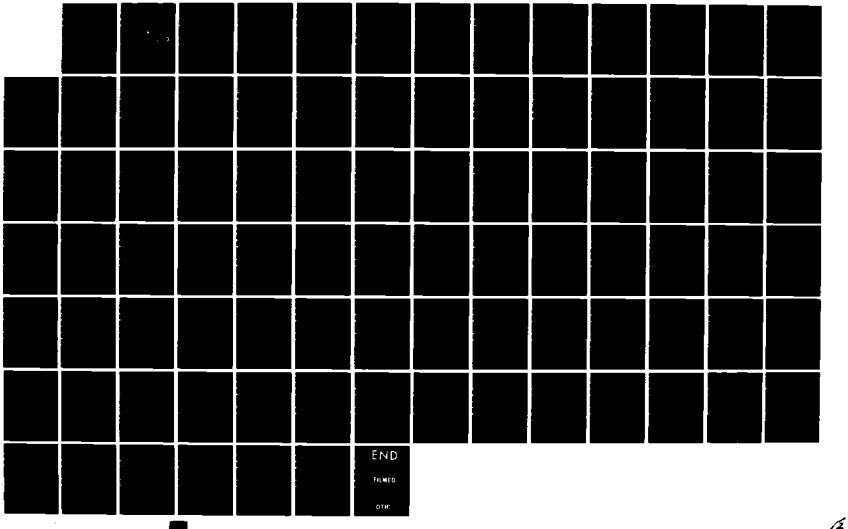
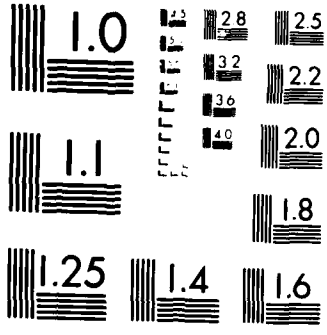
AD-A156 133 BENCHMARK EVALUATION OF PC SIMSCRIPT(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA A A DECKERT MAR 85 1/1

UNCLASSIFIED F/G 9/2 NL





NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

BENCHMARK EVALUATION OF PC SIMSCRIPT

by

Arthur Allen Deckert, Jr

March 1985

Thesis Advisor: Samuel H. Parry

Approved for public release; distribution unlimited

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* Benchmark Evaluation of PC SIMSCRIPT		5. TYPE OF REPORT & PERIOD COVERED Masters Thesis March 1985
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR*(s)* Arthur Allen Deckert, Jr.		8. CONTRACT OR GRANT NUMBER*(s)*
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93943		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93943		12. REPORT DATE
		13. NUMBER OF PAGES 87
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)*
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

SIMSCRIPT, Simulation, modeling, benchmark, PC.

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

As senior defense managers increasingly rely on models and simulations to justify major programs, greater attention is focused on the models and how they were constructed and validated. The SIMSCRIPT language has become the language of choice for military simulations and modeling primarily due to its readability and "world view" use of processes

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-LF-014-6601

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

and events to model the interaction of entities, attributes and sets. This thesis reviews the personal computer release of SIMSCRIPT which was made available during late 1984. Topics include results of hands-on testing, feasability as a teaching tool at the NAVAL POSTGRADUATE SCHOOL, evaluating transportability of programs between different releases of SIMSCRIPT, and benchmark time trials when programs are run on different machine configurations.

Approved for public release; distribution unlimited.

Benchmark Evaluation of PC SIMSCRIPT

by

Arthur Allen Deckert, Jr
Captain, United States Army
B.S.B.A., Washington University, 1976

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
March 1985

Author: Arthur Allen Deckert, Jr

Approved by: Samuel H. Parry, Thesis Advisor

Norman R. Lyons, Second Reader

Willis R. Greer, Jr., Chairman,
Department of Administrative Sciences

Kneale T. Marshall,
Dean of Information and Policy Sciences

ABSTRACT

As senior defense managers increasingly rely on models and simulations to justify major programs, greater attention is focused on the models and how they were constructed and validated. The SIMSCRIPT language has become the language of choice for military simulations and modeling primarily due to its readability and "world view" use of processes and events to model the interaction of entities, attributes and sets. This thesis reviews the personal computer release of SIMSCRIPT which was made available during late 1984. Topics include results of hands-on testing, feasability as a teaching tool at the Naval Postgraduate School, evaluating transportability of programs between different releases of SIMSCRIPT, and benchmark time trials when programs are run on different machine configurations.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to thank Mssrs. Glen Johnson and Hal Duncan of CACI, Inc., for aiding in the significant logistical support necessary to make this thesis possible. I also wish to express my gratitude to Ellen and Jay Roland for their assistance on the VAX/VMS system. The large SIMSCRIPT model used in Chapters 4 and 5 of this thesis was written by ROLANDS & ASSOCIATES CORPORATION in support of advanced research in Airland Combat modeling. Finally, I would like to thank Professor J. Hartman for the invaluable perspective of combat modeling that he provided to me.

I. INTRODUCTION

In recent years, combat modeling has taken on increasing importance as senior defense managers base decisions for new weapon systems, analyze logistics requirements, measure sustainability, and even determine force structure and composition using the results of sophisticated simulation programs. The SIMSCRIPT language has become the standard for simulations, primarily due to its "world view" and use of processes and events to model the interaction of entities, attributes, and sets.

A. THE NATURE OF SIMULATION AND MODELING

As our society has become more complex and diversified, the technical problems associated with managing its operation have also multiplied. Increasingly, managers are seeking new methods and management tools to help them address the complexity of modern day business.

One area that specifically addresses these modern intricate problems is the field of operations research. Operations research evolved from efforts to provide formal, efficient decision-making techniques for the design of an air defense in Britain during World War II. In operations research, techniques are employed that utilize symbolic models and mathematical applications to predict effects of alternative solutions to a problem. [Ref. 1: p. 2]

If the relationships which compose the model are simple enough, mathematical methods may be employed to derive exact information about the model and obtain an analytic solution. However, most real-world systems are too complex to permit realistic models to be solved analytically, and other means must be used for evaluation. One alternative is to use a computer simulation where a model of the system is evaluated numerically over a time period and data are gathered to estimate the true characteristics of the system.

Simulation has proven to be an effective method of pretesting proposed systems, plans, or policies before developing prototypes or conducting field tests. Simulation can also be used to model the effects of varying force structures in large scale combat models. Increasingly, models and simulations have become complex computer programs; written and maintained by project teams with a mixture of specialized professionals possessing either project management, modeling, programming, or functional area expertise.

B. DESCRIPTION AND HISTORY OF SIMSCRIPT II.5

The first version of SIMSCRIPT was developed at The Rand Corporation in the early 1960's by Harry Markowitz. Originally it was an extension of FORTRAN designed to simplify the coding of simulation models. As such, its implementation consisted of a pre-processor or translator

3. Immediate Traceback and Debugger

Syntax errors are only half of the programmer's battle. Often, even more frustrating, is the program that compiles successfully but then cannot execute or escape from an endless loop. Besides detailed compilation error messages, run time errors are also identified and a trace routine automatically invoked. The error description is followed by a snapshot of system variables and the program line number of execution when the fatal error was encountered. A separate catalog of messages also exists for abnormal job termination but traceback is not available here since the result of an abnormal termination is usually a "locked" program.

D. USER DOCUMENTATION

The PC SIMSCRIPT documentation does not attempt to teach the SIMSCRIPT language but does describe the unique features for operation on a PC. It includes a technical description on operation of the compiler, details to be addressed when transporting models to or from the PC, and a complete list of compile messages, run time messages, and abrupt task termination messages.

This is the first release of this 130 page document. Everything in it proved useful to me in my research, however suggestions for incorporation in future revisions would include:

instant during execution of this program. Management might balk at a detailed operations research explanation on why they should increase the number of servicing machines at a particular station, but the meaning of a graphical display picturing an unmanageable queue at a single point speaks for itself.

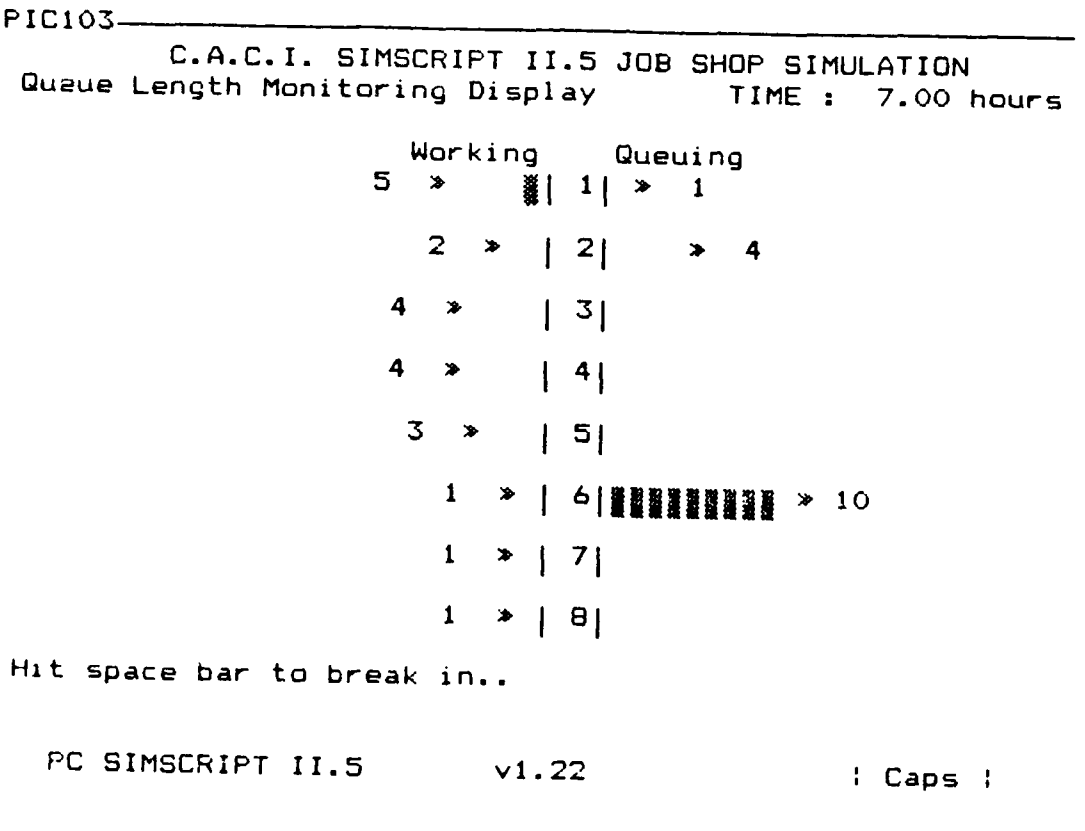


Figure 1 Example of On-Screen Graphics

disabled when a print command is issued to a printer that does not have its own buffer. Another important point is that multi-tasking must not be used to open two sets of processes on the same subdirectory of files. Finally, an Appendix to the PC SIMSCRIPT User's Manual warns that because of multiprogramming, a whole new group of termination possibilities arise. An overload of current tasks can result in one process having all its files "LOCKED" by SimLab if a memory allocation failure cannot be resolved. [Ref. 4: p. C-4]

2. Graphics

It has been said that a picture is worth a thousand words and the graphics capabilities of PC SIMSCRIPT will certainly find their way into both textual and live presentations. A new SIMSCRIPT language command, "DISPLAY", has been implemented for PC SIMSCRIPT that provides full color capability for run time presentation of graphics. The "TALLY" and "ACCUMULATE" commands have always given the capability to easily capture either total or average time data for selected variables and these results may now be displayed graphically. Presentation may either be summary graphs at program conclusion or dynamically changing graphics as the program executes. A sample program provided by CACI displays the varying queueing levels during operation of an eight station assembly line. Figure 1 is a dot-matrix printer copy of the monitor display at one

computers. The flexibility of multi-tasking, graphic color display, and rapid error reconciliation all contribute toward making this a viable system.

1. Virtual Machine/Multi-Windowing

The SimLab overlay allocates available memory enabling both multi-tasking and the processing of programs too large to run in available memory. Program instruction segments are removed from memory and then restored from memory when needed. This removal process can occur intentionally by releasing unneeded entity attributes (arrays) within SIMSRIPT routines or SimLab will perform it dynamically.

At any point during an edit, program execution, or a compilation, a new quarter-screen window may be opened and a different concurrent process SELECTed. The PC's function keys facilitate the transition between tasks by selecting which process to jump to and expanding or contracting windows to the desired size to facilitate work on the SELECTed task. When a process is terminated with the "QUIT" command, the current window "collapses" and any underlying processes automatically become current.

The power of today's PC enables this virtual machine capability, but the implementation is not complete and the user must be aware that there are risks involved. For example all the multi-tasking and windowing features are

according to the de facto standard. Encouraged by that discovery, experimentation revealed that "Ctrl-KB" and "Ctrl-KK" produced full line reverse video blocks that proclaimed block top and block bottom, again the accepted commands.

The editor commands on page three deal with marked block manipulation and exiting the edit mode. These functions are all initiated with the Escape key which then creates a new window requesting the prompt for the exact command desired. All block commands work as stated, but they also can be accomplished without leaving the text using the accepted "Ctrl-KV" to move, "Ctrl-KC" to copy, and "Ctrl-KH" to hide blocks. As with most editors, options exist to "SAVE" a file and continue editing (insurance against a power failure), save a file and "EXIT" (in this case back to SimLab), or to "QUIT" without saving changes. Although most PC editors give a second chance when abandoning an edited file, this "QUIT" command follows the convention of the Digital Equipment Corporation editor and is immediately irrevocable.

C. OTHER FEATURES

By taking advantage of the significant power and simple operating environment of today's microcomputers, PC SIMSCRIPT is able to offer a programming package more advanced in many ways than is possible with mainframe

Files of up to 250,000 bytes are entirely memory-resident so editing operations are nearly immediate. Unfortunately, the price of direct addressing which is necessary for the immediate editing capability, must be paid in the beginning when a file is loaded. A source file of 170,000 bytes required nearly four minutes to retrieve from the SELECTed directory and load into the editor. Typically though, this inconvenience is infrequent since most edits are on individual modules which will be less than 5000 bytes in length. Load time to edit most modules is less than fifteen seconds.

The editing commands are described in three pages of the user's manual. The first page describes the insert toggle and cursor movement in increments of space, word, line, and page. It was a pleasant surprise that the commands resembled the de facto industry standard and were similar to the commands from MicroPro's Wordstar®, Ashton-Tate's dBASE®, and Borland International's Turbo Pascal® and Sidekick®.

The second page of editor instructions addresses character and line delete functions and block marking procedures. Deletion procedures were again the defacto standard but block marking commands were not. At this point some inconsistencies attributable to the newness of this software product began to appear. The commands for marking block top and bottom, "Ctrl-T" and "Ctrl-B," did not work as specified. However "Ctrl-T" did delete word right

The "UNLOCK" command is used to unlock modules that became "locked" as the result of either a KILL command or an abnormal end to a COMPILE operation (program crash).

B. SIMEDIT, THE FULL SCREEN EDITOR

The editor furnished with SIMSCRIPT is a fairly powerful full screen editor. Programs may either be written on a text editor and IMPORTed to SimLab or created directly using SimEdit. However, in nearly all instances the editor should probably be used for making changes after a model is loaded. Of tremendous aid during model building is the incorporation of compiler messages into each module's program source file. When EDITing program modules which have compile warnings or errors, the location of the syntax error is pinpointed by a reverse video block diagnostic message. One caution: the editor allows input of up to 128 characters per line, and the diagnostics may be that long, but the language standards result in the compiler only recognizing the first 80 character positions for each line.

Features of the editor include complete text manipulation to include block seek, mark, move, copy, and delete. A nice feature for writing structured program code is the auto-track indentation which begins every new line at the first character location that was used on the previous line. An annoying result of this however, was the loss of the normal tab function.

or print the current version, modules must be listed individually. Naming the Workfile will not display the current program code if any changes to individual modules have been made. Finally, the "EDIT" command invokes the SimEdit editor on the named module.

2. Program Compilation and Execution

The "COMPILE" command results in compilation for the SELECTed program. One attractive feature of SIMSCRIPT is that during model building, any module which has compiled successfully or even compiled with warnings need not be recompiled unless it has been edited (other than the Preamble which must recompile if any other module has changed). It is gratifying as a new model is constructed to see the compile time drop as successive modules are debugged.

The "RUN" command executes a successfully compiled program. If it is the first execution of a program, a linking routine is automatically invoked. Output resulting from the RUNning of a SELECTed program can be directed to the screen, a printer, a specified DOS file, or as input to another SIMSCRIPT program. Control of a running program is maintained by means of "Ctrl-S" to suspend output, "Ctrl-Q" to resume output and "Ctrl-C" to terminate the currently running program.

The "KILL" command halts the compilation of a program and "locks" all modules which were to be COMPILEd.

imported at once, they must all be contained within the same DOS file since the Workfile is written over each time the "IMPORT" command is executed. The "EXPORT" command permits the transfer to a DOS file of an updated Workfile which contains the current source code for each module within a program. Although not emphasized, this is an extremely important command for three reasons:

1. Removing noncurrent models facilitates fixed disk space management. SIMSCRIPT processing creates many files and requires large amounts of on-line storage. After a series of COMPILE and EDITs a 90 line program 2600 bytes in size may have created a subdirectory containing 60 separate DOS files and occupying 250,000 bytes of storage. If models are to be deleted from the fixed disk to free up space, the source code is easiest to save.

2. The resulting output of program source code is in the format necessary to transport the program for input to another machine.

3. This is the only method for obtaining a continuous listing for the current program. Because the Workfile and other program modules are only recompiled when edited, the "EXPORT" command becomes the only method for obtaining a complete listing of the current version for all modules.

The remaining five file management commands are generally self explanatory. The "DELETE" command erases all files for the named module while the "RECOVER" command restores a module's current source code file to its state before the previous edit (restores DOS file type .BAK). The "TYPE" and "PRINT" commands display the named modules to the screen or printer, respectively. As noted before, to view

interesting result of this is that SimLab assigns the names to the DOS files that it manages. Thus the user is not constrained by DOS conventions that would otherwise limit the length of module names or the use of certain characters such as the period (.) within file names.

An existing SIMSCRIPT program is retrieved, or a new one begun with the "SELECT" command. This command issues the DOS change directory (chdir) command for existing programs or the make directory (mkdir) command if a new program is initiated. When a new program is started, the newly created subdirectory will already contain a Namefile which will become the building block for linking future program modules together.

The status of a particular program is checked with the "STATUS" command. The status of each module within a SELECTed directory is displayed as either edited, old, compiled successfully, compiled with warnings, compiled with errors, not yet written, or locked as the result of an abnormal end to a compilation.

Model building and transportability between machines is facilitated by means of the "IMPORT" and "EXPORT" commands. One or more modules can be brought into the SELECTed model and written to the Workfile module for addition to the model during the next compilation. Although more than one module, or even an entire model, may be

A. SIMLAB, THE OPERATING SYSTEM OVERLAY

SimLab is described as a feature that provides a complete modeling development environment. Actually SimLab is an operating system overlay that resides between SIMSCRIPT's compiler and editor and the PC's disk operating system (DOS). SimLab contains seventeen commands, most of which facilitate either file management or program compilation and execution. However, one of these commands that is unique, is the ampersand symbol (&) which permits direct execution of the DOS commands. Although any DOS command may be executed with the ampersand (to include reformatting your disk), use of this command is limited because any action that results in video screen output may result in a "crash" of the entire system that requires a power-off-power-on sequence to cure. By the same token, underlay programs and "desk organizers" such as Borland International's Sidekick® must not be invoked within SimLab.

1. File Management

File management for SIMSCRIPT programs is probably the reason SimLab was necessary. Each model resides as a separate DOS subdirectory and since programs are modular in design, DOS files are maintained for source code, source backup, object code, and linked object code for each module of a program. In addition each program has several overhead files that tie the modules together. This complex file management function is nearly transparent to the user. One

II. REVIEW OF PC SIMSCRIPT FEATURES

In addition to the SIMSCRIPT compiler, the PC SIMSCRIPT product includes SimLab, an operating system overlay, and SimEdit, a full screen editor. The software operates on an IBM Personal Computer or any of the series of look alikes that operate in a PC-DOS or MS-DOS environment and are based on the Intel 8086/88, 80186/86, or 80286 chip family. The minimum machine configuration must include 320K bytes of main memory, a hard disk with 5 to 40 megabytes of storage, and installation of the appropriate 64-bit floating-point math coprocessor chip (Intel 8087 or 80287). [Ref 4: p. iv]

CACI, Inc. has a reputation for being first class and the user is pleasantly surprised when the PC SIMSCRIPT package arrives overnight via Federal Express. Included with two software disks and a PC SIMSCRIPT users manual are four large textbooks published by CACI. These reference manuals address simulation, model building, and the syntax for the SIMSCRIPT language. The additional textbooks are an absolute necessity for anyone not already versed in SIMSCRIPT programming. The PC SIMSCRIPT user's manual received was Release 1.1 dated October 1984, and the software received was Release 1.02 dated October 1984. In December, software release 1.22 was received and this is the software version that was evaluated.

C. PURPOSE OF THESIS

The purpose of this thesis is to evaluate SIMSCRIPT II.5 and PC SIMSCRIPT with respect to four points: 1) review of PC SIMSCRIPT, features, ease of use, and documentation; 2) evaluate SIMSCRIPT as a teaching tool; 3) examine transportability of SIMSCRIPT code; and 4) benchmark the compilation and run of several models on a Digital Research VAX computer, and an IBM PC.

that read SIMSCRIPT input code and produced FORTRAN language statements. An improved version, SIMSCRIPT I.5, eliminated the dependence on FORTRAN by upgrading the translator to a full scale compiler that directly produced assembly language code. [Ref. 2: p. ii]

In 1975, a major revision was made and now SIMSCRIPT II.5 permits simulation models to use a process-interaction approach in addition to the previously supported event-scheduling approach. The main advantage is that a process is a programming structure that permits a time-ordered sequence of interrelated events separated by passages of time. This approach views a system as consisting of processes, resources, events, attributes, entities, and sets. [Ref 3: p. 128] The language is free-form, English-like, and it embodies current structured design principles such as structured programming and modularity. SIMSCRIPT II.5 is a proprietary product of CACI, Inc., and is available for most major computers manufactured by IBM, CDC, Honeywell, Univac, Digital Equipment, Perkin-Elmer, NCR, PRIME, and also through time-sharing networks. [Ref. 2: p. iii] In 1984 a full implementation of SIMSCRIPT II.5 was released for personal computer (PC) configurations, thus making this powerful language an alternative for a much larger range of users. All further references to SIMSCRIPT in this thesis will actually be referring to the current SIMSCRIPT II.5 language.

1. The final appendix, "Installing PC SIMSCRIPT" is listed on the fourth page of the table of contents. The existence of an installation program that automatically copies the 150 files into the 15 appropriate directories and subdirectories needs to be made a little clearer.

2. The four pages devoted to the description of the SimLab commands listed in alphabetical order is not adequate. They should be grouped by function and include sample statements exercising some available options similar to the format used throughout the SIMSCRIPT II.5 Reference Handbook. Also, perhaps commands should be grouped by function rather than just listed alphabetically.

3. The section on SimEdit commands should be expanded and all features fully explained when the existing documentation errors are corrected.

4. The ten pages that describe the three included sample programs are helpful, and they will certainly be run by all new PC SIMSCRIPT users. They do not, however, constitute a tutorial as the term applies to PC software today.

5. The on-line compile, run, and termination messages are beneficial, yet little was gained by taking 32 pages to list all the messages since most are self explanatory.

6. The section describing the parallel operation of the 8087 and 8088 architecture to emulate an S-machine multitasking computer and the design of the three-pass compiler is very technical. This is nice "gee whiz data" but it is above the level of all but advanced Computer Science personnel who, because of a lack of documentation, are unable to modify any of the supplied routines anyway. Of more benefit would be better instructions to the end user.

III. USE OF SIMSCRIPT AS A TEACHING TOOL

Though SIMSCRIPT is considered a high level programming language, it behaves in a manner that is different from other high level languages such as fourth generation languages (relational data systems). Both languages perform data and file management tasks and relieve the programmer of writing detailed low level routines. However, where these tasks are nearly totally transparent for relational high level languages, SIMSCRIPT has literally hundreds of system controlled functions and variables that are accessible to the programmer and may be addressed for complex modeling requirements. The purpose of this chapter is to comment on how SIMSCRIPT might be incorporated into a simulations course at the Naval Postgraduate School.

A. BASIS FOR EVALUATION

The hands-on experience for the review of PC SIMSCRIPT features in the previous chapter was obtained by preparing solutions for the three projects that were assigned in the OS 3603, Wargaming and Simulation course, taught during the Fall quarter 1984 by Professor J. Hartman. Programming per se was not taught, and students were free to solve the problems using any language they were familiar with. Programming languages used by the students included BASIC,

Pascal, and FORTRAN. Complete problem descriptions, SIMSCRIPT solutions, and program output listings for each of the three projects are contained in Appendices A, B, and C.

B. RESULTS

While much has been written about measuring and estimating programmer productivity [Ref. 5] and numerous works address the processing efficiency of different languages, both of these topics are beyond the scope of this thesis. Instead, typical student's solutions in their preferred languages were selected for comparison to the SIMSCRIPT solutions. It is important to note that none of the students were programmers, that the programs not were written with the goal of optimization, and finally that an individual's solution is just that--one of an infinite number of possible solutions to a problem. To simplify the analysis performed in this chapter, the difficult variation for project one and the simple variation for project three have been eliminated. The review is thus limited to three programs which still assess the widest range of programmer difficulty.

The best method for measuring program length is generally accepted to be lines of code. Table 1 displays the lines of code for each of the projects after all blank lines and comments were removed.

Table 1 Lines of Executable Code

	Project 1	Project 2	Project 3
BASIC	55	70	240
FORTRAN	43	109	186
Pascal	185	176	372
SIMSCRIPT	69	149	102

A review of Table 1 prompts one to wonder why Pascal appears so inefficient when lines of code is used as a yardstick. One answer might be that lines of code is not a good tool when comparing structured languages such as Pascal and SIMSCRIPT with the non-structured formats of FORTRAN and BASIC. Structured languages stress readability and often use self documenting language features on lines by themselves towards this goal. To account for this, Table 2 presents the number of characters of code that were present in each program file after leading and trailing line numbers (BASIC and FORTRAN) and indentation spaces (Pascal and SIMSCRIPT) were deleted. Unfortunately, even with this technique, no clear trends emerge. Thus, the following section describes details of the individual projects and how they were handled by the different languages. Additionally, one interesting fact easily illustrated at this point is the comparative total size and number of DOS files in the SELECTed SIMSCRIPT program subdirectory after one compilation and execution.

Table 2 Characters of Executable Code

	Project 1	Project 2	Project 3
BASIC	2718	1567	6241
FORTRAN	1194	3723	5916
Pascal	5056	4427	8748
SIMSCRIPT (source)	2156	4132	3314
SIMSCRIPT (storage)	53,248	118,784	258,048
(files)	13	27	59

C. REVIEW OF PROJECT SOLUTIONS

Reviewing the number of lines of code does not adequately address the important points of readability and future maintainability which are provided by the Pascal and SIMSCRIPT formats. Similarly, it is difficult to draw meaningful conclusions from the number of characters in the source file since this is so case specific. The BASIC and FORTRAN solutions reviewed for Project One were almost identical, yet a large difference in file size resulted from the use of 25 character descriptive variables in BASIC while the FORTRAN solution used three and four character variables with no self documenting features.

The three projects increased in complexity and programming difficulty as the class topics progressed to more difficult simulation principles. By the last project, the power of SIMSCRIPT was illustrated as the other

languages required significant efforts to create routines for timing, queueing, and statistics compilation.

1. Project One

Project One included two variations and was a simple, deterministic, fixed time step problem. The project was composed of a series of straightforward computations and all solutions were similar.

2. Project Two

Project Two was a stochastic simulation with one active process and required statistical evaluation of results. Once again, most of the problem consisted of simple computations, however, the requirements for random and normal number generators along with the need to calculate mean and variance, which are part of the SIMSCRIPT language, were handled by library calls in FORTRAN, and had to be written for the Pascal and BASIC solutions. Worth noting here, the project required two separate processes, but the assignment was significantly simplified by requiring only one at a time to be active. A true life scenario would have required simultaneous operation of both processes. The SIMSCRIPT solution is easily modified to provide for simultaneous operation of both processes while the program complexity takes on a different dimension for the other solutions.

3. Project Three

Project Three, which also contained two variations, was a stochastic event sequenced simulation with multiple active processes, event queueing, and statistics compilation. There is no question of the programming language of choice for problems like this. SIMSCRIPT provides a system clock, a host of random number generators, dynamic queueing, temporary entities with attributes (arrays), and compilation of sample or time average statistics. Finally, no amount of programmer documentation for the other solutions could match the understandability provided by the SIMSCRIPT structure.

D. CONCLUSIONS

Since SIMSCRIPT is recognized as the standard simulation language for military applications, and because of its powerful modeling capabilities, consideration should be given to increasing the exposure afforded this language. The question of where SIMSCRIPT might fit into the curriculum at the Naval Postgraduate School is not an easy one. The discussion focuses first on which curriculum's students should be targeted, second on what level or levels should the language be taught, and finally what resources will be necessary to give SIMSCRIPT this increased exposure.

Curricula whose students might be considered candidates for learning SIMSCRIPT would include management oriented

ones, such as Operations Research, Computer Systems Management, and Computer Science and tactically oriented curricula such as Command, Control, and Communications, Anti-Submarine Warfare, and Space Systems.

The level at which SIMSCRIPT should be taught is a little easier to express an opinion on. First, there is probably not a need to teach the language on an advanced level. The goal of the school should be widest coverage on a maximum number of topics. The few individuals who anticipate acquiring an in-depth programming knowledge of SIMSCRIPT could probably tailor an individual readings class to accomodate their requirements. On the other hand, a single introductory course on SIMSCRIPT is not a good idea because it would not provide the student with an adequate appreciation for the power of SIMSCRIPT, particularly regarding its advantages over other programming alternatives. My recommendation would be to have two separate courses with the first structured similar to the current OS 3603, Wargaming and Simulation. A following course could then begin with the SIMSCRIPT solution for the last project previously assigned and then proceed onto more advanced SIMSCRIPT projects. An alternative recommendation might be to increase either the number of class hours (currently three) or the number of lab hours (currently one) for the OS 3603 course. This would permit coverage of the same current material with enough time left over to present

the advantages realized if SIMSCRIPT were utilized to solve the last programming project.

Currently at the Naval Postgraduate School, SIMSCRIPT is implemented for the VAX/VMS system on the War Lab's 11/780 and the IBM/VM batch system for the IBM 3033 mainframe. SIMSCRIPT requires a great deal of computational power and usage on the scale for entire classes would probably require either installing it in within the interactive IBM/CMS environment or heavy reliance on the PC configuration.

IV. TRANSPORTABILITY OF SIMSCRIPT CODE

The primary question upon release of PC SIMSCRIPT was if the implementation was complete or if only a subset of the language commands would be available on the PC. In this chapter the transportability of SIMSCRIPT models is evaluated regarding implementation of language features, transportability between machines, and variance of model results when stochastic models are processed on the different hardware and software configurations.

The PC SIMSCRIPT release is intended to be a full implementation of the SIMSCRIPT language. All features in current specifications for the language have been implemented including character string manipulation in the TEXT mode which is a recent update for VAX-11 and IBM S/370 installations. Additionally, PC SIMSCRIPT includes even more compiler warning messages than are implemented on current mainframe versions. New diagnostics include checking use of local variables and monitoring number of arguments passed between routines or functions. Limitations for the PC implementation appear to be only the size of the model and the constructs it employs as confined by the PC memory disk space and the microcomputer addressing limits. Restrictions stated in the user's manual [Ref. 4: p. 5-2] that limit the size of a model that can execute include:

- The maximum data storage space that may be allocated to temporary entities, array storage, and text storage are each 250,000 bytes.

- The maximum size for an array assignment is 4000.

- The maximum number of given and yielding arguments for a routine is 63.

- A maximum of 255, 32-bit variables is allowed per routine. Each DOUBLE variable occupies two 32-bit words of storage.

- The maximum length for each routine's compiled object code is 65,535 bytes.

- A maximum of 3000 routines and functions are allowed within a given application.

The transfer of source code between PC and VAX SIMSCRIPT configurations was easily accomplished during this review, possibly even more easily than moving between different mainframe computers. The typical problem of formatting a tape for the receiving system was bypassed by transferring text files between machines via a dial-up modem. Transfer from the PC to the VAX did, however, result in a line feed symbol <LF> being inserted at the start of each line. Presumably the communications software might have been reconfigured to eliminate this problem. The models thus transferred would not compile until the <LF>s were deleted using the VAX editor. Transfer from the VAX to the PC was error free.

Modeling results must be reproducible and one of today's requirements for a computer program's random number

generator is for the random number stream to be identical given the same set-up conditions or seed values. The SIMSCRIPT language implements ten pseudorandom number generators with floating point precision using the Lehmer technique [Ref. 4: p. 9-9]. Since each of the ten generators receives the same unique seed value as the corresponding generator in other SIMSCRIPT implementations, results in stochastic models should be similar. The programs in Appendices B and C both required the simultaneous use of several random number streams and program results were identical when these programs were executed.

V. BENCHMARK COMPARISON

Benchmarking is a practice usually employed when certifying a computer's capability to run a sample or simulated job mix to the satisfaction and within constraints posed by a prospective end user. Configuration optimization or "tuning" a system involves modification to hardware (often multi-vendor) parameters, the operating system, and data placement algorithms. Hardware considerations include modifications to CPU speed in instructions per second, slave store hit rates, disk rotational latency, controller loadings, and drum address organization. Software considerations include slave store algorithms, virtual store management, logical record management, multiple buffering, and indexing methods. [Ref. 6: p. 33]

Many of these points are relevant for the VAX and PC computers. However, with PC configurations, the user is generally constrained by the limits of the "package." The PC may be configured to a maximum memory expansion but beyond that the only variable would be performance characteristics unique to the manufacturer of the hard disk and controller units. For purposes of this thesis, existing configurations were used without modifications. The time trials in Table 3 are based on wall clock time. This is valid since the PC's are both single user systems. Trials

for the multi-user VAX system were performed when there were no other system users. The IBM-XT (XT) tested contained 640K bytes of memory while the IBM-AT (AT) contained 512K bytes of memory. The VAX was a model 11-780 with 6000K bytes of memory.

The four programs tested include the same three evaluated in Chapter 4 and a significant program that analyzed networking problems associated with Airland Combat modeling. This program contained 43 routines, 4471 lines, and used 169,692 bytes of storage. It is well documented and employed SIMSCRIPT modularity principles so it would probably decrease in size by a third if its size were stated in the same terms as the presentations in Tables 1 and 2 from Chapter 3.

Table 3 Time Trials: IBM-XT / IBM-AT / VAX 11-780
Time in Minutes

	Compile	Link	Execute
Project 1	4/1.9/.3	.1/.2/.3	1.2/ .8/.3
Project 2	10/4.4/.7	.2/.2/.3	3.3/1.9/.3
Project 3	22/9.3/1.2	.8/.6/.4	6.1/3.1/.4
Network	135/54.4/9.7	*	*

*The Network model compiled correctly on the PC's but was not executed because of VAX unique system calls formating for screen input.

GOOD GUYS DIRECT FIRE	GOOD GUYS INDIRECT FIRE	BAD GUYS DIRECT FIRE	BAD GUYS INDIRECT FIRE	TIME (MINUTES)
4.34	100.00	20.23	100.00	128.
4.25	100.00	19.97	100.00	129.
4.16	100.00	19.71	100.00	130.
4.07	100.00	19.45	100.00	131.
3.98	100.00	19.20	100.00	132.
3.89	100.00	18.96	100.00	133.
3.80	100.00	18.71	100.00	134.
3.71	100.00	18.48	100.00	135.
3.64	100.00	18.24	100.00	136.
3.48	100.00	18.02	100.00	137.
3.29	100.00	17.79	100.00	138.
3.12	100.00	17.57	100.00	139.
2.99	100.00	17.36	100.00	140.
2.86	100.00	17.14	100.00	141.
2.73	100.00	16.93	100.00	142.
2.61	100.00	16.73	100.00	143.
2.49	100.00	16.53	100.00	144.
2.37	100.00	16.33	100.00	145.
2.26	100.00	16.14	100.00	146.
2.15	100.00	15.95	100.00	147.
2.04	100.00	15.76	100.00	148.
1.94	100.00	15.57	100.00	149.
1.84	100.00	15.39	100.00	150.
1.74	100.00	15.22	100.00	151.
1.65	100.00	15.04	100.00	152.
1.55	100.00	14.87	100.00	153.
1.46	100.00	14.70	100.00	154.
1.37	100.00	14.53	100.00	155.
1.29	100.00	14.37	100.00	156.
1.21	100.00	14.21	100.00	157.
1.12	100.00	14.05	100.00	158.
1.05	100.00	13.90	100.00	159.

GOOD GUYS		BAD GUYS		
DIRECT FIRE	INDIRECT FIRE	DIRECT FIRE	INDIRECT FIRE	TIME (MINUTES)
13.30	100.00	32.16	100.00	96.
13.39	100.00	31.66	100.00	97.
13.00	100.00	31.16	100.00	98.
12.61	100.00	30.63	100.00	99.
12.24	100.00	30.21	100.00	100.
11.87	100.00	29.75	100.00	101.
11.52	100.00	29.30	100.00	102.
11.17	100.00	28.86	100.00	103.
10.33	100.00	28.42	100.00	104.
10.50	100.00	28.00	100.00	105.
10.18	100.00	27.58	100.00	106.
9.87	100.00	27.17	100.00	107.
9.56	100.00	26.77	100.00	108.
9.26	100.00	26.38	100.00	109.
8.97	100.00	26.00	100.00	110.
8.69	100.00	25.62	100.00	111.
8.41	100.00	25.25	100.00	112.
8.14	100.00	24.89	100.00	113.
7.88	100.00	24.54	100.00	114.
7.62	100.00	24.19	100.00	115.
7.37	100.00	23.85	100.00	116.
7.13	100.00	23.51	100.00	117.
6.89	100.00	23.18	100.00	118.
6.66	100.00	22.86	100.00	119.
6.44	100.00	22.55	100.00	120.
6.22	100.00	22.24	100.00	121.
6.01	100.00	21.94	100.00	122.
5.80	100.00	21.64	100.00	123.
5.60	100.00	21.35	100.00	124.
5.40	100.00	21.06	100.00	125.
5.21	100.00	20.78	100.00	126.
5.02	100.00	20.50	100.00	127.

GOOD GUYS		BAD GUYS		
DIRECT FIRE	INDIRECT FIRE	DIRECT FIRE	INDIRECT FIRE	TIME (MINUTES)
33.39	100.00	55.35	100.00	64.
33.00	100.00	54.36	100.00	65.
32.12	100.00	53.39	100.00	66.
31.26	100.00	52.44	100.00	67.
30.43	100.00	51.50	100.00	68.
29.61	100.00	50.59	100.00	69.
28.82	100.00	49.70	100.00	70.
28.04	100.00	48.83	100.00	71.
27.29	100.00	47.98	100.00	72.
26.55	100.00	47.14	100.00	73.
25.83	100.00	46.33	100.00	74.
25.13	100.00	45.53	100.00	75.
24.44	100.00	44.75	100.00	76.
23.77	100.00	43.98	100.00	77.
23.12	100.00	43.23	100.00	78.
22.49	100.00	42.50	100.00	79.
21.87	100.00	41.78	100.00	80.
21.26	100.00	41.08	100.00	81.
20.67	100.00	40.39	100.00	82.
20.10	100.00	39.72	100.00	83.
19.54	100.00	39.06	100.00	84.
18.99	100.00	38.42	100.00	85.
18.46	100.00	37.79	100.00	86.
17.94	100.00	37.17	100.00	87.
17.43	100.00	36.56	100.00	88.
16.94	100.00	35.97	100.00	89.
16.45	100.00	35.39	100.00	90.
15.98	100.00	34.82	100.00	91.
15.52	100.00	34.27	100.00	92.
15.08	100.00	33.72	100.00	93.
14.64	100.00	33.19	100.00	94.
14.21	100.00	32.67	100.00	95.

GOOD GUYS		BAD GUYS		
DIRECT FIRE	INDIRECT FIRE	DIRECT FIRE	INDIRECT FIRE	TIME (MINUTES)
78.20	100.00	102.34	100.00	32.
76.23	100.00	100.30	100.00	33.
74.30	100.00	98.30	100.00	34.
72.42	100.00	96.35	100.00	35.
70.59	100.00	94.45	100.00	36.
68.80	100.00	92.59	100.00	37.
67.05	100.00	90.77	100.00	38.
65.35	100.00	88.99	100.00	39.
63.69	100.00	87.25	100.00	40.
62.06	100.00	85.55	100.00	41.
60.48	100.00	83.88	100.00	42.
58.94	100.00	82.26	100.00	43.
57.43	100.00	80.67	100.00	44.
55.96	100.00	79.12	100.00	45.
54.52	100.00	77.60	100.00	46.
53.12	100.00	76.11	100.00	47.
51.75	100.00	74.66	100.00	48.
50.42	100.00	73.24	100.00	49.
49.12	100.00	71.86	100.00	50.
47.85	100.00	70.50	100.00	51.
46.61	100.00	69.17	100.00	52.
45.40	100.00	67.87	100.00	53.
44.22	100.00	66.60	100.00	54.
43.07	100.00	65.36	100.00	55.
41.95	100.00	64.15	100.00	56.
40.85	100.00	62.96	100.00	57.
39.78	100.00	61.80	100.00	58.
38.74	100.00	60.67	100.00	59.
37.72	100.00	59.56	100.00	60.
36.73	100.00	58.47	100.00	61.
35.76	100.00	57.41	100.00	62.
34.82	100.00	56.37	100.00	63.

GOOD GUYS		BAD GUYS		
DIRECT FIRE	INDIRECT FIRE	DIRECT FIRE	INDIRECT FIRE	TIME (MINUTES)
175.00	100.00	200.00	100.00	0.
170.70	100.00	195.73	100.00	1.
166.50	100.00	191.55	100.00	2.
162.41	100.00	187.47	100.00	3.
158.41	100.00	183.48	100.00	4.
154.51	100.00	179.59	100.00	5.
150.70	100.00	175.78	100.00	6.
146.98	100.00	172.07	100.00	7.
143.35	100.00	168.44	100.00	8.
139.81	100.00	164.89	100.00	9.
136.36	100.00	161.42	100.00	10.
132.98	100.00	158.03	100.00	11.
129.69	100.00	154.73	100.00	12.
126.48	100.00	151.49	100.00	13.
123.34	100.00	148.33	100.00	14.
120.28	100.00	145.25	100.00	15.
117.30	100.00	142.23	100.00	16.
114.38	100.00	139.28	100.00	17.
111.54	100.00	136.40	100.00	18.
108.76	100.00	133.59	100.00	19.
106.05	100.00	130.84	100.00	20.
103.41	100.00	128.15	100.00	21.
100.83	100.00	125.53	100.00	22.
98.31	100.00	122.96	100.00	23.
95.85	100.00	120.45	100.00	24.
93.45	100.00	118.00	100.00	25.
91.11	100.00	115.61	100.00	26.
88.83	100.00	113.27	100.00	27.
86.60	100.00	110.98	100.00	28.
84.42	100.00	108.74	100.00	29.
82.30	100.00	106.56	100.00	30.
80.22	100.00	104.42	100.00	31.

```
ROUTINE HEADING

PRINT 4 LINES THUS
    GOOD GUYS                          BAD GUYS
DIRECT   INDIRECT              DIRECT    INDIRECT            TIME
 FIRE      FIRE                 FIRE       FIRE           (MINUTES)

END   ''HEADING
ROUTINE SUMMARY

    START NEW PAGE

    PRINT 12 LINES WITH
        TIME.V,
        TOTAL.X1_KILLED_BY_Y1, TOTAL.X1_KILLED_BY_Y2,
        TOTAL.Y1_KILLED_BY_X1, TOTAL.Y1_KILLED_BY_X2,
        Y1, Y2, X̄1, X2 THUS
```


MINUTE ***	:	VICTIM Y1	Y2	X1	X2
K : GOOD GUY DIRECT FIRE (Y1)	:			***.**	
I : GOOD GUY INDIRECT FIRE (Y2)	:			***.**	
L :	:				
L : BAD GUY DIRECT FIRE (X1)	:	***.**			
E : BAD GUY INDIRECT FIRE (X2)	:	***.**			
R :	:				
CURRENT FORCE COMPOSITION	:	***.**	***.**	***.**	***.**

```
END   ''SUMMARY
```

```
        LET BETA = .0002
        LET B = .004
        LET ALPHA = .0001
        LET A = .013
        LET LINES.V = 36

    UNTIL Y1 <= 0 OR X1 <= 0

        DO

            IF LINE.V = 1
                CALL HEADING
            ALWAYS

            PRINT 1 LINE WITH Y1, Y2, X1, X2, TIME.V THUS
***.**      ***.**             ***.**      ***.**             ****

            LET X1_KILLED_BY_Y1 = A * Y1
            LET X1_KILLED_BY_Y2 = ALPHA * Y2 * X1
            LET Y1_KILLED_BY_X1 = B * X1
            LET Y1_KILLED_BY_X2 = BETA * X2 * Y1

            LET TOTAL.X1_KILLED_BY_Y1 = TOTAL.X1_KILLED_BY_Y1 + X1_KILLED_BY_Y1
            LET TOTAL.X1_KILLED_BY_Y2 = TOTAL.X1_KILLED_BY_Y2 + X1_KILLED_BY_Y2
            LET TOTAL.Y1_KILLED_BY_X1 = TOTAL.Y1_KILLED_BY_X1 + Y1_KILLED_BY_X1
            LET TOTAL.Y1_KILLED_BY_X2 = TOTAL.Y1_KILLED_BY_X2 + Y1_KILLED_BY_X2

            LET X1 = X1 - X1_KILLED_BY_Y1 - X1_KILLED_BY_Y2
            LET Y1 = Y1 - Y1_KILLED_BY_X1 - Y1_KILLED_BY_X2

           TIME.V = TIME.V + 1

        LOOP

    END   ''COMBAT
```

```
PREAMBLE

'' *** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ***
'' *** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ***
'' *** Simple Lanchester model to simulate force on force attrition                          ***
'' *** processes.  The good guy defenders possess direct fire of size                       ***
'' *** (Y1) with a Direct Fire Casualty Rate of (A), they are supported                     ***
'' *** by an invulnerable indirect fire force of size (Y2) with an                          ***
'' *** Indirect Fire Casualty Rate of (ALPHA).  The bad guy attackers                        ***
'' *** are of size (X1) with a Direct Fire Casualty Rate of (B).  They                      ***
'' *** also are supported by an invulnerable indirect fire force of size                    ***
'' *** (X2) with an Indirect Fire Casualty Rate of (BETA).                                  ***
'' *** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ***
'' *** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ***

    DEFINE

        X1, X2, Y1, Y2,
        X1_KILLED_BY_Y1, X1_KILLED_BY_Y2,
        Y1_KILLED_BY_X1, Y1_KILLED_BY_X2,
        TOTAL.X1_KILLED_BY_Y1, TOTAL.X1_KILLED_BY_Y2,
        TOTAL.Y1_KILLED_BY_X1, AND TOTAL.Y1_KILLED_BY_X2

    AS REAL VARIABLES

END   ''PREAMBLE
MAIN
    CALL COMBAT
    CALL SUMMARY
END    ''MAIN

ROUTINE COMBAT
    LET X1 = 200
    LET X2 = 100
    LET Y1 = 175
    LET Y2 = 100
```

Assume the battle continues until all direct-fire units are destroyed on one side or the other.

b) Use your model to simulate the scenario on page 2.

c) The program should print out values for each time step as well as a final summary. Print a killer-victim score board as part of the summary.

4. VARIATION: The Y-Force commander, being an NPS grad, gets a bright idea. Since he is being pounded by the X artillery, he chooses to use a fraction (f) of his artillery to fire counterbattery missions against the X2 force instead of support missions against the X1's.

Thus the equation systems change:

$$\frac{dX1(t)}{dt} = -a\ Y1(t) - \text{alpha}\ (1-f)\ Y2(t)\ X1(t)$$

$$\frac{dX2(t)}{dt} = -\text{gamma}\ (f)\ Y2(t)\ X2(t)$$

$$\frac{dY1(t)}{dy} = \text{(as before)}$$

Assume gamma, the counterbattery attrition rate coefficient is given by gamma = 0.0003.

Find the best value of f for the Y-Force commander. Sub-variation, what if the X-Force can also fire counterbattery? How much should each commander use?

Note that, at the moment, no one is attacking the artillery systems. Assume initial strengths X1 = 200, X2 = 100, Y1 = 175, and Y2 = 100. (Force strengths are for illustration purposes only and are not generally considered an adequate force ratio for an attack.) Assume attrition rates (constant throughout the battle) are: a = 0.013, b = 0.004, (Y enjoys the advantage of a prepared position) but alpha = 0.0001, beta = 0.0002 (X gets to shoot at stationary targets).

3. ASSIGNMENT:

a) Write a simulation program to step through time in fixed time steps of 1 minute. For each time step compute casualties to each of the four force components using the Lanchester-type equations.

$$\frac{dX1(t)}{dt} = -a\ Y1(t) - alpha\ Y2(t)\ X1(t)$$

$$\frac{dY1(t)}{dt} = -b\ X1(t) - beta\ X2(t)\ Y1(t)$$

Assume all attrition rates are "per minute" so that a finite difference integration can be used:

$$X1(t + 1) = X1(t) - a\ Y1(t) - alpha\ Y2(t)\ X1(t)$$

and similarly for

$$Y1(t + 1) = Y1(t) - b\ X1(t) - beta\ X2(t)\ Y1(t)$$

and $$\frac{dy(t)}{dt} = -\text{beta}\ y(t)\ x(t)$$

Where casualty rate "beta" = Y casualties / (X firer * Y target * time)

[With x(t) and y(t) = number of surviving combat systems as a function of time]

Some special cases of these equations have analytic closed form solutions, but to obtain real-world modelling fidelity we invariably let the attrition rate coefficients (a, b, alpha, beta) be functions of the combat situation (e.g. range). Then analytic solutions become impossibly difficult, so instead we use numerical solution procedures.

2. SCENARIO: Y, (the good guys) in a prepared defensive position with X attacking. Each of X and Y consists of two components: a) direct fire systems in front line combat and b) supporting artillery systems.

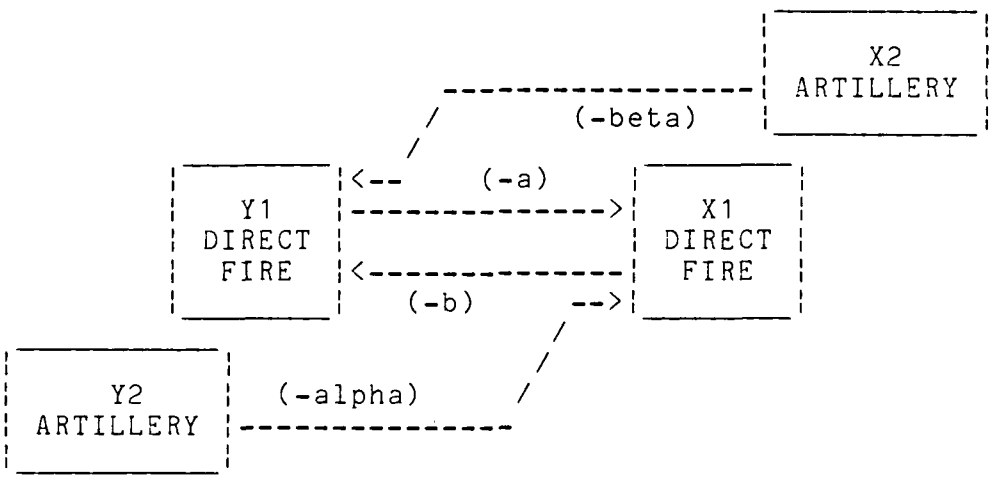


Figure 2 Scenario For Simple Lanchester Model

APPENDIX A

DETERMINISTIC, FIXED TIME STEP SIMULATION OF COMBAT

1. BACKGROUND: F. W. Lanchester, in 1914, presented two differential equation models of combat attrition to describe the difference between ancient (one-on-one) combat and modern warfare where many firers can concentrate fire on a single target. These equations have been developed and elaborated, and are currently used in many large-scale computer simulations of combat to model the combat attrition processes.

a) The "Aimed Fire" square law equations:

$$\frac{dx(t)}{dt} = -ay(t)$$

Where casualty rate "a" = X casualties / (Y firer * time)

and

$$\frac{dy(t)}{dt} = -bx(t)$$

Where casualty rate "b" = Y casualties / (X firer * time)

b) The "Area Fire" linear law equations:

$$\frac{dx(t)}{dt} = -\text{alpha}\ x(t)\ y(t)$$

Where casualty rate "alpha" = X casualties / (Y firer * X target * time)

VI. CONCLUSIONS

PC SIMSCRIPT is a complete implementation of the SIMSCRIPT language for personal computers. By taking advantage of the simple operating system and low cost/high performance ratios for today's PCs, CACI Inc., has potentially introduced a modeling capability to many new users.

Most significant models will continue to operate in a mainframe environment, but the PC will probably impact even on the most serious applications. PCs can be numerous and readily available for development work in designing and transporting individual modules into larger programs. Additionally, PCs introduce the capability for commercial run time applications which will accept new input and execute a previously compiled program to solve the recurring needs of a customer.

Analysis of Table 3 focuses on Compile time. Generally the XT ran over ten times slower than the VAX computer. The AT operated over twice as fast as the XT and processed the sample programs in two to five times the length required for the VAX. Link times were generally insignificant. Execution results include video display rates as a factor affecting total time but generally results paralleled compile time comparisons.

It is tempting at this point to draw conclusions based on dollar/performance ratios comparing a $5000 XT, an $8000 AT, and a $300,000 VAX. Suffice it to say that people in the VAX market will not be satisfied with an AT while people in the AT market cannot be talked up to a VAX. However, the author of this thesis certainly wishes that an AT and not an XT had been available during the review and model development stages of this thesis research. It is frustrating to wait during a long compile cycle. Computers have grown so powerful, so quickly, we forget that not many years ago we submitted programs on cards and received an output listing the next day. Also worth noting, the luxury of being a single user on a powerful VAX system is not common. Processing times for these problems easily doubled when other VAX system users were present.

GOOD GUYS DIRECT FIRE	GOOD GUYS INDIRECT FIRE	BAD GUYS DIRECT FIRE	BAD GUYS INDIRECT FIRE	TIME (MINUTES)
.97	100.00	13.74	100.00	160.
.89	100.00	13.59	100.00	161.
.82	100.00	13.45	100.00	162.
.75	100.00	13.30	100.00	163.
.68	100.00	13.16	100.00	164.
.62	100.00	13.02	100.00	165.
.55	100.00	12.88	100.00	166.
.49	100.00	12.74	100.00	167.
.43	100.00	12.61	100.00	168.
.37	100.00	12.48	100.00	169.
.31	100.00	12.35	100.00	170.
.26	100.00	12.22	100.00	171.
.20	100.00	12.10	100.00	172.
.15	100.00	11.97	100.00	173.
.10	100.00	11.85	100.00	174.
.05	100.00	11.73	100.00	175.
+E-003	100.00	11.61	100.00	176.

MINUTE 177	VICTIM Y1	Y2	X1	X2
K : GOOD GUY DIRECT FIRE (Y1)			37.52	
I : GOOD GUY INDIRECT FIRE (Y2)			100.98	
L :				
L : BAD GUY DIRECT FIRE (X1)	40.39			
E : BAD GUY INDIRECT FIRE (X2)	134.65			
R :				
CURRENT FORCE COMPOSITION	-.04	100.00	11.50	100.00

APPENDIX B

STOCHASTIC SIMULATION OF CRUISE MISSILE DEFENSE

OBJECTIVES:

1. More time step simulation.
2. Random sampling on the computer.
3. Use of library subroutines.
4. Simulating movement.

SCENARIO:

1. Two alternative designs for an air search radar are to be evaluated using computer simulation as a tool. The radar's primary job is early detection of small, fast cruise missiles (CM) attacking a carrier task force.

2. Suppose the radar is located at map coordinates X = Y = 0, and that each CM flies past the radar along a straight line path with constant X coordinate. The CM's move from north to south so their Y coordinate will decrease from some initial value to a critical engagement boundary of Y = 10. If a CM reaches Y = 10 then it has penetrated the defense and the radar has failed. The X coordinate for each missile is different and is determined by a random draw from a probability distribution F(x). (Details of the distribution later.)

3. Each radar has a maximum effective range of RMAX against this type of cruise missile target, so the engagement geometry is as shown in figure B.1.

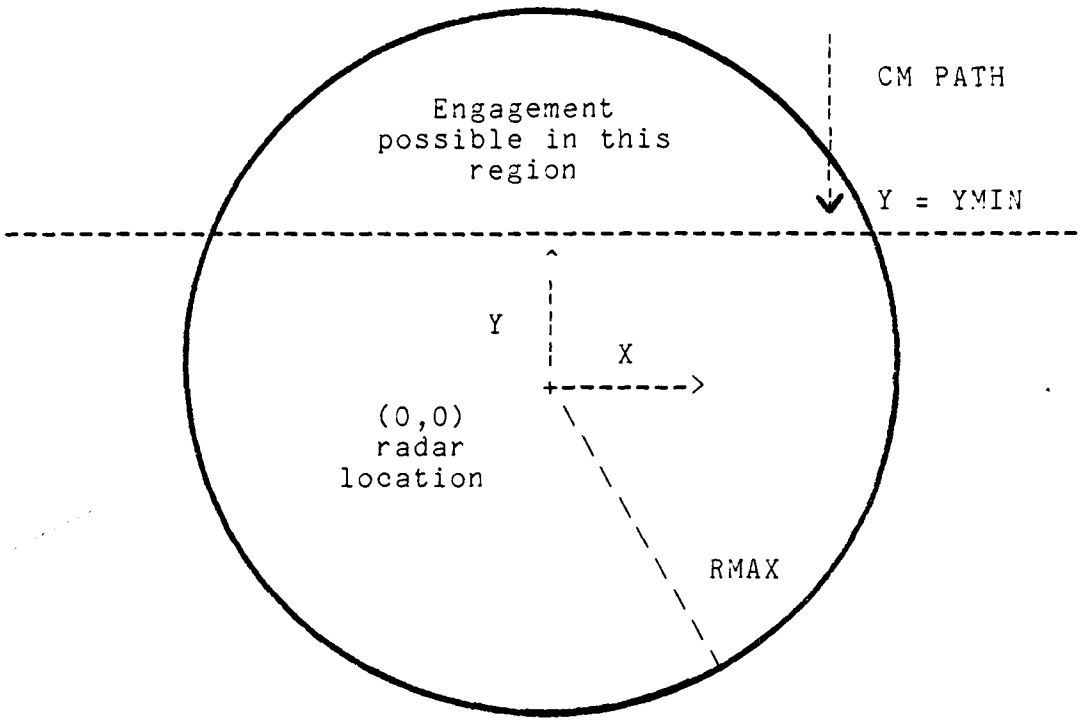


Figure 3 Scenario For Cruise Missile Defense

4. The goal of the radar system is to detect and establish a track on a CM before it penetrates to Y = 10. When a track is established, the radar hands off the CM target to other systems for engagement. Thus there are two measures of performance which the simulation should produce:

a) The fraction of CM encounters which result in detection and tracking (thus successful handoffs).

b) For the successful hand-offs, the average Y coordinate when the handoff occurs.

We will omit from this study any consideration of the effectiveness of the engagement system after the hand-off.

5. For simplicity in this initial study, we will consider the radar's capabilities against single CM targets

only. Leave for a later study the much more complex question of establishing tracks on multiple simultaneously arriving CMs.

RADAR DESCRIPTION:

1. Each radar has a rotating antenna with rotation frequency GRATE. The radar gets one "glimpse" at the target for each rotation of the antenna.

2. The result of a single glimpse is either a "detection" or not (simple Bernoulli trial), single glimpse detection probability PDET. Elaborate models exist for computing PDET as a function of many variables, but for simplicity we will assume that PDET is only a function of range R between the target and the radar. Suppose that we can can approximate PDET by the function:

PDET(R) = 100.0 * PRMIN / (R * R)

Where PRMIN is the detection probability at range R = 10 (Km).

3. The above equation gives PDET for 10 ¼ R ¼ RMAX. For R ½ RMAX assume that PDET = 0.0, and for R ¼ 10 the value of PDET is unimportant since the CM will already have penetrated the defenses.

4. The result of any glimpse is independent (exponential and memoryless) of the result for any other glimpse. In particular, a detect or a string of successive detections does not make detection on the next glimpse any more likely.

5. To establish a track on a CM target, the radar must detect the target for NREQ successive glimpses in a row. Failure to detect on any glimpse means the detect and track process must start all over.

6. When a track is established, then hand-off is assumed to occur immediately and the simulation for this CM is completed.

CRUISE MISSILE PARAMETERS:

1. Cruise missiles attack from North to South along a trajectory with constant X coordinate. The value of X for any given CM is a random variable drawn from a Normal distribution with mean SMU and standard deviation XSIG.

2. The velocity of any given CM is also random. There are three possible velocity values, VEL1, VEL2, and VEL3. They occur with probabilities P1, P2, and P3 respectively. The velocity for a CM is independent of its X coordinate.

SUMMARY OF SCENARIO PARAMETERS:

1. Scenario Parameters are constant for all CM's and radars. Scenario Parameter:

YMIN (Km) -- If a CM gets to YMIN = 10, it has penetrated the defense.

2. Radar Parameters are input separately for each competing system. Radar Parameters:

GRATE (rpm) -- Antenna rotation rate

RMAX (Km) -- max range

PRMIN -- prob detect at R = 10

NREQ -- number successive detects required to track

3. Cruise Missile Parameters are randomly sampled for each CM. CM Parameters:

X (Km) -- X coordinate

VEL (Kph) -- velocity

ASSIGNMENT:

1. Write a time step simulation for evaluating the performance of a radar system against a single cruise missile. For a single CM, the simulation should step through time in time steps corresponding to one rotation of the radar antenna (one "glimpse"). In each time step the simulation will use a random number to determine whether or not a detect occurs. Time steps will continue until either the target is successfully tracked or the target penetrates the defense.

2. Your program should be written in layers:
Control layer -- Do N replications of the simulation, corresponding to N cruise missiles. Tally the results and compute mean and variance of the desired output statistics.
Simulate Layer -- One replication of the simulation creates a new CM and traces its progress one time step at a time from its start to the time when it is engaged or penetrates.
Timestep Layer -- One time step corresponds to the antenna

rotation period for the radar. In the time step the program updates the CM location, computes range and PDET, samples to see if detection occurred, and (if detected) decides if a track is established so that an engagement may occur.

3. Make your program coherent through use of subroutines.

4. The following values ought to be input data:

N = number of replications
YMIN = penetration threshold
XMU,XSIG = parameters for CM X coord distribution
VEL1,VEL2,VEL3,P1,P2,P3 = CM velocity distribution
GRATE,RMAX,PRMIN,NREQ = radar parameters

TEST CONDITIONS:

N = whatever you need to make meaningful conclusions
YMIN = 10 Km

XMU = 6 Km, XSIG = 4 Km
VEL1 = 100 Kph, P1 = 0.5
VEL2 = 150 Kph, P2 = 0.3
VEL3 = 200 Kph, P3 = 0.2

	RADAR A	RADAR B
GRATE:	4 rpm	8 rpm
RMAX:	17 Km	25 Km
PRMIN:	0.9	0.8
NREQ:	2	3

```
PREAMBLE

''*** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ***
''*** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ***
''***  Model to evaluate alternative radar designs for detection of          ***
''***  cruise missiles.  Input values include:  number of different radar   ***
''***   to test, number of missiles per test, target penetration altitude,  ***
''***   glimpse rate, radar range, detection probability, number of tracks  ***
''***   for a successful fix, cruise missile velocity probabilities, and    ***
''***   the mean and variance of the missile's flight path.                 ***
''*** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ***
''*** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ***

    PROCESSES INCLUDE RADAR.TRACK
                EVERY CM HAS A CM.X.TRACK
                         AND A CM.DISTANCE.PER.GLIMPSE
                         AND A CM.RANGE
                         AND A CM.PROB.DET
                         AND A CM.HEIGHT
    DEFINE CM.X.TRACK,
           CM.RANGE,
           CM.PROB.DET,
           CM.DISTANCE.PER.GLIMPSE, AND
           CM.HEIGHT AS REAL VARIABLES

    RESOURCES
        EVERY RADAR HAS A R.GLIMPSE.RATE
                      AND A R.MAX.RANGE
                      AND A R.MIN.RANGE.DET.PROB
                      AND A R.NUM.CONSEC.ACQ
    DEFINE R.MIN.RANGE.DET.PROB AS A REAL VARIABLE
    DEFINE R.GLIMPSE.RATE,
           R.MAX.RANGE, AND
           R.NUM.CONSEC.ACQ     AS INTEGER VARIABLES

    DEFINE .MINUTES TO MEAN UNITS
```

```
    DEFINE P.VEL1, P.VEL2, P.VEL3, AND
           INTERCEPT.HEIGHT

                                AS REAL VARIABLES

    DEFINE NUM.OF.CM.TO.TEST,
           CURRENT.CM.NUMBER,
           NUM.OF.RADAR.TO.TEST,
           XMU, XSIG,
           YMIN,
           VEL1, VEL2, VEL3,
           SUCCESSES, PENETRATIONS, AND
           TRIAL
                                AS INTEGER VARIABLES

    TALLY AVG.HEIGHT AS THE MEAN AND
          DISTRIBUTION AS THE STD.DEV OF INTERCEPT.HEIGHT

END  ''PREAMBLE
MAIN

    CALL INITIALIZE

'' Save seeds so each configuration has the same random number stream
    LET SAVESEED1 = SEED.V (1)
    LET SAVESEED2 = SEED.V (2)

    FOR TRIAL = 1 TO NUM.OF.RADAR.TO.TEST
    DO
        LET SUCCESSES = 0
        LET PENETRATIONS = 0
        LET CURRENT.CM.NUMBER = 0
        RESET TOTALS OF INTERCEPT.HEIGHT
```

```
            LET SEED.V (1) = SAVESEED1
            LET SEED.V (2) = SAVESEED2

            ACTIVATE A CM NOW

            START SIMULATION


            CALL SUMMARY


        LOOP

    END  ''MAIN
    PROCESS CM

    '' LOCAL VARIABLES INCLUDE: CM.SPEED, PICK.VELOCITY

        WHILE CURRENT.CM.NUMBER  <  NUM.OF.CM.TO.TEST
        DO
            CURRENT.CM.NUMBER = CURRENT.CM.NUMBER + 1
            LET CM.X.TRACK(CM) = NORMAL.F (XMU, XSIG, 1)
            LET PICK.VELOCITY = RANDOM.F (2)
            LET CM.SPEED = VEL2
            IF PICK.VELOCITY < P.VEL1
                LET CM.SPEED = VEL1
            ALWAYS
            IF PICK.VELOCITY > (1 - P.VEL3)
                LET CM.SPEED = VEL3
            ALWAYS
            LET CM.DISTANCE.PER.GLIMPSE(CM) =
                        CM.SPEED / (60 * R.GLIMPSE.RATE(TRIAL))

            LET CM.HEIGHT(CM) = SQRT.F (ABS.F ((R.MAX.RANGE(TRIAL) ** 2) -
                                               (CM.X.TRACK(CM) ** 2)))
```

```
        ACTIVATE A RADAR.TRACK NOW

        SUSPEND
''        INTERRUPT THE CM
    LOOP

END  ''CM
ROUTINE HEADING

    PRINT 3 LINES THUS

X-TRACK    HEIGHT   FALL-RATE   RANGE    DETECT-PROB   TRACKS   TIME

END  ''HEADING
ROUTINE INITIALIZE

''  VARIABLES ASSOCIATED WITH RADARS TO TEST AND THEIR PARAMETERS

    LET NUM.OF.RADAR.TO.TEST = 2
    CREATE EVERY RADAR(NUM.OF.RADAR.TO.TEST)

    LET R.GLIMPSE.RATE(1) = 4
    LET R.MAX.RANGE(1) = 17
    LET R.MIN.RANGE.DET.PROB(1) = .9
    LET R.NUM.CONSEC.ACQ(1) = 2
    LET R.GLIMPSE.RATE(2) = 8
    LET R.MAX.RANGE(2) = 25
    LET R.MIN.RANGE.DET.PROB(2) = .8
    LET R.NUM.CONSEC.ACQ(2) = 3

''  VARIABLES ASSOCIATED WITH CRUISE MISSILE DISPERSION AND PERFORMANCE

    LET NUM.OF.CM.TO.TEST = 100
    LET YMIN = 10
    LET XMU = 6
    LET XSIG = 4
```

```
        LET P.VEL1 = .5
        LET P.VEL2 = .3
        LET P.VEL3 = .2
        LET VEL1 = 100
        LET VEL2 = 150
        LET VEL3 = 200

        LET LINES.V = 36

    END   ''INITIALIZE
    PROCESS RADAR.TRACK

    '' LOCAL VARIABLES INCLUDE: CONSECUTIVE.SUCCESSFUL.GLIMPSES

        LET CONSECUTIVE.SUCCESSFUL.GLIMPSES = 0

        UNTIL CONSECUTIVE.SUCCESSFUL.GLIMPSES = R.NUM.CONSEC.ACQ(TRIAL)
            OR    CM.HEIGHT(CM) <= YMIN

        DO
            IF CURRENT.CM.NUMBER = 1 AND  LINE.V = 1  CALL HEADING
            ALWAYS LET CM.RANGE(CM) = SQRT.F (CM.HEIGHT(CM) ** 2  +
                                              CM.X.TRACK(CM) ** 2)
            LET CM.PROB.DET(CM) = 100 * R.MIN.RANGE.DET.PROB(TRIAL) /
                                        (CM.RANGE(CM) ** 2)
            IF CM.PROB.DET(CM) < RANDOM.F (3)
                LET CONSECUTIVE.SUCCESSFUL.GLIMPSES = 0
                ELSE LET CONSECUTIVE.SUCCESSFUL.GLIMPSES =
                                    CONSECUTIVE.SUCCESSFUL.GLIMPSES + 1
            ALWAYS
            IF CURRENT.CM.NUMBER = 1
                PRINT 1 LINE WITH CM.X.TRACK(CM), CM.HEIGHT(CM),
                CM.DISTANCE.PER.GLIMPSE(CM), CM.RANGE(CM),
                CM.PROB.DET(CM), CONSECUTIVE.SUCCESSFUL.GLIMPSES,
                AND TIME.V THUS
 **.**      **.**    *.****    **.****    *.****       **    *****.***
```

```
            ALWAYS
            LET TIME.V = TIME.V + (1 / R.GLIMPSE.RATE(TRIAL))
            LET CM.HEIGHT(CM) = CM.HEIGHT(CM) - CM.DISTANCE.PER.GLIMPSE(CM)
            LOOP
        IF CM.HEIGHT(CM) <=  YMIN
            PENETRATIONS = PENETRATIONS + 1
        ELSE
            SUCCESSES = SUCCESSES + 1
            LET INTERCEPT.HEIGHT = CM.HEIGHT(CM)
        ALWAYS
        REACTIVATE THE CM NOW
''      RESUME THE CM
END   ''RADAR.TRACK
ROUTINE SUMMARY

    START NEW PAGE
    PRINT 7 LINES WITH TRIAL, NUM.OF.CM.TO.TEST, SUCCESSES,
                PENETRATIONS, AVG.HEIGHT, AND DISTRIBUTION THUS
99  SUMMARY STATISTICS FOR RADAR CONFIGURATION NUMBER *

TRIALS  ****            SUCCESSES  ***         PENETRATIONS  ***

AVERAGE INTERCEPT HEIGHT WAS **.***  KILOMETERS
WITH STANDARD DEVIATION OF    *.***

END     ''SUMMARY
```

X-TRACK	HEIGHT	FALL-RATE	RANGE	DETECT-PROB	TRACKS	TIME
.12	17.00	.4167	17.0000	.3114	0.	0.
.12	16.58	.4167	16.5833	.3273	0.	.250
.12	16.17	.4167	16.1667	.3444	0.	.500
.12	15.75	.4167	15.7500	.3628	0.	.750
.12	15.33	.4167	15.3334	.3828	1.	1.000
.12	14.92	.4167	14.9167	.4045	0.	1.250
.12	14.50	.4167	14.5001	.4281	1.	1.500
.12	14.08	.4167	14.0834	.4538	0.	1.750
.12	13.67	.4167	13.6668	.4818	1.	2.000
.12	13.25	.4167	13.2501	.5126	2.	2.250

```
PROCESS P.SERVICE
    WHILE THE PRIORITY.QUEUE IS NOT EMPTY
    DO
        IF N.X.P.OPERATOR = 0
            REQUEST 1 P.OPERATOR(1)
            REMOVE THE FIRST MESSAGE FROM THE PRIORITY.QUEUE
            LET PRIORITY.WAIT = TIME.V - CREATION.TIME(MESSAGE)
            DESTROY THIS MESSAGE
            WORK UNIFORM.F (0.0, 14.0, 3) MINUTES
            RELINQUISH 1 P.OPERATOR(1)
        ELSE
            IF N.X.R.OPERATOR = 0
                REQUEST 1 R.OPERATOR(1)
                REMOVE THE FIRST MESSAGE FROM THE PRIORITY.QUEUE
                LET PRIORITY.WAIT = TIME.V - CREATION.TIME(MESSAGE)
                DESTROY THIS MESSAGE
                WORK UNIFORM.F (0.0, 14.0, 3) MINUTES
                RELINQUISH 1 R.OPERATOR(1)
            ALWAYS
        ALWAYS
    WAIT .1 MINUTE
    LOOP
END    '' P.SERVICE

PROCESS R.SERVICE
    WHILE THE PRIORITY.QUEUE IS NOT EMPTY
    DO
        WAIT .25 MINUTE
    LOOP
    REQUEST 1 R.OPERATOR(1)
    REMOVE THE FIRST MESSAGE FROM THE REGULAR.QUEUE
    LET REGULAR.WAIT = TIME.V - CREATION.TIME(MESSAGE)
    DESTROY THIS MESSAGE
    WORK UNIFORM.F (0.0, 9.8, 4) MINUTES
    RELINQUISH 1 R.OPERATOR(1)
END    ''R.SERVICE
```

```
    ACCUMULATE P.UTILIZATION AS THE AVERAGE OF N.X.P.OPERATOR
    ACCUMULATE R.UTILIZATION AS THE AVERAGE OF N.X.R.OPERATOR
END     ''PREAMBLE


MAIN
    CREATE EVERY P.OPERATOR(1)
    LET U.P.OPERATOR(1) = 1
    CREATE EVERY R.OPERATOR(1)
    LET U.R.OPERATOR(1) = 1

    ACTIVATE A GENERATOR NOW
    START SIMULATION
    CALL SUMMARY
END    ''MAIN


PROCESS GENERATOR
    WHILE TIME.V < .25    ''DAYS
    DO
        WAIT EXPONENTIAL.F(3.0,1) MINUTES
        CREATE A MESSAGE
        LET CREATION.TIME(MESSAGE) = TIME.V
        LET DISTRIBUTION = UNIFORM.F(0.0,1.0,2)
        IF DISTRIBUTION < .4
            FILE THIS MESSAGE IN THE PRIORITY.QUEUE
            ACTIVATE A P.SERVICE NOW
        ELSE
            FILE THIS MESSAGE IN THE REGULAR.QUEUE
            ACTIVATE A R.SERVICE NOW
        ALWAYS
 PRINT 1 LINE WITH TIME.V * 60 * 24,N.PRIORITY.QUEUE,AND N.REGULAR.QUEUE THUS
 ***.****     **       **

    LOOP
END     ''GENERATOR
```

```
PREAMBLE
''*** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ***
''*** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ***
''*** SIMSCRIPT model to evaluate queing of messages and utilization of    ***
''*** operators in a structured message center with a dedicated operator   ***
''*** for priority messages and a second operator who works on regular     ***
''*** messages only if no priority messages are queued.  Messages arrive   ***
''*** exponentially with mean 3 minutes, they are split 60/40 regular/     ***
''*** priority, and service times are uniformly distributed u(0, 9.8)      ***
''*** for regular and u(0, 14) for priority.                               ***
''*** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ***
''*** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ***
    PROCESSES INCLUDE GENERATOR, P.SERVICE, AND R.SERVICE
    TEMPORARY ENTITIES
                EVERY MESSAGE HAS A CREATION.TIME
                              AND MAY BELONG TO THE PRIORITY.QUEUE
                              AND MAY BELONG TO THE REGULAR.QUEUE
    DEFINE CREATION.TIME AS A REAL VARIABLE
    DEFINE PRIORITY.QUEUE AND REGULAR.QUEUE AS FIFO SETS
    THE SYSTEM OWNS THE PRIORITY.QUEUE
    THE SYSTEM OWNS THE REGULAR.QUEUE

    RESOURCES INCLUDE P.OPERATOR AND R.OPERATOR

    DEFINE REGULAR.WAIT AND PRIORITY.WAIT AS REAL VARIABLES

    ACCUMULATE MAX.PRIORITY.QUE AS THE MAXIMUM
           AND AVG.PRIORITY.QUE AS THE MEAN
           OF N.PRIORITY.QUEUE
    ACCUMULATE AVG.PRIORITY.WAIT AS THE MEAN OF PRIORITY.WAIT

    ACCUMULATE MAX.REGULAR.QUE AS THE MAXIMUM
           AND AVG.REGULAR.QUE AS THE MEAN
           OF N.REGULAR.QUEUE
    ACCUMULATE AVG.REGULAR.WAIT AS THE MEAN OF REGULAR.WAIT
```

A 6 HOUR SHIFT IN A SIMPLE MESSAGE CENTER WITH 2 OPERATORS

	PRIORITY MESSAGES	REGULAR MESSAGES
AVERAGE TIME IN QUEUE (minutes)	5.17	9.42
AVERAGE QUEUE LENGTH	.89	2.55
MAXIMUM QUEUE LENGTH	4.00	8.00

THE OPERATORS WERE IDLE 6.70 PER CENT OF THE TIME
COMPLETION TIME OF LAST MESSAGE: 6.26 HOURS

```
PROCESS R.SERVICE
    WHILE THE PRIORITY.QUEUE IS NOT EMPTY
    DO
        WAIT .025 MINUTE
    LOOP

    REQUEST 1 OPERATOR(1)
    REMOVE THE FIRST MESSAGE FROM THE REGULAR.QUEUE
    LET REGULAR.WAIT = TIME.V - CREATION.TIME(MESSAGE)
    DESTROY THIS MESSAGE
    WORK UNIFORM.F (0.0, 9.8, 4) MINUTES
    RELINQUISH 1 OPERATOR(1)
END     ''R.SERVICE
ROUTINE SUMMARY
    PRINT 12 LINES WITH AVG.PRIORITY.WAIT*60*24, AVG.REGULAR.WAIT*60*24,
                        AVG.PRIORITY.QUE, AVG.REGULAR.QUE,
                        MAX.PRIORITY.QUE, MAX.REGULAR.QUE,
        (1 -  UTILIZATION(1) / 2) * 100, AND TIME.V*24
    THUS
      A 6 HOUR SHIFT IN A SIMPLE MESSAGE CENTER WITH 2 OPERATORS

                                PRIORITY MESSAGES     REGULAR MESSAGES
                                -------- --------     ------- --------
   AVERAGE TIME IN QUEUE            ****.**              ****.**
       (minutes)
   AVERAGE QUEUE LENGTH              ***.**               ***.**
   MAXIMUM QUEUE LENGTH              ***.**               ***.**


   THE OPERATORS WERE IDLE **.** PER CENT OF THE TIME
   COMPLETION TIME OF LAST MESSAGE: ***.**  HOURS
END  ''SUMMARY
```

```
MAIN
    CREATE EVERY OPERATOR(1)
    LET U.OPERATOR(1) = 2
    ACTIVATE A GENERATOR NOW
    START SIMULATION
    CALL SUMMARY
END     ''MAIN

PROCESS GENERATOR
    WHILE TIME.V < .25     ''DAYS
    DO
        WAIT EXPONENTIAL.F(3.0,1) MINUTES
        CREATE A MESSAGE
        LET CREATION.TIME(MESSAGE) = TIME.V
        LET DISTRIBUTION = UNIFORM.F(0.0,1.0,2)
        IF DISTRIBUTION < .4
            FILE THIS MESSAGE IN THE PRIORITY.QUEUE
            ACTIVATE A P.SERVICE NOW
        ELSE
            FILE THIS MESSAGE IN THE REGULAR.QUEUE
            ACTIVATE A R.SERVICE NOW
        ALWAYS
 PRINT 1 LINE WITH TIME.V * 60 * 24,N.PRIORITY.QUEUE,AND N.REGULAR.QUEUE THUS
 ***.****     **        **

    LOOP
END     ''GENERATOR

PROCESS P.SERVICE
        REQUEST 1 OPERATOR(1)
        REMOVE THE FIRST MESSAGE FROM THE PRIORITY.QUEUE
        LET PRIORITY.WAIT = TIME.V - CREATION.TIME(MESSAGE)
        DESTROY THIS MESSAGE
        WORK UNIFORM.F (0.0, 14.0, 3) MINUTES
        RELINQUISH 1 OPERATOR(1)
END    '' P.SERVICE
```

```
''*** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ***
''*** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ***
''*** SIMSCRIPT model to evaluate queing and utilization of operators in  ***
''*** a simple message center with priority messages handled before regu- ***
''*** lar ones.  Scenario provides for two operators, messages arrive     ***
''*** exponentially with mean 3 minutes, they are split 60/40 regular/    ***
''*** priority, and service times are uniformly distributed u(0, 9.8)     ***
''*** for regular and u(0, 14) for priority.                              ***
''*** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ***
''*** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ***
PREAMBLE
    PROCESSES INCLUDE GENERATOR, P.SERVICE, AND R.SERVICE
    TEMPORARY ENTITIES
                EVERY MESSAGE HAS A CREATION.TIME
                              AND MAY BELONG TO THE PRIORITY.QUEUE
                              AND MAY BELONG TO THE REGULAR.QUEUE
    DEFINE CREATION.TIME AS A REAL VARIABLE
    DEFINE PRIORITY.QUEUE AND REGULAR.QUEUE AS FIFO SETS
    THE SYSTEM OWNS THE PRIORITY.QUEUE
    THE SYSTEM OWNS THE REGULAR.QUEUE

    RESOURCES INCLUDE OPERATOR
    DEFINE REGULAR.WAIT AND PRIORITY.WAIT AS REAL VARIABLES

    ACCUMULATE MAX.PRIORITY.QUE AS THE MAXIMUM
           AND AVG.PRIORITY.QUE AS THE MEAN
           OF N.PRIORITY.QUEUE
    ACCUMULATE AVG.PRIORITY.WAIT AS THE MEAN OF PRIORITY.WAIT

    ACCUMULATE MAX.REGULAR.QUE AS THE MAXIMUM
           AND AVG.REGULAR.QUE AS THE MEAN
           OF N.REGULAR.QUEUE
    ACCUMULATE AVG.REGULAR.WAIT AS THE MEAN OF REGULAR.WAIT

    ACCUMULATE UTILIZATION AS THE AVERAGE OF N.X.OPERATOR
END     ''PREAMBLE
```

PROJECT B: Now suppose that the wait time for priority messages is considered unacceptably high. As a result, operator 2 is reserved for priority messages only. If there are no priority messages then operator 2 will be idle even if regular messages are in the queue. Operator 1 continues to work on both types of message, still processing priority messages before regular messages. Repeat the simulation and compare the output results.

APPENDIX C

STOCHASTIC EVENT SEQUENCED SIMULATION

SCENARIO: Messages arrive at a message center with exponentially distributed interarrival times. The mean interarrival time is 3 minutes. Message priorities tend to be randomly mixed with 60% regular and 40% priority.

There are two operators who must process all incoming messages. A message which arrives when both operators are busy is held in a queue until an operator is ready to work on it.

When an operator becomes free, she selects a message from a queue (if any) and processes it. If any messages are in the priority queue, then the first priority message is selected. Otherwise, the first regular message is selected.

Message processing times are uniformly distributed: regular are u(0, 9.8) and priority are u(0, 14) minutes in length.

PROJECT A: Simulate this message system using a simulation program which schedules events. Outputs desired include, but are not limited to: 1) average wait in queue for each class of message, 2) average and max queue lengths, and 3) operator idle time %.

SUMMARY STATISTICS FOR RADAR CONFIGURATION NUMBER 2

TRIALS 100 SUCCESSES 63 PENETRATIONS 37

AVERAGE INTERCEPT HEIGHT WAS 13.917 KILOMETERS
WITH STANDARD DEVIATION OF 3.009

X-TRACK	HEIGHT	FALL-RATE	RANGE	DETECT-PROB	TRACKS	TIME
.12	11.25	.2083	11.2503	.6321	0.	172.500
.12	11.04	.2083	11.0420	.6561	1.	172.625
.12	10.83	.2083	10.8337	.6816	2.	172.750
.12	10.62	.2083	10.6254	.7086	3.	172.875

X-TRACK	HEIGHT	FALL-RATE	RANGE	DETECT-PROB	TRACKS	TIME
.12	18.12	.2083	18.1251	.2435	1.	168.375
.12	17.92	.2083	17.9168	.2492	2.	168.500
.12	17.71	.2083	17.7084	.2551	0.	168.625
.12	17.50	.2083	17.5001	.2612	0.	168.750
.12	17.29	.2083	17.2918	.2676	1.	168.875
.12	17.08	.2083	17.0834	.2741	0.	169.000
.12	16.87	.2083	16.8751	.2809	1.	169.125
.12	16.67	.2083	16.6668	.2880	0.	169.250
.12	16.46	.2083	16.4585	.2953	1.	169.375
.12	16.25	.2083	16.2501	.3030	2.	169.500
.12	16.04	.2083	16.0418	.3109	0.	169.625
.12	15.83	.2083	15.8335	.3191	0.	169.750
.12	15.62	.2083	15.6252	.3277	1.	169.875
.12	15.42	.2083	15.4168	.3366	0.	170.000
.12	15.21	.2083	15.2085	.3459	1.	170.125
.12	15.00	.2083	15.0002	.3555	0.	170.250
.12	14.79	.2083	14.7918	.3656	1.	170.375
.12	14.58	.2083	14.5835	.3762	2.	170.500
.12	14.37	.2083	14.3752	.3871	0.	170.625
.12	14.17	.2083	14.1669	.3986	0.	170.750
.12	13.96	.2083	13.9585	.4106	0.	170.875
.12	13.75	.2083	13.7502	.4231	0.	171.000
.12	13.54	.2083	13.5419	.4362	1.	171.125
.12	13.33	.2083	13.3336	.4500	0.	171.250
.12	13.12	.2083	13.1252	.4644	0.	171.375
.12	12.92	.2083	12.9169	.4795	0.	171.500
.12	12.71	.2083	12.7086	.4953	0.	171.625
.12	12.50	.2083	12.5003	.5120	0.	171.750
.12	12.29	.2083	12.2920	.5295	1.	171.875
.12	12.08	.2083	12.0836	.5479	0.	172.000
.12	11.87	.2083	11.8753	.5673	0.	172.125
.12	11.67	.2083	11.6670	.5877	1.	172.250
.12	11.46	.2083	11.4587	.6093	0.	172.375

X-TRACK	HEIGHT	FALL-RATE	RANGE	DETECT-PROB	TRACKS	TIME
.12	25.00	.2083	25.0000	.1280	0.	164.250
.12	24.79	.2083	24.7917	.1302	1.	164.375
.12	24.58	.2083	24.5833	.1324	0.	164.500
.12	24.37	.2083	24.3750	.1346	0.	164.625
.12	24.17	.2083	24.1667	.1370	0.	164.750
.12	23.96	.2083	23.9583	.1394	0.	164.875
.12	23.75	.2083	23.7500	.1418	0.	165.000
.12	23.54	.2083	23.5417	.1443	0.	165.125
.12	23.33	.2083	23.3333	.1469	0.	165.250
.12	23.12	.2083	23.1250	.1496	0.	165.375
.12	22.92	.2083	22.9167	.1523	1.	165.500
.12	22.71	.2083	22.7084	.1551	0.	165.625
.12	22.50	.2083	22.5000	.1580	0.	165.750
.12	22.29	.2083	22.2917	.1610	0.	165.875
.12	22.08	.2083	22.0834	.1640	0.	166.000
.12	21.87	.2083	21.8750	.1672	0.	166.125
.12	21.67	.2083	21.6667	.1704	0.	166.250
.12	21.46	.2083	21.4584	.1737	0.	166.375
.12	21.25	.2083	21.2500	.1772	0.	166.500
.12	21.04	.2083	21.0417	.1807	0.	166.625
.12	20.33	.2083	20.8334	.1843	0.	166.750
.12	20.62	.2083	20.6250	.1881	1.	166.875
.12	20.42	.2083	20.4167	.1919	0.	167.000
.12	20.21	.2083	20.2084	.1959	0.	167.125
.12	20.00	.2083	20.0001	.2000	1.	167.250
.12	19.79	.2083	19.7917	.2042	0.	167.375
.12	19.58	.2083	19.5834	.2086	0.	167.500
.12	19.37	.2083	19.3751	.2131	0.	167.625
.12	19.17	.2083	19.1667	.2178	0.	167.750
.12	18.96	.2083	18.9584	.2226	0.	167.375
.12	18.75	.2083	18.7501	.2276	1.	168.000
.12	18.54	.2083	18.5417	.2327	0.	168.125
.12	18.33	.2083	18.3334	.2380	0.	168.250

SUMMARY STATISTICS FOR RADAR CONFIGURATION NUMBER 1

TRIALS 100 SUCCESSES 72 PENETRATIONS 28

AVERAGE INTERCEPT HEIGHT WAS 12.643 KILOMETERS
WITH STANDARD DEVIATION OF 1.533

```
ROUTINE SUMMARY
    PRINT 14 LINES WITH AVG.PRIORITY.WAIT*60*24, AVG.REGULAR.WAIT*60*24,
                       AVG.PRIORITY.QUE, AVG.REGULAR.QUE,
                       MAX.PRIORITY.QUE, MAX.REGULAR.QUE,
          100*(1 - P.UTILIZATION), 100*(1 - R.UTILIZATION), AND TIME.V*24
    THUS
 A 6 HOUR SHIFT IN A STRUCTURED MESSAGE CENTER WITH 2 SPECIALIZED OPERATORS

                                  PRIORITY MESSAGES     REGULAR MESSAGES
                                  ---------------       ---------------
   AVERAGE TIME IN QUEUE              ****.**               ****.**
      (minutes)
   AVERAGE QUEUE LENGTH                ***.**                ***.**
   MAXIMUM QUEUE LENGTH                ***.**                ***.**


   THE PRIORITY OPERATOR WAS IDLE **.** PER CENT OF THE TIME
   THE REGULAR  OPERATOR WAS IDLE **.** PER CENT OF THE TIME

FINAL MESSAGE COMPLETED AT ***.**  HOURS
END  ''SUMMARY
```

A 6 HOUR SHIFT IN A STRUCTURED MESSAGE CENTER WITH 2 SPECIALIZED OPERATORS

	PRIORITY MESSAGES	REGULAR MESSAGES
AVERAGE TIME IN QUEUE (minutes)	5.36	56.95
AVERAGE QUEUE LENGTH	.82	11.50
MAXIMUM QUEUE LENGTH	5.00	19.00

THE PRIORITY OPERATOR WAS IDLE 36.99 PER CENT OF THE TIME
THE REGULAR OPERATOR WAS IDLE 1.46 PER CENT OF THE TIME

FINAL MESSAGE COMPLETED AT 7.23 HOURS

LIST OF REFERENCES

1. Emshoff, James R. and Sisson, Roger L., Design and Use of Computer Simulation Models, Macmillan, 1970.

2. Russel, Edward C., Building Simulation Models with SIMSCRIPT II.5, CACI, 1983.

3. Law, Averill M. and Kelton, David W., Simulation Modeling and Analysis, McGraw-Hill, 1982.

4. Mullarney, Alasdar, PC SIMSCRIPT II.5 User's Manual, CACI, 1984.

5. Boehm, Barry W., Software Engineering Economics, Prentice-Hall, Inc., 1981

6. Benwell, Nicholas, Benchmarking, Hemisphere, 1975.

BIBLIOGRAPHY

Braun, Jay E., SIMSCRIPT II.5 Reference Handbook, CACI, 1983.

DeVoney, Chris, IBM's Personal Computer, Que Corporation, 1983.

Emshoff, James R. and Sisson, Roger L., Design and Use of Computer Simulation Models, Macmillan, 1970.

Kiviat, P.J., Markowitz, H., and Villaneuva, R., SIMSCRIPT II.5 Programming Language, CACI, 1983.

Hartman, J., Simulation and Wargaming - OS 3603 Class Handout, Fall 1984, U. S. Naval Postgraduate School.

Law, Averill M. and Larmey, Christopher S., An Introduction to Simulation Using SIMSCRIPT II.5, CACI, 1884.

Morris, Michael R. and Roth, Paul F., Computer Performance Evaluation, Van Nostrand Reinhold, 1982.

West, Joel W. III and Johnson, Glen D., SIMSCRIPT II.5 User's Manual VAX/VMS, CACI, 1984.

Wolverton, Van, Running MS-DOS, Microsoft Press, 1984.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943	2
3.	Prof. Samuel H. Parry, Code 55Py Naval Postgraduate School Monterey, California 93943	2
4.	Prof. Norman R. Lyons, Code 54LB Naval Postgraduate School Monterey, California 93943	2
5.	Prof Jim Hartman, Code 55Hh Naval Postgraduate School Monterey, California 93943	1
6.	LCDR Paul Fischbeck, Code 55Fi Naval Postgraduate School Monterey, California 93943	1
7.	Information Systems Directorate Army War College Carlisle Barracks, Pennsylvania 17013	2
8.	Mr. Glen D. Johnson C.A.C.I. 12011 San Vicente Voulevard Los Angeles, California 90049	1
9.	Mr. Hal Duncan C.A.C.I. 3344 North Torrey Pines Court La Jolla, California 92037	1
10.	Cpt Arthur A Deckert, Jr Rt 2 Box 607M White Plains, MD 20695	2

11. Computer Technology Programs, Code 37 1
Naval Postgraduate School
Monterey, CA 93943

END

FILMED

8-85

DTIC