines which are nearly vertical or horizontal, making them precisely so. The tolerance is currently fixed at ± 2 pixels in the x-axis and ± 4 in the y-axis, but it could be advantageous to allow the interactive alteration of these figures, since the



Photograph 6a

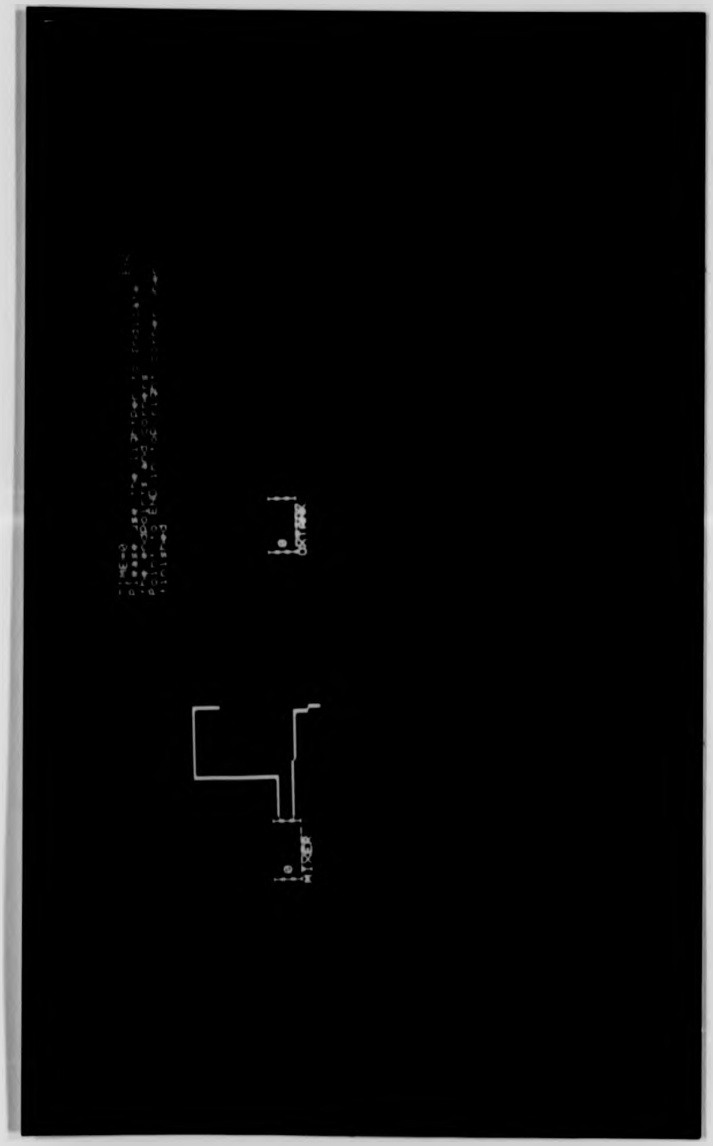
Drawing a link



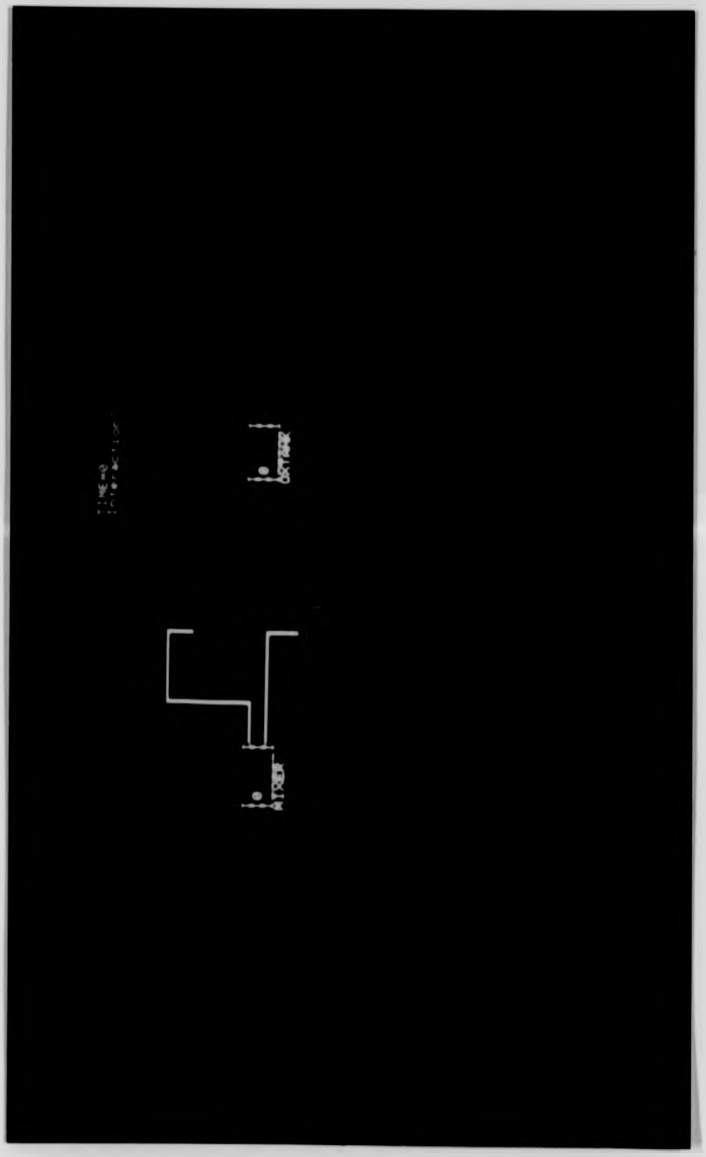
Photograph 6a

Drawing a link

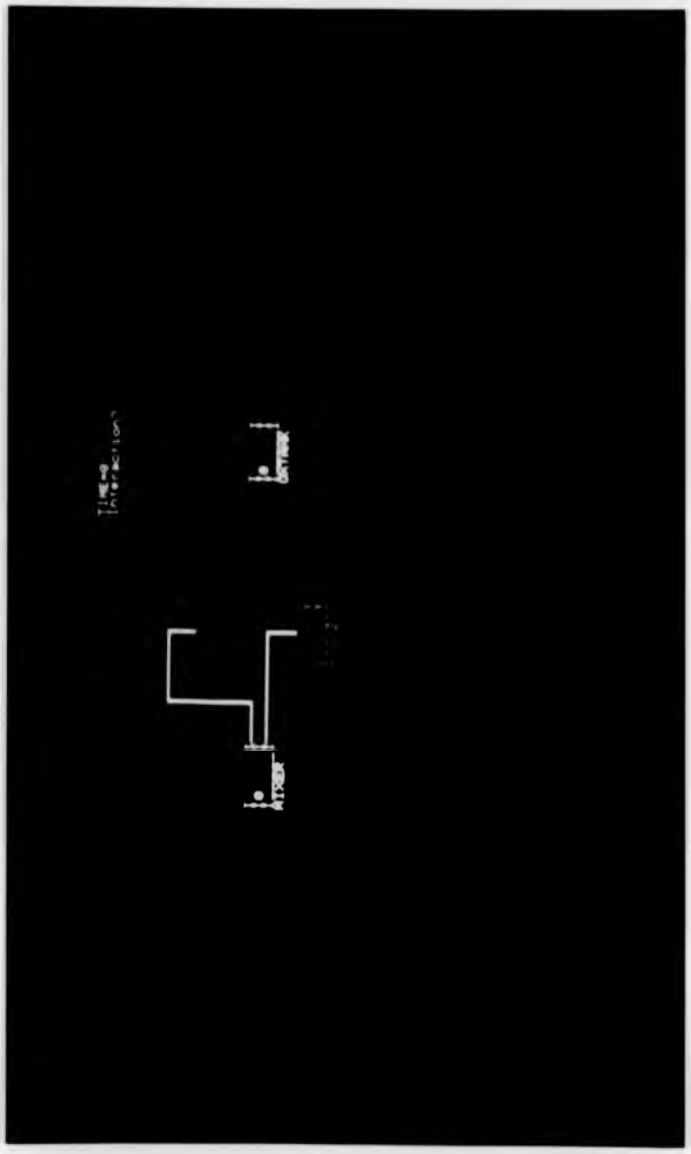
Photograph 6b



Photograph 6b



Photograph 6c



Photograph 6c

precise position of a line may or may not be important, depending on the number of elements already on the screen.

Provision for sources and sinks has also been made. In this example sources will be used to represent the blast furnaces and the oxygen plant, and sinks to collect the manufactured steel and 'used' oxygen. The only difference between a source and a sink is the direction of flow: they are both represented in the system as a vessel with very large maximum and minimum capacities ($\pm 54*10^{10}$ units), with an associated link to connect them to a vessel. When a source or sink is created (using NEXT), the user is asked to which vessel it is to be connected, and the initial rate of flow into or out of the system.

As it is the nature of visual interactive simulation to show the way in which a system proceeds through time, the provision of a means of recording changes within the system has been found useful. This has been achieved by logging the values of one or more variables (such as the quantity of iron in the mixer) at regular intervals, and then plotting this record as a time-series graph. This facility has been included in the interactive development system, and is used by typing 'TSER' in response to the "Type of entity?" question in the NEXT interaction. The routine requests a name for the time-series, the frequency with which recording is to occur, what is to be recorded, the length of the time-series, and the maximum and minimum values the attribute is likely to take. At present it is only possible to record the value of an entity's attribute,

but the evaluation procedure used by the system to update the display may be extended to set an attribute to a queue length, or even some function of a collection of parameters.

In the present example the dialogue is as follows:

```
Interaction? NEXT
Type of entity? TSER
Name of timeseries? MIXERC
Recording interval? 15
Record attribute of which entity? MIXER
Record which attribute of MIXER? 6
MIXER is a vessel. Record System or User attribute? SYSTEM
Length of time-series? 100        (enough for a week's work)
Maximum value? 2000
Minimum Value? 0
Timeseries MIXERC is on LSN 21.
```

Whenever logical screen 21 is switched on, the time-series will be dynamically displayed on the terminal; it may also be inspected by using the TS interaction which produces a temporary static display.

The use of empirical distributions in the steel-making simulation was discussed in Chapter 5, and such distributions may be entered into the model either interactively or by using the existing procedures. When the interactive method is used, the name and number of cells is typed in, followed by the upper bound and the count for each cell (the lower bound of the first cell is

assumed to be zero). Figure 5 shows an example of such a distribution.

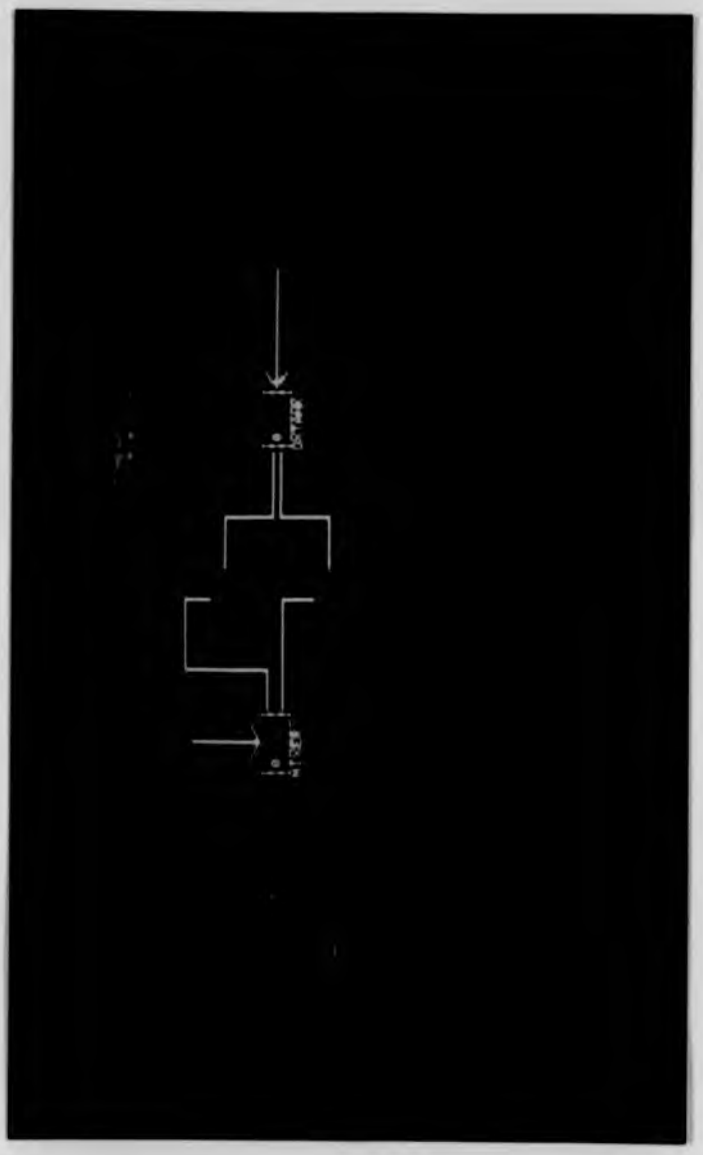
We now have an initial model described by the display shown in Photograph 7. It would now be prudent to save the state of the model by using the DUMP interaction. This writes the information to a disk file, from where it may be retrieved with the REST (restart) command. When a modelling session is finished (using END), the system saves the state as dump 0, and when another session is started it is automatically retrieved. If the model is not in a suitable state when the run is ended this feature may be bypassed by using KILL in place of END.

Typing the command 'RUNNER' puts the simulation system into run mode, where starting conditions may be set (e.g. the contents of the mixer specified). Once this is done, the model may be run under the user's control. There are four ways of proceeding: RUN will cause the model to run forwards through time until an event causes the model to return to the interaction mode, or until the user presses the Break key; GOFO allows the user to specify a (simulated) time at which the interaction mode is entered (unless an event causes this at an earlier time); A advances by one time unit; and E advances to the next event. At this stage it would probably be best to use A to step slowly through time, using other interactions to trigger events as appropriate.

For the purpose of illustration, let us say that a train-load of iron arrives at time=10. The RATE

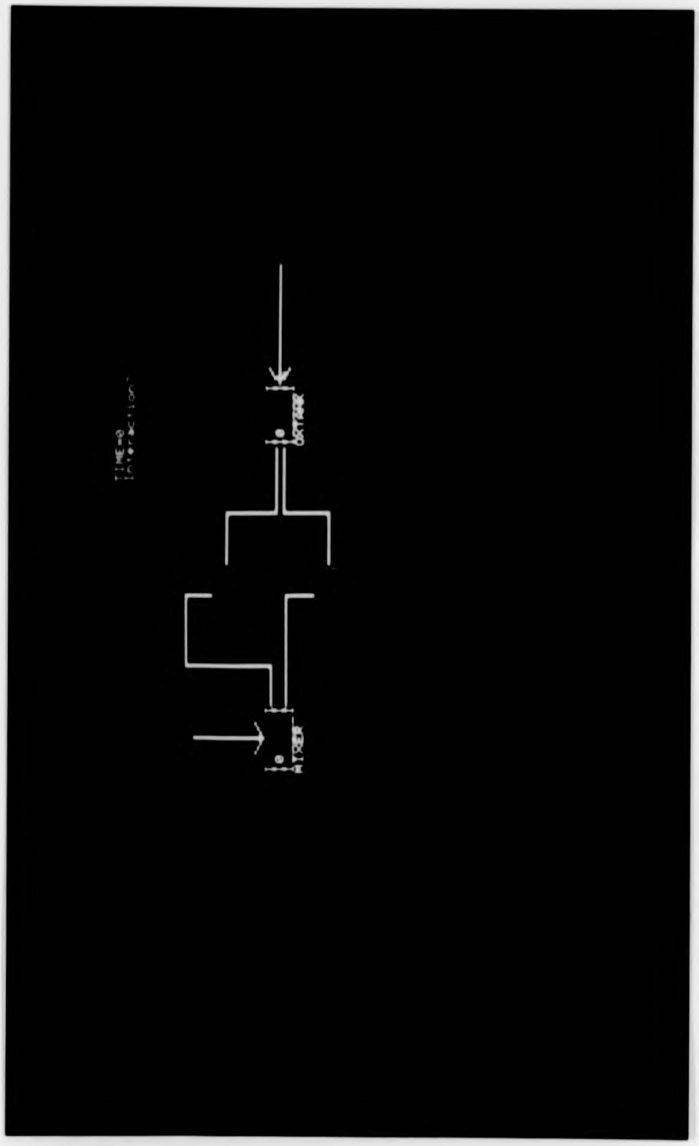
Figure 5

Quantity of oxygen used in second blow



Photograph 7

The completed display



Photograph 7

The completed display

interaction is used to specify some appropriate rate of flow of iron into the mixer (assuming it is not already charging one of the BOSP vessels), and a note made of the time at which the transfer will be complete. At that time the rate is returned to zero. At time=25 the user may decide to start making a batch of steel, as the oxygen tanks are sufficiently full, and the mixer is no longer being topped up with iron. Again, the appropriate rate is set to the required value, but this time there is no need to worry about the time at which the activity ends, since if the BOSP vessel's upper warning limit has been set to the quantity of iron in a batch, a system-generated event will occur, signalling the user to set the rate back to zero. Initial exploration of the problem may proceed in this way, with the sponsor providing the information needed to drive the system forward through time.

There is another feature of the model/display development subsystem that was not illustrated by the above example. The NEW interaction allows the user to define the visual representation of entity types. The user first enters the name of the type being defined, then the screen is cleared and he may build up the desired figure using any displayable character, moving the cursor around the screen with the control keys (up, down, left, right). Characters may be erased by pressing the 'Delete' key; overtyping with a space appears to have the same effect, but it actually enters a space into the design. The inadvertent inclusion of spaces can erase part of the existing display when the

new shape is placed on the screen. Completion of the shape is indicated by typing control-@ (ASCII NUL).

In addition to the static shape of the entity, it is also possible to specify the way attributes of the entity are to be displayed. Each displayed attribute may be shown in one of four ways: text, string, integer, and real. Integer and real are simple enough, giving the numeric value of the attribute with or without rounding. 'Text' converts the bit pattern of the attribute value into six characters. 'String' is more complicated, but offers considerable flexibility. The user can specify the number of values that may be taken by each attribute to be displayed in string format. A string of characters may then be associated with each value. For example, in a simulation of a security system, a type may be defined to represent individual buildings, each with a 'status' attribute which is related to the strings "On Fire!", "Intruder Detected!", or "All Quiet". In this case there are three states (assuming no provision is made for simultaneous burglary and arson), and so the status attribute will take a value from the set {1,2,3}. Whichever display format is chosen, the user also specifies the position in which the attribute is to be displayed, relative to the leftmost character of the top row of the entity's representation (this convention is used to correspond with the fact that the top left corner of the screen of most terminals is the home location, coordinates [0,0]).

Once an entity type has been described in this way, it may be freely used in the construction of new entities.

A useful refinement to the system would be the ability to specify an entity in terms of another entity (which must, of course, already be created), entering only those attributes which are to be different. The dialogue might be along these lines (user input in capitals):

Interaction? NEXT
Type of entity? VESSEL LIKE BOS1
Name of entity? BOS2
Change any attributes? YES
Change display attributes? YES
X-Coordinate is 10. New value? <RETURN> (i.e. no change)
Y-Coordinate is 10. New value? 20
Foreground colour is Red. New colour? <RETURN>
Background colour is Blue. New colour? <RETURN>
Logical Screen Number is 10. New value? <RETURN>
Change any other attributes? NO

A simpler method would be to give the type of the entity as

VESSEL LIKE BOS1 AT 10 20

but the provision of such flexibility would involve input analysis of a far greater sophistication than has yet been provided in the simulation system described above. Indeed, the reply to the question "Type of entity?" is presently limited to six characters, so minor changes would be needed to allow implementation of the dialogue shown. As it is, the provision of 'accelerated input' (i.e. entry of a

sequence of commands separated with commas instead of carriage returns to avoid waiting for the prompts) means that once a user has some experience of using the system the dialogue described could be entered as

NEXT,VESSEL LIKE BOS,BOS2,Y,Y,,20,,,,NO

Note that an empty field is equivalent to typing a null response (i.e. pressing Return alone).

Improvements could also be made to the way a link is initially defined. Since the ends of the link normally adjoin the vessels it is to connect, it should not be necessary for the user to type in the names of those vessels, since this information could be obtained from the model's data structure by searching through the element array to find which vessels (if any) are currently being displayed on the screen in the region of the end of the link (a more sophisticated structure involving multiple linked lists would be more efficient). If more than one vessel is a possible target (or if there is no adjacent vessel) the program would ask the user to make his intentions explicit. As an intermediate solution, there is no reason why the vessels could not be selected by pointing at their names with the light pen rather than using the keyboard. This would only require that input be done through an existing procedure which analyses the data from the terminal to determine its precise source (keyboard, light pen, or function keys), and then strips the prefix characters indicating the source from the message. The

message itself may then be processed in a manner suited to the context.

Further Development of Graphics Facilities

The way in which graphics (in the sense of line drawings rather than the more general meaning of a visual representation) are used within the system deserves further development. As they are only used (at present) for links, it has not been appropriate to put particular emphasis on them. It is likely that one of the next objectives in the development of the system will be to improve the graphics facilities, because with adequate (and easy to use!) software, graphics could replace the current symbol-oriented displays in order to present information more attractively. A significant part of the problem is ensuring a match between the way the user expects to be able to manipulate a shape, and the way the system allows him to do so. Although this also applies to the manipulation of shapes formed from alphanumeric symbols, it becomes more critical in a graphical environment. This may be because an image built from solid, connecting lines is perceived as an integrated whole, whereas the equivalent image formed from characters from the ASCII character set (e.g. minus signs, underscores, lower case L's, and slashes) lacks this cohesion.

As a great deal of work has been carried out on the development of computer graphics, it seems likely that the techniques required to enhance visual interactive

simulation already exist. For example, Sutherland's original Sketchpad system (now some 18 years old) was 'aware' of the individual points making up a line, in the sense that its data structures allowed a user to select a line by aiming a light pen anywhere along the line.

Conclusion

This chapter has discussed some of the ways in which the model development process affects and is affected by the client-analyst relationship. In particular, there are problems of communication which may be alleviated by the ability to produce a first-approximation model quickly and easily. However, there is a risk that the client may expect the model to be completed in an unrealistically brief period, having seen the initial stages carried out with ease (da Silva, 1980).

The design and implementation of an experimental system has been described. This system allows the analyst and his client to interactively create the entities and displays needed to model the situation, and to manipulate the model in simple ways. It is believed that this will have two main effects: the client will be in a better position to explain the nature of the system under study, and the analyst will be freed from much of the tedium of coding the model. Additional benefits lie in the separation of display generation from model logic, as changes to the displays may be made easily and without the need to re-verify the model.

The use of the experimental system to was illustrated by the reconstruction of the steelmaking simulation model described in Chapter 5.

The main shortcoming of the system described is that it requires that either the behaviour of all the entities is pre-defined, or that the user controls the model at a very low level until the program has been re-written to take over this function. Further work is required to develop a fully interactive simulation system which does not impose an excessive burden on its users; some signposts are offered in Chapter 7.

Chapter 7

PROPOSALS FOR THE FURTHER DEVELOPMENT OF VISUAL INTERACTIVE MODEL BUILDING

There seem to be two main directions in which the development of the system could proceed. The first is to carry on as before, making such alterations as appear necessary from time to time, either to allow the use of the system to model some distinctive situation, or as the result of changes to the host computer system. The interactive modelling facilities described in this thesis could be gradually expanded, adding various types of entity to the repertoire. This may be seen in terms of the development of a number of libraries, each relevant to a particular environment, from which the required building blocks can be extracted.

The alternative is to step back from the existing system in order to consider its evolution and potential. Despite its title, Hurrion's thesis does not describe a programming language, but a collection of procedures which eased the task of programming a visual interactive simulation model. The style of the approach was strongly influenced by SIMON: "A simulation program will consist of a number of procedures linked together either by an ALGOL or FORTRAN program" (page 38). Even at this stage, the principle that meeting the decision maker's requirements is more important than those of the analyst had been made explicit.

As work progressed, more features were built into the system. In particular, the inclusion of a scheduling mechanism reduced the amount of coding left to the analyst, and imparted a more easily discernable structure. The non-trivial task of writing procedures in Algol to represent activities was still the analyst's responsibility. The difficulty of this process was compounded by the need to include statements to manipulate the display (except in the case of entity-set operations, which were handled by the system software).

The tedious nature of the task of controlling the display led the current author to develop the extensions to the system that were described above (Chapter 6). This allowed the user of the system to construct skeletal models from previously defined (i.e. built-in) entity types, or to define new types in terms of appearance, but not behaviour. The difficulty is that the system expects 'behaviour' to be defined in terms of what happens in a particular activity, rather than how an entity responds to its environment.

Smalltalk

'Conventional' programming languages are not suited to this way of representing reality. A family of 'Actor' languages has been developed, embodying this world-view (Nelson 1980a, 1980b). Perhaps the most significant of these languages is Smalltalk, developed in the USA by Alan Kay and others at Xerox Palo Alto Research Centre. Although it was intended for use by children, it seems

likely that the power of the language will result in a much wider user base (it has been suggested (Kay, 1977) that children actually require more powerful computers than adults, since they expect ultra-fast responses, kalideoscopic colour, speech synthesis and all the other nice things that the use of Teletypes and time-sharing services conditions adults to do without).

Kay's great contribution was his realisation of the common ground between Simula and Sketchpad, in that both provide a framework for 'instances' of interacting 'objects' which may be created or destroyed as needed. He went on to assert that a computer program could be no more than a collection of objects which communicate with each other by passing messages (Nelson, 1980a), and constructed a computer system with a programming language (Flex) embodying these principles (Kay, 1969).

In addition to Flex, Smalltalk's design was influenced by a language called Logo. The use of Logo and its raison d'etre is described by Papert (1981), suffice it to say that it is intended to provide an environment in which children may learn mathematics without being put off by the coldness of 'normal' maths lessons. Particularly relevant is the intention of stimulating the child's (i.e. the user's) involvement, in much the same way as visual interactive simulation allows a manager to investigate aspects of the system he controls.

How then, does a language like Smalltalk fit into the aim of visual interactive simulation, which is to provide a

manager with an environment in which he may explore problem areas? The answer lies in the match between the structures available in Smalltalk, and those found in the real world. Consider an everyday example, that of a road junction controlled by traffic lights. Who would argue that the lights do not send messages to the cars (or at least to their drivers)? Messages are sent in the opposite direction by the movement of vehicles over sensors embedded in the road. Things get a little strained when we imagine a car being told by the vehicle ahead that it is too close, but the principle is clear: "Look at your message, and act accordingly" (Nelson, 1980a).

The implementors of Rosetta Smalltalk write: "...perhaps its most important characteristic is its friendliness. The notion of communicating with intelligent objects has an anthropomorphic flavour which puts abstract data types in a lively, concrete setting... Our experience and that of Xerox's Learning Research Group show programmers and nonprogrammers alike readily accept the metaphor of active objects communicating by passing messages, and can effectively use the powerful tools for abstraction and extensibility that Smalltalk provides" (Warren and Abbe, 1980, page 158). They also mention that they have produced a simple discrete event simulation system, so it would seem that an attempt to embed a full-blown visual interactive modelling system within the Smalltalk environment would be desirable and within reach.

An important point is that a modelling environment should act like an interpreter, as opposed to a compiler, otherwise the delay while the system assimilates new instructions may inhibit experimentation. Perhaps more importantly, an interpretive system provides opportunities for experimentation with an incomplete model. The current state of the simulation system at Warwick falls between two stools. Users may choose between preconceived alternatives when developing models and during experimentation, but new 'scripts' cannot be written, or existing ones changed without re-writing portions of the total program. This is largely because the simulation software is included in the application program at compilation time, instead of the program acting as input to the simulation software.

In Rosetta Smalltalk, such changes can be made by sending a message telling it that on receipt of a particular message certain actions are to be carried out. Furthermore, the graphical orientation and windowing effects built in to the system are highly relevant to the task of developing visual interactive models. An actor may open a window by sending a message specifying its size and location on the physical screen. The resulting window will overlay the current contents of the specified area, but if it is moved (or closed, or reduced in size), previously occluded portions of other windows will reappear.

This provides a neat way of handling the 'interaction box', which is an area of the screen reserved for dialogues. In the existing system, it permanently occupies

an area 40 characters wide and 5 lines deep at the top right-hand corner of the screen, but locations within the box must be addressed by their absolute screen coordinates. The use of a window has two main advantages in this situation. Since it allows the use of relative coordinates, the position of the interaction box may be varied to suit the rest of the display and/or the user (the latter is especially relevant in order to accomodate left- and right-handers when a light pen is used to pick items from a menu). Secondly, the box need not be a permanent fixture, and its size may vary according to the amount of text to be displayed. Once the model is running, the box can vanish, possibly revealing part of the display which is only relevant during a run. These facilities can of course be implemented within a system written in a 'conventional' programming language like FORTRAN (see da Silva et al, 1981) but this seems a little like trying to turn a family saloon car into something to compete with a Ferrari - it is (just about) possible, but tends to be even more expensive, and being non-standard, difficult to maintain.

Smalltalk will be available very soon. Rosetta, Inc are expected to be the the first company to release a version of the language, although they no longer intend to use the name 'Smalltalk'. This implementation is said to be less powerful than Xerox's, the release of which is expected before the end of 1981 (Nelson, 1981). Digital Equipment Corporation, Apple Computer Company, Tektronix,

and Hewlett-Packard are all known to be "exploring the language's potential" (Morgan, 1981).

An Interpretive Approach

There may be a compromise between the two approaches outlined above. It should be possible to extend the simulation system to allow the specification of a model's logic at run-time by providing an interpreter within the system. This would give the flexibility needed for interactive model building without losing the advantages of a widely known and well supported language. The mechanics of an interpreter have been discussed frequently (e.g. Gries, 1971; and Brown, 1979).

But what should be interpreted? If a model seems destined for regular use (as was the case with Fisher's model), it would be advantageous to allow subsequent compilation of the model in order to avoid the overheads of interpretation. Thus the interpreter should be capable of processing a subset of the simulation procedures and the host language (it will be assumed that this is Algol). This subset should include the input/output routines as well as the simulation procedures for entity and set handling. From the host language, arithmetic, conditional branching (IF...THEN...ELSE...), and looping (FOR...STEP...UNTIL...) appear to be minimal requirements, considering the contents of previously developed event procedures. Local variables would be very useful, and other features could be added if needed.

The interpretation of the system procedures could be done relatively simply. The piece of text being processed could be matched against a list of procedure names, and the number of parameters checked. The parameters would then be evaluated and the procedure called, but this would be satisfactory only in the cases where parameters are called by value. When the value of the parameter is to be altered by the procedure it is called by name. This causes an acute problem, since there is no simple way of relating a compiled program variable with the sequence of characters which was its identifier in the source code. An example is

```
PROCEDURE INPUTR(S,X,Y,SP,Z);
VALUE S,X,Y,SP;
INTEGER S,      %LOGICAL SCREEN NUMBER
        X,      %X-COORDINATE
        Y,      %Y-COORDINATE
        SP;     %NUMBER OF SPACES TO BE CLEARED FOR INPUT
REAL    Z;      %ON EXIT CONTAINS VALUE OF REAL NUMBER
                %   TYPED BY USER
BEGIN
  <procedure body>
END;
```

In this case the problem can be resolved by writing an equivalent procedure which calls all parameters by value and which returns a single result:

```
REAL PROCEDURE INPUTR(S,X,Y,SP);
COMMENT RETURNS THE VALUE OF A REAL NUMBER TYPED BY USER;
VALUE S,X,Y,SP;
INTEGER S,      %LOGICAL SCREEN NUMBER
        X,      %X-COORDINATE
        Y,      %Y-COORDINATE
        SP;     %NUMBER OF SPACES TO BE CLEARED FOR INPUT
BEGIN
   <procedure body>
   INPUTR:=<expression>;
END;
```

Other procedures call parameters by name in order to return more than one result:

```
BOOLEAN PROCEDURE SENSEINPUTR(S,X,Y,SP,Z);
COMMENT RETURNS FALSE IF 'RETURN' PRESSED ALONE; TRUE IF A
NUMBER WAS ENTERED, WITH THE VALUE OF THE NUMBER IN Z;
VALUE S,X,Y,SP;
INTEGER S,      %LOGICAL SCREEN NUMBER
        X,      %X-COORDINATE
        Y,      %Y-COORDINATE
        SP;     %NUMBER OF SPACES TO BE CLEARED FOR INPUT
REAL    Z;      %ON EXIT CONTAINS VALUE OF REAL NUMBER
                %   TYPED BY USER
BEGIN
  <procedure body>
END;
```

In cases like this there is no obvious solution, except to forbid the use of such procedures in interpreted routines!

If the system was running on a microcomputer it would be possible to retain the symbol table produced by the loader and thus permit calling by name by passing addresses with the aid of a machine code subroutine, but this would be practically impossible in a time-sharing system with virtual memory. An inelegant, but workable solution would be the reservation of a small number of global variables for use as call-by-name parameters. These could be matched with their identifying strings in the same way as the procedures themselves. Variables of each data type should be provided to allow maximum flexibility. An example of their use is:

```
TFORM(1,40,4,"Attribute 1 of MIXER is ");
RFORM(1,65,4,ATTRIBUTE(NUMBEROF("MIXER "),1);
TFORM(1,40,5,"New value? ");
IF SENSEINPUTR(1,40,16,10,GLOBALREAL1)
THEN SETATTRIBUTE(NUMBEROF("MIXER "),1,GLOBALREAL1);
```

This has the effect of displaying the first attribute of the mixer, and allowing the user to change it. If the existing value is correct the user simply presses "return".

The use of the procedure NUMBEROF should be explained. An entity created interactively is known to the user only by the name assigned to it. This means that it cannot be referenced within a routine by the name of a variable in the way it is in a conventionally developed model, where the variable is set to the number when the entity is created. For this reason NUMBEROF takes the name of an entity and returns its number. It may well prove

worthwhile to rewrite those procedures which take an entity number as a parameter so they accept names instead.

Scheduling would no longer be a problem, since the scheduling procedure

```
PROCEDURE SCHEDULE(EVENTNO,VAL,MEMBER);
VALUE EVENTNO,VAL,MEMBER;
REAL EVENTNO,   %EVENT NUMBER
     VAL,       %TIME UNTIL EVENT OCCURS
     MEMBER;    %ENTITY NUMBER
```

would be recognised by the interpreter. Any event which could only occur under certain conditions would be processed by rescheduling it after the next event in the timeset. This is the way such situations are generally handled in the existing version of the simulation software. The time advance mechanism would be altered to examine a list of events maintained by the interpreter, this could have a similar format to the EVENTS section of the program in Appendix 3.

When a working model has been produced the interpreted event procedures and the event list would be written to one or more disk files. These would then be merged with the main simulation program and compiled as a complete unit. The compiled code would be used for production runs, but any changes should first be made to the interpreted model as this will facilitate debugging.

The least complicated way of storing the routines to be interpreted is in a string array, with each line of code as a separate element. To avoid the need to rearrange the

array each time a change is made, a system of pointers could be used to indicate the successor of each line. The list of events would also need a set of pointers to the start of each routine on the list. Such techniques are discussed by Foster (1967).

In addition to the interpreter, a number of software tools will be needed. Some kind of editor would be essential to allow the entry and modification of event routines. This could be anything from the simple 're-type the whole line' arrangement found in many Basic interpreters, to a screen-oriented editor with many of the features found in word processors. Bearing in mind the user-oriented approach which has been stressed, an intermediate level of complexity is likely to be best, offering a sufficiently rich editing environment to avoid frustration, but without over-elaborate mechanisms which are so difficult to learn and remember that they are wasted on a casual user. Experience suggests that a good scheme is to provide a system in which a very small number of commands may be used to achieve perhaps 90% of the possible effects, with other commands that facilitate more complicated functions and provide the remaining 10%. Consider ED, the editor supplied with the CP/M operating system (Digital Research, 1978). This has 25 commands, and yet only four (append, type, substitute, and end) are essential. Use of the others may ease editing, but there is no need to learn them at the outset. For example, the line:

THIS IS A LINE

may be changed to

THIS IS A LONG LINE

by substituting "A LONG" for "A" - which takes care of insertions, and deletions are achieved by substituting nothing for something; in this case something could be "THIS IS ", resulting in

A LONG LINE

While it is not suggested that ED is a model editor, it provides an example of the distinction between features that are essential and those which are desirable.

The main advantage of a screen-oriented editor is that the effect of changes are immediately visible. In general, non-screen editors do not display the changed line after a function has been executed, instead it is left to the user to list the affected lines to ensure that his command had the effect he expected. Since the simulation software has the routines needed to display text at specific screen locations, and there is already a screen editor for the design of entities, the development of a screen oriented text editor would be less arduous than it would be if starting from scratch. Provision of a screen-oriented editor would also be consistent with the display editor.

Certain other aids would be valuable. These include the display of the current values of any or all variables, single stepping and/or program tracing, and the listing of routines to the screen or the printer.

Conclusion

The simulation software developed at Warwick started as a collection of subroutines designed to aid the construction of a visual interactive simulation model. Further development resulted in the software becoming what is effectively a set of macro-instructions, with a relatively small amount of code written in Algol (or other language) by the analyst in order to represent model-specific logical relationships, calculations, and the like.

It is generally accepted that the task of debugging a computer program is made easier when the programming language is interpreted rather than compiled (e.g. Gries, 1971, page 328). The adoption of an interpretive approach within the simulation software has merit, since the development process is one of successive refinement coupled with validation and verification.

Two possible ways of achieving this are discussed. The first is to select a language which has a world-view which corresponds to the nature of the systems likely to be modelled, and which provides a pleasant programming and experimental environment. The Smalltalk language is suggested as a candidate.

The alternative is to build an interpreter within the existing software. This would allow the user to describe and modify the behaviour of entities and other parts of the system at run-time. The features and mechanisms needed to achieve this are discussed.

Chapter 8

CONCLUSIONS

The usefulness of visual interactive simulation has been demonstrated by the work reviewed in Chapter 1. The projects described in this thesis add further weight to the evidence. The advantages of visual interactive simulation over conventional modelling techniques arise from the provision of a powerful mechanism for communicating the nature of a model and the results of experimentation with it to the client (typically a manager). In addition the possibility of interactive experimentation means that the skills, insights, and knowledge of a manager can be combined with the computer's speed and tirelessness to achieve better results than either might achieve in isolation. It is important that these models must be perceived by their potential audience as usable (as well as worthwhile) tools, otherwise the effort involved in their construction will be wasted. For this reason attention must be paid to ergonomic factors. It is insufficient to consider the design and use of the hardware alone, the conversational interface between the system and the user must be humane.

These issues apply to the acceptance and use of a model, but what of its development? Once it is there, such a model can bridge the gulf between client and analyst, but some points of contact must be found in order to construct it. It is suggested that the provision of facilities for

on-line model development could allow an initial 'quick and dirty' model to be built. As the parties come to understand each others world-view and language, and the analyst learns about the system under study, successive refinements could be made until an acceptable representation has been produced. This would then be used to investigate the problem. A subsidiary benefit would be that the client may gain understanding of the system as a result of studying it in sufficient detail to construct a model, whereas with conventional development techniques this is the perogative of the analyst. This would avoid the analyst's problem of communicating his new-found knowledge to his client, and generally speaking, the client is in a better position to utilise such insights.

An experimental system has been described which allows the interactive development of the entities, displays, and other elements that make up a model. Other facilities allow the manipulation of such a model in a manner akin to hand simulation. Once the analyst has learned enough about the working of the real system the operating rules may be programmed into the model, without undoing the work already carried out.

The drawback with this scheme is that this programming cannot be carried out interactively within the modelling environment. This could be achieved by either embedding the simulation software in an interpretive programming system (such as Smalltalk, which also offers certain

advantages due to its structure), or by providing an interpreter within the simulation system.

An approach which could be used to follow the latter route is outlined, along with some of the potential pitfalls. Its advantages are that it uses a well known and supported language, enhanced by the simulation procedures that have already been developed, and that programs developed within such a system could be subsequently compiled (for greater machine efficiency and hence faster running).

REFERENCES

ANASTASI, A. (1978) Psychological Testing. Third Edition. Macmillan, London.

BAZJANAC, V. (1976) Interactive Simulation of Building Evacuation with Elevators. Record of Proceedings, 9th Annual Simulation Symposium, March 1976, pages 17-19.

BIRD, P.F. (1977) 'Digilux' Touch Sensitive Panel. Paper presented at I.E.E. conference on 'Displays for man-machine systems', 4-7th April.

BOOTHROYD, H. (1978) Articulate Intervention. Taylor and Francis, London.

BOWEN, H.C., FENTON, R.J., CONNAUGHTON, D.E., FISHER, M.W.J., and HURRION, R.D. (1979a) An Interactive Aid to Improve Plant Control. I.E.E. Third International Conference on Trends in On-Line Computer Control Systems, 27-29th March, Sheffield. Pages 37-40.

BOWEN, H.C., FENTON, R.J., ROGERS, M.A.M., HURRION, R.D. and SECKER, R.J.R. (1979b) Interactive computing as an aid to decision makers. In Haley, K.B. (ed.), Operational Research '78. North-Holland Publishing Company, Amsterdam, pages 829-842.

BROWN, J.C. (1978) Visual Interactive Simulation: Further Developments Towards A Generalised System And Its Use In Three Problem Areas Associated With A High-Technology, Manufacturing Company. M.Sc. Thesis, University of Warwick.

BROWN, P.J. (1979) Writing Interactive Compilers and Interpreters. John Wiley and Sons, Chichester.

BURCHI, R.S. (1980) Interactive Graphics Today. IBM Systems Journal, 19,3, pages 292-313.

BUXTON, J.N. (ed) (1968) Simulation Programming Languages. North-Holland Publishing Company, Amsterdam.

CAKIR, A., HART, D.J., and STEWART, T.F.M (1979) The VDT Manual. Inca-Fiej Research Association, Darmstadt, West Germany.

CHRIST, R.E. (1975) Review and analysis of color coding research for visual displays. Human Factors, 17, 6, pages 542-570.

CIARCIA, S. (1978) Let Your Fingers Do The Talking, Byte, 3, 8, (August 1978) pages 156-165.

CLEMENTSON, A.T. (1980a) Extended Control and Simulation Language Users Manual. University of Birmingham.

CLEMENTSON, A.T. (1980b) ECSL/CAPS Detailed Reference Manual. University of Birmingham.

CONRAD, R. and HULL, A.J. (1968) The Preferred Layout For Numeric Data Entry Keysets. Ergonomics, 11, 2, pages 165-173.

DA SILVA, C.M. (1980) Personal communication.

DA SILVA, C.M., HURRION, R.D., SWANN, W.H., and TOSNEY, P.J. (1981) A decision support system for the planning and control of complex and interrelated manufacturing units. Paper presented to the Conference of the International Federation of Operational Research Societies, Hamburg, July 1981.

DAVIES, N.R. (1979) Interactive Simulation Program Generation. In ZEIGLER, B.F., ELZAS, M.S., KLIR, G.J. and OREN, T.I. (eds) Methodology in Systems Modelling and Simulation. North-Holland Publishing Company, Amsterdam, pages 179-200.

DEMARS, S.A. (1975) Human Factors Considerations for the use of Color in Display Systems, NASA Report TM-X-72196, John F. Kennedy Space Center.

DIGITAL RESEARCH (1978) ED: A context editor for the CP/M disk system. User's guide. Digital Research, Pacific Grove, California.

DOYLE, E. (1969) Manufacturing Processes and Materials for Engineers. Second Edition. Prentice-Hall, Englewood Cliffs, New Jersey.

EDEN, C. and JONES, S. (1980) Publish or Perish? - A Case Study. Journal of the Operational Research Society, 31, pages 131-139.

ELLISON, D. (1980) Apples in the Centre in Simulation. Simulaton Newsletter, 5 (October 1980), pages 2-3.

ENGLISH, W.K., ENGLEBART, D.C., and BERMAN, M.L. (1967) Display Selection Techniques for Cursor Manipulation. I.E.E.E. Transactions on Human Factors in Electronics, HFE-8(1), pages 5-15.

FISHER, M.W.J. (1981) An Application Of Visual Interactive Simulation In The Management Of Continuous Process Chemical Plants. Draft Edition. To be submitted as a Ph.D Thesis, University of Warwick.

FORRESTER, J. (1969) Urban Dynamics. M.I.T. Press and Wiley, Cambridge, Massachusetts.

FOSTER, J.M. (1967) List Processing. Macdonald-Elsevier, London.

GOODWIN, N.C. (1975) Cursor positioning on an electronic display using lightpen, lightgun, or keyboard for three basic tasks. Human Factors, 17, 3, pages 289-295.

GORDON, G. (1978) System Simulation. Second Edition. Prentice-Hall, Englewood Cliffs, New Jersey.

GREENBERGER, M. and JONES, M.M. (1967) On-line simulation in the Ops system. Proceedings of the 21st National Conference, Association for Computing Machinery, pages 131-138.

GRIES, D. (1971) Compiler Construction for Digital Computers. John Wiley and Sons, New York.

HOPKIN, V.D. (1977) Colour displays in air traffic control. I.E.E. Third International Conference on Trends in On-Line Computer Control Systems, 27-29th March, Sheffield, pages 46-49.

HURRION, R.D. (1976) The Design, Use, and Required Facilities of an Interactive Visual Computer Simulation Language to Explore Production Planning Problems. Ph.D. Thesis, University of London.

HURRION, R.D. (1978) An investigation of visual interactive simulation methods using the job-shop scheduling problem. Journal of the Operational Research Society, 29, 11, pages 1085-1093.

HURRION, R.D. (1980) An Inplementation of Visual Interactive Simulation Using a Microcomputer. Omega, 8, 2, pages 237-238.

HURRION, R.D. and DA SILVA, C.M. (1981) Dialogue and Display Management in a Decision Support System for Production Planning. To be published as a Centre for Industrial, Economic and Business Research Discussion Paper, University of Warwick, Coventry.

JOHNSON, E.A. (1967) Touch Displays: a programmed man-machine interface. Ergonomics, 10, pages 271-277.

KAY, A. (1969) The Reactive Engine. Ph.D. Thesis, University of Utah.

KAY, A. (1977) Microelectronics and the Personal Computer. Scientific American, September 1977.

KNOWLTON, K. (1975) Virtual Pushbuttons as a Means of Person-Machine Interaction. Proceedings of Conference on Computer Graphics, Pattern Recognition, and Data Structure. I.E.E.E., New York.

LONG, J.B., DENNETT, J., BARNARD, P.J., and MORTON, J. (1977) Effect of Display Format on the Direct Entry of Numerical Information by Pointing. I.E.E. Conference Publication 150, 'Displays for Man-Machine Systems', pages 134-137.

MCCORMICK, E.J. (1964) Human Factors Engineering. McGraw-Hill, New York.

MCEWING, R.W. (1977) Touch displays in industrial computer systems, in I.E.E. Conference Publication Numbr 150, Displays for Man-Machine Systems, pages 79-81.

MORGAN, C. (1981) Smalltalk: A Language for the 1980's. Byte, 6,8 (August 1981), pages 6-7.

NAKATANI, L.H. and O'CONNOR, K.D. (1980) Speech feedback for touch-keying. Ergonomics, 23, 7, pages 643-654.

NAYLOR, T.H., BALINTFY, J.L., BURDICK, D.S., and CHU, K. (1966) Computer Simulation Techniques. John Wiley and Sons, New York.

NELSON, S.S. (1979) CONSIM: a study of control issues in conversational simulation. Computer Journal, 22, 2, pages 119-126.

NELSON, T. (1980a) Symposium On Actor Languages. Creative Computing, 6, 10 (October 1980), pages 61-86.

NELSON, T. (1980b) Actor Languages: Part 2. Creative Computing, 6, 11 (November 1980), pages 73-94.

NELSON, T. (1981) Appallogy & Ruminescence. Creative Computing, 7, 1 (January 1981) pages 68-72.

PALME, J. (1977) Moving Pictures Show Simulation to User. Simulation, 29, 6, pages 204-209.

PAPERT, S. (1980) Mindstorms: children, computers and powerful ideas. Harvester Press, Brighton.

POULTON, E.C. and EDWARDS, R.S. (1980) Search for noisy coloured tracks camouflaged by noisy coloured backgrounds. Ergonomics, 23, 3, pages 247-261.

RUBENS, G.T. (1979) A Study of the Use of Visual Interactive Simulation for Decision-Making in a Complex Production Environment. M.Sc. Thesis, University of Warwick.

SECKER, R.J.R. (1977) That Visual Interactive Simulation Offers a Viable Technique for Examining Complex Production Planning and Scheduling Problems. M.Sc Thesis, University of Warwick.

SECKER, R.J.R. (1979) Visual Interactive Simulation Using a Mini-Computer. Journal of the Operational Research Society, 30, 4, pages 379-381.

SnONTZ, W.D., TROMM, G.A. and WILLIAMS, L.G. (1971) Color Coding for Information Location. Human Factors, 13, 3, pages 237-246.

SINGLETON, W.T. (1969) Display design: principles and procedures. Ergonomics, 12, 4, pages 519-535.

SPEARMAN, M.L. (1980) Dynamic Interactive System Simulation: An Inexpensive Approach to Solving Logistics Problems. Proceedings - 1980 Spring Annual Conference, American Institute of Industrial Engineers, Inc, pages 546-548.

STAMMERS, R.B. and BIRD, J.M. (1980) Controller Evaluation of a Touch-Input Air Traffic Data System. Human Factors, 22, 5, pages 581-589.

SUTHERLAND, I.E. (1963) Sketchpad, a Man-Machine Graphical Communication System. Proceedings of the Spring Joint Computer Conference, Detroit, Michigan, May 21-23. American Federation of Information Processing Societies.

TESLER, L. (1981) The Smalltalk Environment. Byte, 6, 8, (August 1981), pages 90-147.

UMBERS, I.G. (1976) A Review of the Human Factors Data on Input Devices Used for Process Computer Communication. Warren Spring Laboratory, Report Number LR242.

WARREN, S.K. and ABBE, D. (1980) Presenting Rosetta Smalltalk. Datamation, May 1980, pages 145-158.

WEAR, L.L. and DORF, R.C. (1970) An Interactive Keyboard for Man-Computer Communication. AFIPS Conference Proceedings, 36, pages 607-612.

WHITEFIELD, D., BALL, R.G. and ORD, G. (1980) Some Human Factors Aspects of Computer-Aiding Concepts for Air Traffic Controllers. Human Factors, 22, 5, pages 569-580.

BIBLIOGRAPHY

AAKER D.A. and WEINBERG, C.B. (1975) Interactive Marketing Models. Journal of Marketing, 39, pages 16-23.

ALEMPARTE, M., CHHEDA, D., SEELEY, D. and WALKER, W. (1975) Interacting with discrete simulation using on-line graphic animation. Computers and Graphics, 1, pages 309-318.

ALTER, S. (1977) Why is man-computer interaction important for decision support systems? Interfaces, 7,2, pages 109-115.

BEADLE, R.B. (1979) Planning for production of the new Mini. Paper presented at the Operational Research Society's Annual Conference, 1979.

BOOTHROYD, H. (1978) Articulate Intervention. Taylor and Francis, London.

BOXER, P.J. (1979) Designing simulators for strategic managers. Journal of Management Studies, 16, 1, pages 30-44.

BROWN, J.C. (1978) Visual Interactive Simulation: Further Developments Towards A Generalised System And Its Use In Three Problem Areas Associated With A High-Technology, Manufacturing Company. M.Sc. Thesis, University of Warwick.

BURCHI, R.S. (1980) Interactive Graphics Today. I.B.M. Systems Journal, 19, 3, pages 294-313.

BUXTON, J.N. (ed) (1968) Simulation Programming Languages. North-Holland Publishing Company, Amsterdam.

CAKIR, A., HART, D.J., and STEWART, T.F.M (1979) The VDT Manual. Inca-Fiej Research Association, Darmstadt, West Germany.

CHATTERGY, R. and POOCH, U.W. (1977) Integrated Design and Verification of Simulation Programs. Computer, 10, 4, pages 40-45.

DEGREENE, K.B. (1970) Systems Psychology. McGraw-Hill, New York.

DEMARS, S.A. (1975) Human Factors Considerations for the use of Color in Display Systems. NASA Report TM-X-72196, John F. Kennedy Space Center.

DONOVAN, J.J., JONES, M.M. and ALSOP, J.W. (1969) A graphical facility for an interactive simulation system. In MORRELL, A.J.H. (ed.) Information Processing 68. North-Holland Publishing Company, Amsterdam, pages 593-596.

DUNN, R.M. (1974) Computer Graphics - The Present. Proceedings of the Society for Information Display, 15, 1, pages 24-32.

EDEN, C. and HARRIS, J. (1975) Management Decision and Decision Analysis. Macmillan, London.

FISHER, M.W.J. (1981) An Application Of Visual Interactive Simulation In The Management Of Continuous Process Chemical Plants. Draft Edition. To be submitted as a Ph.D Thesis, University of Warwick.

FISHMAN, G.S. (1978) Principles of Discrete Event Simulation. Wiley-Interscience, New York.

GAINES, B.R. and FACEY, P.V. (1975) Some experience in interactive system development and application. I.E.E.E. Proceedings, 63, 6, pages 894-911.

GILB, T. and WEINBERG, G.M. (1977) Humanized Input. Winthrop Publishers, Inc, Cambridge, Massachusetts.

GOILLAU, P.J. and LAIOS, L. (1977) Factors affecting human decision-making in a simulated process control task. I.E.E. Conference publication number 150, Displays for man-machine systems, 4th-7th April 1977, pages 69-72.

GORDON, G. (1978) System Simulation. Second Edition. Prentice-Hall, Englewood Cliffs, New Jersey.

HURRION, R.D. (1976) The Design, Use, and Required Facilities of an Interactive Visual Computer Simulation Language to Explore Production Planning Problems. Ph.D. Thesis, University of London.

JONES, C.H., HUGHES, J.L. and ENGVOLD, J.L. (1970) A comparative study of management decision-making from computer terminals. A.F.I.P.S. Conference proceedings, 36, pages 599-605.

JONES, P.F. (1978) Four Principles of Man-Computer Dialogue. Computer-Aided Design, 10, 3, pages 197-202.

KENNEDY, T.C.S. (1974) The design of interactive procedures for man-machine communication. International Journal of Man-Machine Studies, 6, pages 309-334.

KISHI, S (1976) Plant Monitoring by Color CRT Displays for Boiling Water Reactor. Hitachi Review, 25, 8, pages 265-270.

LEES, R. (1977) The man-display relationship. I.E.E. Conference publication number 150, Displays for man-machine systems, 4th-7th April 1977, pages 73-74.

LITTLE, J.D.C. (1970) Models and Managers: the concept of a decision calculus. Management Science, 16, 8, pages B466-B485.

MCCORMICK, E.J. (1964) Human Factors Engineering. McGraw-Hill, New York.

MAHER, P.K.C. and BELL, H.V. (1977) The man-machine interface - a new approach. I.E.E. Conference publication number 150, Displays for man-machine systems, 4th-7th April 1977, pages 122-125.

MARTIN, J.T. (1973) Design of Man-Computer Dialogues. Prentice-Hall, New Jersey.

MEADOW, C.T. (1970) Man-Machine Communication. Wiley-Interscience, New York.

MILLER, L.A. and THOMAS, J.C. Jnr. (1976) Behavioural issues in the use of interactive systems. In BLASER, A. and HACKL, C., Interactive Systems: Proceedings, 6th Informatik Symposium, I.B.M. Germany, Bad Hamburg, v.o.H., September 1976. Springer-Verlag, Berlin.

MILLER, R.B. (1965) Psychology for a man-machine problem-solving system. Data Systems Division Report TR00.1246, I.B.M. Poughkeepsie, New York.

NELSON, T. (1980a) Symposium On Actor Languages. Creative Computing, 6, 10 (October 1980), pages 61-86.

NELSON, T. (1980b) Actor Languages: Part 2. Creative Computing, 6, 11 (November 1980), pages 73-94.

NELSON, T. (1981) Appallogy & Ruminescence. Creative Computing, 7, 1 (January 1981) pages 68-72.

ORLICKY, J. (1969) The Successful Computer System. McGraw-Hill, New York.

RUBENS, G.T. (1979) A Study of the Use of Visual Interactive Simulation for Decision-Making in a Complex Production Environment. M.Sc. Thesis, University of Warwick.

SCHOBER, F. (1976) Interactive simulation models in planning. In BLASER, A. and HACKL, C., Interactive Systems: Proceedings, 6th Informatik Symposium, I.B.M. Germany, Bad Hamburg, v.o.H., September 1976. Springer-Verlag, Berlin.

SCOTT MORTON, M.S. (1967) Interactive visual display systems and management problem solving. Industrial Management Review, 9, pages 69-81.

SECKER, R.J.R. (1977) That Visual Interactive Simulation Offers a Viable Technique for Examining Complex Production Planning and Scheduling Problems. M.Sc Thesis, University of Warwick.

SHACKEL, B. (1969) Man-computer interaction - the contribution of the human sciences. Ergonomics, 12, 4, pages 485-499.

SINGLETON, W.T. (1969) Display design: principles and procedures. Ergonomics, 12, 4, pages 519-535.

SNYDER, J.E. (1968) Why is top management difficult to convince? Second Conference on Applications of Simulation - Papers, pages 1-5.

SULONEN, R.K. (1972) On-line simulation with computer graphics. ONLINE 72, International conference on online interactive computing, 4th-7th September 1972, Brunel University, Uxbridge.

SUTHERLAND, I.E. (1963) Sketchpad, a Man-Machine Graphical Communication System. Proceedings of the Spring Joint Computer Conference, Detroit, Michigan, May 21-23. American Federation of Information Processing Societies.

TOCHER, K.D. (1963) The Art of Simulation. English Universities Press, London.

TREU, S. (1975) Interactive command language based on required mental work. International Journal of Man-Machine Studies, 7, pages 135-149.

UMBERS, I.G. (1976) A Review of the Human Factors Data on Input Devices Used for Process Computer Communication. Warren Spring Laboratory, Report Number LR242.

YOUNG, L.F. (1978) Another look at man-computer interaction. Interfaces, 8, 2, pages 67-69.

APPENDIX 1

User manual for the Rolls-Royce Vision Scheduling Demonstrator

VISION SCHEDULING DEMONSTRATOR

User Manual

1. Introduction

The simulation model represents a two-stage system where raw materials are 'bought-in' and subsequently made into a finished product. The object of the model is to show how difficult it is to schedule batches of work in progress and orders for material to meet the demand for the product without incurring excessive stockholding costs, and with as few changes to the schedule as possible.

To run the model, log on to the Burroughs B6700 at the University (using the appropriate usercode and password), and type:

RUN D/PROG

The program will first ask for the type of terminal being used - valid replies are CIFER, MELLOR, ISC, ISCICI, and ISCTEK. Note that your reply must be typed in upper case.

After a short delay the main display will appear, with the prompt 'INTERACTION?' at the top right-hand corner of the screen. If the default parameters are to be used, simply type RUN, and follow the subsequent prompts; otherwise type OMANUA and the program will request values for each parameter in turn. If only a few parameters are to be altered, it is quicker to use specific interactions (such as O$FORC to change the forecast function constants). Details of these commands are given in the INTERACTIONS section of this document.

At the start of each accounting period (a.p.) the user must decide upon schedules for both stock and materials.

At this time the program asks 'OLD, NEW, OR RECOMMENDED SCHEDULE?'. If 'O' is typed the old schedule is carried over, 'R' causes the program to produce a schedule based on the state of the system, and 'N' allows the user to enter a new schedule on a form-filling basis. It is also valid to type 'I' (for Interaction), which returns to the outermost level where any interaction may be used.

If the main display is not switched on at the start of an a.p. the recommended schedule is used. This allows the model to run in batch mode.

2. Displays

The user may select any one or a combination of displays (see the description of interaction DISP). Refer to the following table for screen numbers and their associated displays.

Screen no	Display
3	The main display, showing the schedules, forecasts, stocks, work in progress, etc.
21	Timeseries of demand for the product.
22	Timeseries of demand for materials.
23	Timeseries of unsatisfied demand for the product.
24	Timeseries of unsatisfied demand for materials.
25	Timeseries of forecast demand for the product.
26	Timeseries of forecast demand for materials.
27	Timeseries of stock plus work in progress for the product, plus stock of material.
28	Timeseries of service level.

3. Histograms

A number of static histograms are provided to illustrate certain measures of performance. The following table gives the names and descriptions of the histograms.

Instructions for displaying histograms may be found in the Interactions section of this manual.

Name	Description
CH.C.A	Shows net changes to the schedule for the product, in the 'current' a.p.
CH.2.A	As CH.C.A but for a.p.s 1 and 2.
CH.4.A	As CH.C.A but for a.p.s 3 and 4.
CH.C.C	As CH.C.A but for the material schedule.
CH.2.C	As CH.C.C but for a.p.s 1 and 2.
CH.4.C	As CH.C.C but for a.p.s 3 and 4.
BAPOPA	Shows number of operations against batch/a.p.s for the product stage.
BAPOPC	As BAPOPA but for materials.
TOTOPA	Shows total number of operations against batch/a.p.s for product.
TOTOPC	As TOTOPA but for materials.
HIST.1	Shows total number of operations completed in the product stage in the previous a.p.
HIST.2	As HIST.1 but for materials.
HIST.4	Aggregate of HIST.1 and HIST.2.
HIST.5	Shows total load on system contributed by product in a.p. (i.e. [number of operations*batch size] summed over all batches.)
HIST.6	As HIST.5 but for materials.
HIST.8	Aggregate of HIST.5 and HIST.6.

4. Forecasting Function

The forecasting function is the same for both stages of the system, and is defined as follows:

$$F_1=F_0*(1-k)+(k*D)$$

where

F_1 is the present forecast of demand

F_0 is the previous forecast

D is the present demand for the product

k is some constant

The value of the constant (k) can be changed by interaction (see O$FORC in the Interaction section). The initial values are given in the Initialisation section.

5. Minimum Stock Level Function

The minimum stock level function is the same for both stages of the sytem, and is defined as follows:

$$S=k*F$$

where

S is the minimum stock level

F is the present forecast for the product

k is some constant

The value of the constant (k) can be changed by interaction (see O$FORC in the Interaction section). The initial values are given in the Initialisation section.

6. Lead Times

The lead times for both stages can be changed by interaction (see O$LEAD in the Interaction section). The initial values are given in the Initialisation section.

7. Priority Function

At each review each batch of work in progress or stock on order is given a priority rating described in words as follows:

Priority Rating=Expected time to completion -
Time remaining until batch is required

The priority rating is then matched against a function which computes the actual number of operations carried out on that batch during the a.p. The function is described by four limits (see interaction O+PGRF in the Interaction section) which can be changed by interaction (see O$PLIM and O+PLIM). The initial values are given in the Initialisation section.

8. Demand Generation

The demand for the product is generated randomly as part of each review. The number may be drawn from one of three distributions - rectangular (range X to Y), Poisson (mean Z), or Gamma (mean W and standard deviation S). The initial settings are given in the Initialisation section, but they may be changed interactively (see ORNDM, OPOIS, and OGAMM in the Interaction section).

9. Initialisation

The model may be set up manually or automatically. If if is set up automatically the values of the critical parameters are as specified below:

		Product	Material
Forecast Function Constant		0.1	0.1
Minimum Stock Level Constant		2	3
Lead Times (days)		140	80
Priority Function:	Upper X	40	20
	Lower X	-40	-20
	Upper Y	25	22
	Lower Y	15	18
Amount in Stock		0	0
Previous Unsatisfied Demand		*	*
Previous A.P. Forecast		10	10

Demand Generator is Gamma with mean=10, standard deviation=10

* Variable, determined by random sample

10. Interaction Facilities

Whenever 'RUNNING' appears at the top right corner of the screen the user may press the 'Break' key in order to enter the Interaction mode. In this mode displays may be changed, parameters altered, and the state of the model 'dumped'. If the model has already been dumped, it is possible to restart it from that state. This is commonly used to compare the effects of different operating strategies.

When the prompt 'INTERACTION?' is displayed, the user may reply with one of the following commands:

RUN

The model will run until either the 'Break' key is pressed, the user's interaction is required, or the program exceeds the process time limit.

GOTO

Allows the user to cause the model to run until a specified time is reached.

OREFO

Clears the screen(s) and reforms it (them) in the current mode of display. Useful when unwanted messages appear, or when transmission errors occur on telephone lines.

HIST

Asks for the name of a histogram, and (if it exists) displays it.

DISP

Allows the user to turn displays on and off by specifing the screen numbers.

OMANUA

May only be used at the start of a run. Allows the user to specify the starting values of all parameters.

COMP

Used mainly to compare timeseries graphs. Allows the user to return to a previously dumped state, change one or more parameters, and then re-run the model with the new timeseries being overlaying the original. This can be an effective way of comparing strategies.

ORAND

Changes the random number stream to that specified.

ORNDM

Allows the user to specify values for the range of the rectangular random sampling mechanism.

OPOIS

Allows the user to specify the mean of the Poisson random samples.

OGAMM

Allows the user to specify the mean and standard deviation of the Gamma random samples.

ODGEN

Allows the user to specify the type of distribution to be used when generating the demand for the product.

OBATC

Causes the model to run without updating the screen for the specified period of time. This can be up to two orders of magnitude faster, and to avoid confusion the screen is cleared when operating in this mode.

KILL

Terminates the program.

O+PGRF

Displays a representation of the Priority function at a position specified by the user.

O*PGRF

Clears the Priority function graph from the screen, if it is currently being displayed.

O+PLIM

Displays the upper and lower limits of the priority function at a position specified by the user.

O*PLIM

Clears the Priority function limits from the screen, if they are currently being displayed.

O+LEAD

Displays the leadtimes for both stages of the system at a position specified by the user.

O$LEAD

Allows changes to be made to the leadtimes.

O*LEAD

Clears the leadtimes from the screen, if they are currently being displayed.

O+DCON

Displays the minimun stock level function constants for both stages of the system at a position specified by the user.

O$DCON

Allows changes to be made to the minimum stock level function constants.

O*DCON

Clears the minimum stock level function constants display from the screen, if they are currently being displayed.

O+WINP

Displays the current state of work in progress at a position specified by the user.

O*WINP

Clears the work in progress display from the screen, if it is currently being displayed.

O+FORC

Displays the forecast function constants for both stages of the system at a position specified by the user.

O$FORC

Allows changes to be made to the forecast function constants.

O*FORC

Clears the forecast function constants from the screen, if they are currently displayed.

OCL 1

Clears the master screen.

OCL 2

Clears the subsidiary screen, if it is in use.

OREF1

When both screens are in use, this command is used to reform the master screen without affecting the subsidiary screen.

OREF2

Reforms the subsidiary screen without affecting the master screen.

SCRN

Allows the user to bring a subsidiary screen into operation. The station name of the subsidiary terminal must be known.

OSOFF

Switches the secondary screen off, but does not affect the master screen or the running of the program.

OPAUSE

Allows the user to switch on or off an optional pause in program output. Used to allow assimilation of an otherwise rapidly changing display.

DUMP

Allows the user to dump the current state of the model into a file. This state may be recovered by using the OREST interaction (see below).

OREST

Restarts the model from some pre-dumped state.

APPENDIX 2

Listings of programs used for speech input

```
 10 REM  *** VOICE TERMINAL ***
 20 REM
 30 REM      STEPHEN  WITHERS
 40 REM        JANUARY 1979
 50 REM  **********************
 60 REM    WRITTEN IN APPLE
 70 REM       INTEGER BASIC
 80 REM  SET LOMEM:5500 BEFORE
 90 REM         RUNNING
100 D$="": REM D$=CONTROL-D
110 DIM FILE$(6),INTERACT$(6)
120 PRINT D$;"NOMON C,I,O"
125 REM CLEAR THE SCREEN
130 CALL -936
135 REM INITIALISE SPEECHLAB
140 PRINT D$;"PR#3"
145 PRINT
150 PRINT D$;"PR#0"
155 PRINT "ENTER FILENAME";
160 INPUT FILE$
165 REM LOAD THE FIRST VOICE-PRINT FILE
170 PRINT D$;"BLOAD ";FILE$;"1"
175 PRINT "READY"
180 REM GET A COMMAND
190 PRINT D$;"IN#3"
200 INPUT INTERACT$
210 REM CHECK THAT COMMAND WAS RECOGNISED
220 IF INTERACT$="RUN" THEN GOTO 440
230 IF INTERACT$="KILL" THEN GOTO 440
240 IF INTERACT$="O" THEN GOTO 440
250 IF INTERACT$="N" THEN GOTO 440
260 IF INTERACT$="R" THEN GOTO 440
270 IF INTERACT$="HIST" THEN GOTO 440
280 IF INTERACT$="REFO" THEN GOTO 440
290 IF INTERACT$="OK" THEN GOTO 440
400 REM IF WORD NOT RECOGNISED...
410 PRINT "PARDON?" : REM USE A CONTROL-G TO SOUND BLEEP
420 GOTO 190
430 REM SEND COMMAND TO BURROUGHS
440 PRINT D$;"PR#2"
450 PRINT INTERACT$
460 PRINT D$;"PR#0"
470 REM HAVE WE FINISHED?
480 IF INTERACT$#"KILL" THEN GOTO 190 :REM FOR NEXT COMMAND
490 PRINT D$;"IN#0"
500 PRINT D$;"PR#0"
510 END
```

```
 10 REM  ****************************
 15 REM
 20 REM  BUILD INTERACTION VOICEFILE
 30 REM   WRITTEN IN APPLE INTEGER
 40 REM            BASIC
 50 REM        STEPHEN WITHERS
 60 REM          JANUARY 1979
 70 REM  ****************************
 80 D$="" : REM D$=CONTROL-D
 90 REM INITIALISE SPEECHLAB
100 PRINT D$;"PR#3"
110 PRINT
120 PRINT D$;"PR#0"
130 REM MAKE TWO COPIES OF EACH COMMAND
140 FOR I=0 TO 1
150 PRINT "RUN"
160 PRINT D$;"PR#3"
170 PRINT "RUN"
180 PRINT D$;"PR#0"
190 PRINT "KILL"
200 PRINT D$;"PR#3"
210 PRINT "KILL"
220 PRINT D$;"PR#0"
230 PRINT "OLD"
240 PRINT D$;"PR#3"
250 PRINT "O"
260 PRINT D$;"PR#0"
270 PRINT "RECOMMENDED"
280 PRINT D$;"PR#3"
290 PRINT "R"
300 PRINT D$;"PR#0"
310 PRINT "NEW"
320 PRINT D$;"PR#3"
330 PRINT "N"
340 PRINT D$;"PR#0"
350 PRINT "HIST"
360 PRINT D$;"PR#3"
370 PRINT "HIST"
380 PRINT D$;"PR#0"
350 PRINT "SCREENS"
360 PRINT D$;"PR#3"
370 PRINT "REFO"
380 PRINT D$;"PR#0"
390 PRINT "OK"
400 PRINT D$;"PR#3"
410 PRINT "OK"
420 PRINT D$;"PR#0"
430 NEXT I
440 REM NOW SAVE PATTERNS IN A DISK FILE
450 DIM FILENAME$(10)
460 PRINT "ENTER FILENAME ";
470 INPUT FILENAME$
480 PRINT D$;"BSAVE ";FILENAME$;"1";", A2048, L3451"
490 END
```

APPENDIX 3

Skeletal program for interactive model development

```
 $SET ERRLIST LINEINFO INSTALLATION 339
 $RESET NOBINDINFO LIST
BEGIN
 $INCLUDE "(UWBSRHX)DESIGN"
 $INCLUDE "DISPLAY/PROCS"
  VBEGIN;
  VINITIALISE;
  EVENTS
      1: DUMMYEVENT;
    994: LAG;
    995: UPDATEVESSEL;
    996: MAXCAPINTERACT;
    997: MAXWARINTERACT;
    998: MINWARINTERACT;
    999: MINCAPINTERACT;
  VEND;
END.
```

APPENDIX 4

A Description Of The Software Used To Define A New Entity Type

In order to clarify the way in which the software processes a new entity type, the method is described with the aid of an example. This should be read in conjunction with the listing in Appendix 5 (some line numbers are provided in the text as an aid to cross-referencing), and Figure 4 is reproduced here to assist further.

A string array is used to represent the display. This consists of 48 elements, each of which initially contain 80 Null characters (lines 23700-23900). The user is prompted to give the name of the new type, which is checked against a list of those already used to ensure no duplication occurs (24300-24800). If the name is accepted, a new display set is created (25100) to contain the display elements resulting from the design process.

Procedure GETIMAGE (1500-9800) handles the user's input for the new type's representation. Two pointers are used to maintain the position of a logical cursor within the screen image display, corresponding to the position of the real cursor on the screen. Provision has been made for alternative terminals (5100-6400), since different devices assign varing codes to the cursor control keys. Single character input mode is used so that the program responds to each key depression (normally input is buffered until a carriage return is received indicating the end of the record). When a key is pressed one of five things happens. If it is a cursor control key, the pointers are adjusted to reflect the new position of the real cursor. For instance,

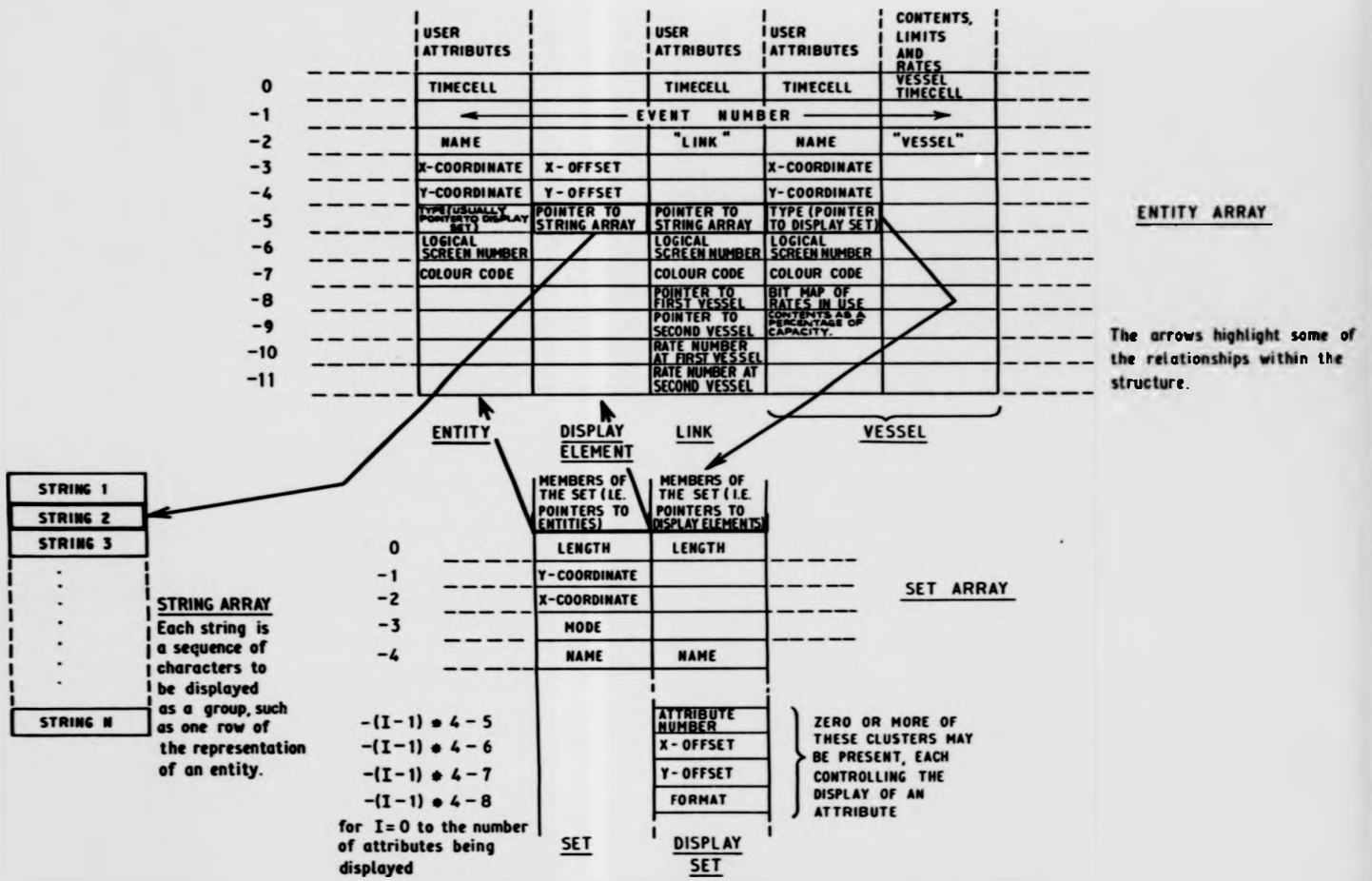


Figure 4

Data structure used in the extended system

pressing 'cursor right' adds 1 to the value of the x-pointer, leaving the y-pointer unchanged.

Selecting a printable character sets the current cell of the screen image array (indicated by the x- and y-pointers) to the chosen character, and the x-pointer incremented (7600-8900).

For reasons which will be explained later, unwanted characters may not be erased by overtyping with the space bar, so the Delete key is used for this purpose. This resets the current cell to Null, and clears the current postion on the display. The cursor position is not altered (7000-7500).

The only other control character recognised by the routine is Null (control-@). This is used to signal that the shape has been completed (6700) and that the program should return to normal input mode.

The final contents of the screen image display are analysed. Each row is considered in turn. First, the position of the leftmost non-Null character is noted (10900) and then the leading Nulls dropped. Next, a subpicture is formed from the leftmost part of the string which includes no Nulls (11300). This is why spaces are only entered when they are actually wanted as part of a representation (perhaps to ensure that part of another item is erased) and not to delete unwanted characters. The subpicture is then compared with existing subpictures (11900-12400), and only if no match is found is it entered into the subpicture array (12500-13200), otherwise the pre-

existing one is referenced. In either case a new display element is created (13300) and added to the display set (13400). Three of the display element's attributes are used to store the corresponding subpicture, and its position relative to that of the total shape. This process is repeated until all 80 characters have been discarded as Nulls or included in display elements.

For example, if the following shape was defined

with no spaces being typed within the triangle, the following situation results:

Display Set	relative position		subpicture
	x	y	pointer
	+2	0	1
	+1	+1	2
	+4	+1	3
	0	+2	2
	+5	+2	3
	0	+3	4

Subpicture 1	"/\"
2	"/"
3	"\"
4	"------"

It is often desirable to label specific instances of an entity type with their names, or to show the value of one or more attributes (e.g. the contents of a vessel). For this reason the final stage in the process allows the user to specify which (if any) of the attributes of entities of the new type are to be displayed, and the manner in which they are to be shown. Having entered the number of attributes to be displayed (18800-18900), the user is asked to specify them by number, along with the format. The choice is between integer, real, text (the 48 bit value of the attribute interpreted as 6 characters) or string (19200-20100). If string format is selected the user must also specify the number of states the attribute may take, and the string to be associated with each value (20600-22400). Whichever format is chosen, it is necessary to state the relative position at which the attribute is to be displayed.

It can be seen from the above description that the display design facility is built around the entity-set paradigm used in simulation modelling, thus minimising the need for additional software.

APPENDIX 5

Listing Of The Software Described In Appendix 4

```
 100  PROCEDURE EDIT(DISPLAYELEMENTNUMBER);
 200  VALUE DISPLAYELEMENTNUMBER;
 300  INTEGER DISPLAYELEMENTNUMBER;
 400  COMMENT THIS PROCEDURE IS USED TO CREATE NEW ENTITY TYPES
 500          OR TO ALTER EXISTING TYPES.
 600          SINGLE CHARACTER INPUT MODE  IS USED TO MAP THE SCREEN INTO A
 700          STRING ARRAY. THE SUBPICTURES REQUIRED ARE ISOLATED FROM THIS
 800          'MAP' AND DISPLAY ELEMENTS ARE PRODUCED TO STORE THE INFORMATION
 900          SJ WITHERS, JULY 1980;
1000  BEGIN
1100    REAL TYPE;
1200    INTEGER I,SETNO;
1300    BOOLEAN NEWTYPE;
1400    STRING ARRAY IMAGE [0:47];
1500    PROCEDURE GETIMAGE;
1600    COMMENT GETS THE IMAGE BEING DRAWN ON THE SCREEN CHARACTER BY
1700            CHARACTER, MAPPING IT INTO ARRAY IMAGE;
1800    BEGIN
1900      INTEGER X,Y;
2000      STRING UP,DOWN,LEFT,RIGHT,CH,ENDOFMESSAGE,DEL;
2100      PROCEDURE INSERT(WHICHSTRING,CHAR,POSITION);
2200      VALUE POSITION;
2300      COMMENT REPLACES THE POSITION'TH CHARACTER OF WHICHSTRING WITH CHAR;
2400      STRING WHICHSTRING,CHAR;
2500      INTEGER POSITION:
2600      BEGIN
2700        STRING BEGINNING,ENDING;
2800        IF POSITION<0 OR POSITION>LENGTH(WHICHSTRING)-1
2900        %REMEMBER STRINGS NUMBER FROM 1, SCREEN COORDS FROM 0
3000        THEN BEGIN
3100               STRING DUMMY;
3200               CLEARBOX;
3300               TFORM(INT,41,1,"Error in INSEPT.");
```

```
3400              TFORM(INT,41,2,"Length=");
3500              IFORM(INT,48,2,LENGTH(WHICHSTRING));
3600              TFORM(INT,41,3,"Position=");
3700              IFORM(INT,50,3,POSITION);
3800              TFORM(INT,41,4,"Press RETURN");
3900              READ(ONECHAR,<A1>,DUMMY);
4000            END
4100        ELSE BEGIN
4200              BEGINNING:=TAKE(WHICHSTRING,POSITION+1);
4300              ENDING:=DROP(WHICHSTRING,POSITION+2);
4400              WHICHSTRING:=BEGINNING CAT CHAR CAT ENDING;
4500            END;
4600      END OF PROCEDURE INSERT;
4700      X:=0;
4800      Y:=0;
4900      SETSINGLECHAR;
5000      ENDOFMESSAGE:=48"00";
5100      IF DEVICE[1]="ISCTEK" OR DEVICE[1]="CIFER "
5200      THEN BEGIN
5300             UP:=48"0B";
5400             DOWN:=48"25";
5500             LEFT:=48"16";
5600             RIGHT:=48"05";
5700           END
5800      ELSE IF DEVICE[1]="ISC    "
5900           THEN BEGIN
6000                  UP:=48"1C";
6100                  DOWN:=48"25";
6200                  LEFT:=48"3F";
6300                  RIGHT:=48"19";
6400                END;
6500      DEL:=48"07";
6600      CH:=EMPTY8,
```

```
6700      WHILE CH NEQ ENDOFMESSAGE
6800      DO BEGIN
6900           INPUTSTRINGCHAR(INT,X,Y,CH);
7000           IF CH=DEL
7100           THEN BEGIN
7200                  INSERT(IMAGE[Y],NULL,X);
7300                  CFORM(INT,X,Y," ");
7400                  CURSOR(INT,X,Y);
7500                END
7600           ELSE IF CH GEQ " "
7700                THEN BEGIN
7800                       INSERT(IMAGE[Y],CH,X);
7900                       X:=X+1;
8000                       IF X>79
8100                       THEN BEGIN
8200                              X:=79;
8300                              CURSOR(INT,X,Y);
8400                            END;
8500                     END
8600                ELSE BEGIN
8700                       IF CH=UP
8800                       THEN Y:=IF Y=0 THEN 0 ELSE Y-1;
8900                       IF CH=DOWN
9000                       THEN Y:=IF Y=47 THEN 47 ELSE Y+1;
9100                       IF CH=LEFT
9200                       THEN X:=IF X=0 THEN 0 ELSE X-1;
9300                       IF CH=RIGHT
9400                       THEN X:=IF X=79 THEN 79 ELSE X+1;
9500                     END;
9600        END;
9700      SETMULTICHAR;
9800      END OF PROCEDURE GETIMAGE;
9900      PROCEDURE STORESUBPICTURES(SETNUMBER);
```

```
10000       VALUE SETNUMBER; INTEGER SETNUMBER;
10100     COMMENT BREAKS THE SCREEN IMAGE INTO SUBPICTURES, AND STORES
10200             THE INFORMATION, CREATING NEW SUBPICTURE ELEMENTS AS NEEDED;
10300       BEGIN
10400         INTEGER I,X,OLDX,Y,NUMBER,PICNO;
10500         BOOLEAN NEWSUBPICTURE;
10600         STRING SUBPIC;
10700         FOR Y:=0 STEP 1 UNTIL 47 DO
10800         BEGIN
10900           X:=LENGTH(HEAD(IMAGE[Y],NULL));
11000           IMAGE[Y]:=TAIL(IMAGE[Y],NULL);
11100           WHILE LENGTH(IMAGE[Y])>0
11200           DO BEGIN
11300                SUBPIC:=HEAD(IMAGE[Y],NOT NULL),
11400                IMAGE[Y]:=TAIL(IMAGE[Y],NOT NULL);
11500                OLDX:=X-1;
11600                X:=X+LENGTH(HEAD(IMAGE[Y],NULL))+1;
11700                IMAGE[Y]:=TAIL(IMAGE[Y],NULL);
11800                NEWSUBPICTURE:=TRUE;
11900                FOR I:=1 STEP 1 UNTIL SUBPICCOUNT
12000                DO IF SUBPIC=SUBPICTURE[I]
12100                   THEN BEGIN
12200                          NEWSUBPICTURE:=FALSE;
12300                          PICNO:=I;
12400                        END;
12500                IF NEWSUBPICTURE
12600                THEN BEGIN
12700                       SUBPICCOUNT:=*+1;
12800                       PICNO:=SUBPICCOUNT;
12900                       SUBPICTURE[PICNO]:=SUBPIC;
13000                       WRITE(PICFILE[PICNO+1],/,SUBPIC);
13100                       WRITE(PICFILE[0],/,SUBPICCOUNT);
13200                     END,
```

```
13300              VDISPENTITY("SUBPIC",NUMBER);
13400              VADDLAST(NUMBER,SETNUMBER);
13500              SETATT(NUMBER,-3,OLDX);
13600              SETATT(NUMBER,-4,Y);
13700              SETATT(NUMBER,-5,PICNO);
13800              IF DEBUG
13900              THEN BEGIN
14000                     CLEARBOX;
14100                     TFORM(INT,41,1,"DISPLAYENTITY #");
14200                     IFORM(INT,56,1,NUMBER);
14300                     TFORM(INT,60,1,"JUST CREATED.");
14400                     TFORM(INT,41,2,"X=");
14500                     IFORM(INT,43,2,ATTRIBUTE(NUMBER,-3));
14600                     TFORM(INT,41,3,"Y=");
14700                     IFORM(INT,43,3,ATTRIBUTE(NUMBER,-4));
14800                     TFORM(INT,41,4,"SUBPICTURE=");
14900                     IFORM(INT,52,4,ATTRIBUTE(NUMBER,-5));
15000                     TFORM(INT,41,5,"Press RETURN");
15100                     ERRORPAUSE;
15200                     CLEARBOX;
15300                   END;
15400            END;
15500        END;
15600      END OF PROCEDURE STORESUBPICTURES;
15700      PROCEDURE INSERTAT(X,Y,SOURCE);
15800      VALUE X,Y;
15900      INTEGER X,Y;
16000      STRING SOURCE;
16100      COMMENT OVERLAYS A PORTION OF ARRAY IMAGE WITH SOURCE, MAINTAINING
16200              THE TOTAL LENGTH OF THE STRING. Y AND X SPECIFY THE ARRAY
16300              ROW AND THE STARTING POSITION RESPECTIVELY;
16400      BEGIN
16500        STRING LEFTBIT,RIGHTBIT;
```

```
16600         INTEGER SOURCELENGTH;
16700         SOURCELENGTH:=LENGTH(SOURCE);
16800         LEFTBIT:=TAKE(IMAGE[Y],X-1);
16900         RIGHTBIT:=DROP(IMAGE[Y],X-1+SOURCELENGTH);
17000         IMAGE[Y]:=LEFTBIT CAT SOURCE CAT RIGHTBIT;
17100       END OF PROCEDURE INSERTAT;
17200     PROCEDURE ANYDYNAMICS;
17300     COMMENT ALLOWS THE USER TO SPECIFY WHICH, IF ANY, OF THE ATTRIBUTES
17400             OF THE  TYPE BEING DEFINED ARE TO BE DISPLAYED DYNAMICALLY,
17500             AND THE MODE OF DISPLAY TO BE  USED;
17600     BEGIN
17700       REAL REPLY,MODE,J,STATES;
17800       INTEGER HOWMANY,I,ATTRIBNO,XOFFSET,YOFFSET;
17900       CLEARBOX;
18000       TFORM(INT,41,1,"Do you wish to dynamically display any");
18100       TFORM(INT,41,2,"attributes? "),
18200       DO INPUTW(INT,72,2,6,REPLY)
18300       UNTIL REPLY="Y      " OR REPLY="YES    " OR REPLY="N      "
18400             OR REPLY="NO     ";
18500       IF REPLY="Y      " OR REPLY="YES    "
18600       THEN BEGIN
18700              WRITE(ATTRIBFILE,<"ENTITY TYPE: ",A6>,TYPE);
18800              TFORM(INT,41,3,"How many attributes? "),
18900              LIMITINPUTI(INT,62,3,3,HOWMANY,0,DYNAMICS),
19000              CLEARBOX;
19100              IF HOWMANY>0
19200              THEN FOR I:=1 STEP 1 UNTIL HOWMANY
19300              DO BEGIN
19400                   TFORM(INT,41,1,"Which attribute? ");
19500                   LIMITINPUTI(INT,58,1,6,ATTRIBNO,-7,ENTCOLS);
19600                   SETARR[SETNO,-(I-1)*4-5]:=ATTRIBNO;
19700                   TFORM(INT,41,2,"Is it to be displayed in text");
19800                   TFORM(INT,41,3,"(type 1), integer (2), or real (3) ");
```

```
19900                TFORM(INT,41,4,"format, or used as a pointer to a");
20000                TFORM(INT,41,5,"string (4)?");
20100                LIMITINPUTI(INT,53,5,1,MODE,1,4);
20200                CLEARBOX;
20300                CLEARFROM(INT,41,5);
20400                IF MODE NEQ 4
20500                THEN SETARR[SETNO,-(I-1)*4-8]:=MODE
20600                ELSE BEGIN
20700                       TFORM(INT,41,1,"How many states associated");
20800                       TFORM(INT,68,1,"with this");
20900                       TFORM(INT,41,2,"attribute? ");
21000                       INPUTI(INT,52,2,6,STATES);
21100                       WRITE(ATTRIBFILE,<"ATTRIBUTE NUMBER ",I3>,I);
21200                       CLEARBOX;
21300                       SETARR[SETNO,-(I-1)*4-8]:=-LABCOUNT;
21400                       TFORM(INT,41,1,"Attribute   Corresponding");
21500                       TFORM(INT,41,2,"Value       String");
21600                       FOR J:=1 STEP 1 UNTIL STATES DO
21700                       BEGIN
21800                         IFORM(INT,45,3,J);
21900                         INPUTSTRING(INT,53,3,26,LABELS[LABCOUNT],26);
22000                         CLEARFROM(INT,41,3);
22100                         WRITE(ATTRIBFILE,/,J,LABELS[LABCOUNT]);
22200                         LABCOUNT:=LABCOUNT+1;
22300                       END;
22400                     END;
22500                CLEARBOX;
22600                TFORM(INT,41,1,"X-offset from location of entity? ");
22700                LIMITINPUTI(INT,75,1,3,XOFFSET,-79,79);
22800                SETARR[SETNO,-(I-1)*4-6]:=XOFFSET;
22900                TFORM(INT,41,2,"Y-offset? ");
23000                LIMITINPUTI(INT,51,2,3,YOFFSET,-47,47);
23100                SETARR[SETNO,-(I-1)*4-7]:=YOFFSET;
```

```
23200                END;
23300              CLEARBOX;
23400            END;
23500      END OF PROCEDURE ANYDYNAMICS;
23600       %BODY OF PROCEDURE EDIT STARTS HERE;
23700       IMAGE[0]:=REPEAT(NULL,80);
23800       FOR I:=1 STEP 1 UNTIL 47
23900       DO IMAGE[I]:=IMAGE[0];
24000       IF DISPLAYELEMENTNUMBER=0
24100       THEN BEGIN
24200              CLEARBOX;
24300              TFORM(INT,41,1,"Name of element type?");
24400              INPUTW(INT,63,1,6,TYPE);
24500              NEWTYPE:=TRUE;
24600              FOR I:=1 STEP 1 UNTIL SETCOUNT
24700              DO IF SETARR[I,-4]=TYPE
24800                 THEN NEWTYPE:=FALSE;
24900              IF NEWTYPE
25000              THEN BEGIN
25100                     VDISPSET(TYPE,SETNO);
25200                     GETIMAGE;
25300                     STORESUBPICTURES(SETNO);
25400                     ANYDYNAMICS;
25500                   END
25600              ELSE BEGIN
25700                     CLEARBOX;
25800                     TFORM(INT,41,1,"An element type with that name has");
25900                     TFORM(INT,41,2,"already been defined.");
26000                     TFORM(INT,41,3,"Press RETURN");
26100                     ERRORPAUSE;
26200                   END;
26300            END
26400       ELSE BEGIN
```



```
26500              FOR I:=1 STEP 1 UNTIL SIZEOF(DISPLAYELEMENTNUMBER)
26600              DO BEGIN
26700                   INTEGER X,Y,MEMBER,POYNTER;
26800                   MEMBER:=IDENTITY(DISPLAYELEMENTNUMBER,I),
26900                   X:=ATTRIBUTE(MEMBER,-3),
27000                   Y:=ATTRIBUTE(MEMBER,-4);
27100                   POYNTER:=ATTRIBUTE(MEMBER,-5);
27200                   INSERTAT(X,Y,SUBPICTURE[POYNTER]);
27300                   DISPLAYELEMENTFORM(INT,0,0,MEMBER);
27400                 END;
27500              STORESUBPICTURES(DISPLAYELEMENTNUMBER);
27600            END;
27700       END OF PROCEDURE EDIT;
```