(1)

A COMPUTER BASED EDUCATIONAL AID FOR THE INSTRUCTION OF COMBAT MODELING

THESIS

Richard S. Moore
Captain, USAF

AFIT/GOR/ENS/92M-21

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

1

AFIT/GOR/ENS/92M-21

A COMPUTER BASED EDUCATIONAL AID FOR THE INSTRUCTION OF COMBAT MODELING

THESIS

Richard S. Moore
Captain, USAF

AFIT/GOR/ENS/92M-21

Approved for public release; distribution unlimited

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE March 1992	3. REPORT TYPE AND DATES COVERED Master's Thesis
4. TITLE AND SUBTITLE A COMPUTER BASED EDUCATIONAL AID FOR THE INSTRUCTION OF COMBAT MODELING		5. FUNDING NUMBERS
6. AUTHOR(S) Richard S. Moore, Captain, USAF		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology, WPAFB OH 45433-6583		8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/~~GST~~/ENS/92M-21 GOR
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Department of Defense AU CADRE/WGTA Maxwell AFB, AL		10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES		
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited		12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Abstract

The Air Force Institute of Technology is one of few institutions that teaches combat modeling. Combat models are typically dynamic computer simulations of specialized dynamic processes. Great difficulty exists in portraying these dynamic and recurrent processes with conventional static diagrams.

This thesis produced a computer-based instructional aid which presents animated examples of processes common to combat models, demonstrates a simple few-on-few aerial combat model, and encourages student exploration and interaction with these presentations. This aid provides dynamic examples of random number generation, detection processes, target selection, and target destruction. A complete combat model demonstrating a few-on-few air duel is animated and includes features which permits students to monitor the internal processes and continuously changing states of the simulation. Generated model output displays typical measures of products often used in verification and validation of combat models. A follow- up questionnaire challenges the student and their understanding of combat modeling methodologies and encourages their curiosity to explore the inner workings of the model.

As with any developmental software package, improvements and enhancements can always be made. This thesis provides the foundation to support these welcomed improvements and enhancements.

14. SUBJECT TERMS Simulations; War games; Models; Training			15. NUMBER OF PAGES 222
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

THESIS APPROVAL

STUDENT: Captain Richard S. Moore CLASS: GOR 92-M

THESIS TITLE: A Computer Based Educational Aid For The Instruction of Combat Modeling

DEFENSE DATE: 27 FEB 92

COMMITTEE:	NAME/DEPARTMENT	SIGNATURE
Advisor	Major Michael W. Garrambone, USA/ENS	[illegible]
co-Advisor	Major Paul F. Auclair/ENS	Paul F. Auclair

AFIT/GOR/ENS/92M-21

A COMPUTER BASED EDUCATIONAL AID FOR THE INSTRUCTION OF COMBAT MODELING

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Masters of Science in Operations Research

Richard S. Moore, B.S.

Captain, USAF

March,1992

Acknowledgments

I am indebted to my thesis advisor, Major Michael W. Garrambone, USA, for his ever present assistance and guidance in the development of this thesis. I also wish to thank Major Paul F. Auclair for his technical assistance and thorough critique of this document. Finally, I thank my family, my wife Sherri, our sons Alex and Aaron, and daughter Alysha who continued to supply their uplifting support and encouragement in light of their own personal sacrifices.

Richard S. Moore

Table of Contents

Page

List of Figures

List of Tables

AFIT/GOR/ENS/92M-21

Abstract

The Air Force Institute of Technology is one of few institutions that teaches combat modeling. Combat models are typically dynamic computer simulations of specialized dynamic processes. Great difficulty exists in portraying these dynamic and recurrent processes with conventional static diagrams.

This thesis produced a computer-based instructional aid which presents animated examples of processes common to combat models, demonstrates a simple few-on-few aerial combat model, and encourages student exploration and interaction with these presentations. This aid provides dynamic examples of random number generation, detection processes, target selection, and target destruction. A complete combat model demonstrating a few-on-few air duel is animated and includes features which permits students to monitor the internal processes and continuously changing states of the simulation. Generated model output displays typical measures of products often used in verification and validation of combat models. A follow- up questionnaire challenges the student and their understanding of combat modeling methodologies and encourages their curiosity to explore the inner workings of the model.

As with any developmental software package, improvements and enhancements can always be made. This thesis provides the foundation to support these welcomed improvements and enhancements.

A COMPUTER BASED EDUCATIONAL AID FOR THE INSTRUCTION OF COMBAT MODELING

I. Introduction

1.1 Background

"Simulation is one of the most widely used techniques in operations research and management science, and by all indication its popularity of use is on the increase" (24:2). In fact, modeling and simulation are the principal tools used by the Department of Defense (DoD) as aids in studying their most complex problems. Specifically, combat models and simulations are applied to problems of battle planning, wartime operations, weapons procurement, force sizing, human resource planning, logistics planning, and national policy analysis (22:4).

As one would expect, a diverse wealth of literature exists on the subject of combat modeling. That is, there are books describing how to build models, memorandums on how to use them, and critiques discussing their unfortunate and unintentional misuse. The General Accounting Office (44) emphasized in their report, *Models, Data, And War: A Critique Of The Foundation For Defense Analysis*, the need for the decisionmaker to insist that models and simulations used to support decision making be easily understood (transparent), validated and verified (appraised), and consistent with the stated problem (44:49). To meet this demand, the model developers and analyst must have superior knowledge of the development and application of DoD models. Unfortunately, gaining insight into a models construction and purpose is no easy task.

One reason for the difficulty may be that models evolve over time. That is, as the "what if" questions arise for an analyst to assess,"new and improved" models are developed to answer such questions. This action then adds to the model's size and complexity. Today, a small combat model may have thousands of lines of computer code while larger models might have hundreds of thousands of lines. It typically takes an analysts weeks to learn how to operate a small model, and several years to learn the "guts"—the internal workings—of the more modern ones (42). In too many cases, this learning has been a time prohibitive endeavor.

Unfortunately, in these cases, the simulation model used in application is treated like the mysterious "black box". The model input is fed into the box and the resultant output is unquestionably taken, without knowledge of how the information was manipulated within the box (see Figure 1). To ensure proper model application, immediate measures through education must be taken to break this "black box" syndrome. It may be impractical to expect the decisionmaker and analyst to be educated on every line of code in the model, but learning the key elements such as the objective of the model, scenario, data requirements, and output analysis methodology is essential in preventing the unintentional misuse of combat models (28:5).

The Air Force Institute of Technology (AFIT) has taken positive measures in preventing this unintentional misuse of combat models. AFIT offers a two course combat modeling sequence which is designed to educate students on a combat model's key elements and the techniques used in dynamic modeling of combat. The students are presented with multiple static illustrations of these dynamic processes throughout the course sequence. Following successful completion of the courses, the students have a better understanding of modeling combat and are equipped with the tools necessary for investigating or breaking open the aforementioned "black box". In this light, AFIT has attempted to provide greater understanding of combat modeling to students and future decisionmakers.

Combat models are typically dynamic computer simulations of specialized processes which are difficult to illustrate with conventional static diagrams. To better present these processes, it was suggested that a simple demonstrative combat model be used as an educational aid (18). This

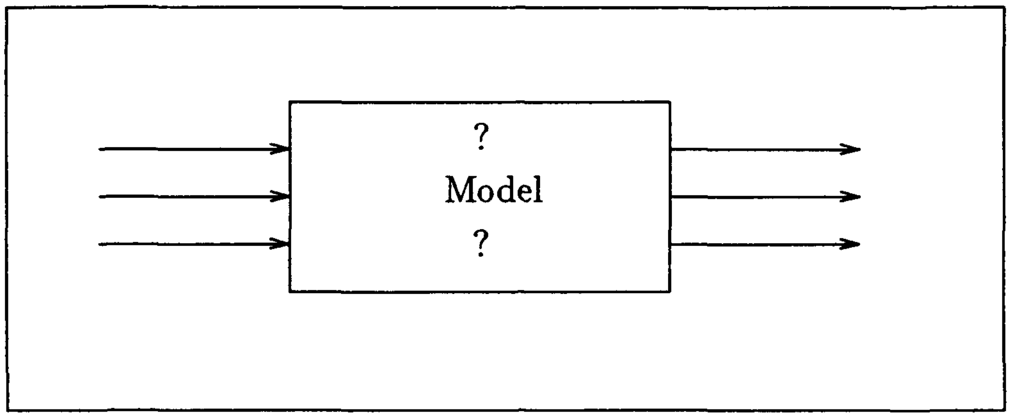


Figure 1. The "Black Box" Simulation

model should include features that permit dynamic illustrations or animation of the key processes of a combat model such as searching, target identification and engagement. By providing such a tool, the student is given the opportunity to visualize, investigate, and experiment with these processes to gain a better and longer lasting understanding of how the phenomena may be properly portrayed.

1.2 Problem Statement

Currently AFIT lacks a dynamic media to teach and train students about the key elements of combat modeling and simulation tools.

1.3 Thesis Objective

The objective of this thesis was to develop a simple and portable air–to–air, few–on–few combat simulation which demonstrates the fundamental concepts of air combat, combat modeling, and simulation in a form that supports student learning of model design and use in DoD studies.

1.4 Scope and Limitation

In light of the objective, this thesis describes the approach to development, design, and application of such an educational aid. The purpose of this aid is for the education and training of combat modeling students who will become the analysts and decision makers of tomorrow. The combat model described in this thesis is intended only for demonstration and is not a tool for analysis. Unfortunately, the time permitted for this thesis constrained the development of the demonstration's full potential as an education aid. Due to this constraint, the number of dynamic examples and novel model features were limited. In most cases, the limitations involved necessary simplifying assumptions. These assumptions are addressed in appropriate sections throughout this document.

Although, the detail of the combat model demonstration is limited in its ability to capture the full realism of an air-to-air duel, the absence of some "real world" realism serves as valuable instruction. In the employment of this aid, the student may be challenged to review the combat demonstration again and again to identify these limitations. Identifying that which is absent is indeed a more difficult task than observing what is present, challenging the students' understanding of modeling combat.

1.5 Definitions

Before a discussion of the literature on the topic of combat modeling is presented, a short glossary of terms germane to the subject is provided. Some of these terms, such as combat model and war game, are interchanged on occasion. Therefore, the following definitions are offered to clarify the meaning of such terms and to highlight the subtle distinctions between them.

> *War Game.* A simulation of a military operation involving two opposing forces, using rules, data, and procedures designed to depict an actual or assumed real-life situation. War games may be manual, with all decisions, assessments, and book-keeping functions performed manually; computer-assisted; or completely computerized. (14:227)

Analytical Model. A simulation model comprised of sets of mathematical equations as models of all the basic events and activities in the process being described. (14:151)

High Resolution Combat Model. A model which includes detailed interactions of individual combatants or weapon systems. Each combatant in a high resolution model has its own vector of state variables which describe its unique situation and its unique perception of the battlefield as the battle progresses. Interactions among combatants are resolved at the one-on-one engagement level–often computing separately the results of each individual shot fired in the battle. The engagement models include terrain and environmental effects as well as the states of firer and target. (19:Sec 1-6)

Simulation. Simulation is the use of a numerical model of a system to study its behavior as it operates over time. Discrete event simulation deals specifically with modeling of those systems in which the system state is deemed to change instantaneously at discrete points in time, rather than continuously. (43:217)

II. Literature Review

2.1 Introduction

The purpose of this section is to review pertinent literature available on the principal subject of combat modeling. The reader should keep in mind that the purpose of this thesis was not to develop a combat analysis tool, but to develop a computer-assisted instruction (CAI) aid which demonstrates techniques used to simulate some of the basic phenomena which occur in combat.

The wide range of material addressed in this thesis is divided into five sections. The first section investigates the advantages and development of computer-assisted instructions (CAI). This section provides a definition of CAI and guidelines for the development of a CAI system. The definition as well as the guidelines assisted in defining the desired objective of the CAI.

The second section addresses the role of computer generated graphics. This section provides support for the use of graphics as a teaching aid as well as, a valuable tool for model verification and validation.

The third section describes the classification of combat models by purpose, qualities, and construction. Classification of a combat model is the first and most important step in model development. Classification establishes a model foundation and provides the skeletal structure for model development.

The fourth section addresses simulation of several key combat processes such as search, detection, target selection, and target destruction. The importance of a simulation scenario is also addressed in this section.

Finally, the fifth section addresses the selection of an appropriate high order computer language.

2.2 Computer-Assisted Instruction

Edmounds described computer-assisted instruction (CAI) as

> ... an area of education that involves any training technique in which a student interacts with a computer system. The system may be based on a large mainframe-type computer, a microcomputer such as a personal computer, or anything in between. The basic approach used by CAI requires the student to sit at a computer terminal of small computer and, through a keyboard or some other student-controlled technique, respond to messages that are displayed on the monitor. Computer-assisted instruction goes under several other names, such as computer-based training (CBT) and computer-aided instruction (also abbreviated CAI). (15:104)

The definition above makes no reference to the type of instructions that the CAI pertains to nor does it characterize the receiver of the instructions. A significant feature of CAI is its flexibility. CAI may be applied to nearly every learning environment ranging from elementary schools through the college level.

Advantages offered by CAI include (15:104- 105):

- *Availability:* The instructions are as available as the computer it runs on.
- *Paced Learning:* Student can control the pace of the instruction.
- *Efficiency and Cost:* A large number of people can be trained at one time with CAI techniques. Once the software program is developed, the cost per student tends to decrease, especially when CAI does not require personal instruction.
- *Psychological:* The student's training can be done privately without peer judgement.
- *Interaction:* The computer enhances student interaction through its high speed of response (21:102).
- *Demonstration:* Computer graphics enhance dynamic demonstrations through use of animation.

A multitude of studies on the value and worth of CAI have been accomplished. In a literature review of the worth of CAI, Hathaway reported that CAI generally produced as much or more learning in less time than conventional methods of instruction. However, he warns that poorly developed and administered computer based educational environments may in fact be detrimental to learning (20:7).

To ensure the proper development of an effective CAI system, the role of the system and the intended user must be defined. The purpose of the system could be to teach one or more lessons, perform rote drills, perform simulations, act as a tutorial, or function as a game (11:1-2). In terms of the user, the system must account for ability level, information needs, and learning environment (38:2-3).

Thus, properly developed computer based instruction systems have been shown to be worthwhile and applicable to a wide range of subjects. It remains plausible then, that CAI may enhance the presentation of combat modeling concepts.

2.3 Use of Graphics and Animation

Using visualization has been an effective pedagogical technique to achieve instructional goals. In his article on animation, Robert Duisberg recognized the potential of graphics as a communication interface with the user.

> The perceptual endowments of people are strongly optimized for real-time image processing, and interactive graphics can immediately communicate multidimensional information about the internal state of a complex dynamic process.(12:276)

The introduction of animation into simulation gives the viewer a strong visual representation (36:363-370), and, as Knuth put it, "An algorithm must be seen to be believed" (23:4).

Graphics not only aid in achieving instructional goals, it can assist in model construction, debugging, verification, and validation. For the model builder, graphical presentations such as animations, charts, and plots provide the "seeing is believing". In short, animating a simulation strongly enhances the verification of the model code and its performance (40:19-20);(35:33);(37:40).

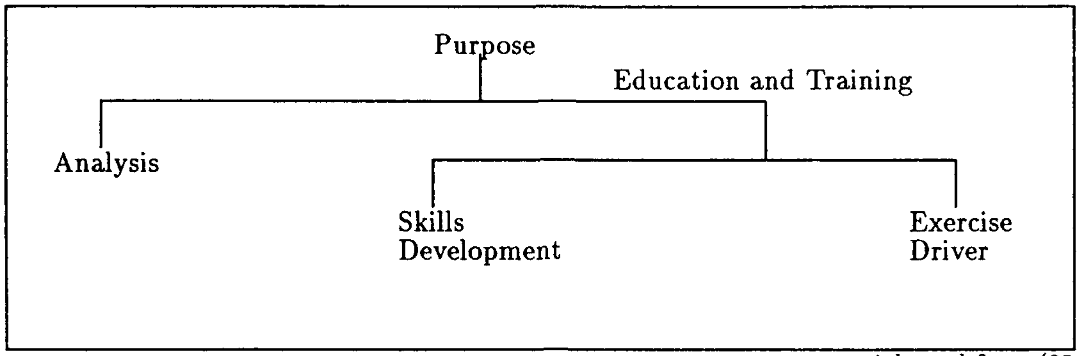
2.4 Combat Simulation Model Classification

Each combat model can be described in terms of its purpose, qualities, and construction. One such classification scheme is found in *SIMTAX: A Taxonomy for Warfare Simulation*, an appendix to the Department of Defense's Catalog of Wargaming and Military Simulation Models (10).

The classifications contained within SIMTAX are significant to this research endeavor as they provide a conceptual framework during the design and development of the instructional combat model. Generally, these classifications branch into sub-classes which are further sub-divided as warranted. The intent here is not to explore every possible sub-classification presented in SIMTAX, but to discuss only those classifications that are germane to this thesis.

2.4.1 Classification By Purpose. According to SIMTAX, a wargame or military simulation model may be used for either analysis or education and training as depicted in Figure 2. SIMTAX defines the education and training models as those whose purpose "is to transfer or reinforce a lesson or relationship that is already known"(25:3). The training and education models are further categorized as those that contribute to skills development or control various wargame exercises (25:3-5). Reflecting on the objective of this thesis, it is clear, that the purpose of this combat model is to develop skills through education and training.

2.4.2 Clcssification By Qualities. "The qualities dimension of a military model are those real entities and processes which the model represents" (25:7). For example, these entities may be aircraft or airbases. A process, on the other hand, refers to an action or activity such as distributing supplies or providing attack warning. The qualities defined in the SIMTAX are:



Adapted from (25:3)

Figure 2. Classification By Purpose

1. Domain: "The physical or abstract space in which the entities and processes operate" (25:7). The space may be air, land, sea, or a combination depending the phenomena to be modeled (25:7). In the case of the air domain, the physical space is three dimensional—vehicle latitude, longitude, and altitude—assuming that no representation of ground based units such as an airbase is made (18). However, when a ground element such as an airbase becomes a part of the simulation, multiple domains exist—air and ground.

2. Span: "The scale of the domain" (25:7). The span, like the domain, depends on the phenomenon to be modeled. In the case of global warfare, the span is obviously global. When modeling the one-on-one, close in, air-to-air engagement, the span is defined as individual. Description of the span is often subjective (25:7).

3. Environment: "The texture or detail of the domain" (25:7). The environment contains the conditions that simulated combat forces are subject to, such as, types of weather, day/night, and terrain features (25:7).

4. Force Composition: "The mix of forces that can be portrayed by the model" (25:7). A force, as defined in the model, may depend on the domain and span of the model. In a regional air

model, the force may be comprised of a squadron or flight of aircraft. On the other hand, in an individual air model, the force composition may be a single aircraft (18).

5. Scope of Conflict: "The category of weapons" (25:7). In a global warfare simulation, the scope may be nuclear ballistic missiles. At the individual level, the scope may be a single conventional air-to-air missile (18).

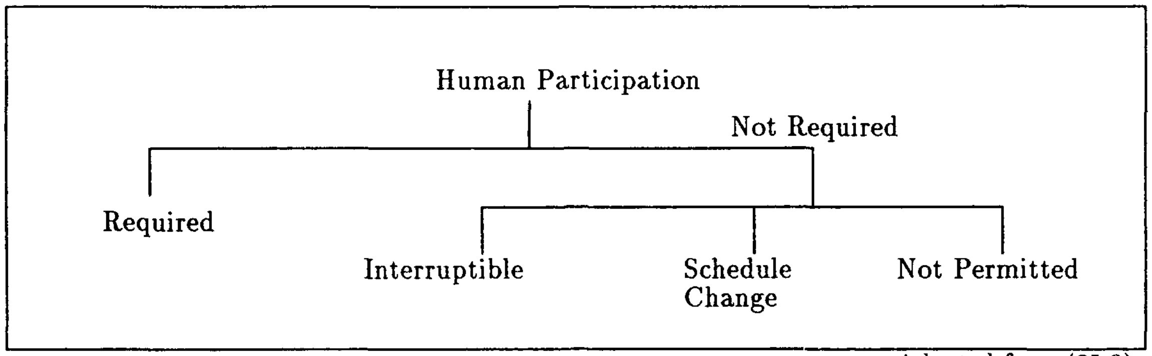
6. Mission Area: "Recognized combination of weapons and procedures used to accomplish a specific objective" (25:7). In the global span, mission area may be defined as global air lift. In the individual span, the mission area can be defined as a combat air patrol (18).

7. Level of Detail: "For the purpose intended, it needs to be like the real thing" (7:5).

2.4.3 Classification by Construction. Construction refers to combat model features such as human participation, time processing, treatment of randomness, and sidedness (25:9-11). Classification of the model's construction identifies the need for a user/model interface as well as key simulation mechanisms.

SIMTAX defines human participation as "the extent to which a human presence is allowed or required to influence the operation of the model" (25:9). Human participation, as shown in Figure 3, may be either required or not required. Even though the latter category does not require human participation, participation may be permitted as shown in Figure 3. This "not required" category is further subdivided into classes of permissible participation, such as, program interruption, scheduled changes, and participation not allowed. A program interrupt permits the user to halt the simulation and possibly alter a modeled condition.

A scheduled change permits the user to alter the current modeled conditions. One example would be to move resources at the end of each battle cycle. The last category, "not permitted," is used to classify a model which does not permit any user involvement once the simulation has begun (25:9).



Adapted from (25:3)

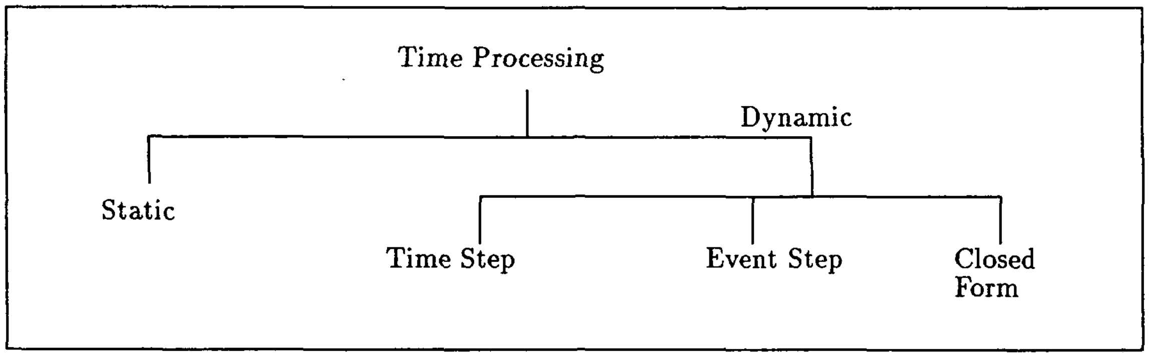
Figure 3. Classification By Human Participation

A model can also be classified by the methods used to simulate the passage of time. As shown in Figure 4, two major classes of methods exist–static and dynamic. Within the static class of models, time advance is not represented. However, dynamic models do represent the passage of time and employ three possible techniques to accomplish this. These techniques are the fixed time step, event step, and closed form techniques. The time step and event step methods step through time. With fixed time step, the step size is constant; although, there may exists several internal time step mechanisms embedded within the model. Each of these embedded mechanisms may use a different time step length.

The event step technique is more efficient, in most cases, than the fixed step. The timing mechanism of the event step "looks ahead" to see when the next event will occur. The time is then advanced to coincide with this event's occurrence.

The closed form technique uses a set of differential or difference equations. This technique applies to modeled events that have a close form solution (25:9-10).

There are advantages and disadvantages in both the fixed and event time step mechanisms. The greatest advantage of the fixed time step is its simplicity in computer program coding. Unfortunately there are three major disadvantages of using this technique (19:Sec 2,7).



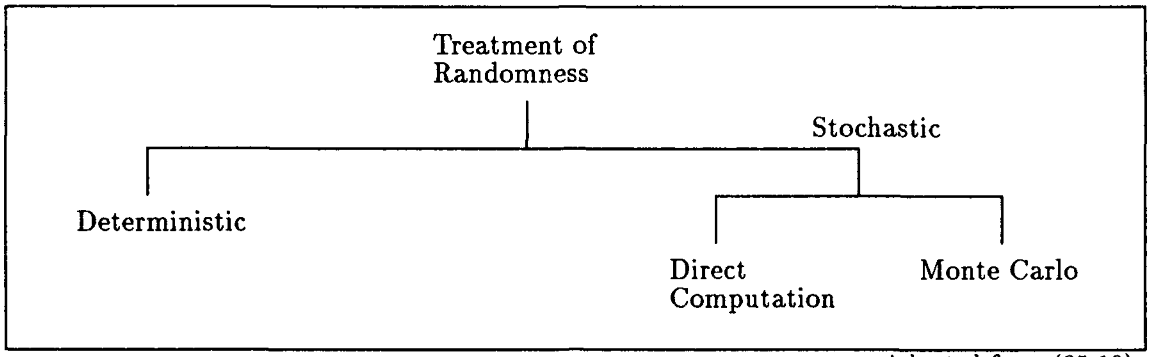
Adapted from (25:10)

Figure 4. Classification By Time Processing

1. If the model has periods of inactivity in which state variables are not changing, computer processing time is wasted by checking these variables at each time step (19:Sec 2,8).
2. If a change in a state variable does occur, the recording of the event takes place at the end of the time step interval and not at the "actual" time within the interval it would have occurred (19:Sec 2,8).
3. If at the end of the interval, mutual events occur, a determination as to which event occurred first cannot be performed. For example, it would be impossible to determine which of two pilots fired first in an engagement (19:Sec 2,8).

The last two disadvantages can be overcome by making the step size sufficiently small. However, this action only amplifies the first disadvantage listed above (19:Sec 2,8).

Hartman wrote, "Most modern high resolution models do not use the fixed time step approach" (19:Sec 2,7). This comment by Hartman was directed at combat models in general. However, this statement appears to apply primarily to ground combat models. A review of the *J8 Catalog of Wargaming and Military Simulation Models* revealed that the majority of the air-to-air high resolution models used the fixed time step technique in modeling aerial engagements (10:M30-



Adapted from (25:10)

Figure 5. Treatment of Randomness

M32). These highly dynamic state variables require continual updates of aircraft performance parameters and location. This constant need for updates presents an ideal condition for use of the fixed time step technique. In other words, since the aircraft must always be on the move, the state of the simulation is always changing.

The advantage of the event step technique is that it overcomes the disadvantages of the fixed step. The event step jumps over periods of inactivity; events are recorded at the "actual time" of occurrence, and multiple occurrence of events are distinguishable. The disadvantage of the event step is it is more difficult to develop. An additional disadvantage occurs when the frequency of the simulation demands for updates increases. At some point, the efficiency of the event step is lost, and the run time of the simulation will greatly increase due to the constant scheduling of updates.

As shown in Figure 5, a model's construction can also be classified by the method used to treat randomness. A model is categorized as deterministic if there is no consideration of randomness in the simulation. If randomness is addressed, the model is then categorized as stochastic (25:10).

The stochastic method employs the use of pseudo random number generated uniformly over an interval from zero to one. A model which uses this technique of random number generation is generally referred to as a Monte Carlo model. The following quote from SIMTAX states the condition by which a model is labeled as Monte Carlo model.

> If any part of a model draws even one random number for use in determining a realization of a random variable (i.e., uses the Monte Carlo method), then that part is Monte Carlo, and if any part of a model is Monte Carlo then the model as a whole is Monte Carlo. (25:A12)

The Monte Carlo process requires a random number to be generated. A great majority of random number generators use the linear congruential method to create a stream of random numbers to be used in the simulation. This method is described below (24:424).

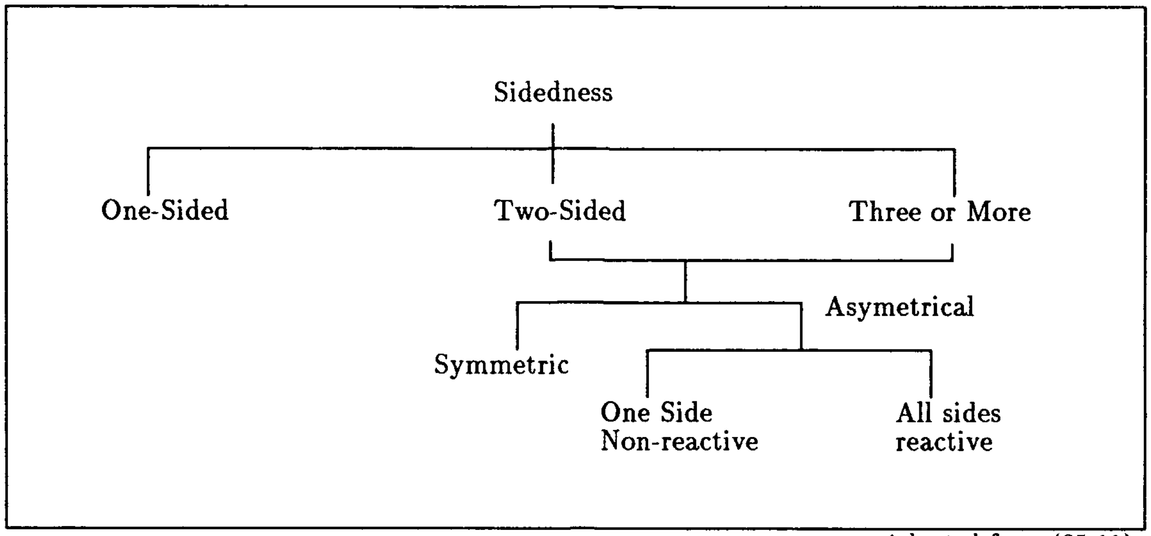
A sequence of integers $Z_1, Z_2, \cdots$ is defined by the recurrent formula

$$Z_i = (aZ_{i-1} + c)(\bmod m)$$

where m (the *modulus*), a (the *multiplier*), c (the *increment*), and Z_0 (the *seed* or *starting value*) are nonnegative integers.

The final category of classification by construction is the sidedness of the model shown in Figure 6.

Sidedness refers to "the number of collections or alliances of resources working on or through the model toward a common goal" (25:11). This category is divided into three classes. First is the one sided classification where all the assets belong to one side. The second and third classes are two-sided and three or more sided. These last two classes are divided into two subclasses–symmetric and asymmetric. For a model to be symmetric, three conditions must exist. First, a simulated resource belongs to one side only, not shared. Second, resources on either side must be equitable. For example, if one side, say side one, has the capability to defeat a threat owned by side two, then side two must also have an equitable defense against a threat possessed by side one. The third



Adapted from (25:11)

Figure 6. Classification By Sidedness

condition states that the interaction which is simulated between side one's defense and side two's threat must exists for side two's defense against side one's threat. When there is an imbalance between these conditions, the model is then classified as asymmetrical–one force is superior to the other (25:11).

Asymmetric is further divided as either one side reactive or both sides reactive. In the case of one side being non-reactive, that side will not fire back in defense nor take action, such as taking cover to prevent being detected or hit. In the reactive case, both sides react to stimulus such as shooting and defending.

2.5 *Designing Combat Models and Wargames*

In his research memorandum, *Wargame Design, Development, And Play*, Peter Perla posed fundamental questions that designers of wargames must ask and answer. These questions are also appropriate to the design of this instruction aid. These questions are (28:2):

1. What does the sponsor [investigator] want to learn from the player?
2. What does the sponsor want to say to the players?
3. Who are the players that will be involved or that need to be involved, and what are their interests or concerns?
4. How can the sponsor's and player's goals best be linked? In particular, what information must the game provided, and how can it be structured to do so?

Responding to these questions provides rudimentary guidance for the development of a modeling effort. Based on these questions, Perla also presents a step-by-step set of specific processes to guide the development of a wargame. These processes were then synthesized with the questions addressed above to establish the following useful guidelines (28:3-25);(32:11):

- Specifying the Objective
- Defining the infrastructure
- Assembling the information
- Devising the mechanics
- Model Translation
- Verification
- Validation
- Documentation

The list above does not necessarily represent a sequence, but only a list of activities. Many of these activities in fact may be accomplished simultaneously.

Specifying the objective and defining the infrastructure are addressed by the classification of the model's purpose and qualities. The assembling of information is the process of defining the wargame scenario and collecting data. From Perla, "The term 'scenario' has its origins in the theatrical world, where it refers to an outline or synopsis of the plot of a play, novel, or other work". A scenario may define an aircraft entity's role or mission, the tactics which enable the aircraft to meet its mission objectives, and the logistics to support the aircraft (28:8-9).

The mechanics, as Perla defines them, are the math models and the procedures. A math model serves a function. For example, a math model may be used to calculate the dynamic air pressure at a missile's current velocity and altitude. In contrast, a procedure is comprised of order dependent steps that may contain a math model in one of those steps. An example of a procedure may be the steps in simulating the flight of an air-to-air missile. First the missile location must be advanced. Second, the missile's range to target must be calculate. Third, if within range of the target, the missile detonatess. The fourth procedure then determines if the target was eliminated (28:21).

The process of model translation transforms the objective, infrastructure, scenario, and procedures of an abstract model into acceptable computer code (32:12). This begins by decomposing the conceptual model into smaller modules. Each of these modules may be the steps of a procedure like those described in the above paragraph. For each of these smaller modules, pseudo code is developed–a literal description of the computer program flow. From the pseudo code, flow charts are developed and finally the actual coding of the model is accomplished (16:8-9). This approach of breaking a large job into smaller more visible tasks leads to the development of structured computer code.

Validation and verification are evaluation processes. These tasks respectively consist of "... determining that the translated model executes on the computer as the modeler intended.", and "... determining that the simulation model is a reasonable representation of the system being simulated." The verification and validation process is key in establishing the worth and credibility the model (32:12).

The final process is the documentation of the model. Documentation is a written record describing the model or its use. It may take the form of comments embedded in the model's computer code or that of a manual. Whatever form it takes, documentation should occur throughout the model's development.

In addition to Perla's set of guidelines, Dunnigan provides two simple rules in developing a combat model: keep it simple and plagiarize. Plagiarizing is Dunnigan's way of saying "use available techniques", a point of view shared also by Perla (13:236-237);(30:187).

2.6 *Combat Model Processes*

The intent of the this section is to present specific processes germane to combat engagements and the techniques used to simulate those processes. Several of the techniques addressed in the following paragraphs have been adopted from the *Piloted Air Combat Analysis Model* (PACAM), a high resolution, air-to-air, many-on-many, close-in combat simulation model (1).

The flow of a combat model can be defined as a simple recurrence between two distinct processes. Hartman has labeled these processes as "searching" for and "engaging" the enemy. If the search process does not identify an enemy, the process loops until an enemy is identified. Each of these processes are composed of distinct events. Hartman labels the searching events as "search" for targets, "detect" them, and "select" one. In the case of the engaging process, he labels the events as "fire" at the target and assess the "impact" (19:Sec 2,20). The process described by Hartman appears to be standard for high resolution combat simulations (18).

2.6.1 Searching. Models which simulate the process of searching focus on a situation where a target's location within the search area is unknown. The target is only in the view of the sensor for a limited time as the field of view of the sensor pans the region occupied by the target. This process of searching continues in cycles. The cycles begins with initiation of the search and terminates when the sensor either detects a target or has reached the limits of the search pattern (19:Sec

5,1). "Search models try to describe the probability distribution of the amount of time required to find the target under these circumstances" (19:Sec 5,1). The outcome of the search process is dependent on the effective range of the sensor, the position of the sensor, and the position of the target. Measurement of the sensor position and that of the target is accomplished by establishing a relative coordinate system that is centered on the sensor. As the target approaches and enters the effective range of the sensor, detection can occur (19:Sec 5,4).

2.6.2 Detection. The simplest means of simulating the process of search and follow-on detection is by the deterministic "definite range law" or "cookie cutter" method. This method simulates a sensor which has perfect coverage within the search region as defined by the sensor's sweep limits and maximum range. As soon as the target enters the search region, the target is detected with a probability of 1.0 (19:Sec 5,6). Another approach to detection is a continuous search method. Unlike the cookie cutter, this method is stochastic. The continuous search model used in PACAM is a function of the nominal range of the sensor and the range to the target. The nominal range is a function of the target's aspect. This detection model is show below

$$P_d = e^{\ln(.5)*(R/RNom)^4}$$

where R is the range to target and RNom is the nominal range of the radar with respect to the target's orientation relative to the radar's antenna. When RNom data is not available, the value of one is used in its place. Once the target is subject to detection, a random number is drawn and compared to the detection probability. If the random number drawn does not exceed the detection probability, the target is detected (4:149).

2.6.3 Target Selection. Following the detection of a target, the target is added to a target list, this list may contain one or more targets. Hartman wrote, "An unfortunate fact about target selection modeling is that there is no basic seminal theory known to this writer [Hartman]" (19:Sec

6,1). Hartman acknowledges that each combat model performs the selection of targets based on the scope of the combat model.

Hartman addresses two applicable target selection situations. The first situation is the single autonomous firer. The firer engages the target based only on what he knows about the target. If more than one target is present, the single firer must have a means to prioritize the target selection. Factors to consider in the prioritization of the targets are range, target type, threat, and ammunition availability (19:Sec 6,1-3).

In air models, prioritization of air targets is based on their location within segments of the air space surrounding the aircraft. These segments are defined by the target range and the angle between the aircraft's velocity vector and the line of sight vector terminating at the target. Selection of targets is then based on the presence of targets in prioritized segments. Generally, the segment immediately forward of the aircraft (a position known as the forward quarter) is the preferred region for attacking a target. If an enemy has been detected, and is in some other region, the pilot will maneuver in such a way to bring the target aircraft into the forward quarter (1:84-98).

The second situation defined by Hartman is where several firers coordinate their target selection to mass their fire on a target. To simulate this activity requires a set of fire coordination rules which generally express the execution of some tactical doctrine (19:Sec 6,10). PACAM uses such a set of coordinated procedures defined by the user as an input to the model. These procedures define the tactics to be used in order to concentrate fire on a target (1:84-98).

2.6.4 Impact Assessment. After firing upon the target, the task of simulating the outcome follows. The simulation of a weapon's accuracy assumes that the impact point is a normally distributed random variable (19:Sec 7,2- 3).

The randomness of the impact point is due to a couple factors. These include the human component, such as aim, and the physical component such as deviation in the projectile's ideal performance (19:Sec 7,3). To model the randomness, normal random deviates are generated using the polar method (described in Chapter IV) to simulate impact points, Hartman provides the following.

> Statistical theory suggest that the Normal distribution might be a reasonable model for the impact point. This has been confirmed by thousands of test firing of numerous weapons, and thus the Normal distribution is universally used as the appropriate stochastic model of firing accuracy. (19:Sec 7,3).

2.6.5 Measures of Dispersion. After firing a weapon repeatedly at a target, a pattern around the intended impact point will develop. This pattern illustrates the accuracy of the weapon system. The Normal probability distribution function (pdf) is the most common means of describing the dispersion pattern. The two parameters of the Normal pdf are the mean and the variance of the data. In this case the mean is the round-to-round systematic error component and the variance or dispersion is the round-to-round independent error component (19:Sec 7,3).

The most common metric for classifying the accuracy of the weapon's impact point is the weapons circular error probable "The circular error probable (CEP) is that radial distance from the aim point within which half the impacts are expected to land" (19:Sec 7,4). For the simple case that assumes a circular target, circular impact distribution, no bias, and equal down and cross range dispersion error, the CEP may be approximated by

$$CEP = 1.1774\sigma \tag{1}$$

where σ is the standard deviation of the weapon's dispersion pattern (19:Sec 7,5). Unfortunately, the assumptions of Eq (1) are often overly restrictive. In many cases, σ can not be represented by a single value because the dispersion error along the cross range axis is not the same as that across the firing axis. In addition, most weapons tend to exhibit some degree of weapon bias. To treat these less than ideal cases, the CEP may be estimated by

$$CEP = CEP_{MPI} \cdot (1.0039 - .0528 \cdot V + .4786 \cdot V^2 - .0793 \cdot V^3) \tag{2}$$

where

$$V = Bias/CEP_{MPI} \tag{3}$$

$$CEP_{MPI} = 0.614\sigma_s + 0.563\sigma_l \tag{4}$$

"Bias" is the radial distance between the aim point and the mean point of impact, CEP_{MPI} is the CEP of the mean point of impact, and σ_s is the smaller of the two dispersion measurements (31).

By substituting the representative values for the conditions of Eq (1) into the corresponding parameters of Eqs (2),(3), and (4) the resultant function is that of Eq (1). Therefore, Eq (2) is more robust than that of Eq (1) and it allows treatment of the less than ideal parameters which characterize a weapons accuracy performance.

Probability of Hitting a Target. The CEP can be used in determining the probability of hit for any target size. This probability can be generated with use of the function

$$P_H = 1 - e^{-0.693147 * R^2/CEP^2} \tag{5}$$

where the constant term of Eq (5) is one half of the square of the constant term in Eq (1), and R is the radius of the target. The equation's use of the ratio of target radius to CEP is ideal for expressing the relationship of CEP and target radius (33:Sec 3-25).

2.7 *Simulating Aircraft Performance*

One essential element of combat that Hartman did not include in his two step combat process was movement. For aircraft to stay aloft, they must move continuously. The aircraft movement capability is determined by several performance measures. The performance measures are dependent on several physical conditions such as altitude and aircraft velocity.

PACAM determines several performance factors such as lift, drag, thrust, and fuel consumption by the use of look-up tables of aircraft performance factors. Several of these look-up tables provide a surface of discrete points representing one aspect of the aircraft's performance, say fuel consumption. The method of double interpolation is used to determine fuel consumption which is a function of the two independent factors, velocity and altitude. (3:281); (1:28-36). This method is quite effective in capturing the performance aspects of the aircraft; however, the constant double interpolation of several performance tables adds significantly to the simulation time.

2.8 Missions and Tactics

The scenario of the combat model is what drives the simulation. The scenario may be driven by the mission, its objective, tactics, and the environment to be simulated (18). The type of mission selected to support this thesis is contained in the scenario as a Combat Air Patrol (CAP).

This type of mission comes under the heading of defensive counter air. The defensive counter air mission as defined in *United States Air Force Basic Doctrine, AFM 1-1* is the process of "detecting, identifying, intercepting, and destroying enemy air forces that are trying to attack friendly forces or enter friendly air space" (9:3-3). Shaw describes conditions favorable to the use of CAP. One such condition exists when "attacks by aircraft armed with long-range, stand-off weapons can be launched many miles from their target" (41:325). The advantage of the CAP mission is that airborne aircraft have a greater chance of intercepting an intruder than ground-alert interceptors in time critical circumstances. This, of course, assumes that the approach direction of the attacker is known with some degree of accuracy. (41:326).

The stationing of the CAP is dependent on several factors, such as, fuel, patrol velocity, altitude, and patrol pattern. Shaw describes one such pattern as the Lufbery circle. This pattern, as shown in Figure 7 is an elongated racetrack and provides the best sensor coverage when two aircraft are on the station (41:328).

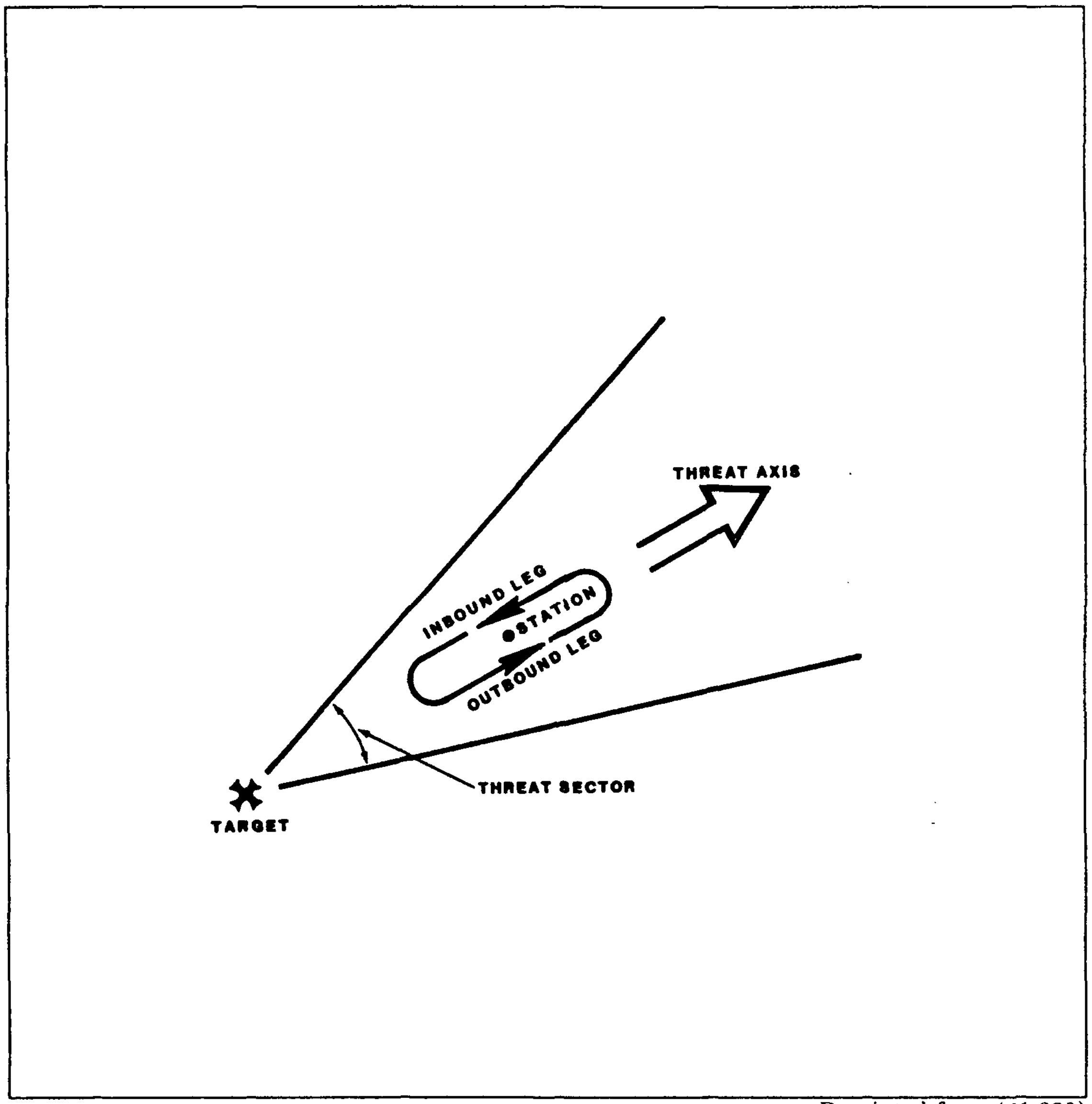
The role of the aircraft flying the CAP mission is to intercept intruders. The method of interception, in most cases, is dependent on the situation. To simulate this activity, research into the type of maneuvers for a given situation was required. Again, Shaw provides the insight as to the type of maneuver used in the typical interception.

The intercept begins with a two part maneuver. The first maneuver begins with the forward quarter intercept. In this intercept position, the interceptor, offset by some prescribed distance, flies a parallel track towards the oncoming intruder. At some point along the interceptor track, a conversion point is reached where the pilot begins the stern conversion maneuver toward the intruder. If performed correctly, this maneuver will allow the interceptor to keep his nose and weapons pointed at the intruder throughout the entire turn. The maneuver results in a position where the intruder is slightly ahead of the interceptor and within weapons range (41:351).

The forward quarter and stern conversion maneuvers require some guess work and simple calculations. To begin, the pilot must determine what is the required displacement between the interceptor and intruder flight paths. This displacement represents the turning circumference for his given velocity, altitude, and sustainable load on the aircraft. This displacement, is determined by the following calculation

$$Displacement = 100 \cdot TAA \cdot Range$$

where TAA is the angle between the target's heading and the line-of-site vector, and range is the distance between the two aircraft. Once the displacement is known, the conversion range must be determined. This is the target range at which the interceptor begins his stern conversion. This range is a function of the length of the arc defined by the current turn radius of the interceptor, the velocity of the two aircraft, and the distance the pilot wants to be behind the intruder when



Reprinted from (41:328)

Figure 7. Combat Air Patrol

he engages weapons. Conversion ranges for specific displacements, velocities, and altitudes are specified in the aircraft's performance charts. (41:348-353). Not all intercepts fly this combination of maneuvers when intercepting an intruder; however, the simulation of this combination is ideal and within the scope of this thesis effort.

2.9 Selection of Computer Languages

In paragraph 2.4, a discussion of the use of graphics as an effective pedagogical technique to achieve instructional goals was presented. However, developing the means of presenting the graphics on a computer screen could become an difficult task without special provision. These provisions, such as functions for drawing a line or circle, are often provided in some higher order computer languages and will be discussed in the paragraph below.

This author selected the BASIC language computer language from among several high order languages to use in this thesis primarily for its ease of coding and graphical function capabilities. In addition to graphical functions in a computer language, a means to provided animation is equally desired. The BASIC computer language features both graphics and animation constructs. Animation in BASIC is achieved by alternating virtual pages of graphic images on the screen. As cne page is displayed, another page is being created. As the graphics images change from page to page and the pages are move sequentially on and off of the computer screen, animation of objects takes place. Additionally, this feature is ideal when there is a need to switch from a graphics format to a text format without losing the graphic images or having to recreate them (45).

BASIC provides additional features other than those of graphics and animation. Functions which enable the altering the color of text and the generation of aural cues permits the development of special features which can enhance the man-machine interface.

The new ANSI BASIC i les many features now common to later generation high order languages. New program flow features include the "select case", "do while", and "if-then-else, and elseif" statements. Another new feature allows alphanumeric labels to direct jumps in program flow. The most outstanding feature between the old and new BASIC is the ability to compile the BASIC code and link to other object files written in different languages. Some reviewers of the new BASIC claim that many of these new features have outdone Pascal, Modula-2, and even C (39:295). Fortunately, in light of the all the new features that BASIC offers, the best of the old features have been retained, such as the graphics functions, multipage screens, program interrupts, and simple syntax.

2.10 Conclusion

A wide range of literature had to be reviewed in order to develop this thesis. Given this wide range, this review was divided into five sections. The first section investigated the worth and development of computer aided instruction. In the second section, the role of computer generated graphics was addressed. The third section laid the foundation for structuring a combat model by classifying a model's purpose, qualities, and construction. Presented in the fourth section was a review of simulating key combat process. This review included the processes of search, detection, target selection, impact assessment, and movement. Finally, the fifth section addressed the issue of computer languages and specific advantages these languages provide.

III. Approach

3.1 Introduction

This chapter presents the approach taken in developing the animated tutor for aerial combat simulation (ATACS). The approach began by defining the objective, breaking it into two distinct sub-objectives and then identifying the tasks associated with each sub-objective to be performed.

3.2 Objective

The objective of this thesis was to develop an aid which would perform visual demonstrations of combat modeling concepts and present a high resolution air-to-air combat simulation. The approach taken in the design of this thesis was to provide instructors of combat modeling with aid that would complement their course material. The aid not only had to complement the course material through demonstration, but also provide the "seeing is believing" realism to key modeling processes. To fulfill this objective, the needs of the instructor and the students of combat modeling remained the foremost guide in this approach.

3.3 Sub-objectives

Two sub-objectives were identified in the problem statement. The first was the need to develop a shell around the actual combat simulation. This shell needed to communicate instructions to the students, prompting them for information as well as responding to their input. Another term typically used to describe this function is "interface"; the shell needed to act as an interface between the student and the computer. The shell also needed to perform the function of "integrator". That is, the shell needed to demonstrate or animate specific learning objectives in concert with the material presented by the instructor. For this purpose, the shell would integrate the lesson material into a dynamic and visual context.

The second sub-objective was the design and implementation of the high resolution combat model. Specifically, the sub-objective required the design of an air-to-air, combat simulation that would demonstrate the basic fundamentals and key elements of combat modeling in a form that could support student learning of model design and use. To support student learning, the model had to be transparent. In this sense, transparency means the model had be visible to the student, much like the clear plastic automotive model engines which allow their viewers to look inside at moving components to see how the parts worked together. Here the student has the ability to look inside the simulation. To permit such an activity, a portal or window had to be developed that would display the current status of the model's parts. These parts are, in fact, the model variables which characterize the changing states of the simulation over time.

3.4 The Shell

As mentioned, the shell had to interface between the student and the computer while integrating lesson material with animated illustrations. The interface process required the development of some means to prompt the student's actions. The shell had to provide a means to present the available presentations to the student and permit flexibility in the timing or selection of ATACS options. Two feasible solutions where considered. The first was that of a predetermined sequential presentation approach much like a guided tutorial. In this case the student would have been presented with concept number one followed by concept number two followed by the remaining concepts in sequence. This approach would have met the first requirement to present all presentations; however, it would have failed the second requirement of flexibility.

The second approach was to develop an interactive hierarchial system of menus. This approach would permit the display of all the programmed presentations as well as allowing the flexibility to view a presentation or simulation from any point in the program. This approach also provided a degree of flexibility for the instructor as to when the material should be presented and the order of presentation. The menu approach met both requirements of presentation and flexibility and was selected as the preferred means by which the shell would operate.

3.5 Menus

The menus present all of ATACS available options as well as provide the student with the flexibility to revisit a specific option or skip around it. The following discussion addresses these menus and the options each provide.

3.5.1 Opening Menu. The opening menu permits selection from the following options:

- Examples of Processes
- Run Combat Demonstration
- Display Input/Output
- Terminate

Once an option is selected from the opening menu, a subordinate menu is displayed. The subordinate menu presents the selections associated with the selected main menu option. The following paragraphs describe the features of each option.

3.5.2 Examples of Processes. Following the selection of *Examples of Processes,* six examples of processes common to high resolution combat models are offered. These processes are:

- Random Number Generation
- Search
- Detect
- Target Selection
- CEP Demonstration

• Track Angle

Each of these options introduces a process used in air-to-air combat simulation. Once an example is selected, a narrative of the process is displayed and student interaction is solicited. Most examples require student interaction, prompting them to manipulate and explore the process being examined. In most of the examples, animation is used to support the presentation and the point of the example.

3.5.2.1 Random Number Generation. The first option, *Random Number Generation*, presents a brief narrative describing the useful purpose of the pseudo random number generator. The student is prompted to enter an initial random number seed and the number of samples to be generated. A test of the generator is then performed to demonstrate the random number generator's uniform probability density function. This feature allows the student to observe graphically the impact of sample size and seed values on the test results. It also presents a means of validating the uniform random number generator to the student.

3.5.2.2 Search. The second option, *Search*, presents a brief narrative that describes how the process of search may be simulated as an attempt to detect.

3.5.2.3 Detect. The third option, *Detect*, presents a narrative that describes two methods to simulate the detection process. The two methods presented are the deterministic "cookie cutter" method and a stochastic continuous method. An illustration contrasting both methods is presented to the student.

To illustrate the cookie cutter method, a step function is plotted on the screen. It shows the range at which the probability of detection steps from zero to one. After the student enters a range–to–target in response to a prompt, the program places a maker on the step function corresponding to that range and displays the outcome of the detection event.

Following the example of the cookie cutter, the program enters the second detection example. This example presents a stochastic detection model. A plot of the model's output is superimposed over the earlier plotted step function as a means of comparing these two approaches. The student is again prompted for a target range. Once entered, the probability is determined, and a marker is plotted on the probability curve. The program then selects a random number. The random number drawn is used to determine if the target was detected at the range entered by the student. The outcome of the detection event is then reported to the student.

The point of the detection example is to highlight the fundamental differences between two common modeling techniques. Understanding these differences allows the student to determine which of these techniques is best suited for a specific simulation task.

3.5.2.4 Target Selection. The fourth option, *Target Selection*, is a narrative describing a means to simulate the target selection process in the air-to-air combat environment. It addresses the technique of rule based target prioritization and selection as well as the simulation of identification friend or foe interrogation.

3.5.2.5 CEP Demonstration. The fifth option, *CEP Demonstration*, presents a narrative pertaining to the calculation and use of this factor. In this example, the student is prompted to enter the number of sample weapons firings. The down range and cross range miss distance for each firing is then plotted on a graph. At the completion of the firing, mean and variance of the cross range and down range miss distances are calculated. Based on these statistics, the circular error probable (CEP) is calculated and a circle depicting the CEP is drawn around the impact points.

A plot of the probability of hit, based on the calculated CEP is then plotted. The student may enter various target radii and observe the impact target radius may have on the hit probability.

To contrast the impact of CEP on the probability of hit, the student may enter additional CEP values to generate new cumulative distribution curves.

The circular error probable (CEP) example was designed to illustrate the concept and application of the CEP factor. The point to be illustrated is that as different CEPs are used, the probability of hitting a target of constant size changes.

3.5.2.6 Track Angle. The final example, *Track Angle*, is another narrative example. It discusses the concept of the track angle, specifically describing how it is determined and applied.

3.5.3 Combat Demonstration. This selection begins the simulation of air-to-air combat. The simulation utilizes most of the processes demonstrated in the narrative and animated examples. Immediately following the selection of the simulation option, the student must select an "express" or "edit" data entry option. The express option loads the scenario file or user requested scenario file and begins the simulation. The edit option permits the student to load a scenario file and edit its contents. The option of editing the scenario file invites the student to explore the model's flexibility and its ability to respond to scenario changes.

During the combat simulation, the student will have the option to view the simulation graphically as an animation on the computer screen or switch to the *Attribute Screen*. The Attribute Screen displays the status of several characteristics of the entities as they change over the period of the simulation. A sample of these characteristics include the aircraft velocity, fuel consumption, location, and status such as engaged or not engaged.

3.5.4 Display Output. When the simulation is complete, the student may select the *Display Output* option. Output reports from a simulation provide the opportunity to compare the results with an expected outcome. Output reports are one of many valuable tools used to verify and validate simulation models.

The output reports available to the student are the following:

- Reflected/Echo Input Report
- Detailed Summary Report
- Aircraft Performance Factors
- Significant Event Summary Report

3.5.4.1 Reflected Input Report. The first option, *Reflected Input Report*, produces a listing of all the input data, both user entered such as the scenario file and embedded, such as the aircraft performance data file. This report exemplifies the magnitude of data required for a small simulation such as this. Furthermore, this option demonstrates the extent of data that may be embedded in the "black box" without the prior knowledge of its existence. This option emphasizes the extent of embedded data within models which may not always be apparent to the model user.

3.5.4.2 Detailed Summary Report. The *Detailed Summary Report* produces a listing of selected variables and their values at each time pulse of the simulation. This report gives the student an option of capturing an account of what actually took place inside the model.

3.5.4.3 Aircraft Performance Factors. This report lists the model-generated aircraft performance factors. The display lists such factors as aircraft thrust, fuel consumption rates, and turn performance. This report highlights the fact that the entities, aircraft in this case, must be characterized by their performance, such as their ability to move and consume.

3.5.4.4 Significant Event Summary. The final report, *Significant Event Summary*, provides a listing of the significant events and their time of occurrence. Such events include detection of an intruder, target identification, and target destruction. The advantage of such a report permits examination of the occurrence of major events directly without having to uncover the event from the profusion of data found in the extensive detailed listing. These reports are indicative of those found in most high resolution models and provide an added view into the operation of the simulation. Also, these are the typical reports used by modelers in debugging and validating models.

3.6 The Air-to-Air Combat Model

The air-to-air combat model is the core of this educational aid. Discussions of key factors and concepts, such as the input data, combat scenario, entities, and program flow are presented in the following paragraphs.

3.6.1 Data. The data used to drive the simulation was adapted from the Piloted Air Combat Analysis Model (PACAM). This data was provided by the Survivability/Vulnerability Information Analysis Center located at Wright-Patterson AFB, OH. The primary data includes aircraft and missile flight performance and fuel consumption data. The performance data relates specific attitudes (velocity and altitude) to consumption and performance measures. PACAM and the data was validated by comparison with 37 scripted flight tests.

Two possible approaches for the application of this data were explored. The first approach was to calculate aircraft performance by interpolation of data arrays as is done in PACAM. The disadvantage of this approach is the excessive processing time required at each time step during the simulation. This excessive processing would have rendered real time animation of the simulation impossible. The second approach was to fit the data to a surface, but that did not prove to be feasible. For example, fuel consumption rates are based on two parameters, altitude and velocity. A surface plot of these parameters revealed a very difficult and complex surface to regress.

Both approaches presented serious limitations; however, in light of the scope of this research, a modification to the first approach was adopted. The aircraft performance factors based on a single altitude and two velocities are computed at the models initialization. Unfortunately, this limits the aircraft to a single altitude and two, user selected, velocities.

3.6.2 Scenario. Before the flow of the model can be presented. the scenario surrounding the model and the roles played by each of the characters must be discussed. The scenario is what drives the simulation. Without a clearly defined scenario, design of the simulation would proceed without direction. Hence, a scenario had to be defined for this simulation which included defining combat objectives or missions, command relationships between aircraft commanders in the simulation, and resources available to both Red and Blue forces (28:29- 30).

In chapter II, a review of the defensive counter air mission and a subset of that mission, the combat air patrol (CAP) was presented. It is the CAP mission that is simulated in this model.

The scenario begins with two Blue air interceptors (AI) flying a circle profile over a forward operating location in defense of a Blue airbase. Upon detection of a possible hostile intruder or intruders by the airbase's tactical surveillance radar, the AI are vectored to intercept. Guidance to the intruder is controlled by the ground controller until the interceptor's sensors have locked on to the intruder. To demonstrate decision processes and maneuvering, a rule of engagement (ROE) embedded within the model requires the interceptor to maneuver within visual range of the intruder to confirm its identity.

At the moment of the intruder's detection, Blue accelerates to the intruder until he is within the visual range of the intruder. The intruder continues on a course toward the airbase target unaware of Blue's advance. Upon visual identification of the intruder by the Blue aircraft's commander, Blue fires a missile at the intruder. At the time of missile launch, location of the target is passed to the missile and missile begins its flight toward the target. During missile flyout, the missile is continually updated on the location of the target. When the missile has reach the location of its target, an impact assessment is performed. If the intruder is eliminated, the interceptor resumes flying the CAP. If the intruder survives, the interceptor prepares for another missile launch. If the intruder makes it to the intended target before being eliminated, it destroys the target.

Each Blue aircraft will continue to track the intruding Red forces unless a " Bingo " condition is experienced. A Bingo condition is when an aircraft has reached a point where only enough fuel remains to safely return to base. If such a condition occurs the Blue aircraft will break off any attempt to intercept the attacking Red force. The one exception to this event is in the case where a Blue aircraft has a target within visual range. If this condition exists, the Blue aircraft remains engaged with the intruder until either the Red or Blue combatant is destroyed.

The simulation ends in one of two ways. The student may end the simulation via a *Quit* feature or the simulation times out.

Once the simulation has terminated, the student is offered the option of reviewing a list of questions addressing the models features. An example of these questions is provided in Appendix C.

3.6.3 Entities and Attributes. The aircraft, missile, and airbase entities have certain characteristics referred to as attributes. Entities in ATACS exist as a collection of the attributes listed in Table 1; the entity is created by assigning values to these attributes. Since an aircraft is an entity, this entity can be characterized by its sidedness such as "Blue" or "Red". Each aircraft can be characterized by its velocity—forward velocity, and component velocities in the X and Y directions. Since the aircraft is moving in a direction, then the heading becomes another means of characterizing the entity. The aircraft is traveling in three dimensional space; therefore, its location as defined in a three dimensional coordinates system will also describe the entity. The aircraft's list of targets and their location is another attribute which can further describe the entity. Finally, the status of the entity such as "patrolling", "intercepting", "detected", "not detected", or "killed in action" are also used to characterize the entity.

Other entity types mentioned above include the missiles, and the airbase. For the missiles, many of the attributes that characterize the aircraft apply to the missiles as well. As for the airbase, its defined by its X and Y coordinates, posture, and the range of the search radar which it owns.

Aircraft	Missiles	Airbase
Side	Owner	X Location
Aircraft Status	Missile Number	Y Location
X Location	Missile Status	Posture
Y Location	X Location	Radar Range
Heading	Y Location	
Fuel Level	Heading	
Velocity	Flight Time	
Thrust Mil	Velocity	
Thrust AB	Tracker Range	
Fuel Mil	Kill Probability	
Fuel AB	Target	
Sustained Gs		
Max Gs		
Min Turn Radius		
Wing Area		
Max weight		
Min weight		
Radar Range		
Optical Range		
Target		
Number of Missiles		

Table 1. Entities and attributes

3.7 Generalized Model Flow

Using the scenario and the entity's characteristics, the following generalized model flow was developed.

- Begin simulation, initialize variables
- Assign Red's Target
- For each second of simulated time perform the following

   1. SEARCH
      - Fly CAP mission
      - Search for intruders
      - Intruder detected, begin TARGET SELECTION, otherwise return to SEARCH
   2. TARGET SELECTION

- Assign available Blue aircraft to Red targets based on closes proximity

3. PURSUIT

   - Vector Blue to Red intruders

   - If firing conditions are met, begin FIRE, otherwise continue PURSUIT

4. FIRE

   - Fire missile at intruder

   - Assess impact

   - If target was not destroyed and firing conditions met, FIRE

   - If intruders are destroyed, return to CAP

The above describes the action taken primarily by the Blue forces. In this model, the Red forces are non-reactive. They have a single minded goal to attack the Blue airbase.

3.8 Verification and Validation

Verification. Verification is the process of verifying that the code, as written, is correct in syntax and logic. The verification process was performed throughout the development of ATACS by module prototyping, graphical methods, animation, and review of output products.

In many cases, a prototype module which simulated a specific process was written and tested before introduction into the model. The module's output was then verified against existing values when available. The development of the module which calculates the aircraft performance factors was such a case. Each factor produced by ATACS was verified against a related performance chart found in the aircraft's operating manual.

A second method used to verify specific analytical processes made use of MATHCAD, a mathematical software package with graphics support. Using this method, the output from the developed module was plotted and compared to the results from MATHCAD. This method was used to verify the module's representation of the modeled analytical process.

The third method used in verifying processes such as movement and scale considerations was ATACS animation feature. This method allowed visualization of the processes such as the aircraft turning in the direction of the calculated heading, as well as the distance the aircraft moved in one time pulse. An unreasonable outcome here indicated a breakdown in logic or scale of units.

The fourth method used another ATACS feature. The attribute screen was used to monitor attribute values during the simulation, and ATACS' generated output products were reviewed following a simulation. These built in features quickly identified improperly assigned values to variables and logic errors.

These last two methods of verification clearly identified the strengths of the significant animation and attribute screen features of ATACS. These features are intended to served as a means to expose the simulation's "guts" to the student. Proof of the ability of these features to provide their instructional value lies in the constant reliance on these features by this author when verifying many of ATACS' modeled processes.

Validation. Validation is the process of ensuring the model is reflecting the "real world" to the extent possible. For example, an aircraft that continues to fly with no fuel or performs maneuvers inconsistent with a specific situation or condition does not reflect this real world image. Once again, ATACS' ability to animate the simulation and display the dynamic states of the simulation were relied upon in the validation process also.

Validation lies with the experts, the experts being the individual or group who have experienced the environment being simulated. In this case, the scenario and animation were reviewed by experienced pilots whose comments and recommendations were reviewed and if possible, incorporated.

3.9 Hardware Requirements

Returning to the objective for a moment, a requirement for the system was that it needed to be portable. So a means by which the system could be made portable had to be considered. One type of computer system had to be selected to base the program on. It was decided that the design should be compatible with an IBM AT system or clone with the Mircosoft Disk Operating System. This choice would permit portability to the largest population of computers.

The entire system is capable of being transported on two 360 kilobyte floppy diskettes. The operating speed of the machine clearly influence the run time of the program. An 80286 class or better machine produces reasonable performance, but use of an XT requires the virtue of long-suffering. The program is compatible with the enhanced graphics adapter (EGA) and the video graphics array (VGA).

3.10 Graphic Requirements

Being IBM compatible does not ensure the compatibility of controlling and presenting screen graphics. The presentation of graphics is controlled by the computers graphics adaptor. The two more common adapters are the EGA and VGA adapters. Each of these have design limitation on screen resolution (the detail of the image that can be presented). The EGA detail is less than that of the VGA. Thus programs written explicitly for VGA graphics are not downward compatible with an EGA system. However, the reverse is not true, that is, a graphics program written for the EGA is most often compatible with a VGA system.

One approach to be taken to meet the condition of portability was to develop the graphics software to be compatible with the EGA. This in turn would permit the system to run on both EGA and VGA based systems. The cost of this solution would be a limitation of screen resolution for the VGA user. The graphics would always be constrained to that of the EGA limitations. A second approach was to develop a means by which the computer program would interrogate the system it was installed on and determine the type of graphics hardware of the computer host. Once the hardware configuration was known, the program would automatically configured itself for the existing hardware. This latter approach was selected based on the flexibility it offered.

3.11 Animation

During the air-to-air combat model demonstration, the positions of the graphic images, such as aircraft icons, are computed and updated. Refreshing the screen with the updated image position provides animation of the icons. Two possible methods to create the animation were possible and examined. Selection between these methods was based principally on the quality of the animation, the computational efficiency, and absence of distracting screen flicker.

The first method was based on a straight forward sequential operation. Starting with as much as four icon aircraft images the process would begin by drawing the first, second, third, and then the fourth icon. In each iterative loop, the first icon is erased from its current position then redrawn based on its updated position. The second icon is then erased and redrawn based on its updated position then on to the third and fourth icons. This approach requires the image to be drawn on the screen as the image is being viewed. This process could be quite distracting for a computer operating with a clock speed of eight mega-hertz or less.

The second method makes use of a special screen feature provided by the high order computer language BASIC. This feature permits the drawing of images on a virtual screen page in the computer's memory while presenting an earlier drawn image to the viewer. When it is time to update the screen image, the virtual page in memory is presented, already drawn. The drawing of pages in memory permits a smooth transition between update icon positions.

Unfortunately this option also comes with a cost. The minimum number of pages for the animation would be two. When the page feature is used, a portion of the computer's graphics memory is allocated to these pages. As the number of pages increase, say four, the resolution of each page decreases. So the cost of this option is in resolution of the icon images if 2 or more pages are used. With only two pages allocated, the resolution is equivalent to that of EGA graphics.

Use of the screen page options would limit the screen to that of the EGA requirements. Currently, BASIC does not support this feature for VGA graphics. Therefore, until BASIC can support the page feature for VGA, the sequential operation will remain as the preferred structure for animation.

3.12 User Documentation

To assist in operating ATACS, user documentation was developed. A copy of the documentation is attached as Appendix B. The documentation is divided into two major sections. The first section describes ATACS' system files and installation procedures. The second describes the ATACS model options such as the animated examples, the combat demonstration, and model output display.

A listing of the files which makeup ATACS are provided below.

- **ACS.EXE:** ATACS executable file
- **ACS.BAS:** The aircraft combat simulation source code.
- **MAINMENU.BAS:** The menu driver module.
- **MODRND.BAS:** The random number example module.
- **MODPDET.BAS:** The probability of detection example module.
- **MODCEP.BAS:** Impact assessment example module.
- **MODMSG.BAS:** Message module.
- **MODOUT.BAS:** Output module.
- **MODTITLE.BAS:** Generate ATACS opening screen.

- **B1PERF.DAT:** Blue one's aircraft performance data.
- **R1PERF.DAT:** Red one's aircraft performance data.
- **B2PERF.DAT:** Blue two's aircraft performance data.
- **R2PERF.DAT:** Red two's aircraft performance data.
- **B1PERF.DOC:** Blue one's documented aircraft performance data.
- **R1PERF.DOC:** Red one's documented aircraft performance data.
- **B2PERF.DOC:** Blue two's documented aircraft performance data
- **R2PERF.DOC:** Red two's documented aircraft performance data
- **MASTER.DAT:** Default scenario data file
- **ACPERF.DAT:** Calculated aircraft performance factors. This file is produced during the simulation.
- **DETAIL.DAT:** A Second-by-second detailed listing of activities that occurred during the simulation. This file is produced during the simulation.
- **OUTPUT.DAT:** A listing of significant events that occurred during the simulation. This file is also produced during the simulation.

Additional documentation is available when using ATACS. Accompanying each set of menus is information addressing the purpose of each displayed option. As an introduction to the combat demonstration, a scenario synopsis is presented to the student. Following the synopsis, a simple reminder of the purpose of each of the function keys is displayed. Upon selection of an output report, a short paragraph addressing the type of report and its typical use is presented before the report is actually displayed.

One last source of documentation is the source code itself. The source code is provided to the student as another instructional aid. The student is free to enhance or embellish the code in any fashion which will support further understanding of simulating combat. For this reason the code has over 900 embedded comments. Many of these comments are program variable definitions, descriptions of conventions used in naming variables, descriptions of sequences, and warnings.

3.13 Enhancements

The processes and order in which these sub-objectives were accomplished are illustrated in Figure 8. Enhancement iterations continued until the remaining time was insufficient to complete another full iteration. Further enhancements may possibly include greater interaction between the user and the model, additional examples, additional aircraft, and an increase number of screen options.

3.14 Methods Summary

The following paragraphs below summarize the methods used in the design and development of this educational aid. These methods consisted primarily of research, identification and use of existing combat model computer algorithms, interviews, and use of operation research techniques.

The scenario had to demonstrate, to some degree, Air Force doctrine, rules of engagement, tactics, and decision rules. Development of the scenario was supported by interviews with rated Air Force officers and allied officers to gain insight into mission planning, electronic sensor use, maneuvering, and tactics employed during air-to-air combat.

The use of class notes from Military System Simulation, Military Systems Analysis, and Combat Modeling High Resolution and Aggregated combat modeling were the core sources of simulation techniques and applications of combat models.

A review of air-to-air combat models' operating manuals provided valuable insight into program structure and algorithms used to represent the combat. From these manuals, the necessary characteristics to sufficiently depict entities and their objectives were identified.

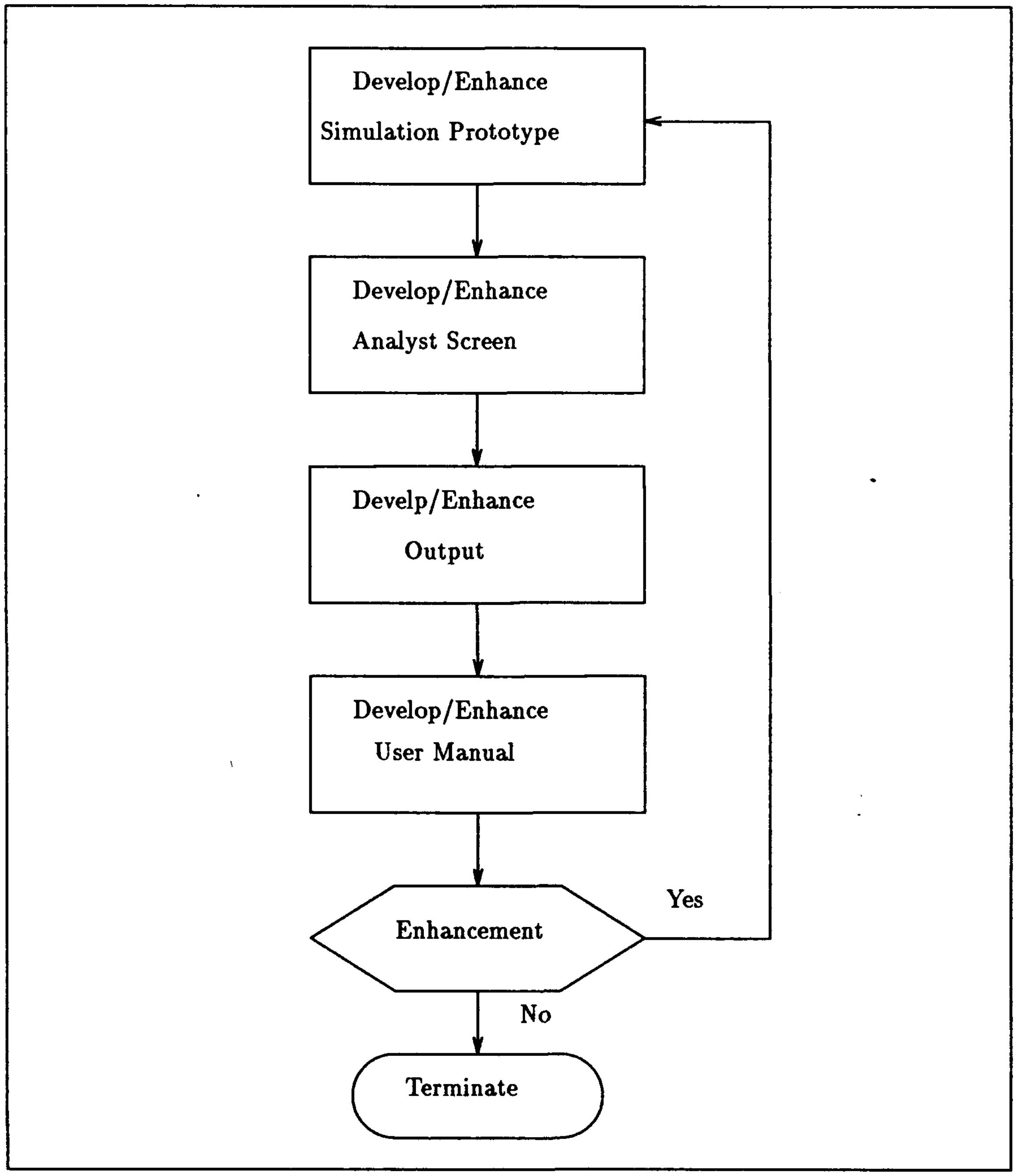


Figure 8. Enhancement Flow

The methods used by air-to-air manual "board" wargames were studied as alternatives to simulate movement, search, and detect processes associated with air-to-air combat. It was hoped that these board games might reveal computationally efficient process algorithms. Unfortunately, they did not.

To establish what ATACS should emphasize, instructors and students were interviewed. The interviews focused on finding areas of combat modeling instructions that could be complemented by a computer based demonstration. The instructor's and students' responses provided the guidance in developing the animated examples found in ATACS.

3.15 Summary

This section presented the approach taken in the design of ATACS. In many cases, design options that would impact the value and performance of this aid were presented and argued. Presentations of several key features were provided and accompanied by an explanation of the value each feature. An outline of the combat scenario was presented as well as the concepts of the entity and its characteristics or attributes. The essential verification and validation approach and its results were fully discussed. Finally, the case of hardware consideration and compatibility was presented and argued. Chapter IV will discuss the design of the many BASIC subprograms within ATACS.

IV. ATACS Construction

4.1 Introduction

The purpose of this chapter is to present an informative discussions of how the animated tutor for aerial combat simulation (ATACS) simulates key processes common to combat models. The following sections highlight the major processes used in generating visual examples, animation, combat simulation, and output generation.

Discussions of the logic flow of ATACS have been limited to the pseudo code level. Unfortunate as it may seem to limit the discussion in such a fashion, the alternative to address every line is impractical. Variable definition for the air combat model and a complete documented source listing have been included in Appendix D and Appendix E. Whenever reference to a subprogram or function is made, the actual cryptic name of the program will be used. This should provide useful guidance for those who wish to follow the pseudo code discussion using the provided source code.

4.2 ATACS Specifications

Throughout the discussion of ATACS' program structure, the reader will find that ATACS is module and subprogram orientated. A module was used to compartmentalize the many options, such as the process examples, the combat model, and output generation offered within ATACS. Within a module, subroutines and functions were used to separate tasks. All together, ATACS consists of eight modules and 50 subprograms. In terms of lines of source code, ATACS consists of 5,074 lines. Of these lines, 3,520 lines consist of one or more executable statements. As Microsoft BASIC 4.5 permits more than one executable statement per line, the 3,520 lines contain 4,514 executable statements. During the discussion of the user manual in Section III, the use of embedded comments was mentioned. ATACS has 976 lines of white space, 321 comments, comments which share the same line as an executable statement, and 582 full-line comments. All together, not counting white space, there are 903 embedded comments in ATACS' source code. In terms of data and program files, the system consists of one executable file, five input data files, four documented input files, and eight module source code files. The system will fit on two, low density, 360 kilobyte, floppy diskettes.

4.3 Menus and Dynamic Examples

This section briefly presents the operation of ATACS' menu feature followed by a detailed discussion of the flow of each of ATACS animated examples.

4.3.1 Menus. Figure 9 illustrates ATACS' hierarchial structure of menus and available options. The lines connecting the menus define the routes between the main menu and the subordinate menus. The purpose of the menu system is to direct execution of subroutines which correspond to the student's selected option. Figure 10 illustrates the basic logic flow of the menu system. The main menu is the initial starting point for ATACS. A selection from the main menu will cause the display of a subordinate menu. A selection from subordinate menu in-turn executes the subprogram supporting the selected option.

4.3.2 Animated Examples. Of the six examples provided under the *Examples* option, only three are addressed here. These examples are: Random Number Generator, Detection, and CEP Demonstration. Those that are not addressed are static examples presenting text only.

4.3.2.1 Random Number Generator (RNG). The subroutine RNDDEMO performs all the necessary student interfaces, calculations, and graphical operations required to run this example. The only exception is the use of the functions RND and RANDOMIZE, which are provided by Microsoft BASIC. Figure 11 is an abbreviated logic flow diagram of RNDDEMO.

The subprogram is called by the main program. Upon entering the program, a page of introductory text is displayed. followed by the initialization of variables and graphic parameters.

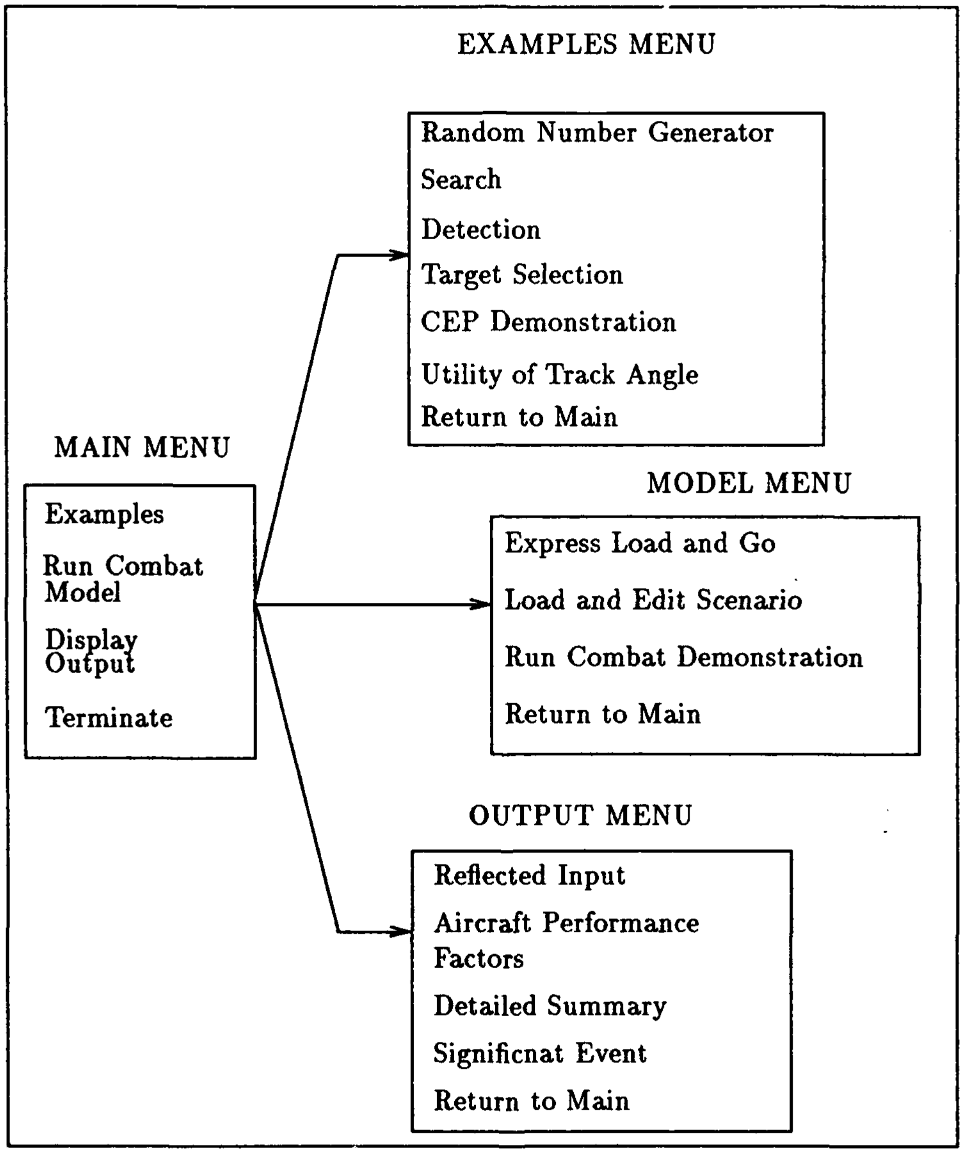


Figure 9. Menu Options

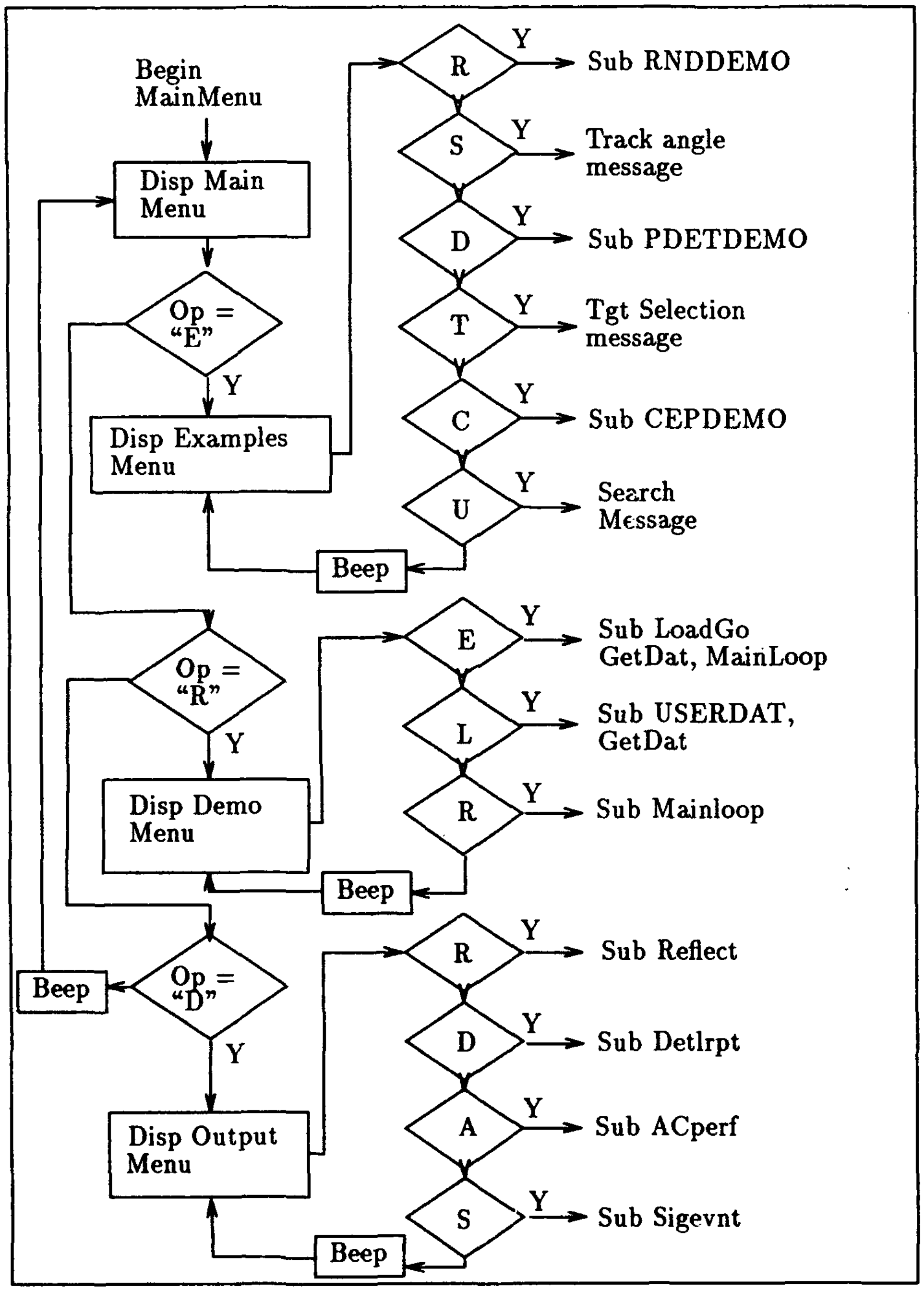


Figure 10. Menu Logic Flow

The graphical parameters include establishing the viewport (a graphic window display) and window dimensions which orient the viewport coordinates.

Following initialization, the student is then prompt to enter a seed value for the RNG and the number of samples to be drawn from the RNG. Once these entries have been made, the program enters a recursive loop. The operations within the loop include the drawing of the random number, finding the cell in which it belongs, drawing and stacking a bar in that cell, and updating the cell frequency. There are 10 cells; each cell is 0.1 unit wide. Once the random number is assigned to the appropriate cell, a bar is added to the cell stack to animate a dynamic histogram. Additionally, for each cycle of the loop, the cell frequency is computed and presented. The cell frequencies are displayed along the X axis. Once the number of samples has been reached, the program exits the loop and prints the output statistics. The student is then prompt again to choose another sample size or to exit the example. The screen image displayed to the student is shown in Figure 12.

4.3.2.2 Detection. The subroutine PDETDEMO performs all the necessary student interfaces, calculations, and graphical operations required to run this example. The only exceptions are calls to functions PDET and RND. Figures 13 and 15 are abbreviated logic flow diagrams of subroutine PDETDEMO. Figure 13 illustrates the flow of the deterministic detection model, and Figure 15 illustrates the flow of the stochastic model. Both methods are featured as animated examples.

PDETDEMO is called by the main program whenever the student selects the *Detection Methods* example. The first part of this program demonstrates the *cookie cutter* method. Upon entering the subroutine, variables are initialized and graphical parameters are set according to procedures similar to those described in Section 4.3.2.1.

Following the initialization, a step function representing the cookie cutter range is plotted in the graphics window. The student is then prompted to enter a *Range to target* value. The program

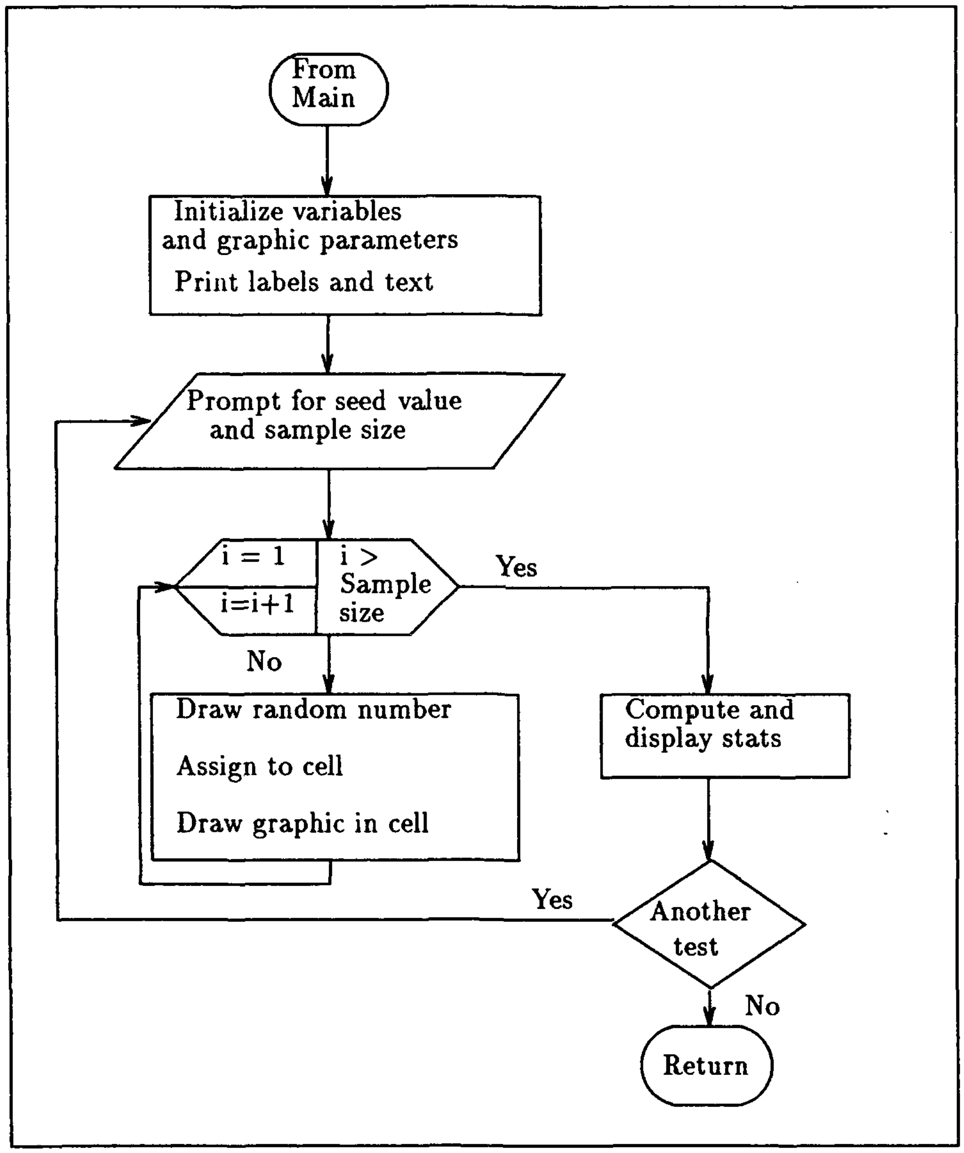
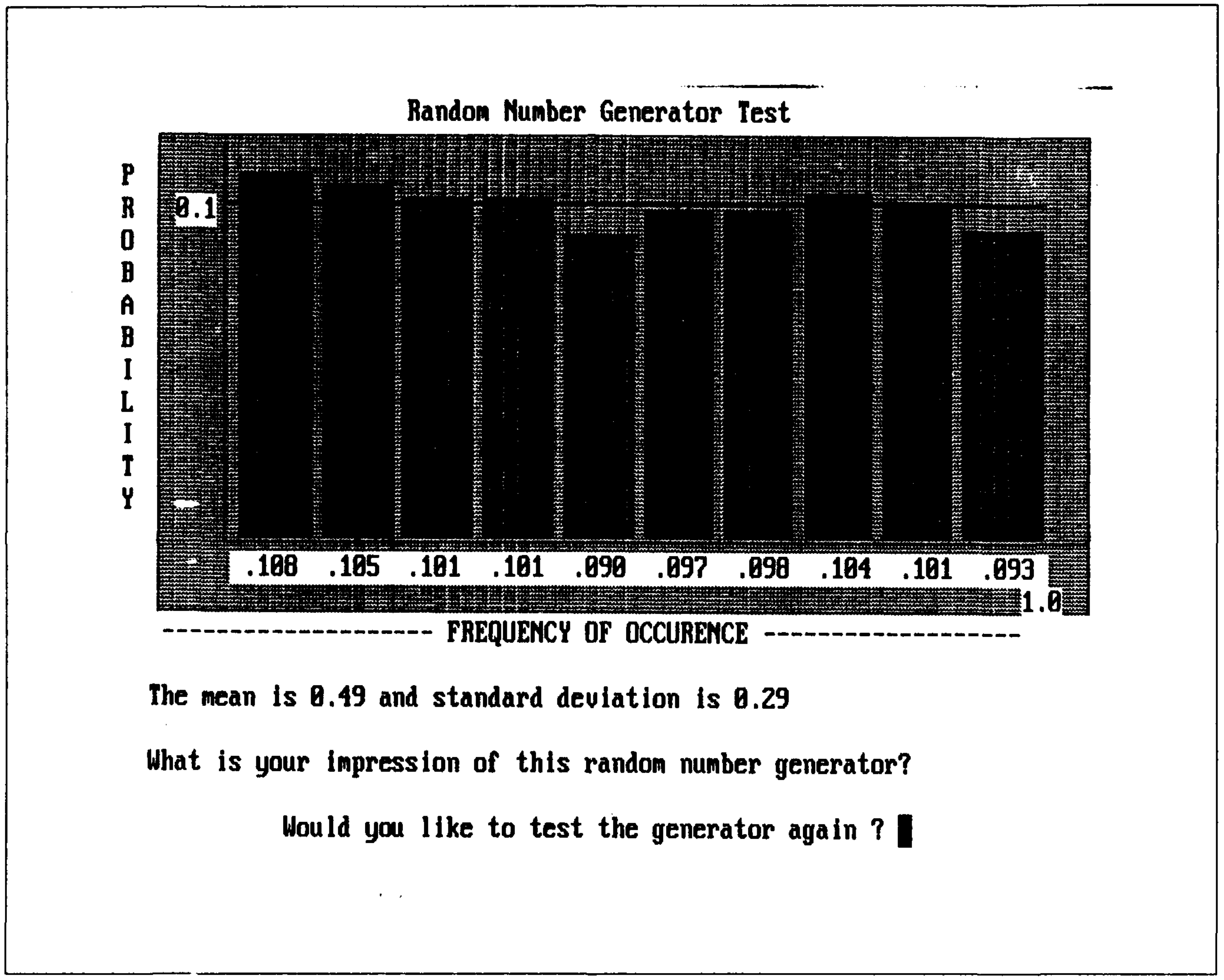


Figure 11. RNDDEMO Subroutine


Random Number Generator Test
PROBABILITY
0.1
.108 .105 .101 .101 .090 .097 .098 .104 .101 .093
1.0
-------------------- FREQUENCY OF OCCURENCE -------------------

The mean is 0.49 and standard deviation is 0.29

What is your impression of this random number generator?

Would you like to test the generator again ?

Figure 12. Random Number Generator Example Display

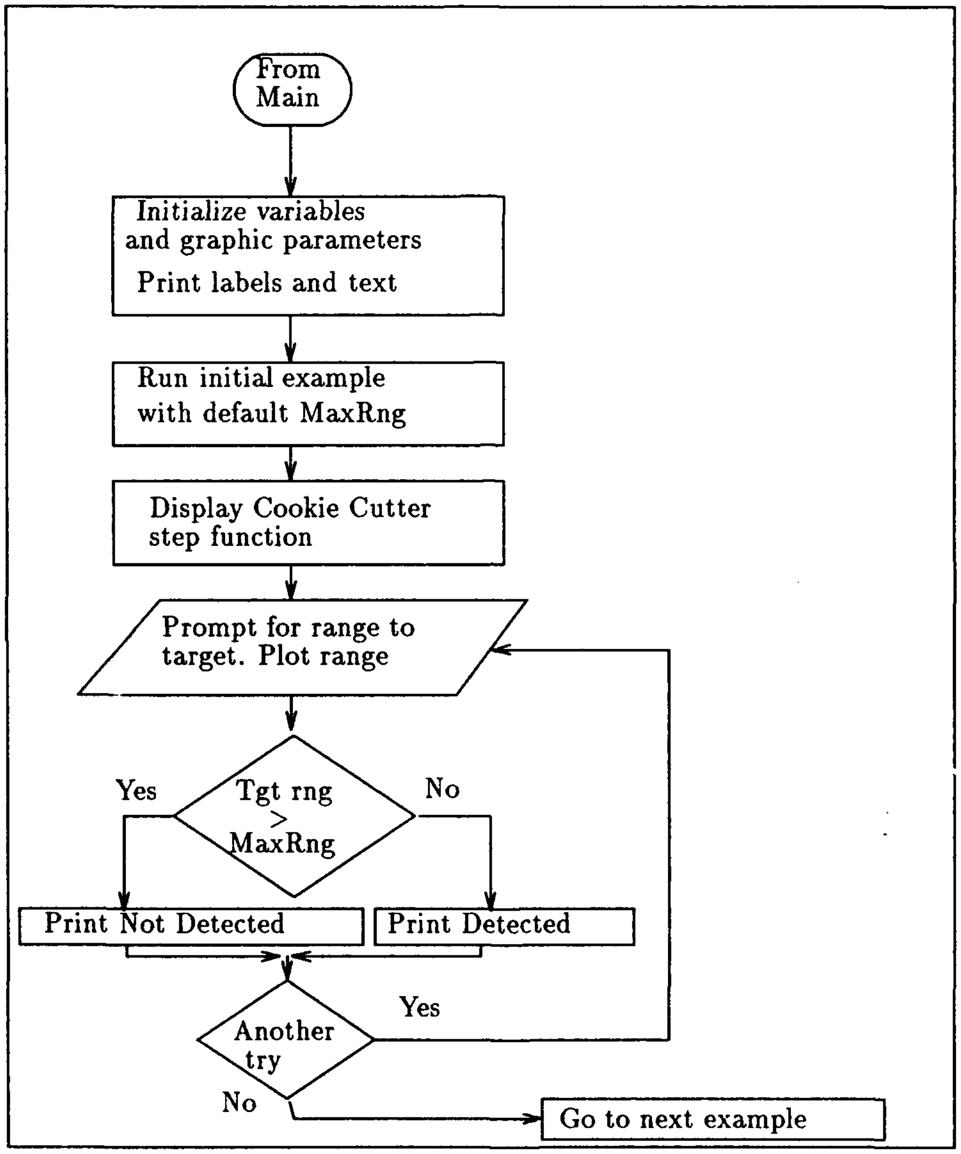

From Main
Initialize variables and graphic parameters
Print labels and text
Run initial example with default MaxRng
Display Cookie Cutter step function
Prompt for range to target. Plot range
Tgt rng > MaxRng
Yes
No
Print Not Detected
Print Detected
Another try
Yes
No
Go to next example

Figure 13. PDETDEMO Subroutine (Deterministic)

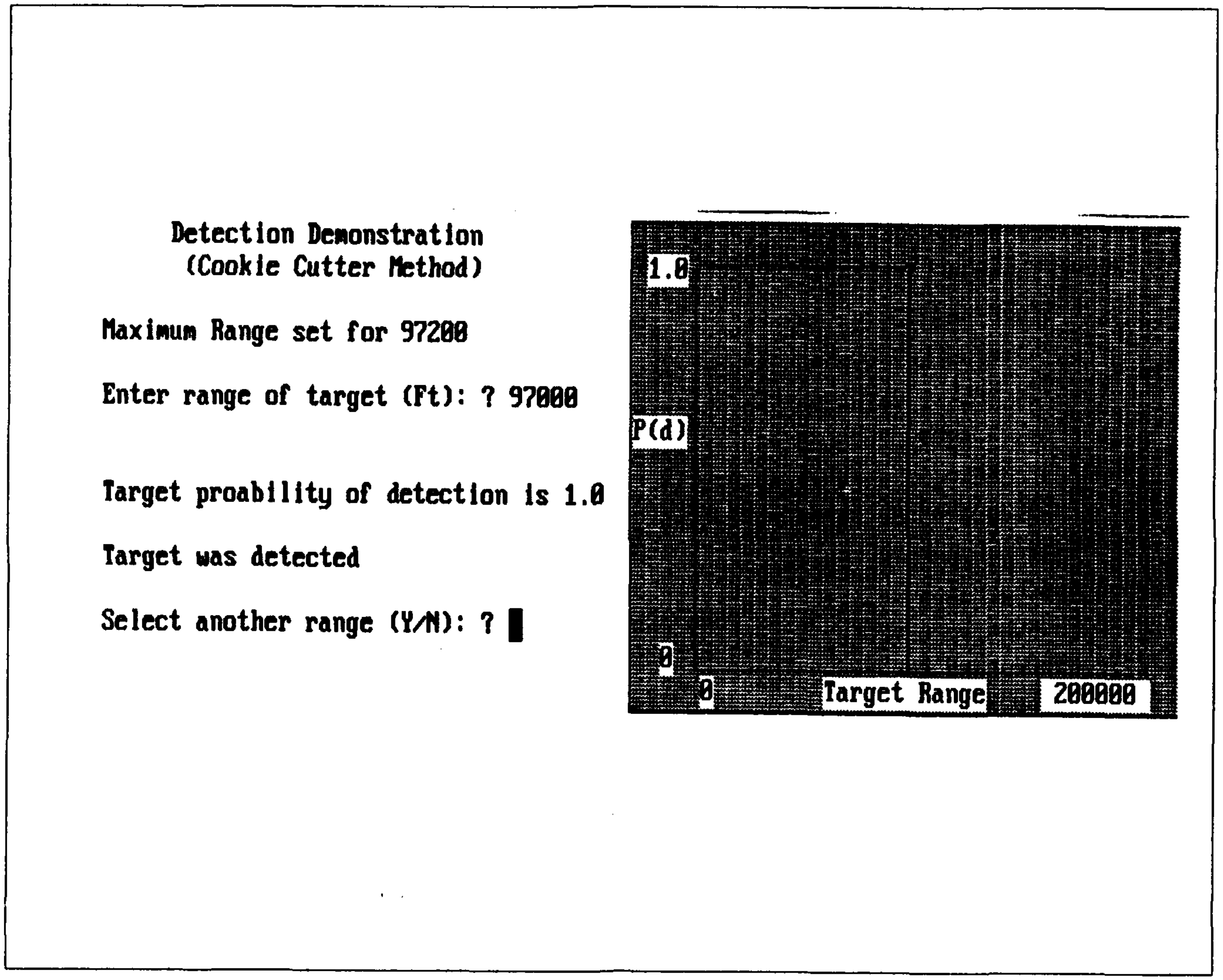
plots this point on the curve and reports the outcome of the detection process. Figure 14 shows an example of the display presented to the student.

To contrast the deterministic and stochastic methods, the program superimposes the graph of the exponential detection function over the previous plotted step function. Again the student is prompted for a range to target. Once this value is entered, the program plots the point on the curve, computes the probability of detection, makes a draw from the RNG, and prints the detection results. The results displayed to the student are: probability of detection, the value of the random number generated, and whether the target was detected or not. Figure 16 shows an example of the display presented to the student.

In both, the deterministic and stochastic examples, the student may enter as many range-to-target values as they wish. In the case of the stochastic example, reentering the same range emphasizes the element of chance. By superimposing the graph of one function over the other, the student is presented a visual comparison of these functions. By entering the same range-to-target for in both examples, the student can observe the different characteristics between these two detection techniques. Understanding these differences is the point of this example and allows the student to determine which of these techniques is best suited for future simulation tasks.

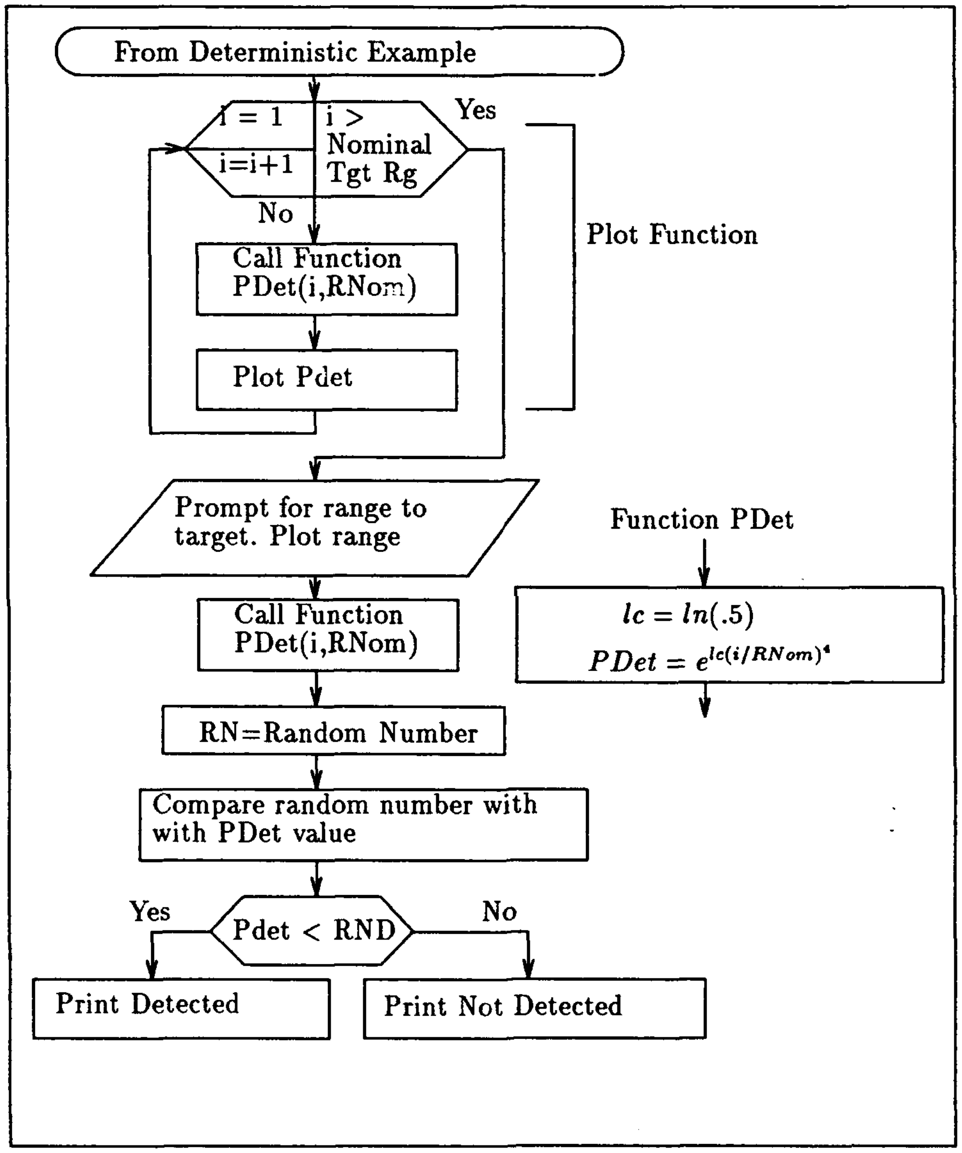
The *CEP demonstration* is the third and last example to be discussed. This example is based on a test firing of a simulated weapon where measurements of the impact points are made and collected. From the collected data, a point estimate of the circular error probable (CEP) and mean impact point is calculated and displayed. The animation of the weapon test firing and a displayed narrative explaining how the CEP is used in simulations provides the basis for understanding simulated kills. Figure 17 illustrates the logic flow of this example.

The example begins by asking the student to enter the number of test firings to be performed. Once the student enters the sample size, the program enters into a recursive loop. With each pass



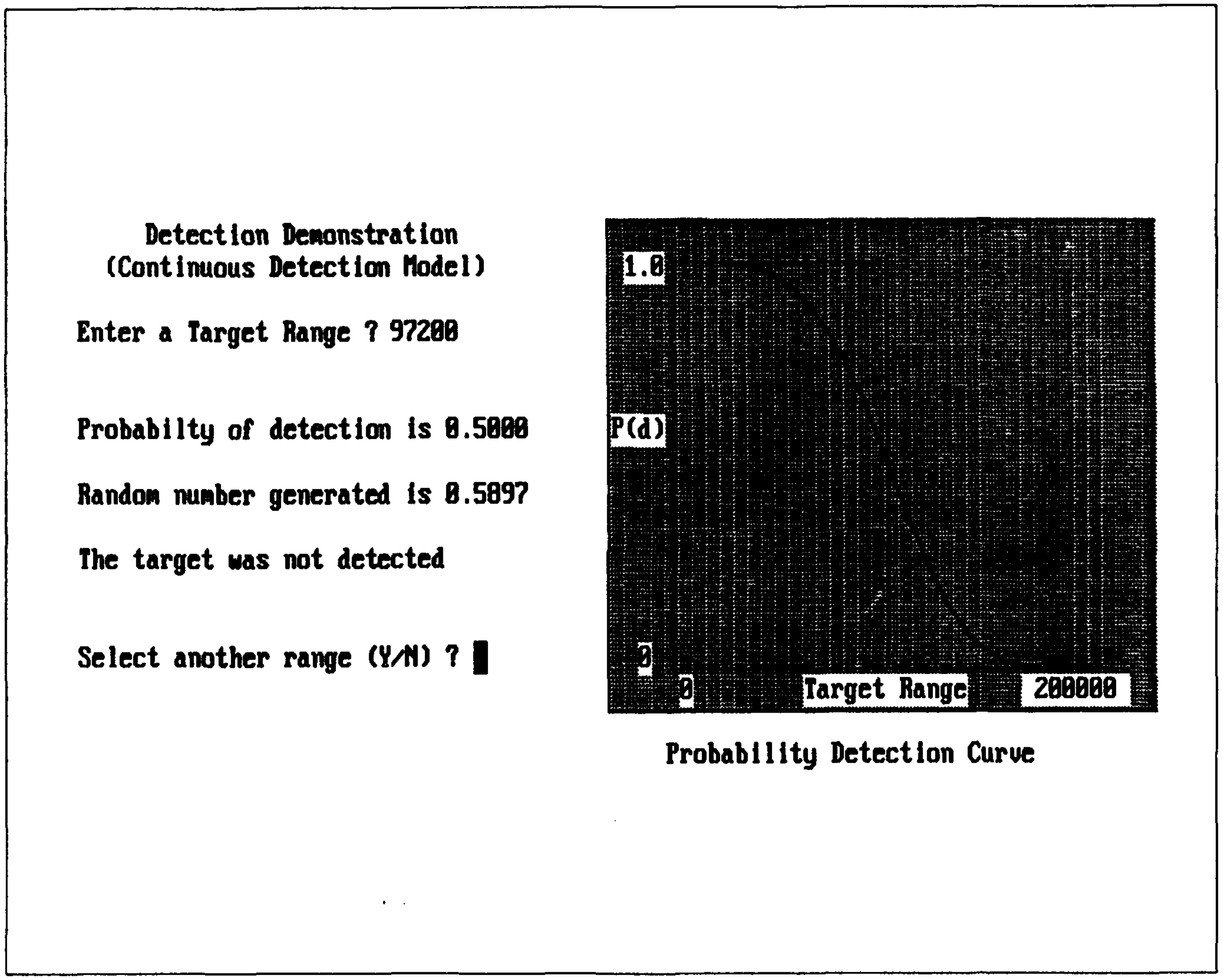
```
          Detection Demonstration
           (Cookie Cutter Method)

Maximum Range set for 97200

Enter range of target (Ft): ? 97000


Target proabillity of detection is 1.0

Target was detected

Select another range (Y/N): ?
```

Figure 14. Deterministic Example Screen Display



From Deterministic Example
i = 1
i=i+1
i > Nominal Tgt Rg
Yes
No
Plot Function
Call Function PDet(i,RNom)
Plot Pdet
Prompt for range to target. Plot range
Function PDet
$lc = ln(.5)$
$PDet = e^{lc(i/RNom)^4}$
Call Function PDet(i,RNom)
RN=Random Number
Compare random number with with PDet value
Pdet < RND
Yes
No
Print Detected
Print Not Detected

Figure 15. PDETDEMO Subroutine (Stochastic)



```
     Detection Demonstration
   (Continuous Detection Model)

Enter a Target Range ? 97200


Probabilty of detection is 0.5000

Random number generated is 0.5897

The target was not detected


Select another range (Y/N) ?
```

Figure 16. Stochastic Example Screen Display

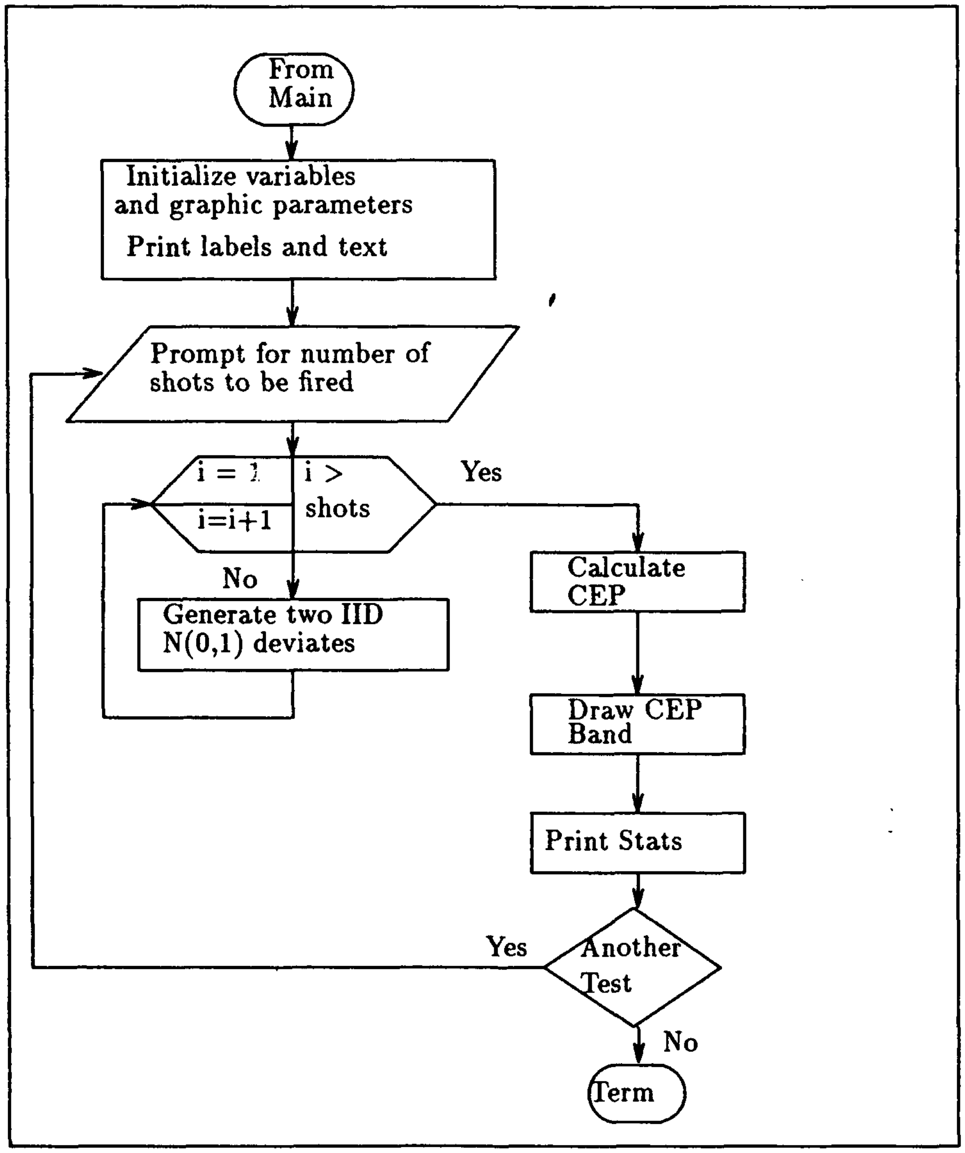

From Main
Initialize variables and graphic parameters
Print labels and text
Prompt for number of shots to be fired
i = 1
i=i+1
i > shots
Yes
No
Generate two IID N(0,1) deviates
Calculate CEP
Draw CEP Band
Print Stats
Another Test
Yes
No
Term

Figure 17. CEPDEMO Logic Flow

through the loop, the polar method is used to simulate normally distributed impact points. The following steps outline the process of generating the random impact points (24:491).

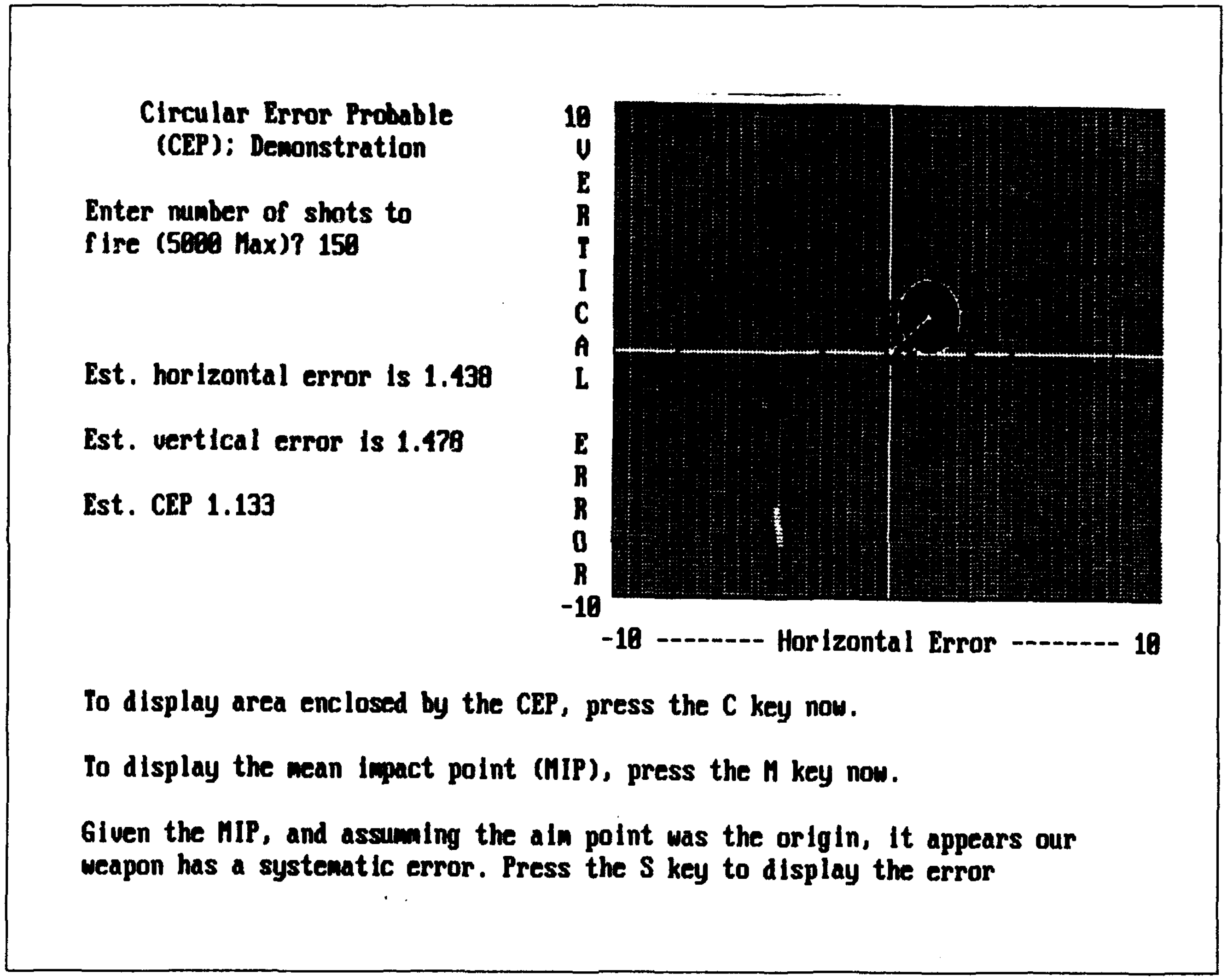
1. Generate two independent identically distributed (IID) uniform random numbers from using the system RNG. Say U_1 and U_2
2. As an intermediate step, set temporary values $V_i = 2 * U_i$ for $i = 1,2$ and let $W = V_1^2 + V_2^2$.
3. If W is greater than 1, start over. A value greater than one would cause a negative argument under the radical in the following step
4. Let $Y_{i,j} = V_i \mu \sigma_j \sqrt{(-2lnW)/W}$, where μ is the bias and σ_i for $i = 1,2$ are the cross and down range errors. In this CEP demonstration, $\sigma_1 = \sigma_2$.

$Y_{i,1}$ and $Y_{i,2}$ are now (IID) Normal(0,1) random variates of which only one is used. The process is then repeated a second time and another coordinate generated. The results produces a set of random X and Y coordinates for a single impact point.

Once the number of sample firings is completed, the CEP is calculated. The output statistics include the estimated mean impact point and the value of the CEP. Figure 18 shows an example of the display screen presented to the student.

4.4 Scenario Creation

The ATACS scenario is somewhat rigid. That is, the scenario is based on one or two Blue aircraft orbiting around a combat air patrol station. The Red forces which are comprised of one or two aircraft attempt to attack the Blue force's airbase. Once the Red forces are detected, the Blue aircraft are directed to intercept and destroy the intruders. This scenario can not be changed by the student. However, several of the initial conditions such as the location of the airbase, CAP, and intruders may be altered by the student. The student may also change the aircraft's altitude, velocities, sensor performance, fuel, and number of missiles carried.



```
Circular Error Probable
 (CEP): Demonstration

Enter number of shots to
fire (5000 Max)? 150


Est. horizontal error is 1.430

Est. vertical error is 1.470

Est. CEP 1.133

To display area enclosed by the CEP, press the C key now.

To display the mean impact point (MIP), press the M key now.

Given the MIP, and assumming the aim point was the origin, it appears our
weapon has a systematic error. Press the S key to display the error
```

Figure 18. CEP Example Screen Display

To load or create a scenario, the student selects *Run Demonstration*. After making this initial selection, the program displays the simulation menu. The student may select *Express Load and Go* to load and execute available scenario files or *Load and Edit Scenario* to edit a scenario file. In either case, the student has the option to view a synopsis of the scenario. The synopsis introduces the scenario to the student as well as explains what they will see during the simulation.

When editing a scenario, the first display in this sequence prompts the student with an option to either load the system default scenario or modify an existing scenario file. If the default scenario is selected, the ATACS will read in the master scenario and perform the necessary preprocessing. If the alternative is selected, the student is asked for the name of the scenario file to be modified. This latter option permits the student to build a library of scenario files if desired.

A secondary benefit of the scenario editor is the enhanced visibility of the model. The editor permits the student to view and alter all the scenario components. Through observation of the simulation, the student can determine how an altered scenario component affects the simulation outcome.

4.5 Flight Performance Preprocessor

The flight performance preprocessor, subroutine GETDATA, initializes the remaining data elements, particularly aircraft flight performance elements. GETDATA uses these elements to calculate the aircraft performance parameters.

To capture as much of the physical realism as possible, the following parameters are determined.

- Atmospheric density
- Dynamic pressure at velocity
- Speed of sound

- Mach number
- Dynamic pressure at speed of sound
- Coefficient of lift
- Drag coefficient
- Thrust in military power and afterburner
- Fuel consumption in both military power and afterburner
- Sustain normal force for sweeping turns
- Radius of turn and turn rate.

Some of the above values are calculated directly while others such as coefficient of lift, drag coefficient, thrust, and fuel consumption are interpolated from data arrays.

Once GETDATA has completed reading in the performance data elements, it enters a pre-processing phase. It first calculates an aircraft's mach number based on the velocity entered by the student. This calculation requires a call to function SPDS. Function SPDS then calculates the speed of sound at the scenario specified altitude. Mach is then calculated by dividing the aircraft's velocity by the speed of sound. GETDATA then calculates two pressure parameters–dynamic pressure at the specified velocity and the dynamic pressure at the speed of sound.

Aircraft performance factors are determined by the process of double interpolation. A generalized flow of this process is shown below.

$$Slope1 = Mach - MachVal_i/(MachVal_{i+1} - MachVal_i) \tag{6}$$

$$Slope2 = Alt - AltVal_j/(AltVal_{j+1} - AltVal_j) \tag{7}$$

$$\overline{Slope2} = 1 - Slope2 \tag{8}$$

$$Temp1 = \overline{Slope2} * Val_{i,j} + Slope2 * Val_{i,j+1} \quad (9)$$

$$Temp2 = \overline{Slope2} * Val_{i+1,j} + Slope2 * Val_{i+1,j+1} \quad (10)$$

$$Result = Temp1 + Slope1 * (Temp2 - Temp1) \quad (11)$$

MachVal and *AltVal* in Eqs (6) and (7) are the marginal row and column values of the performance data tables. *Mach* and *Alt* in the same equations are the student supplied values based on velocity and altitude. *Val* terms in Eqs (9) and (10) are the table values from the aircraft performance array, such as fuel consumption. Each of the *Temp* values in Eqs (9) and (10) are values which lie between two sequential values in the array occupying adjacent columns. Then the result, Eq (11) is simply a linear interpolation between these two intermediate *Temp* values.

This sequence is performed to estimate the aircraft's thrust, drag, and fuel consumption in afterburner and military power throttle settings. A value for each of these parameters is estimated for both velocities specified for each aircraft.

Sustained Turn Performance: This performance measure is defined at the point where thrust equals drag in a level turn at a specified power setting. The aircraft sustained turning performance may be lift limited, structural limited, or thrust limited (4). Since velocity is limited to two values and altitude is held constant, thrust is then a constant. This leaves lift is the only variable that remains to be calculated. Once lift is determined, then the sustained normal force is calculated by

$$SusG = MAX(1, DesireLift) * U * Mach$$

where *DesireLift* is the lift obtainable given current velocity, altitude, and thrust. However, to sustain flight, this value is bound at the lower end to one gravitational constant (G). *U* is the dynamic pressure at the speed of sound, and Mach is the mach number. SusG is then used to determine the RadTurn (radius of turn). RadTurn is calculated as follows

$$RadTurn = V^2/(G * Min(3, SusG))$$

where V is current velocity and G is the gravitational constant. In the denominator, SusG is bounded at the upper end at three Gs, as a human factors consideration (4).

The calculation of the above aircraft performance factors is accomplished as part of the initialization of the combat model. Initialization of icons is also performed at this time. These icons are used in the animation of the aircraft, missiles, and airbase.

4.6 Animation

Without a doubt, graphics provides a sense of entertainment for the student using ATACS. However, the intended purpose of the graphics is to provide a visual interface between the dynamic simulation and the student.

Most often, an object such as an aircraft or missile in a high resolution combat model is treated as a point mass. This point mass has an associated coordinating vector which defines its relative location. Changes in the vector represents the transition of the point mass. ATACS uses the same convention with the addition of real-time graphics. Once the graphic features were introduced to ATACS, the point mass not only had a set of coordinating points defined by the vector, it also had a two dimensional shape which had an orientation governed by the aircraft's heading. Providing that shape, and orientating the shape to the aircraft's heading is the subject of this section.

First, a short discussion of the layout of the graphics screen. At initialization, the graphics screen represents an airspace of 430 miles across the screen along the X axis, and 300 miles down the screen along the Y axis. A Cartesian coordinate system is used with the origin at the center of the screen. A zero heading—North–points to the center of the viewers right hand side of the screen.

Figure 19 shows the icons used for both Blue and Red aircraft, missiles, and the airbase. The same aircraft icon is used for both the Blue and Red forces; only their color distinguishes them from one another. The creation of the icon is accomplished by orientating the image at the screen origin. The coordinate points which define the beginning and ending points of each line segment are initialized and stored at startup. For each change in position and heading, the icon line segments are translated from the origin to the current relative position of the point mass and rotated to the aircraft's heading.

The sensor fan which protrudes from the nose of the aircraft follows a slightly different process. For reasons of appearance, the aircraft image size was not constrained to the scale of the map. However, the range of the radar had to be scaled to the graphics map since it represented an actual range. For example, if the student selects a radar fan with a maximum range of 15 miles, the fan displayed on the map will extend 15 miles in front of the aircraft. Had the icon been scaled to the graphics map; the icon would have appeared as a dot. Since a dot provides less character than the icon shown in Figure 19, drawing the icon out of scale was justified.

The following steps describe the process of initiating and animating the icon image.

1. Initialize icon coordinates $X_2, \ldots, X_{13}$ and $Y_2, \ldots, Y_{13}$ and $X_{i,14}, \ldots, X_{i,17}$ and $Y_{i,14}, \ldots, Y_{i,17}$ where i=1 to the number of aircraft.
2. For each change in position and heading do the following.
   (a) Erase the old image of icon with background color.
   (b) For each aircraft, translate and rotate all coordinate points.

$$xx_i = (X_i * \cos(ACHead) - Y_i * \sin(ACHead)) + AC_x$$

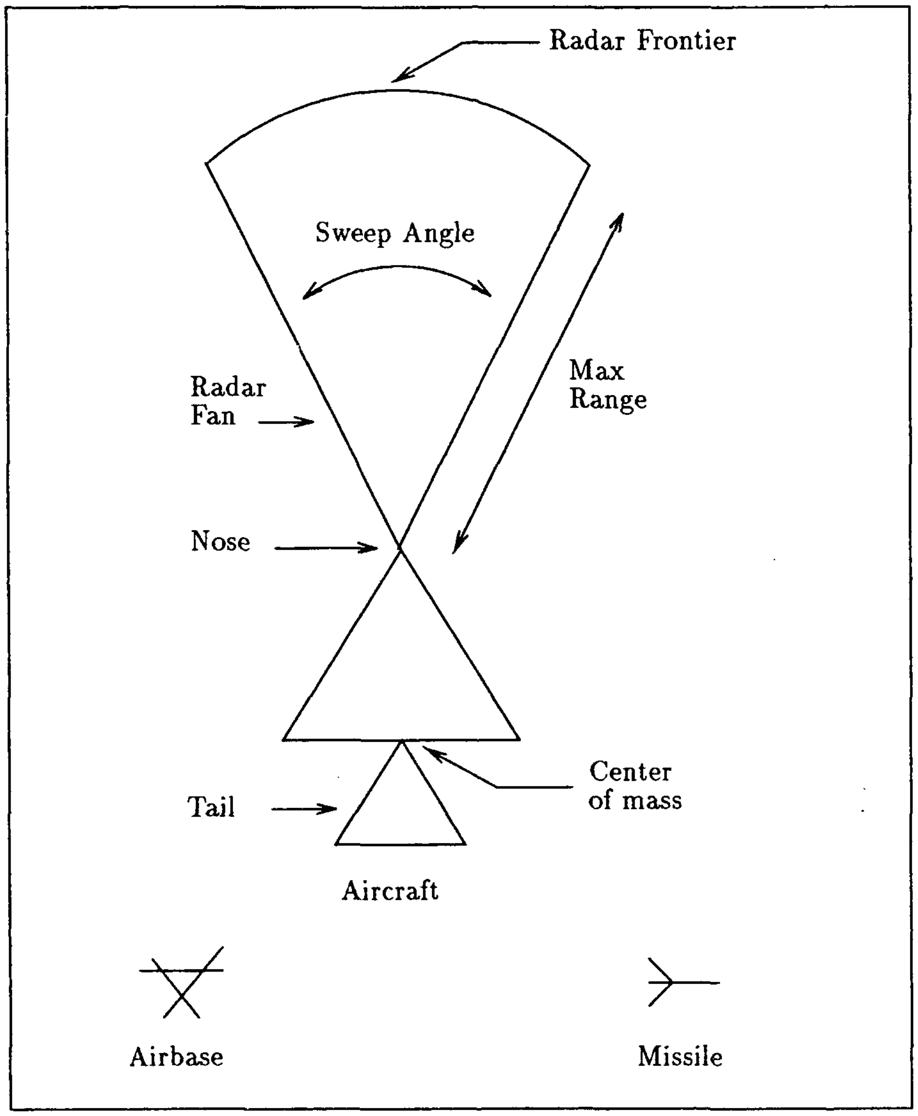


Figure 19. ATACS Icons

$$yy_i = (X_i * \sin(ACHead) - Y_i * \cos(ACHead)) + AC_y$$

where AC_x and AC_y are the X and Y coordinates of the point mass.

(c) Assign a color depending on known or unknown sidedness.

(d) Draw each line segment using BASIC's LINE function.

(e) Capture coordinates. By capturing the old coordinates, they can be used for erasing the image during next pass.

Radar Fan. The displaying of the radar fan is a little more complex. The arc which represents the frontier of the maximum range is drawn using BASIC's CIRCLE function. The function's argument consist of a center point (the nose of the aircraft), the radius (the radars maximum range), a radial measurement indicating the relative starting point of the circle, and a terminating point. Because of the convention used by BASIC to define the start and stop points of the arc, the frontier is drawn in two arcs, left and right of the aircraft's nose. Additionally, checks to see if values greater than 2π or less than zero are performed before they are entered as members of the CIRCLE argument. In these cases, 2π is either subtracted or added to the start or stop value.

Missiles. Launched missiles are also mobile entities that must be displayed. With the exception of fewer line segments to draw and no radar fan, the animation process is quite similar to that of the aircraft. If it is determined the missile has killed the target, the missile and the target images are removed from the display.

Airbase, Radar, and Grid There are three images displayed which are not mobile. These are the airbase, the tactical surveillance radar maximum range frontier, and a grid overlay. The airbase is displayed by drawing three line segments to create the image of runways. The tactical surveillance radar frontier is created by drawing a circle centered on the airbase with radius equal to the maximum range of the radar. The grid overlay is a pattern of dots with an interspacing scaled to 15 miles. The purpose of the overlay is to provide scaled reference system to the viewer.

Additional graphic features provided by the ATACS are the Zoom-In and Zoom-Out functions. The zoom in function provides the option to zoom in on the center of the screen which magnifies the images within the view. Of course, the zoom out function performs the opposite operation. These functions are particularly useful when the aircraft are in close proximity of each other and closer observation is desired. If the initial conditions position the attacking aircraft well outside the screen view, the zoom out function provides the option to reduce the scale, which expands the area being viewed.

4.7 Model's Principal Architecture

The model basically consists of an initialization step, a main recursive loop, and a termination procedure. The initialization step includes the data-gathering and initiating of system variables. The data-gathering step includes importing and building the scenario data file as well as computing aircraft performance parameter as discussed in Section Section 4.5.

Following initialization, the program executes a recursive loop. For each pass through the loop, the program directs control to subprograms which perform such key processes as search, detect, steer, and shoot. The following outline describes the program flow that is illustrated in Figure 20.

1. Search for intruders. Check for detection of intruders. If intruders are present, assign targets and pursue. Otherwise, fly on station.
2. Determine aircraft's headings.
3. Compute course corrections.
4. If intruder is present, check firing conditions.
5. If firing conditions are met, fire missile.
6. If missile is fired, begin a quarter second time step within missile loop.

(a) Compute missile course correction.

(b) Check range to target.

- If target range is within one half the distance of missile advance in a quarter second, declare a target hit.

(c) Draw missile image.

(d) Advance missile.

7. Draw aircraft images.

The following discussion in this section focuses on the above tasks with the exception of the drawing the images.

4.7.1 Search. The process of *Search* in ATACS is really the process of trying to detect. Each pass through the main loop advances the simulation clock by one second. During each pass, the model determines if an intruder has come within sensor range of the blue forces.

Within each pass through the main loop, the subprogram DETECT is called. Within DETECT, the distances between the intruders and their intended target, as well as, the distances between intruders and interceptors are determined. The distance measurements are preformed by function DIST. The X and Y coordinates of the airbase location, and an intruder aircraft are passed as arguments to Dist. The distance is then computed based on the familiar distance formula

$$Dist = \sqrt{(X_2 - X_1) + (Y_2 - Y_1)}$$

If the distance between the base and an intruder is equal to or less than the tactical surveillance radar range, the intruder is detected, the posture of the airbase goes to *High*, the status of the blue forces changes to ***pursuit***, and the status of the intruding aircraft is(are) changed to *Detected*. Otherwise, blue forces continue to fly their CAP mission unaware of the approaching enemy aircraft.

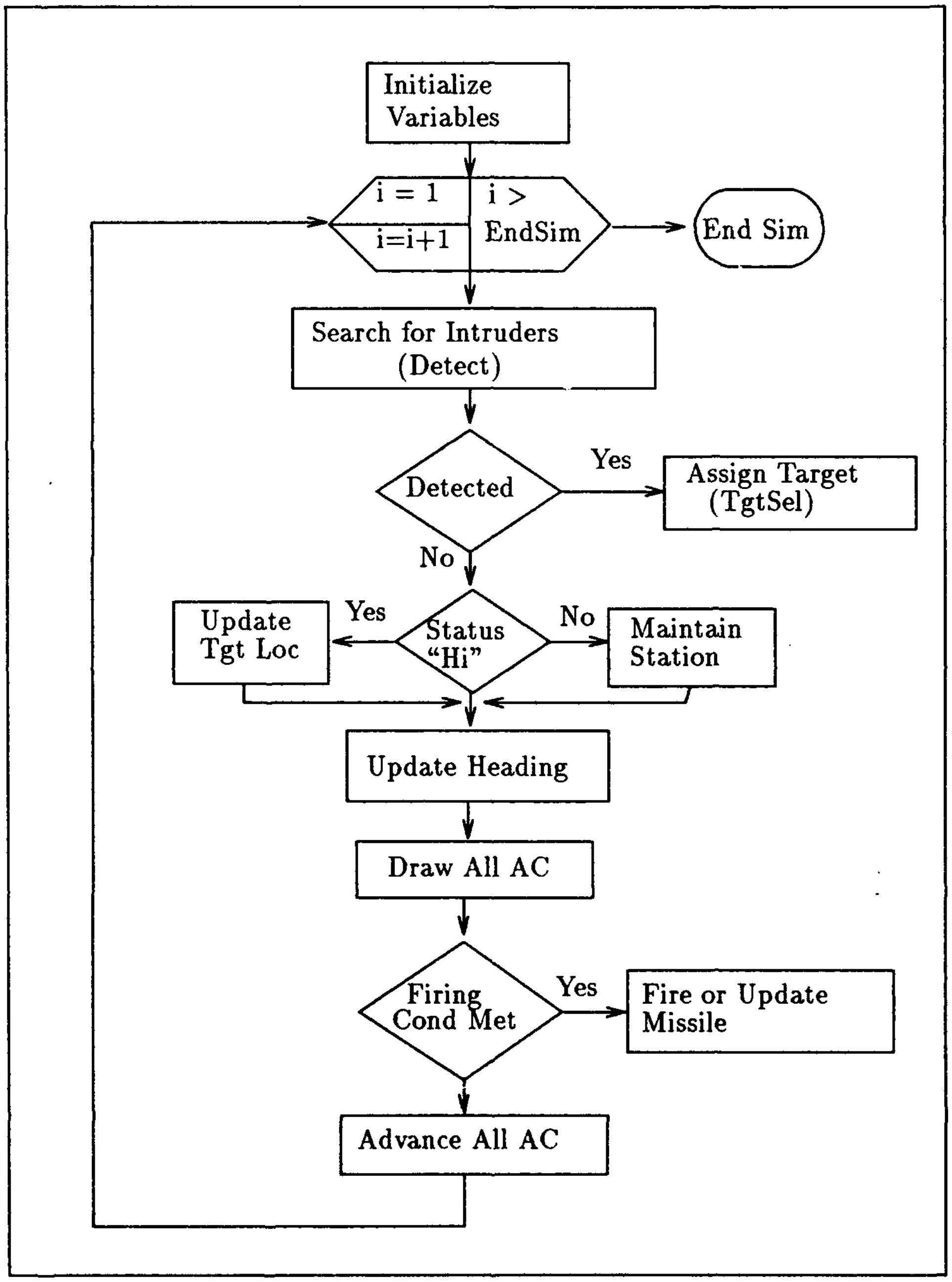


Figure 20. Main Control Loop of ATACS

Once an intruder has been detected, it becomes the target of the blue forces. If the scenario calls for two intruders, the second intruder will not be targeted until detected.

Targeting. Targeting is based on an internal *Rule of Engagement* (ROEs). This rule states that intruders will be targeted based on minimum range. That is, if Blue 1 and Blue 2 are in pursuit of a Red target and another Red intruder is detected, the Blue aircraft which is the closest to the first Red target remains engaged while the other Blue aircraft will break off and pursue the newly detected intruder. Once one of the Red targets has been eliminated, the newly freed Blue interceptor will be reassigned to the remaining Red target. The logic flow of the target assignment algorithm is shown in Fig 21.

4.7.2 Assignment of Headings. For each pass through the loop, an initial heading is computed for each aircraft by function MHEAD. Function MHEAD begins by finding the relative distance between the aircraft and its target in both X and Y directions. In this fashion, a coordinate system is established with the aircraft at the origin and the target in one of four quadrants. To determine which quadrant the target is located in, several checks are performed based on the position of the target relative to the aircraft.

Once the checks are complete, the heading is computed by taking the arctangent of the relative Y distance over the relative X distance.

The following describes the sequence of the final stage of establishing a heading.

```
If $Target_y > AC_y$ then (Target is ahead of aircraft)
        $Heading = \arctan(distance_y / distance_x) + \pi$
Else
        $Heading = \arctan(distance_y / distance_x)$
End If $Target_x < AC_x$ then (Target is left of aircraft)
        $Heading = \arctan(distance_y / distance_x) + \pi$
```

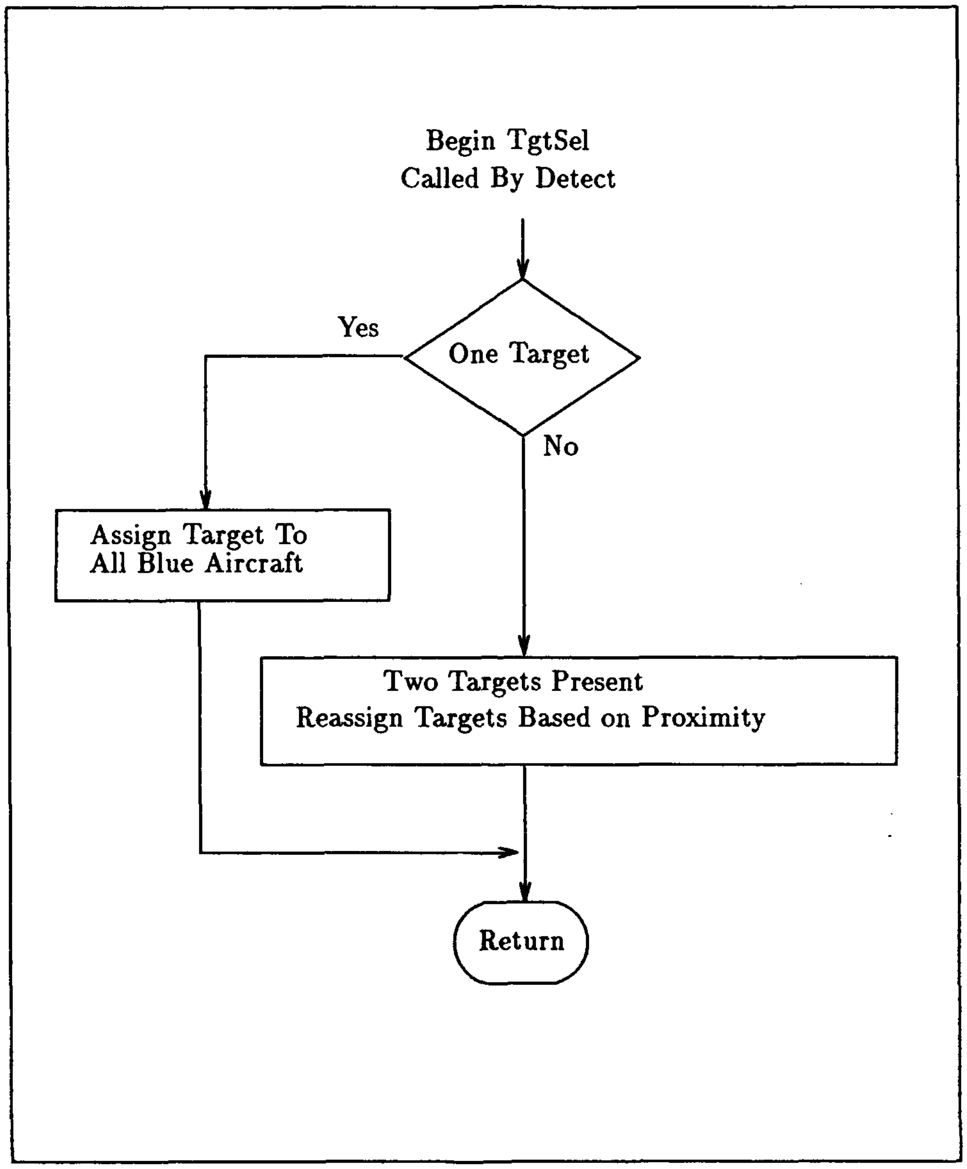

Begin TgtSel
Called By Detect
One Target
Yes
No
Assign Target To All Blue Aircraft
Two Targets Present
Reassign Targets Based on Proximity
Return

Figure 21. Target Assignment

```
Else
        $Heading = \arctan(distance_y / distance_x)$
End
```

The final step in assigning a heading is to verify the heading is within 0 and 2π. Following the verification, the function ends and control is returned to mainloop.

4.7.3 Track Angle. A common term found in high resolution aircraft combat models is the term ***track angle***. This term is defined as the angle between the aircraft's velocity vector and the line of site (LOS) vector. The velocity vector generally extends in the same direction as the heading. The LOS vector originates from the nose of the aircraft and terminates at the target. Figure 22 illustrates this definition.

ATACS makes use of the track angle concept in three sperate applications. These are: turn limiting, determining turn direction, and target detection.

Up to this point in the main control loop, a new heading has been computed, but none of the aircraft have moved during this cycle. The newly established heading points directly at the target. This is the heading the pilot wants to steer to. The old heading is the current direction the aircraft is pointing in. The difference between these two directions is the track angle.

The track angle is computed using the inside angle, hence angle lies between 0 and π radians. Checks are performed to verify the angle lies in this range. An example is provided to illustrate this procedure. Say the old heading was 30 degrees, and a new heading of 300 degrees was computed. The difference between these two directions is 270 degrees, a value greater than π. The angle of 270 degrees is clearly not the inside angle. By subtracting this angle from 2π, the desired inside angle of 90 degrees is obtained.

Turn Limiting. During the scenario input, the student selects an altitude, and two operating velocities. The altitude and velocities determine the capable radius of turn for the aircraft. From

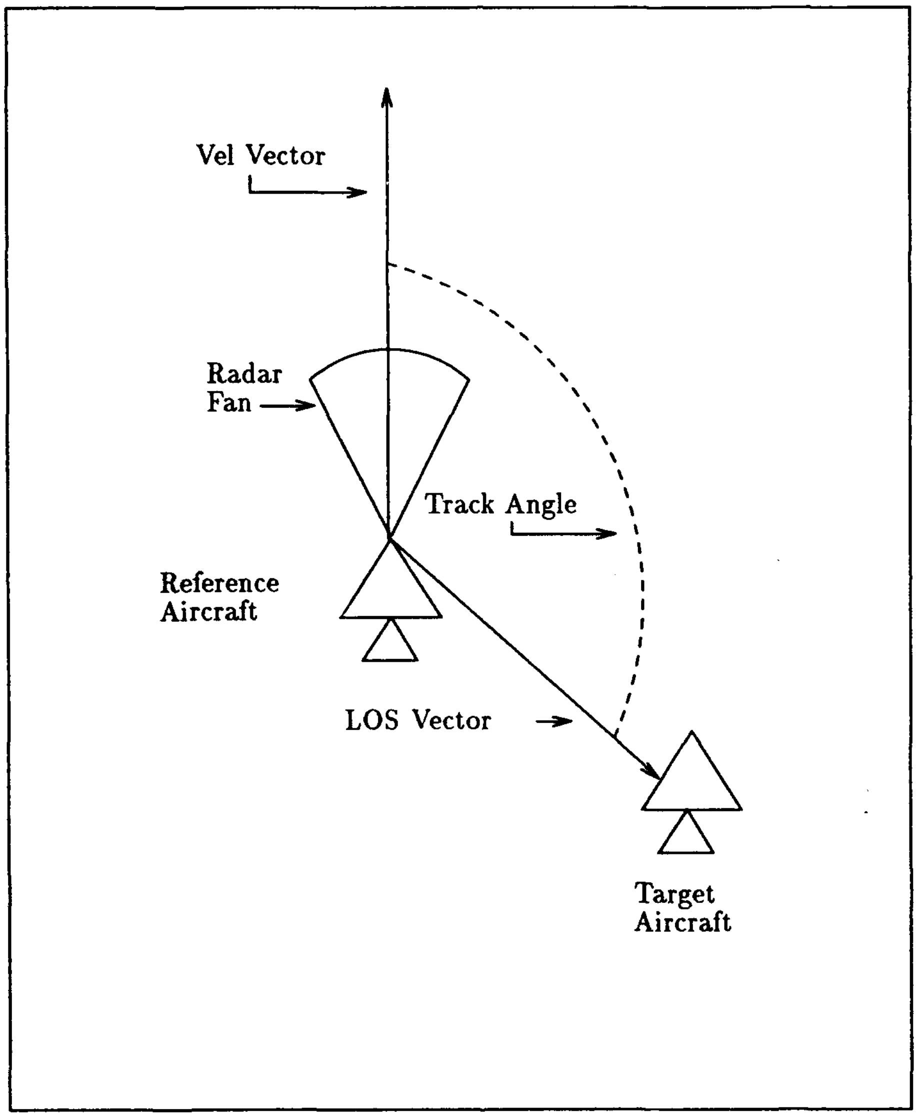


Figure 22. Track Angle

the radius of turn, a turn rate is computed and is expressed as radians per second. Now, if the tracking angle is 90 degrees (1.57 radians) as in the above example, and the turn rate is only 0.13 radians per second, its obvious the complete turn can not occur in the one second cycle. Therefore, the turn will need to be constrained to the turn rate. During each cycle, the updated track angle is compared to the turn rate. If the magnitude of the track angle drops within the constraint of the aircraft's turn rate, the aircraft will turn an amount equal the magnitude of the track angle. If the track angle is greater than the turn rate, the turn remains constrained.

Direction of Turn. Once an intruder has been detected and a new heading assigned, the simulated pilot must decide whether a left or right turn is the best. For example, say at the time the new heading is issued, the aircraft is flying a 180 degree due south heading. A new heading of 90 degrees due east is issued by the ground controller. The best turn, if unobstructed, would be to turn to the left. For ATACS to decide this, it must sample a left turn (much like looking to the left) by adding one unit of turn rate to the old heading. If this action reduces the track angle (the objective of the optimal turn), then a left turn will be performed. Otherwise, a right turn is initiated.

Track Angle and Target Detection.

The track angle is also used for target detection by the on-board radar. However, in this application, the direction of the LOS vector is determined. Once the direction of the LOS vector is known, then simply subtracting the LOS direction from the current heading yields the track angle. The track angle is then compared to the aircraft's on-board radar sweep angle. A target will be detected if it is within the sweep angle and the maximum range of the radar.

4.7.4 Missile Launch, Track, and Aircraft Kill. In ATACS, simulating the flight of a guided missile to its intended target is very much like simulating the flight of an aircraft to its intended target. However, there are two distinct difference. First, the simulated time base for the missile is every 0.25 seconds as compared to one second for the aircraft. Second, target detection and target selection is known. This decreases the number of required operations in the performance of the simulation. The entire flow of the missile launch, flight, and destroy sequence is illustrated in Figure 23

Launch Conditions and Missile Initialization Within MainLoop, the opportunity for a missile launch is checked with each aircraft. Before an aircraft can launch a missile, three conditions must be met. First, a missile must be available. Second, a target must be present. Third, the target must be within optical range of the shooting aircraft. Once the initial conditions for firing are met, subprogram MISSILE is called. Upon entering into the subprogram, the missile is initialized. The coordinates of the target are passed to the missile, and the missile's rocket motor timer is set.

Missile Flight. Every simulated quarter second the missile is advanced towards its intended target. Once the missile is advanced, its new location is compared to that of the target. When the missile's range to target is within one half of the distance it travels in a quarter second, the target is declared hit, and a call to subprogram PKILL is made. Otherwise, when four quarter seconds have passed, a return to mainloop is executed.

Target Kill. In subprogram PKILL, a random number is drawn and compared to the missile's probability of kill (Pk). If the random number value is less than or equal to the Pk of the missile, subprogram DESTROY is called to destroy the target. If the random number is greater than the Pk, the target is missed. If the target is missed and the firing conditions for the shooter still exist, the shooter will launch another missile.

Destroying the Target. Destroying the target in the simulation means destroying the entity. This operation is performed by subprogram DESTROY. DESTROY zeros out any numerical attributes of the destroyed target, changes status attributes of the target to reflect its current status of "Killed in Action (KIA)", and frees the shooter to engage another target or return to the patrol station. As a visual indication of the targets destruction, a call to subprogram EXPLODE is performed to animate the explosion of the target aircraft.

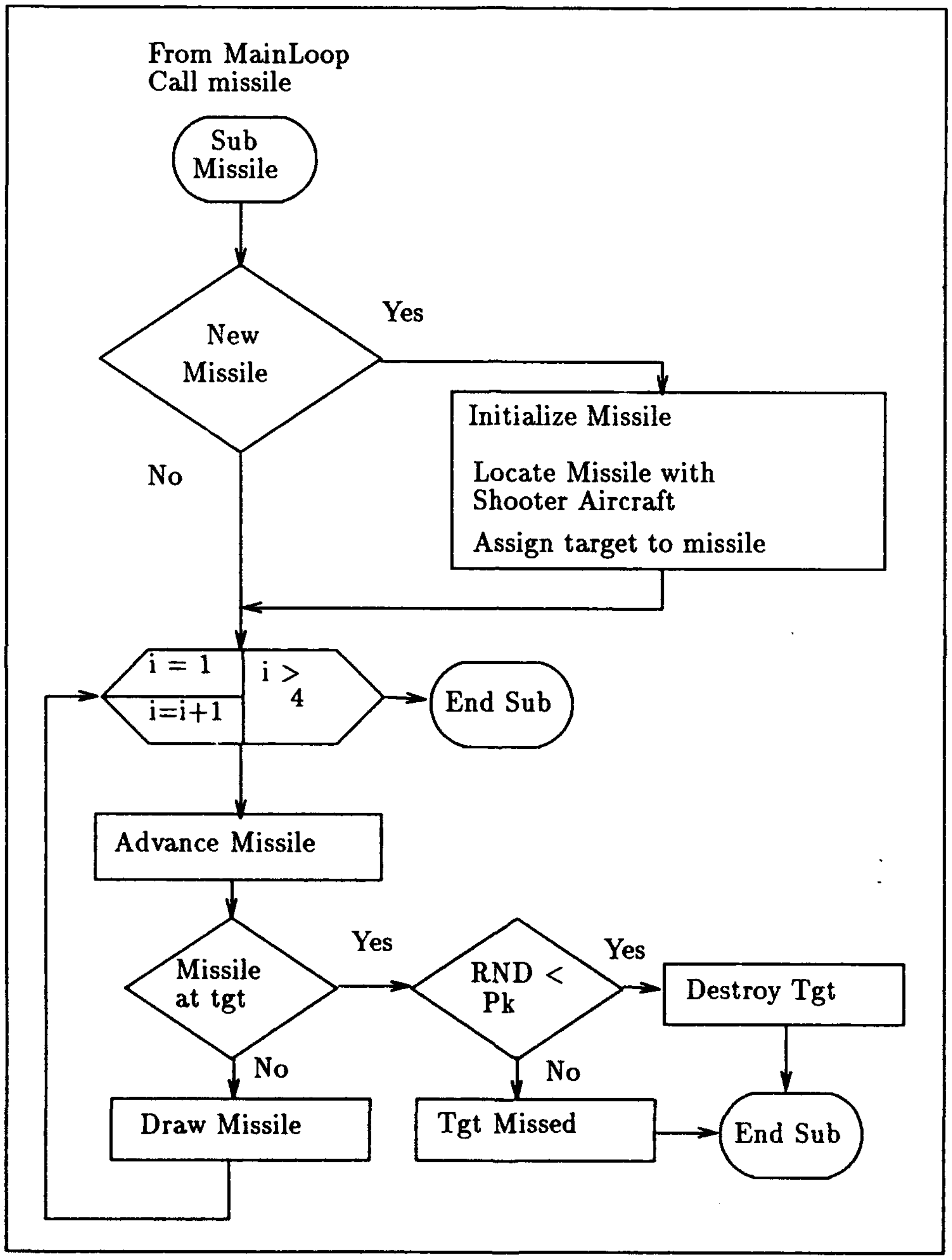


Figure 23. Missile Flow

Animation of the Missile. The drawing of the missile is performed by subprogram MSL-DRAW. The procedure to draw the missile is identical to the procedures described in Section 4.6. The only exception to this is the fewer number of line segments required to draw the missile.

Missile Termination. If either the target aircraft is destroyed or the missile's rocket motor burns out, the missile is terminated and the screen is refreshed. Missile termination is performed by subprogram MSLTERM. The termination is accomplished by simply advancing the missile counter belonging to the shooter. As a result of program structure, this procedure, in effect, cuts the missile out of the simulation.

4.7.5 Aircraft Advance. This process begins by a call to subprogram ADV from the main loop. The following outlines the steps taken in ADV. These steps are also illustrated in a flow diagram shown in Fig 24.

1. Select appropriate velocity based on status
2. Compute the turn rate and fuel consumption based on velocity
3. Find the unit X and Y components of the new heading.
4. Compute new coordinates
5. Check fuel status

Fuel consumption is based on the fuel consumption rates computed in the GETDATA subprogram. With each cycle, the status of the aircraft is determined. The status determines which velocity the aircraft will operate at and the rate fuel is consumed. Since the consumption rates are in pounds per second, and the time step based on the second, a unit consumption rate can be added directly to the current consumption total. Following the sum, a new fuel-remaining percentage is computed. In the unfortunate event the fuel-remaining percentage drops below zero, subprogram DESTROY is called and the aircraft is removed from the simulation.

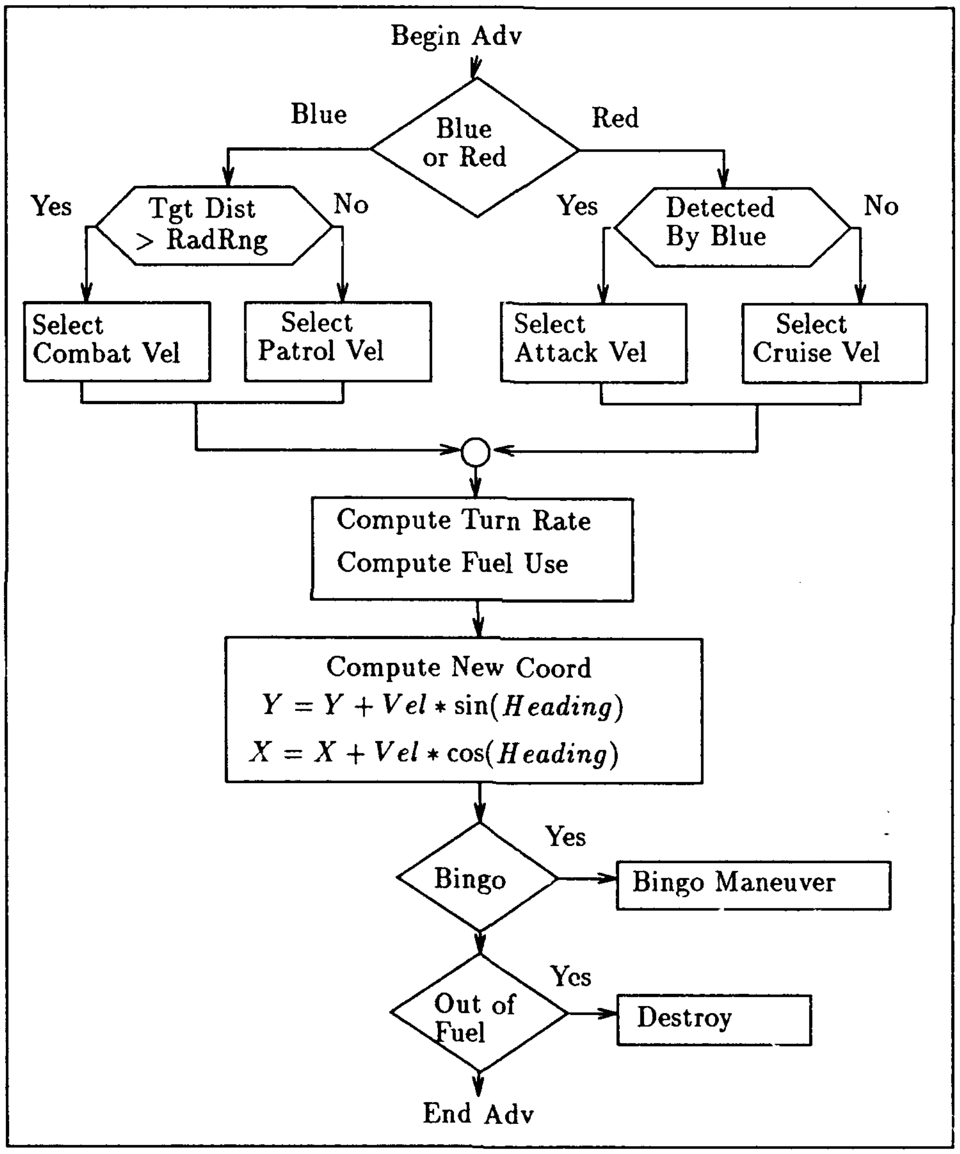


Figure 24. Subprogram ADV

For Blue aircraft, if the fuel level drops below the student's defined "Bingo" level, subprogram BING is called to perform a "Bingo" maneuver. If the aircraft does not have a target within optical range, subprogram BING removes targets from the aircraft's target list and vectors the aircraft back to its base.

4.7.6 Attribute Screen. ATACS provides a feature unlike any other provided by aircraft combat models. This feature is an option for the student to view inside the simulation while it is running.

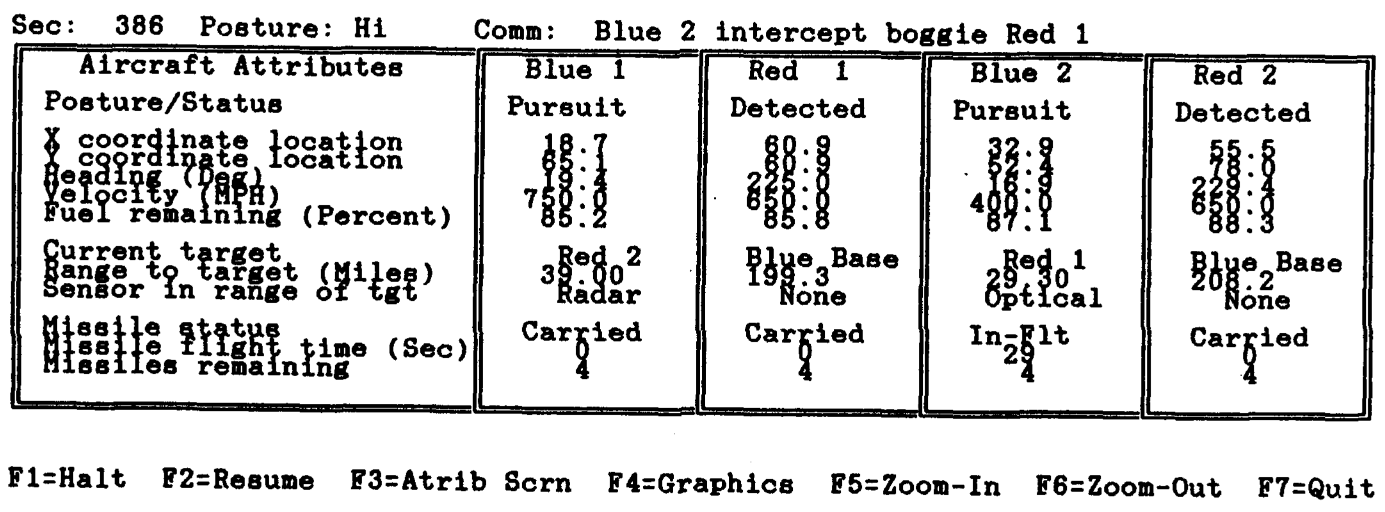
The *Attribute Screen* option provides the student the option to view the changing status of the simulation in numerical and conditional terms. An image of the attribute screen display is shown in Figure 25. As seen by the many attributes listed here, the student is able to look into simulation and observe, verify, and validate the many processes occurring during the simulation's operation.

To display the attribute screen, the student selects the option, function key F3, from the options displayed at the bottom of the screen. Subprogram FRAME is called to create the borders around the displayed data. The values of the aircraft attributes are then presented. Since the attribute values change faster than most observers can absorb, they are updated on only every tenth pass through the main loop.

4.7.7 Additional Features. The following discussion addresses additional features provided by ATACS.

Top Line Status Board. At the top of both the animation screen and the Attribute Screen, the current simulated time, the posture of the airbase (Low or High), and communication between the airbase and the Blue aircraft is displayed.

Bottom Line Menu. At the bottom of both display screens, a menu of options are displayed. These options include:



```
Sec:  386  Posture: Hi      Comm:  Blue 2 intercept boggie Red 1
    Aircraft Attributes           Blue 1     Red  1      Blue 2     Red 2
  Posture/Status                 Pursuit    Detected    Pursuit    Detected

  X coordinate location           18.7       60.9        32.9       55.5
  Y coordinate location           65.1       60.9        52.4       78.0
  Heading (Deg)                   19.4      225.0        16.9      229.4
  Velocity (MPH)                 750.0      650.0       400.0      650.0
  Fuel remaining (Percent)        85.2       85.8        87.1       88.3

  Current target                  Red 2     Blue Base    Red 1     Blue Base
  Range to target (Miles)        39.00     199.3        29.30     208.2
  Sensor in range of tgt          Radar      None       Optical     None

  Missile status                 Carried    Carried     In-Flt     Carried
  Missile flight time (Sec)         0          0          29          0
  Missiles remaining                4          4           4          4

F1=Halt  F2=Resume  F3=Atrib Scrn  F4=Graphics  F5=Zoom-In  F6=Zoom-Out  F7=Quit
```

Figure 25. Attribute Screen

- F1 Halt: halt the simulation
- F2 Resume: resume the simulation
- F3 Atrib Scrn: display the attribute screen
- F4 Graphics: display the graphics screen
- F5 Zoom-In: zoom in on center of the screen
- F6 Zoom-Out: zoom-out from the center of the screen
- F7 Quit: quit the simulation

Monitor Selection. ATACS is initially setup for a video graphics array graphics based system. However, this may present a hardware compatibility problem for some students. ATACS attempts to adjust for such a problem.

ATACS auto[illegible]tically determines the type of monitor present. Upon entering this process during initialization, a test of the BASIC graphics function PSET is conducted. Failure of the function indicates to ATACS the program is not installed on a VGA compatible system. In this event, ATACS reconfigures itself to be compatible with the enhanced graphics adaptor (EGA) graphics display. If this fails, ATACS informs the student of the hardware incompatibility.

4.8 Output

ATACS can generate up to four types of reports. These are the ***Reflected Report***, the ***Detailed Summary Report***, the ***Aircraft Performance Report*** and the ***Significant Event Summary Report***.

To generate the reflected report, the scenario and aircraft performance data is retrieved and formatted for display.

The data for the detailed summary report is generated at each pass through the loop. During each pass, the status and values of aircraft attributes are written to a file to be later retrieved and displayed.

The aircraft performance factors report is generated during the calculation of these factors. Once the factors are calculated, their values are formatted for display.

When any significant event such as an intruder being detected, missile launch, or kill occurs, simulation time and participants are recorded for later review in the significant event report.

4.9 Model Limitations

The ATACS combat demonstration is a combat model employing the simplest techniques to simulate specific phenomena. In an attempt to allow the simulation to operate while displaying graphics, some limitations on data accessing and reality had to be imposed. Altitude, although selectable by the student, is constrained to a single horizontal plane. Velocity is similarly limited to two constant values: military power and afterburner. Even though turn rate is considered, roll rate and pitch was not.

Detection of targets make exclusive use of the cookie cutter method. Although this is an acceptable method, it lacks the element of chance.

The missiles in ATACS are simple point masses directed to move to a location specified by the missiles assigned target. No missile performance factors akin to those of the aircraft are considered in ATACS. No specific missile guidance such as infrared or radar is modeled in the ATACS. No proximity fuzing or warhead dynamic fragmentation patterns are modeled. Although these limitations are significant, their impact is minimal given the intentional application of ATACS.

4.10 Summary

This section has presented the construction, operation, and use of ATACS' many features. The discussion was organized into the three major areas of student menus, graphics generation, and the actual combat model. Discussion of the logic flow of several of the principal ATACS operations was presented. Where needed, pictorial illustrations were provided to enhance the discussion. The final sections addressed the limitations of the air combat demonstration and the generation of the output products.

V. Conclusions

5.1 Summary

The objective of this thesis was to develop a simple and portable air-to-air, few-on-few combat simulation which demonstrates the fundamental concepts of air combat, combat modeling, and simulation in a form which supports student learning of model design.

This objective was met. The animated tutor for air combat simulation (ATACS) fulfills this objective.

ATACS is an aid which complements the AFIT combat modeling course material through demonstration and provides the "seeing is believing" realism to key modeling processes.

ATACS is a portable system which can be operated with any IBM compatible desk top computer supporting an enhance graphics adapter or better and color monitor.

ATACS is a menu driven system which hosts several dynamic and animated examples of modeling processes used in simulating combat activities. These examples are designed to support student learning of the more difficult combat phenomena simulated in combat models.

ATACS is a demonstration of simulated air-to-air combat. Features within the simulation permit the student to view an animation of the simulation and monitor its continuously changing states. Both of these features work in unison to support concepts of model scenario, target detection, target selection, attack, and elimination. Additionally, both the animation and monitoring features enforce techniques used in model verification and validation.

ATACS is a sample of output products which demonstrate those found accompanying most large scale combat models. The output products reinforces student learning of output analysis and model verification and validation.

Finally, ATACS is an example of proper program documentation. Contained in Appendix B is ATACS' user manual. This manual addresses the concerns a student may have such as: identifying

ATACS' hardware requirements, operating ATACS, and defining ATACS' many program files. This thesis document, in total, serves as an analyst/programmers manual for the student who desires to learn more through exploration and modification of the program's code. Within the program code lies another source of documentation as well. Over 900 comments have been included in ATACS' computer code. These comments serve to classify variables, explain usage, and provide warning of sensitive variable conventions. The extensive ATACS' documentation is yet one more pedagogical feature provided by this dynamic educational aid.

5.2 Development of ATACS

The development of ATACS began by identifying the needs of instructors and students of combat modeling.

The following paragraphs below summarize the methods used in the design and development of ATACS. These methods consisted primarily of research, identification and use of existing combat model computer algorithms, interviews, and use of operation research techniques.

1. The scenario had to demonstrate to some degree Air Force doctrine, rules of engagement, tactics, and decision rules. Development of the scenario was supported by interviews with rated Air Force officers and allied officers to gain insight into mission planning, electronic sensor use, maneuvering, and tactics employed during air–to–air combat.
2. The use of class notes from Military System Simulation, Military Systems Analysis, and Combat Modeling High Resolution and Aggregated combat modeling were the core sources of simulation techniques and applications of combat models.
3. A review of air–to–air combat models' operating manuals provided valuable insight into program structure and algorithms used to represent the combat. From these manuals, the necessary characteristics to sufficiently depict entities and their objectives were identified.

4. The methods used by air-to-air manual "board" wargames were studied as alternatives to simulate movement, search, and detect processes associated with air-to-air combat. It was hoped that these board games might reveal computationally efficient process algorithms. Unfortunately, they did not.

5. To establish what ATACS should emphasized, instructors and students were interviewed. The interviews focused on finding areas of combat modeling instructions that could be complemented by a computer based demonstration. The instructor's and students' responses provided the guidance in developing the animated examples found in ATACS.

5.3 Recommendation

Although this thesis met the original objective, there exists room for enhancement. This thesis established a baseline from which a future ATACS may be developed. More time and additional research could produce a tool that may extend beyond the classroom and into the offices of analysts and decisionmakers. ATACS could evolve into a modeling primer offering fundamental insight into the design, development, and application of high resolution combat models. This insight may permit a more thorough understanding of a model's scope and intended purpose. In-turn, a future ATACS could meet the challenge set forth by the General Accounting Office .

Specific recommendation to ATACS include the following:

- Dynamic Examples
  - Include such processes as event time step scheduling. Contrast both, fixed time step and event time step timing mechanisms.
  - Include an animated search example which points out the probabilistic nature of the search activity.

  - To enforce the many components that make up the probability measure, ***probability of kill***, an improved impact assessment example is recommended. This should point out that the probability of eliminating a target may include aircraft reliability, launch reliability, guidance reliability, fuzing reliability, and detonation reliability to mention a few.

- Air Combat Demonstration
  - Permit intruding Red forces to be reactive
  - Model aircraft maneuvers employing and demonstrating typical air tactics.
  - Permit a full three degrees of freedom (X,Y,Z) of the aircraft's point mass.
  - Find more efficient methods of determining aircraft performance parameters.
  - Use methods other than the cookie cutter to simulate detection.
  - Improve coordinate tatics between interceptors operators.
  - Enhance missile flight performance. Consider missile flight dynamics and various guidance and seeker techniques.

Several of these recommendations are features this author would have liked to introduce, but time constraints prohibited such development. This author believes the accomplishment of any one of these recommendations will greatly increase the instructional value of ATACS.

5.4 Conclusion

In conclusion, ATACS complements the instruction of combat modeling and possibly simulation in general. ATACS dynamically illustrates common processes found in combat models and employs these processes in an animated demonstration of simulated air combat. ATACS has the capability to immediately communicate complex multi-dimensional information about modeling processes common to combat models. ATACS certainly has the capability to enhance the effectiveness of instruction in combat models.

ATACS SIMTAX

TITLE: Animated Tutor for Air Combat Simulation

MODEL TYPE: Education and Training

PROPONENT:Air Force Institute of Technology Department of Operational Science

PURPOSE: To provide a demonstration of basic fundamentals of air-to-air combat modeling to support student learning of modeling combat.

DESCRIPTION:

- Domain: Land and Air
- Span: Local
- Force Composition: Blue and Red
- Scope of Conflict: Air-to-Air Conventional
- Mission Area: Combat Air Patrol
- Level of Detail of Processes and Entities: Processes are of sufficient detail to demonstrate the phenomena. Blue force entities maneuver and attack Red air targets. Red force entities attack a predefined ground target.

CONSTRUCTION:

- Human participation: Animated examples require student participation. Scenario creation may require student participation. No student participation required once the combat demonstration begins.
- Treatment of Randomness: Monte-Carlo

- Sidedness: Two sided, Blue forces reactive only

LIMITATIONS: Model is a first draft. Purpose of the model is to provided demonstration only. Some activities are treated as first order processes only.

INPUT: Optional

OUTPUT: Detail and summary reports

HARDWARE AND SOFTWARE:

- Computer: IBM AT Compatible
- Peripherals: VGA or EGA color monitor
- Language: Compiled Microsoft Basic 4.5
- Documentation: User manual

SECURITY CLASSIFICATION: Unclassified

GENERAL DATA:

- Date Implemented: March 1992
- Data Base: Provided.
- CPU time per Cycle: Based on the default scenario file, actual demonstration of air-to-air combat requires 1.5 minutes using an IBM AT equipped with a math coprocessor and operating at 12 MHz.
- Data Output Analysis: N/A
- Frequency of Use: Instructor Dependent
- Users: AFIT

ATACS User Manual

ATACS (animated tutor for air combat simulation) is a simple and portable, air-to-air, few-on-few combat simulation which demonstrates the basic fundamentals of air combat and key elements of combat modeling and simulation in a form which supports student learning of model design and use in DoD studies.

Since ATACS is an educational tool, much effort was devoted to designing a simple to use system requiring very little reference to a user manual. The primary purpose of this manual is to identify hardware requirements, system files, introduce you to ATACS, and explain menu options and screen displays in greater than that provided by the ATACS program.

Hardware Requirements

ATACS will operate on any IBM compatible XT or AT machine with an enhance graphics adaptor (EGA) or with video graphics array (VGA)

graphics capability. Although ATACS is compatible with an XT, it is not recommended due to the slower processing speed of an XT. Although a math coprocessor is not required it is recommended.

ATACS is provided on a 5 1/4 low density disk. To load ATACS on to your hard disk simply copy all files over to the hard disk. While using ATACS, you have the option of creating new scenario files. Additionally, ATACS must access data files and output files during the combat demonstration. Due to this type of input/output operation, it is recommended that you create a subdirectory for ATACS to be loaded in to.

ATACS Files

The files which makeup ATACS are listed below.

- **ATACS.EXE:** ATACS executable file.

- **ATACS.BAS:** The aircraft combat simulation source code.
- **MAINMENU.BAS:** The menu driver module.
- **MODTITLE.BAS:** The title generator
- **MODRND.BAS:** The random number example module.
- **MODPDET.BAS:** The probability of detection example module.
- **MODSCRH.BAS:** The search example module.
- **MODPKIll.BAS:** Impact assessment example module.
- **MODMSG.BAS:** Message module.
- **MODOUT.BAS:** Output module.
- **B1PERF.DAT:** Blue one's aircraft performance data.
- **R1PERF.DAT:** Red one's aircraft performance data.
- **B2PERF.DAT:** Blue two's aircraft performance data.
- **R2PERF.DAT:** Red two's aircraft performance data.
- **B1PERF.DOC:** Blue one's documented aircraft performance data.
- **R1PERF.DOC:** Red one's documented aircraft performance data.
- **B2PERF.DOC:** Blue two's documented aircraft performance data
- **R2PERF.DOC:** Red two's documented aircraft performance data
- **MASTER.DAT:** Default scenario data file

The above files are provided with the ATACS disk. The files with the BAS extension are source code files. ATACS is written in Microsoft BASIC 4.5. All of ATACS source code is provided and available for your review using a text editor or the Microsoft BASIC 4.5 editor.

The following files may be created when ATACS is used.

- **ACPERF.DAT:** Calculated aircraft performance factors.
- **DETAIL.DAT:** A Second–by–second detail listing of activities that occurred during the simulation.
- **OUTPUT.DAT:** A listing of significant events that occurred during the simulation.

Running ATACS

Simply enter **ATACS** at the DOS prompt. ATACS is menu driven. The available menus are shown in FIG 26.

Accompanying each displayed menu is a short explanation of each option. The opening menu permits you to select the following options:

- Examples of Combat Modeling Processes
- Run Demonstration
- Display Input/Output
- Terminate the Program

Examples of Processes

Following the selection of *Examples of Processes* you are offered six examples of processes inherent to almost all dynamic high resolution combat models. These processes are:

- Random Number Generation
- Target Searching
- Various Detection Methods
- Target Identification and Selection
- CEP Assessment

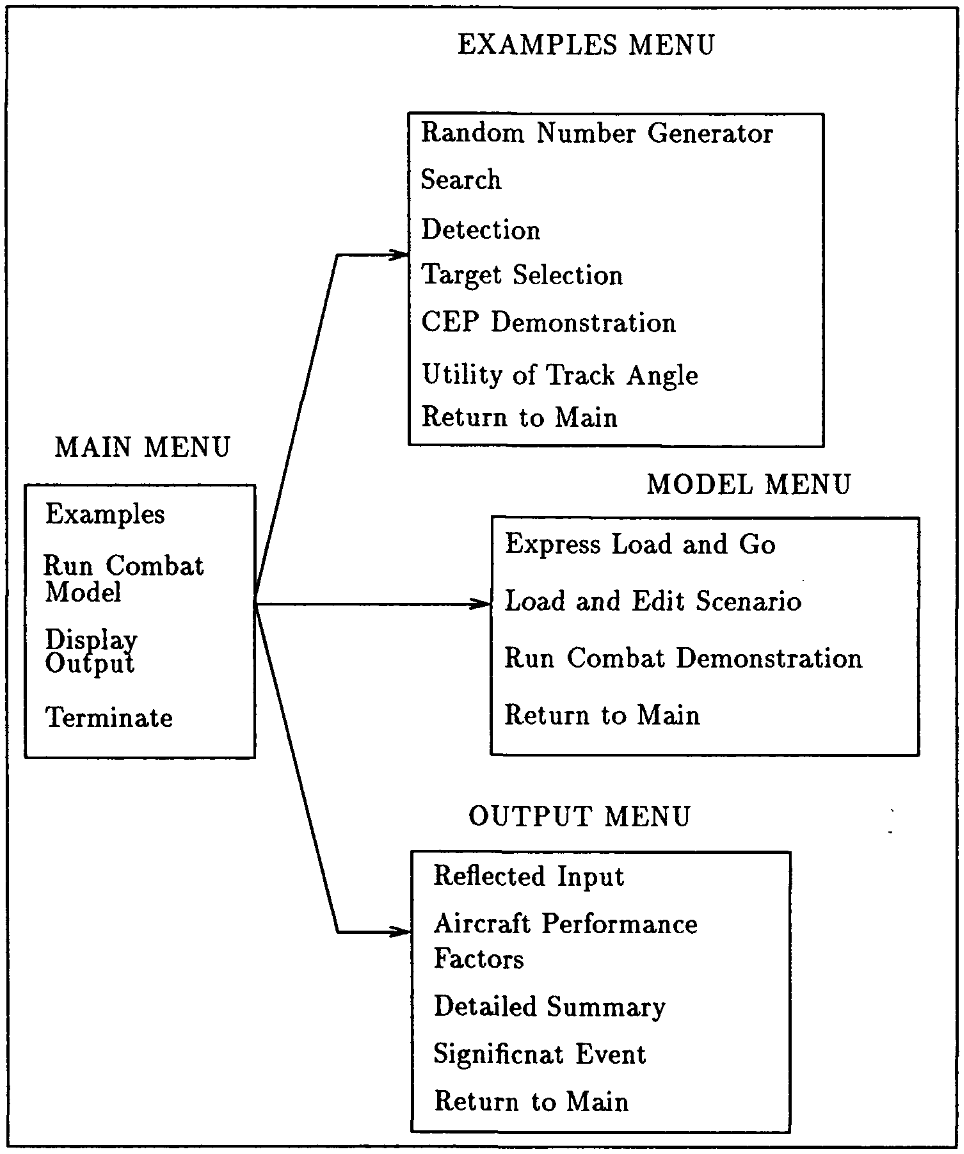


Figure 26. ATACS menus

- Utility of Track Angle

Random Number Generation: This option presents a brief narrative describing the useful purposes of the random number generator. You are prompted to enter the number of samples to be taken from the random number generator and a seed to begin the generator with. A test of the generator is then performed and the animated results displayed.

Target Searching: The second option presents a narrative on the simulation of search.

Detection Methods: The third option presents to you a narrative describing two methods by which to simulate the detection process. The two methods presented are the deterministic "cookie cutter" and a stochastic model for search radars. An illustration of both methods is presented. The first is the "cookie cutter" method. In this illustration a step function is plotted on the screen, showing the range at which detection occurs and the probability step from zero to one at that range. You are prompted to enter a range-to-target value, the program evaluates the range and places a maker on the curve corresponding to the range and probability of detection.

A second example of a detection method follows the cookie cutter illustration. This example presents stochastic probability of detection model. A plot of the model is superimposed over the step function as a means of comparing between these two simple approaches to modeling detection. You are again prompted for a target range. Once entered, the probability is determined, and a marker is plotted on the curve. The program then selects a random number. The random number drawn is used to determine if the target was detected at the range entered you entered.

Target Identification and Selection: The fourth option displays a narrative describing possible methods used to simulate this process as it pertains to the air-to-air combat environment.

Assessment of CEP: This option presents a narrative pertaining to various methods used in the simulating this event. In this example the student is prompted to enter the number of test firings. Each firing is then plotted on a target. At the completion of the firing the systematic weapon's bias and cross range and down range dispersion of the impact points are calculated.

Based on these statistics the circular error probable (CEP) is calculated an a circle depicting the CEP is drawn around the impact points. "The circular error probable is that radial distance from the aim point within which half of the rounds will land (in a probability sense)". Following this demonstration a plot of the probability of hit based on the calculated CEP is plotted. You may enter target radii to determine what effect CEP and target size has on the on the probability of hit.

Track Angle: The final example is also a narrative explaining the use of track angle in air combat models.

Run Combat Simulation: Following this selection from the main menu, you are offered three options. However, selection of these options is dependent on an order of selection. For example a scenario file must be loaded first before the simulation can begin.

Express Load and Go: This option will load the default scenario file or a file of your choice and begin the air combat simulation. When entering a scenario file name of your choice, remember to include the filename extension such as "DAT." Before the simulation begins, a brief synopsis of the scenario and function key definition is displayed.

Load and Edit Scenario File: This option will also allow you to select either the default scenario file or one of your choice.

Before the scenario editor is displayed a brief synopsis of the scenario is presented. The scenario file is then loaded into the editor. You are given the opportunity to change any value used in the scenario file. However, you must observe the limitations on some data values. Simply enter a new value at the prompt or press Enter/Return to accept the default value. The editor consists of six screens. Once completed, you are returned to the menu. At this point you may select the **Run Simulation** option.

Run Simulation: Once this option is selected you are required to enter a seed to begin the random number generator. Once entered, the function key definition is displayed followed by the start of the simulation begins. The most important of the function keys is the *Atrib Scrn* key F3. This option opens up the simulation by displaying the attribute values of several of the simulated entities.

ATACS animates such entities as the airbase, aircraft, and missiles. The icons used to animate these entities are shown in Fig 27.

The simulation will automatically terminate at the end of the default simulation time of 3000 seconds or the simulation time entered by you. However, the simulation may be terminated at any time by pressing the F7 *Quit* key.

Display Output

Once a simulation has been executed, output is available for display. From the main menu select **Display Output**. The available output are the following:

- Reflected/Echo Input Report
- Detailed Summary Report
- Aircraft Performance Factors
- Significant Event Summary Report

Reflected Input/Echo Report

This option produces a listing of all the input data, both user entered such as the scenario file and embedded, such as the aircraft performance data file.

Detailed Summary Report This option produces a listing of selected variables and their values collected at each time pulse of the simulation.

Aircraft Performance Factors

This report lists the aircraft performance factors calculated by the model. The display lists such factors as aircraft thrust, fuel consumption rates, and turn performance.

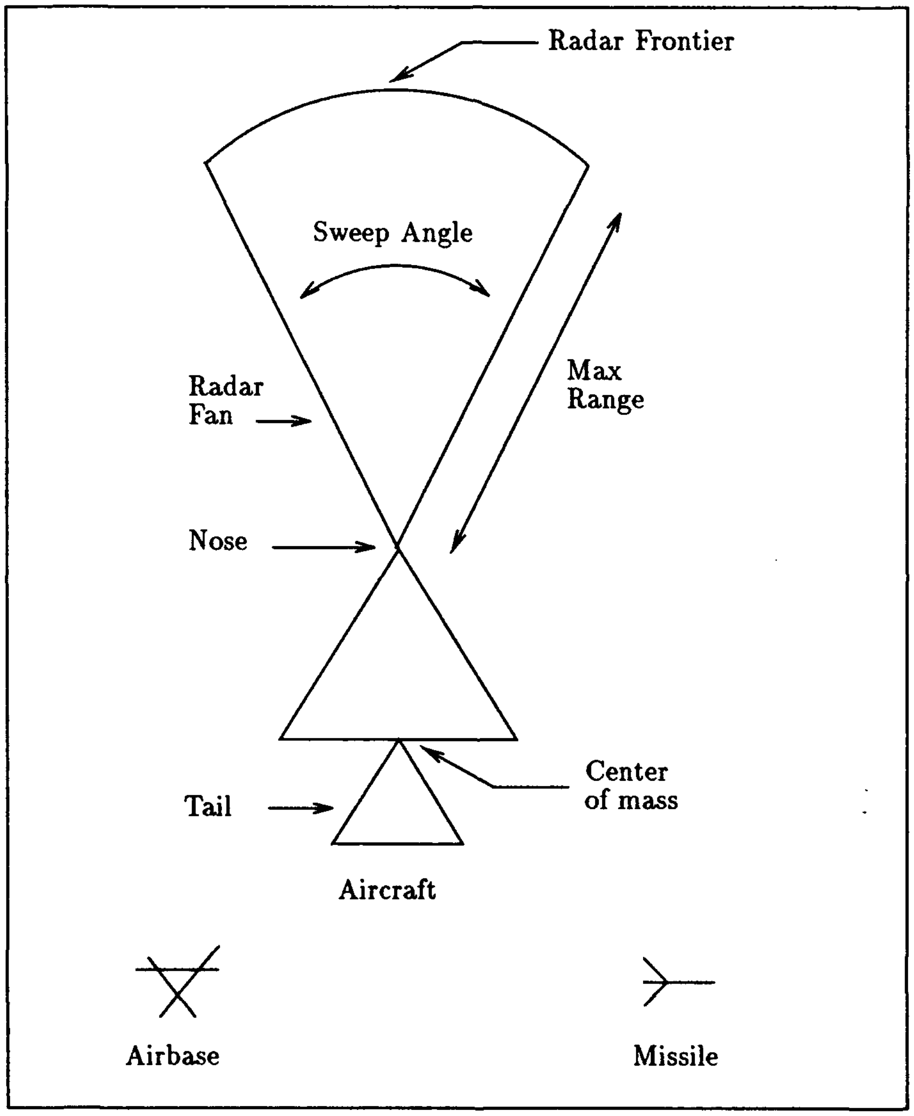


Figure 27. ATACS Icons

Significant Event Summary This final report provides a listing of the significant events and their time of occurrence. Such events include detection of an intruder, target identification, and target destruction.

Summary

ATACS offers you the opportunity to examine a combat model up close. To take it apart and look inside. These opportunities are not likely to be offered with other combat models. As mentioned in the introduction, much effort was devoted to designing a simple to use, self–documented system requiring very little reference to this manual. Once you have used ATACS you will find this to be true.

ATACS Questionnaire

Model Classification

Was the purpose of the combat demonstration–*education and training* or *analysis* ?

Classification by Qualities

What was the model's *domain*; was *land* in the domain?

What was the model's *span*?

What was the model's *environment*?

Was darkness modeled, and if not, would it make a difference in the outcome?

Was weather modeled, and if not, would it make a difference in the outcome?

What was the model's *force composition*?

What was the model's *scope of conflict*?

What was the model's *mission area*?

What was the model's *level of detail of processes and entities*?

Classification by Construction

Was your involvement required during the combat demonstration?

What time advanced mechanism (*fixed step*) or *event step* was used in the combat demonstration?

If fixed step, what was the size of the step, and was there more than one?

If event step, which events advanced the simulation time?

What time advance mechanism is best for this type of simulation and why?

Is the combat demonstration a *deterministic* or *stochastic* model?

Is the combat demonstration a *Monte-Carlo* simulation?

How many *sides* were represented in the demonstration?

Was treatment of the combatants *symmetric* or *asymmetric*?

If asymmetric, was only one side reactive?

Entities and Attributes

What entities were simulated by the combat demonstration?

What attributes did these entities posses?

Was the existence of these entities explicitly defined in the simulation or did they exist as a collection of attributes?

How were entities destroyed?

What happens to entities when they are destroyed?

Aircraft

Was altitude a factor in the simulation of the aircraft performance?

Was the aircraft velocity a factor in the simulation?

What aircraft performance measures were used to simulate the aircraft's flight and maneuverability?

What sensors were did the aircraft posses?

Were these sensors active?

What aircraft resources were represented?

Were these resources consumed?

What was the outcome if these resources were expended?

Missiles

What missile performance measures were used in the simulation?

Did missiles share the same time mechanism as the aircraft?

If so, did the missile share the same time step?

Did missiles actively track their targets?

How was missile effectiveness (killing ability) modeled?

Could a missile miss a target?

Scenario

What was the scenario?

Was the scenario plausible and/or flexible?

Were you able to customize the scenario?

Search

What methods were used to simulate the process of *search*?

Were the processes of search explicit or implied?

Which simulated sensors, either explicitly or implicitly simulated, were actively searching for intruders?

Were sensors modeled as entities or attributes of a higher entity?

Detection

What method was used to simulate the process of *detection*?

Was the method deterministic or stochastic?

Could the method be classified as *Monte-Carlo*?

Target Assignment

Was the target assignment random or was there an underlying rule?

Would the simulation permit the same target to be assigned to more than one shooter?

How many simultaneous targets could be assigned to one shooter?

Communication, Command, and Control

Was C^3 present?

If so, who was in command; what did they command; and was C^3 actively used to control entity activities?

Could shooters coordinate or concentrate fire (more than one shooter shooting at a target)?

Advance

Did the advance mechanism for aircraft and missile use a fixed rate throughout the simulation or was the rate altered by changes in entity status?

Target Destruction

To destroy a target, was the missile required to impact the target (Hint: *think of the time* advance mechanism and advance mechanism)?

Variable Definitions For MODACS

Single Dimension Variables

Basex, Basey: Airbase location

GCIRg: Tactical surveillance radar range

Format: Flag for attribute screen format

AScreen: Attribute/animation screen flag

Scale: Graphics screen scaling factor

Wx, Wy: Graphics window dimensions

AR: Aspect ratio

endsim: Simulation run time

StatPtx, StatPty: Combat Air Patrol (CAP) station point

OEF: CAP orbit expansion factor

AltaArg: Number of altitude elements in altitude array

MachArg: Number of mach elements in mach array

SysStat: System posture (Hi/Low)

Icon Arrays

X(),Y(): Coordinate points of icon line segments

xxold(),yyold(): Old translated and rotated coordinates of line segments

OldEndTop(), OldEndBot(): Old translated and rotated coordinates of radar frontier image

Mode: Color indicator for icons

Aircraft Location, Heading, Distance, Track Angle, and Targets Arrays

ACx,ACy: Location of aircraft

Tgtx, Tgty: Location of the target

ACHead,OldHead: Current and previous assigned heading

TDist: Distances between entities

Stat, IStat: Character and integer status variables

Trkang: Track angle of indexed aircraft

Tgt, ITgt: Target character and integer identifiers

TgtDetLvl: Indicator of sensor in range

Senario and Aircraft Performance Variables

Dat: Number of scenario data elements

Vel, Vp, TV, VelMax: Low velocity, hi velocity, temporary holding variable for present velocity, and maximum permissible velocity

WgArea: Aircraft's wing area

MaxWt, MinWt: Maximum and minimum aircraft combat weight

MaxAlt, MinAlt: Maximum and minimum combat altitudes

MaxETA: Maximum normal force

ThrKF, DragKF: Multipliers for thrust and drag table factors

ThrMP, ThrAB: Military and afterburner thrust table values

THMP, THAB: Index aircraft thrust in military and afterburner

TurnRate, RadTurn, TTurn, RTmiles: Aircraft turn factors

FCMP, FCAB: Table values for fuel consumption

MPFC, ABFC: Fuel consumption rate for indexed aircraft

Bingo: Bingo fuel level for indexed aircraft

BurnRate: Fuel consumption rate based on current velocity

Sensor variables

OptRng: Optical maximum range

RadRng: Radar maximum range

Missile variables

NumMSL: Number of missile on indexed aircraft

MslV: Missile velocity

MslSenRg: Missile sensor range

MX, MY: Missile icon line segment coordinates

MF: Number of missiles fired from indexed aircraft

MTNOW: Missile timer

Pk: Probability of kill for missile

IMslStat: Integer value for missile status

IMTgt: Missile integer target identification

Appendix E. *ATACS Source Code Listings*

MODTITLE

```
'***********************************
'ATACS Title Generator

'Author: Capt R. Moore

'Date: 15 Feb 91

'***********************************


COMMON SHARED MonMode%, mennum%

SUB title
'' Drawing of ATACS (Animated Tutor for air combat simulation

'MonMode% = 12

IF MonMode% = 12 THEN

SCREEN 12
VIEW (0, 0)-(620, 460)
offset = 0 '70
lnum = 100
LOCATE 4, 20: PRINT "ANIMATED TUTOR FOR AIR COMBAT SIMULATION"
LOCATE 22, 14: PRINT "DESIGNED TO COMPLEMENT THE TEACHING OF
COMBAT MODELING"
LOCATE 22 + 5, 28: PRINT "PRESS ANY KEY TO BEGIN"
ELSE
SCREEN 9, , 1, 0
offset = 0' 50
lnum = 80
VIEW (0, 0)-(620, 340)
LOCATE 3, 20: PRINT "ANIMATED TUTOR FOR AIR COMBAT SIMULATION"
LOCATE 18, 14: PRINT "DESIGNED TO COMPLEMENT THE TEACHING OF
COMBAT MODELING"
LOCATE 18 + 5, 28: PRINT "PRESS ANY KEY TO BEGIN"
END IF

letwd = 10
letht = 100
clr = 3


'' Letter a

start = 25
eend = 125
delta = (eend - start)
mid = start + delta / 2
midht = lnum + (letht / 2)

FOR i = 1 TO letwd + 3
LINE (mid + i, lnum)-(start + i, lnum + letht), clr
LINE (mid + i, lnum)-(eend + i, lnum + letht), clr
NEXT i

FOR i = 1 TO (letwd - 3)
LINE (start + .3 * delta, lnum + 50 + i)-(eend - .2 * delta, lnum
+ 50 + i), clr
NEXT i

'' letter t

lnum = lnum + offset

'' Horizontal start and end
start = 142
eend = 242
mid = start + (eend - start) / 2

FOR i = 0 TO letwd
 LINE (start, lnum + i)-(eend, lnum + i), clr     'Top of T
 LINE (((mid - letwd / 2) + i), lnum)-(((mid - letwd / 2) + i),
lnum + letht), clr
 LINE (start, lnum + letwd + i)-((start - i + letwd), lnum +
letwd + i), clr'Let tail
 LINE ((eend - (letwd - i)), lnum + letwd + i)-(eend, lnum +
letwd + i), clr'Right tail
NEXT i


" Letter a

lnum = lnum + offset   '- 35

start = 245
eend = 345
delta = (eend - start)
mid = start + delta / 2
midht = lnum + (letht / 2)

FOR i = 1 TO letwd + 3
LINE (mid + i, lnum)-(start + i, lnum + letht), clr
LINE (mid + i, lnum)-(eend + i, lnum + letht), clr
NEXT i

FOR i = 1 TO (letwd - 3)
LINE (start + .3 * delta, lnum + 50 + i)-(eend - .2 * delta, lnum
+ 50 + i), clr
NEXT i

" letter c

lnum = lnum + offset + 10

start = 385
eend = 485
delta = (eend - start)
mid = start + delta / 2
midht = lnum + (letht / 2)
cstr = .017453 * 60
cstp = .017453 * 320

IF MonMode% = 12 THEN
r1 = 38: r2 = 48
ELSE
r1 = 56: r2 = 68
END IF

FOR i = r1 TO r2 STEP .1
CIRCLE (mid, midht - 10), i, clr, cstr, cstp
NEXT i

" letter s

lnum = lnum + offset

start = 490
eend = 590
delta = (eend - start)
mid = 10 + start + delta / 2
midht = lnum + (letht / 2)
tcstr = .017453 * 45
tcstp = .017453 * 270
bcstr1 = .017453 * 0
bcstp1 = .017453 * 90
bcstr2 = .017453 * 195
bcstp2 = .017453 * 360
tmid = lnum + .25 * letht
bmid = lnum + .7 * letht

IF MonMode% = 12 THEN
r1 = 17: r2 = 27
ELSE
r1 = 23: r2 = 35
END IF


FOR i = r1 TO r2 STEP .1
CIRCLE (mid, tmid - 8), i, clr, tcstr, tcstp
CIRCLE (mid, bmid - 8), i, clr, bcstr1, bcstp1
CIRCLE (mid, bmid - 8), i, clr, bcstr2, bcstp2
```

```
NEXT i

IF MonMode% = 9 THEN
SCREEN 9, , 0, 1
END IF

DO WHILE INKEY$ = "": LOOP

END SUB
```

MAINMENU

```
'*********************************
'Program: Menu module

'Author: Capt R. Moore

'Date: 15 Feb 91

'*********************************
DECLARE SUB Reflect ()
DECLARE SUB Sigevnt ()
DECLARE SUB ACperf ()
DECLARE SUB Detlrpt ()
DECLARE SUB msg (MsgNum%)
DECLARE SUB USERDAT ()
DECLARE SUB LoadGo ()
DECLARE SUB MainLoop (MonMode%)
DECLARE SUB GetData ()
DECLARE SUB CEPDEMO (MonMode%)
DECLARE SUB RNDDEMO (MonMode%)
DECLARE SUB SCRHDEMO (MonMode%)
DECLARE SUB PDETDEMO (MonMode%)
DECLARE FUNCTION Menu% (choices$())
DECLARE SUB DisplayMenuBox (choiceList$(), leftCoord%, prompt$,
ok$)
DECLARE SUB Frame (Left%, Right%, Top%, Bottom%)

COMMON SHARED MonMode%, mennum%, com$

CONST false% = 0, true% = NOT false%

DATA Examples of Processes
DATA Combat Demonstration
DATA Display Output
DATA Terminate
DATA Random Number Generator
DATA Search
DATA detect
DATA target selection
DATA CEP demonstration
DATA Utilty of track angle
DATA main menu
DATA Express load and go
DATA Load and edit scenario file
DATA run demonstration
DATA main menu
DATA reflected input
DATA detailed summary
DATA Aircraft Performance Factors
DATA Summary of significant events
DATA main menu

SUB DisplayMenuBox (choiceList$(), leftCoord%, prompt$, ok$)
'///////////////////////////////////////////////////////
' Program Display Menu Box
' Author: Capt R. Moore
' Date: 14 Oct 92
'
' Source: Micro-Soft Quick Basic 2nd Edition
'
' Description:  This subprogram list the menu options and
' thenmakes
'  a call to frame.  The argument values passed to frame are the
'  dimensions based on the size of the longest word in the option
'  list.  Additionally, the prompt is created in the subprogram.
'
' Called by Function Menu
'
' Calls Sub Frame
'
'///////////////////////////////////////////////////////
'
' Find number of choices and initialize variables
'
numChoices% = UBOUND(choiceList$)
prompt$ = " "
ok$ = ""
lngChoice% = 0
```

```
'
' Prepare the prompt string and the string of legal input
characters (ok$).
' Find the length of the longest option (LngChoice%)
'
'
FOR i% = 1 TO numChoices%
  first$ = UCASE$(LEFT$(choiceList$(i%), 1)) 'Grab the first
letter
  ok$ = ok$ + first$
  prompt$ = prompt$ + first$ + " "
  longTemp% = LEN(choiceList$(i%))    ' Find the length of
eachchioce
  IF longTemp% ¿ lngChoice% THEN lngChoice% = longTemp%
NEXT i%

lngChoice% = lngChoice% + 1
prompt$ = prompt$ + "-¿ "

' Check and see if prompt string is longer than longest option

IF LEN(prompt$) ¿= lngChoice% THEN lngChoice% = LEN(prompt$) + 1
'
' Determine the coordinates for the menu frame based on
menuoptions
'
leftCoord% = 37 - lngChoice% " 2
rightCoord% = 80 - leftCoord%
topCoord% = 3
bottomCoord% = 10 + numChoices%
Frame leftCoord%, rightCoord%, topCoord%, bottomCoord%
'
'
'
FOR i% = 1 TO numChoices%
   LOCATE 6 + i%, leftCoord% + 3
   COLOR 14, 0: PRINT UCASE$(LEFT$(choiceList$(i%), 1))
   LOCATE 6 + i%, leftCoord% + 4
   COLOR 7, 0: PRINT MID$(choiceList$(i%), 2)
NEXT i%

COLOR 3
SELECT CASE mennum%
 CASE 1
   LOCATE 4, 35: PRINT "Main Menu"
 CASE 2
   LOCATE 4, 29: PRINT "Dynamic Examples Menu"
 CASE 3
   LOCATE 4, 31: PRINT "Demonstration Menu"
 CASE 4
   LOCATE 4, 34: PRINT "Output Menu"
END SELECT
COLOR 7



line$ = STRING$(lngChoice%, 196)  ' 196 is the "'", print it
longChoice times
LOCATE 5, leftCoord% + 3: PRINT line$  ' 3 is the margin from
border
LOCATE 7 + numChoices%, leftCoord% + 3: PRINT line$ 'print
bottomsingle line
'
' Print the input prompt
'
LOCATE 9 + numChoices%, leftCoord% + 3: PRINT prompt$;

END SUB

SUB Frame (Left%, Right%, Top%, Bottom%)
'/////////////////////////////////////////////////////////////-
/////
' Program Frame
' Author: Capt R. Moore
' Date: 14 Oct 92
'
' Source: Micro-Soft Quick Basic 2nd Edition
```

```
'
' Description:  The frame subprogram draw a double-line
rectangular
'               frame on the screen to enclose menu options.
'
' Called by Sub DisplayMenuBox
'
'////////////////////////////////////////////////


' Draw the four corners using the text-graphics character set
from
' extended ASCII character set.

LOCATE Top%, Left%: PRINT CHR$(201)
LOCATE Top%, Right%: PRINT CHR$(187)
LOCATE Bottom%, Left%: PRINT CHR$(200)
LOCATE Bottom%, Right%: PRINT CHR$(188)

' Draw vertical lines

FOR Vert% = Top% + 1 TO Bottom% - 1
  LOCATE Vert%, Left%: PRINT CHR$(186)
  LOCATE Vert%, Right%: PRINT CHR$(186)
NEXT Vert%

' Draw horizontal lines

Horz% = Right% - Left% - 1
Hline$ = STRING$(Horz%, 205)
LOCATE Top%, Left% + 1: PRINT Hline$
LOCATE Bottom%, Left% + 1: PRINT Hline$

END SUB

SUB mainmenu
'////////////////////////////////////////////
' Program: MenuOptions
' Author: Capt R. Moore
' Date: 14 Oct 92
'
' Source: Micro-Soft Quick Basic 2nd Edition
'
' Description:  This program sets up the menu options based on
'               selections from other menus
'
'/////////////////////////////////////////////////////////////-
/////
DIM menuOpt1$(4), menuOpt2$(7), menuOpt3$(4), menuOpt4$(5)

CLS

SCREEN 0


' MenuOption Data


FOR i% = 1 TO 4
  READ menuOpt1$(i%)
NEXT i%

FOR i% = 5 TO 11
  j% = i% - 4
  READ menuOpt2$(j%)
NEXT i%

FOR i% = 12 TO 15
  j% = i% - 11
  READ menuOpt3$(j%)
NEXT i%

FOR i% = 16 TO 20
  j% = i% - 15
  READ menuOpt4$(j%)
NEXT i%
```

```
CLS
DO
Head1:
msg 101
mennum% = 1
 SELECT CASE Menu%(menuOpt1$())
 CASE 1
   CLS
   GOTO Head2
 CASE 2
   CLS
   GOTO Head3
 CASE 3
   CLS
   IF GotData% = 2 THEN
      GOTO head4
   ELSE
      BEEP
      msg 1011
      GOTO Head3
   END IF
 CASE 4
   CLS : END
 CASE ELSE
   done% = true%
END SELECT

Head2:
msg 102
mennum% = 2
 SELECT CASE Menu%(menuOpt2$())
 CASE 1
   msg 110
   SCREEN MonMode%
   RNDDEMO MonMode%
   SCREEN 0
 CASE 2
   msg 111
   SCREEN MonMode%
'   SCRHDEMO MonMode%
   SCREEN 0
 CASE 3
   msg 112
   SCREEN MonMode%
   PDETDEMO MonMode%
   SCREEN 0
 CASE 4
   msg 113
   SCREEN MonMode%
   CLS
   SCREEN 0
 CASE 5
   msg 114
   SCREEN MonMode%
   CEPDEMO MonMode%
   SCREEN 0
 CASE 6
   msg 115
   SCREEN MonMode%
   SCREEN 0
   CLS
 CASE 7
   GOTO Head1
 CASE ELSE
   done% = true%
 END SELECT
 GOTO Head2

Head3:
msg 103
mennum% = 3
SELECT CASE Menu%(menuOpt3$())
 CASE 1
   CLS : LoadGo
   CLS : GetData
   IF GotData% ¡ 2 THEN : msg 200: msg 210
   MainLoop MonMode%
```

```
    GotData% = 2
    SCREEN 0
    GOTO head4
  CASE 2
    CLS : USERDAT
    GetData
    GotData% = 1
    CLS
    CLOSE #5
  CASE 3
    IF GotData% = 1 THEN
       msg 210
       MainLoop MonMode%: CLS
       CLOSE #10
       GotData% = 2
       SCREEN 0
       GOTO head4
    ELSEIF GotData% = 2 THEN
       BEEP
       msg 1032
       GOTO Head3
    ELSE
       BEEP
       msg 1031
       GOTO Head3
    END IF
  CASE 4
    GOTO Head1
  CASE ELSE
    done% = true%
END SELECT
GOTO Head3

head4:
msg 104
mennum% = 4
 SELECT CASE Menu%(menuOpt4$())
 CASE 1
   CLS : msg 1041: Reflect: CLS
 CASE 2
   CLS : msg 1042: Detlrpt: CLS
 CASE 3
   CLS : msg 1043: ACperf: CLS
 CASE 4
   CLS : msg 1044: Sigevnt: CLS
 CASE 5
   GOTO Head1
 CASE ELSE
   done% = true%
 END SELECT
 GOTO head4

Loopit:
  IF NOT done% THEN
    PRINT
    PRINT "Press the spacebar to continue."

    DO
      ch$ = INKEY$
    LOOP UNTIL ch$ = " "  'Indefinite Pause
    CLS
  END IF
  RETURN


LOOP UNTIL done%  ' End of main loop
END

END SUB

FUNCTION Menu% (choices$()) STATIC
'/////////////////////////////////////////////////////////////////'

' Program Menu
' Author: Capt R. Moore
' Date: 14 Oct 92
'
```

```
' Source: Micro-Soft Quick Basic 2nd Edition
'
' Description: This function displays the options. Prompts fcr a
'  choice and returns an integer value representing that choice.
'
' Called by MainMenu
'
' Calls DispMenuBox
'
'///////////////////////////////////////////////////////////
'
listlength% = UBOUND(choices$)  'Find length of choice array

' Set up the menu display


DisplayMenuBox choices$(), leftmargin%, promptStr$, okStr$

' Get main menu chice. Validate and verify the choice

controlKeys$ = CHR$(13) + CHR$(27)
DO
  LOCATE , , 1
  charPos% = 0
  DO
   answer$ = UCASE$(INKEY$)  'Return the uppercase of letter
slected
   IF answer$ ¡¿ "" THEN
      charPos% = INSTR(okStr$, answer$)  'Match and return char
position
      IF charPos% = 0 THEN BEEP
   END IF
  LOOP UNTIL charPos% ¿ 0

  PRINT answer$
  COLOR 14: LOCATE 11 + listlength%, 23, 0
  PRINT "¡Enter¿ to confirm; ¡Esc¿ to reslect."
  COLOR 7
  inchoice% = charPos%
  charPos% = 0
  DO
    answer$ = INKEY$
    IF answer$ ¡¿ "" THEN
      charPos% = INSTR(controlKeys$, answer$)
      IF charPos% = 0 THEN BEEP
    END IF
  LOOP UNTIL charPos% ¿ 0

  IF charPos% = 1 THEN
    done% = true%
    CLS
  ELSE
    done% = false%
    LOCATE 11 + listlength%, 23: PRINT SPACE$(35)
    LOCATE 9 + listlength%, leftmargin% + 3 + LEN(promptStr$):
PRINT " ";
    LOCATE , POS(0) - 1
  END IF
LOOP UNTIL done%

Menu% = inchoice%

END FUNCTION



DECLARE SUB Reflect ()
DECLARE SUB Sigevnt ()
DECLARE SUB ACperf ()
DECLARE SUB Detlrpt ()
DECLARE SUB msg (MsgNum%)
DECLARE SUB USERDAT ()
DECLARE SUB LoadGo ()
DECLARE SUB MainLoop (MonMode%)
DECLARE SUB GetData ()
DECLARE SUB CEPDEMO (MonMode%)
DECLARE SUB RNDDEMO (MonMode%)
DECLARE SUB SCRHDEMO (MonMode%)
DECLARE SUB PDETDEMO (MonMode%)
DECLARE FUNCTION Menu% (choices$())
DECLARE SUB DisplayMenuBox (choiceList$(), leftCoord%, prompt$,
ok$)
DECLARE SUB Frame (Left%, Right%, Top%, Bottom%)

COMMON SHARED MonMode%, mennum%, com$

CONST false% = 0, true% = NOT false%

DATA Examples of Processes
DATA Combat Demonstration
DATA Display Output
DATA Terminate
DATA Random Number Generator
DATA Search
DATA detect
DATA target selection
DATA CEP demonstration
DATA Utilty of track angle
DATA main menu
DATA Express load and go
DATA Load and edit scenario file
DATA run demonstration
DATA main menu
DATA reflected input
DATA detailed summary
DATA Aircraft Performance Factors
DATA Summary of significant events
DATA main menu

SUB DisplayMenuBox (choiceList$(), leftCoord%, prompt$, ok$)
'///////////////////////////////////////////////////////////////-
/////
' Program Display Menu Box
' Author: Capt R. Moore
' Date: 14 Oct 92
'
' Source: Micro-Soft Quick Basic 2nd Edition
'
' Description: This subprogram list the menu options and
thenmakes
'  a call to frame. The argument values passed to frame are the
'  dimensions based on the size of the longest word in the option
'  list. Additionally, the prompt is created in the subprogram.
'
' Called by Function Menu
'
' Calls Sub Frame
'
'///////////////////////////////////////////////////////////////-
/////
'
' Find number of choices and initialize variables
'
numChoices% = UBOUND(choiceList$)
prompt$ = " "
ok$ = ""
lngChoice% = 0
'
' Prepare the prompt string and the string of legal input
characters (ok$).
' Find the length of the longest option (LngChoice%)
'
'
FOR i% = 1 TO numChoices%
  first$ = UCASE$(LEFT$(choiceList$(i%), 1)) 'Grab the first
letter
  ok$ = ok$ + first$
  prompt$ = prompt$ + first$ + " "
  longTemp% = LEN(choiceList$(i%))    ' Find the length of
eachchioce
  IF longTemp% ¿ lngChoice% THEN lngChoice% = longTemp%
NEXT i%
```

```
lngChoice% = lngChoice% + 1
prompt$ = prompt$ + "-¿ "

' Check and see if prompt string is longer than longest option

IF LEN(prompt$) ¿= lngChoice% THEN lngChoice% = LEN(prompt$) + 1
'
' Determine the coordinates for the menu frame based on
menuoptions
'
leftCoord% = 37 - lngChoice% " 2
rightCoord% = 80 - leftCoord%
topCoord% = 3
bottomCoord% = 10 + numChoices%
Frame leftCoord%, rightCoord%, topCoord%, bottomCoord%
'
'
'
FOR i% = 1 TO numChoices%
  LOCATE 6 + i%, leftCoord% + 3
  COLOR 14, 0: PRINT UCASE$(LEFT$(choiceList$(i%), 1))
  LOCATE 6 + i%, leftCoord% + 4
  COLOR 7, 0: PRINT MID$(choiceList$(i%), 2)
NEXT i%

COLOR 3
SELECT CASE mennum%
 CASE 1
  LOCATE 4, 35: PRINT "Main Menu"
 CASE 2
  LOCATE 4, 29: PRINT "Dynamic Examples Menu"
 CASE 3
  LOCATE 4, 31: PRINT "Demonstration Menu"
 CASE 4
  LOCATE 4, 34: PRINT "Output Menu"
END SELECT
COLOR 7



line$ = STRING$(lngChoice%, 196)  ' 196 is the "'", print it
longChoice times
LOCATE 5, leftCoord% + 3: PRINT line$  ' 3 is the margin from
border
LOCATE 7 + numChoices%, leftCoord% + 3: PRINT line$ 'print
bottomsingle line
'
' Print the input prompt
'
LOCATE 9 + numChoices%, leftCoord% + 3: PRINT prompt$;

END SUB

SUB Frame (Left%, Right%, Top%, Bottom%)
'///////////////////////////////////////////////////////////////-
/////
' Program Frame
' Author: Capt R. Moore
' Date: 14 Oct 92
'
' Source: Micro-Soft Quick Basic 2nd Edition
'
' Description:  The frame subprogram draw a double-line
rectangular
'               frame on the screen to enclose menu options.
'
' Called by Sub DisplayMenuBox
'
'///////////////////////////////////////////////////////////////-
/////


' Draw the four corners using the text-graphics character set
from
' extended ASCII character set.

LOCATE Top%, Left%: PRINT CHR$(201)
LOCATE Top%, Right%: PRINT CHR$(187)
LOCATE Bottom%, Left%: PRINT CHR$(200)
LOCATE Bottom%, Right%: PRINT CHR$(188)

' Draw vertical lines

FOR Vert% = Top% + 1 TO Bottom% - 1
  LOCATE Vert%, Left%: PRINT CHR$(186)
  LOCATE Vert%, Right%: PRINT CHR$(186)
NEXT Vert%

' Draw horizontal lines

Horz% = Right% - Left% - 1
Hline$ = STRING$(Horz%, 205)
LOCATE Top%, Left% + 1: PRINT Hline$
LOCATE Bottom%, Left% + 1: PRINT Hline$

END SUB

SUB mainmenu
'//////////////////////////////////////////////////////////////-
/////
' Program: MenuOptions
' Author: Capt R. Moore
' Date: 14 Oct 92
'
' Source: Micro-Soft Quick Basic 2nd Edition
'
' Description:  This program sets up the menu options based on
'               selections from other menus
'
'//////////////////////////////////////////////////////////////-
/////
DIM menuOpt1$(4), menuOpt2$(7), menuOpt3$(4), menuOpt4$(5)

CLS

SCREEN 0


' MenuOption Data


FOR i% = 1 TO 4
  READ menuOpt1$(i%)
NEXT i%

FOR i% = 5 TO 11
  j% = i% - 4
  READ menuOpt2$(j%)
NEXT i%

FOR i% = 12 TO 15
  j% = i% - 11
  READ menuOpt3$(j%)
NEXT i%

FOR i% = 16 TO 20
  j% = i% - 15
  READ menuOpt4$(j%)
NEXT i%

CLS
DO
Head1:
msg 101
mennum% = 1
 SELECT CASE Menu%(menuOpt1$())
 CASE 1
   CLS
   GOTO Head2
 CASE 2
   CLS
   GOTO Head3
 CASE 3
   CLS
   IF GotData% = 2 THEN
     GOTO head4
   ELSE
     BEEP
     msg 1011
     GOTO Head3
   END IF
 CASE 4
   CLS : END
 CASE ELSE
   done% = true%
END SELECT

Head2:
msg 102
mennum% = 2
 SELECT CASE Menu%(menuOpt2$())
 CASE 1
   msg 110
   SCREEN MonMode%
   RNDDEMO MonMode%
   SCREEN 0
 CASE 2
   msg 111
   SCREEN MonMode%
'   SCRHDEMO MonMode%
   SCREEN 0
 CASE 3
   msg 112
   SCREEN MonMode%
   PDETDEMO MonMode%
   SCREEN 0
 CASE 4
   msg 113
   SCREEN MonMode%
   CLS
   SCREEN 0
 CASE 5
   msg 114
   SCREEN MonMode%
   CEPDEMO MonMode%
   SCREEN 0
 CASE 6
   msg 115
   SCREEN MonMode%
   SCREEN 0
   CLS
 CASE 7
   GOTO Head1
 CASE ELSE
   done% = true%
 END SELECT
 GOTO Head2

Head3:
msg 103
mennum% = 3
SELECT CASE Menu%(menuOpt3$())
 CASE 1
   CLS : LoadGo
   CLS : GetData
   IF GotData% ¡ 2 THEN : msg 200: msg 210
   MainLoop MonMode%
   GotData% = 2
   SCREEN 0
   GOTO head4
 CASE 2
   CLS : USERDAT
   GetData
   GotData% = 1
   CLS
   CLOSE #5
 CASE 3
   IF GotData% = 1 THEN
     msg 210
     MainLoop MonMode%: CLS
     CLOSE #10
```

```
      GotData% = 2
      SCREEN 0
      GOTO head4
    ELSEIF GotData% = 2 THEN
      BEEP
      msg 1032
      GOTO Head3
    ELSE
      BEEP
      msg 1031
      GOTO Head3
    END IF
  CASE 4
    GOTO Head1
  CASE ELSE
    done% = true%
END SELECT
GOTO Head3

head4:
msg 104
mennum% = 4
 SELECT CASE Menu%(menuOpt4$())
 CASE 1
    CLS : msg 1041: Reflect: CLS
 CASE 2
    CLS : msg 1042: Detlrpt: CLS
 CASE 3
    CLS : msg 1043: ACperf: CLS
 CASE 4
    CLS : msg 1044: Sigevnt: CLS
 CASE 5
    GOTO Head1
 CASE ELSE
    done% = true%
 END SELECT
 GOTO head4

Loopit:
  IF NOT done% THEN
    PRINT
    PRINT "Press the spacebar to continue."

    DO
      ch$ = INKEY$
    LOOP UNTIL ch$ = " "  'Indefinite Pause
    CLS
  END IF
  RETURN


LOOP UNTIL done%  ' End of main loop
END

END SUB

FUNCTION Menu% (choices$()) STATIC
'////////////////////////////////////////////////////////////-
/////
' Program Menu
' Author: Capt R. Moore
' Date: 14 Oct 92
'
' Source: Micro-Soft Quick Basic 2nd Edition
'
' Description: This function displays the options. Prompts for a
'  choice and returns an integer value representing that choice.
'
' Called by MainMenu
'
' Calls DispMenuBox
'
'////////////////////////////////////////////////////////////-
/////
'
listlength% = UBOUND(choices$)  'Find length of choice array
```

```
' Set up the menu display


DisplayMenuBox choices$(), leftmargin%, promptStr$, okStr$

' Get main menu chice. Validate and verify the choice

controlKeys$ = CHR$(13) + CHR$(27)
DO
  LOCATE , , 1
  charPos% = 0
  DO
   answer$ = UCASE$(INKEY$)  'Return the uppercase of letter
slected
   IF answer$ ¡¿ "" THEN
      charPos% = INSTR(okStr$, answer$)  'Match and return char
position
      IF charPos% = 0 THEN BEEP
   END IF
  LOOP UNTIL charPos% ¿ 0

  PRINT answer$
  COLOR 14: LOCATE 11 + listlength%, 23, 0
  PRINT "¡Enter¿ to confirm; ¡Esc¿ to resiect."
  COLOR 7
  inchoice% = charPos%
  charPos% = 0
  DO
    answer$ = INKEY$
    IF answer$ ¡¿ "" THEN
      charPos% = INSTR(controlKeys$, answer$)
      IF charPos% = 0 THEN BEEP
    END IF
  LOOP UNTIL charPos% ¿ 0

  IF charPos% = 1 THEN
    done% = true%
    CLS
  ELSE
    done% = false%
    LOCATE 11 + listlength%, 23: PRINT SPACE$(35)
    LOCATE 9 + listlength%, leftmargin% + 3 + LEN(promptStr$):
PRINT " ";
    LOCATE , POS(0) - 1
  END IF
LOOP UNTIL done%

Menu% = inchoice%

END FUNCTION
```

MODRND

```
'///////////////////////////////////////////////////////////////-
/////
' Program: Random Number Generator Test
' Date: 19 Oct 91
'
' Source: Maj Garranbone.
'
' Description:  Program makes n samples of BASICs random number
'               generator and plots the results
'
' Called By:  Main Menu
'
' Program Calls:
'
'///////////////////////////////////////////////////////////////-
/////

DECLARE SUB RNDDEMO (MonMode%)

RNDDEMO (MonMode%)

END

SUB RNDDEMO (MonMode%)
'
' Screen set-up

CLS
Restart:

IF (MonMode% = 12) THEN
  VIEW (46, 20)-(590, 252), 3, 4'Coordinates of screen disignated
for graphics
ELSE
  VIEW (46, 15)-(590, 220), 3, 4'Coordinates of screen disignated
for graphics
END IF

WINDOW (-.08, 1.2)-(1.08, -.2)  'Coordinates disignated within
the viewport
LINE (0, 1.18)-(0, 0)           'Vertical  Axis
LINE (0, 0)-(1.1, 0)            'Horizontal Axis

IF (MonMode% = 12) THEN

'Axis labels

LOCATE 17, 6: PRINT " ———————— FREQUENCY OF OCCURENCE
———————— "
LOCATE 3, 4: PRINT "P"
LOCATE 4, 4: PRINT "R"
LOCATE 5, 4: PRINT "O"
LOCATE 6, 4: PRINT "B"
LOCATE 7, 4: PRINT "A"
LOCATE 8, 4: PRINT "B"
LOCATE 9, 4: PRINT "I"
LOCATE 10, 4: PRINT "L"
LOCATE 11, 4: PRINT "I"
LOCATE 12, 4: PRINT "T"
LOCATE 13, 4: PRINT "Y"

ELSE

EGAAdj = 2
LOCATE 17, 10: PRINT " ———————— FREQUENCY OF OCCURENCE
———————— "
LOCATE 3, 4: PRINT "P"
LOCATE 4, 4: PRINT "R"
LOCATE 5, 4: PRINT "O"
LOCATE 6, 4: PRINT "B"
LOCATE 7, 4: PRINT "A"
LOCATE 8, 4: PRINT "B"
LOCATE 9, 4: PRINT "I"
LOCATE 10, 4: PRINT "L"
LOCATE 11, 4: PRINT "I"
LOCATE 12, 4: PRINT "T"
LOCATE 13, 4: PRINT "Y"
```

```
END IF

' Print Graph Lables

LOCATE 4, 8: PRINT "0.1"
LOCATE 16, 70: PRINT "1.0"

' Print text

LOCATE 1, 25: PRINT "Random Number Generator Test"

' Prompt for input

LOCATE 19, 6: INPUT "Enter number of samples "; num
LOCATE 21, 6: RANDOMIZE
LOCATE 19, 6: PRINT "
"
LOCATE 21, 6: PRINT "
"
LOCATE 20, 25: PRINT "Number of random draws: "

' Define constants

SumSq = 0!
Mean = 0!

' Define end points for bargraph

x1l = .02: x1r = .11
x2l = .125: x2r = .21
x3l = .225: x3r = .31
x4l = .325: x4r = .41
x5l = .425: x5r = .51
x6l = .525: x6r = .61
x7l = .625: x7r = .71
x8l = .725: x8r = .81
x9l = .825: x9r = .91
x10l = .925: x10r = 1.025
y1 = .005: y2 = .005
y3 = .005: y4 = .005
y5 = .005: y6 = .005
y7 = .005: y8 = .005
y9 = .005: y10 = .005
s1 = 0: s2 = 0: s3 = 0: s4 = 0: s5 = 0
s6 = 0: s7 = 0: s8 = 0: s9 = 0: s10 = 0

' Scale window

Scale = 10 / num

FOR i = 1 TO num

LOCATE 20, 50: PRINT i

  Sample = RND(1)
  SumSq = SumSq + Sample ^ 2   'Capture Statistics
  Mean = Mean + Sample          ' Ditto

SELECT CASE (Sample)
  CASE IS ¡= .1
    s1 = s1 + 1
    ss1 = s1 / num
    LOCATE 15, 12: PRINT USING " .### "; ss1
    y1 = y1 + Scale
    LINE (x1l, y1)-(x1r, y1), 4
  CASE IS ¡= .2
    s2 = s2 + 1
    ss2 = s2 / num
    LOCATE 15, 18: PRINT USING " .### "; ss2
    y2 = y2 + Scale
    LINE (x2l, y2)-(x2r, y2), 4
  CASE IS ¡= .3
    s3 = s3 + 1
    ss3 = s3 / num
    LOCATE 15, 24: PRINT USING " .### "; ss3
    y3 = y3 + Scale
    LINE (x3l, y3)-(x3r, y3), 4
```

```
 CASE IS ¡= .4
  s4 = s4 + 1
  ss4 = s4 / num
  LOCATE 15, 30: PRINT USING " .### "; ss4
  y4 = y4 + Scale
  LINE (x4l, y4)-(x4r, y4), 4
 CASE IS ¡= .5
  s5 = s5 + 1
  ss5 = s5 / num
  LOCATE 15, 36: PRINT USING " .### "; ss5
  y5 = y5 + Scale
  LINE (x5l, y5)-(x5r, y5), 4
 CASE IS ¡= .6
  s6 = s6 + 1
  ss6 = s6 / num
  LOCATE 15, 42: PRINT USING " .### "; ss6
  y6 = y6 + Scale
  LINE (x6l, y6)-(x6r, y6), 4
 CASE IS ¡= .7
  s7 = s7 + 1
  ss7 = s7 / num
  LOCATE 15, 48: PRINT USING " .### "; ss7
  y7 = y7 + Scale
  LINE (x7l, y7)-(x7r, y7), 4
 CASE IS ¡= .8
  s8 = s8 + 1
  ss8 = s8 / num
  LOCATE 15, 54: PRINT USING " .### "; ss8
  y8 = y8 + Scale
  LINE (x8l, y8)-(x8r, y8), 4
 CASE IS ¡= .9
  s9 = s9 + 1
  ss9 = s9 / num
  LOCATE 15, 60: PRINT USING " .### "; ss9
  y9 = y9 + Scale
  LINE (x9l, y9)-(x9r, y9), 4
 CASE IS ¡= 1!
  s10 = s10 + 1
  ss10 = s10 / num
  LOCATE 15, 66: PRINT USING " .### "; ss10
  y10 = y10 + Scale
  LINE (x10l, y10)-(x10r, y10), 4
 END SELECT
NEXT i

'Draw the 0.1 line

LINE (0, 1.005)-(1.025, 1.005)

Mean = Mean / num
Sigma = SQR((SumSq - num * Mean ^ 2) / num)
LOCATE 20, 20: PRINT "
     "
LOCATE 19, 6: PRINT USING "The mean is #.## and standard
deviation is #.##"; Mean; Sigma
LOCATE 21, 6: PRINT "What is your impression of this random
number generator?"

'Do it again
LOCATE 23, 16: INPUT "Would you like to test the generator again
"; Ans$
IF UCASE$(Ans$) = "Y" THEN
 Ans$ = " "
 FOR i = 19 TO 25
  LOCATE i - EGAAdj: PRINT "
                 "
 NEXT i
 GOTO Restart
END IF
CLS
END SUB
```

MODPDET

```
'/////////////////////////////////////////////////////////////
' Program: PDETDEMO
' Date: 19 Oct 91
'
' Source: PACAM & Hartman
'
' Description: This program demonstrates the probability
ofdetecting
'            a target given an aspect ratio of one (head on
basecase)
'
' Called By:
'
' Program Calls:
'
'////////////////////////////////////////////////////////////
DECLARE FUNCTION PDet! (i!, RNom!, LnConst!)
DECLARE SUB PDETDEMO (MonMode%)

PDETDEMO MonMode%

END

FUNCTION PDet (i, RNom, LnConst)
PDet = EXP(LnConst * (i / RNom) ^ 4)
END FUNCTION

SUB PDETDEMO (MonMode%)
'
' Input Parameters for Cookie Cutter

R = 40000

' Input Parameters for Probability model

RMax = 97200!      ' Rng for 0.5 probability of detection
OffSet = .3 * RMax  ' Used for the graph display
Aspect = 1!
Pwr = .25          ' Sqr*Sqr
SizeFact = 1!
TgtRng = 200000!    ' Target Range
Res = 10           ' Resolution of Graph


' Intermediate Calculations for Proability model

Rfact = Aspect * SizeFact ^ Pwr
RNom = Rfact * RMax
LnConst = LOG(.5)
'MaxAx = RNom * (LOG(.001) / LOG(.5)) ^ .25

CLS

' Establish screen

IF (MonMode% = 12) THEN
  VIEW (319, 1)-(639, 241), 3, 4'Coordinates of screen disignated
for graphics
  WINDOW (-1 * OffSet, 1.1)-(TgtRng + (TgtRng * .1),
-.1)'ViewportCoordinates
ELSE
  VIEW (319, 1)-(639, 210), 3, 4'Coordinates of screen disignated
for graphics
  WINDOW (-1 * OffSet, 1.1)-(TgtRng + (TgtRng * .1),
-.1)'ViewportCoordinates
END IF

' Print Text

LOCATE 1, 7: PRINT "Detection Demonstration"
LOCATE 2, 8: PRINT "(Cookie Cutter Method)"

' Temp text for cookie cutter, may read a data file later


' Print Axis
```

```
LINE (0, 0)-(TgtRng, 0)'X Axis
LINE (0, 0)-(0, 1!)

' Print lables and Values

LOCATE 2, 42, 0
PRINT "1.0"
LOCATE 15, 71, 0
PRINT TgtRng
LOCATE 7, 41, 0
PRINT "P(d)"
LOCATE 15, 55, 0
PRINT "Target Range"
LOCATE 15, 46, 0
PRINT "0"
LOCATE 14, 43
PRINT "0"

' Graph step function
LINE (0, 1!)-(RMax, 1!)
LINE (RMax, 1!)-(RMax, 0!)

LOCATE 4, 2: PRINT "Maximum Range set for 97200"

Restart1:

LOCATE 6, 2: INPUT "Enter range of target (Ft): "; R

IF R ¿ 97200 THEN
 PSET (R, 0), 4
 CIRCLE (R, 0), 500, 4
 LOCATE 9, 2: PRINT "Target proability of detection is 0.0."
 LOCATE 11, 2: PRINT "Target was not detected."
ELSE
 PSET (R, 1)
 CIRCLE (R, 1), 500, 4
 LOCATE 9, 2: PRINT "Target proability of detection is 1.0"
 LOCATE 11, 2: PRINT "Target was detected"
END IF

' Prompt for another sample

LOCATE 13, 2: INPUT "Select another range (Y/N): "; Ans$
IF UCASE$(Ans$) = "Y" THEN
  FOR i = 6 TO 13
    LOCATE i: PRINT "                              "
  NEXT i
  GOTO Restart1
END IF

' clear out cookie cutter text

FOR i = 2 TO 13
    LOCATE i: PRINT "                              "
NEXT i


' clear out graphics

LOCATE 1, 7: PRINT "Detection Demonstration"
LOCATE 2, 4: PRINT "(Continuous Detection Model)"
LOCATE 17, 45, 0
PRINT "Probability Detection Curve"

' Now plot the function.

FOR i = 1 TO TgtRng STEP Res
PSET (i, PDet(i, RNom, LnConst))
NEXT i

' Do the interactive portion
10
LOCATE 4, 2
INPUT "Enter a Target Range "; R

Pd = 0
Pd = PDet(R, RNom, LnConst)
```

```
LOCATE 7, 2: PRINT USING "Probabilty of detection is #.####"; Pd

' Locate an tag the point

PSET (R, Pd), 4
CIRCLE (R, Pd), 500, 4

Test = RND(1)

LOCATE 9, 2: PRINT USING "Random number generated is #.####";
Test

LOCATE 11, 2
IF Test ¡= Pd THEN
  PRINT "The target has been detected"
ELSE
  PRINT "The target was not detected"
END IF

' Do it again

LOCATE 14, 2: INPUT "Select another range (Y/N) "; Ans$
IF UCASE$(Ans$) = "Y" THEN
  FOR i = 2 TO 16
   LOCATE i
   PRINT "                                     "
  NEXT i
  GOTO 10
END IF
END SUB
```

MODCEP

```
'////////////////////////////////////////////////
' Program: ModCEP
' Date: 15 Oct 91
'
' Source: Law and Kelton, "Simulation Modeling & Analysis", 2nd
ed.
'
' Description: This program demonstrates the simulation of the
testing
'           of a weapon system firing on a target "num" times.
'           Based on the results, CEP is collected and
displayed
'
'           The generation on the random normal deviate is
based
'           based on the Polar Method.
' Called By:
'
' Program Calls:  CEPDEMO
'
'//////////////////////////////////////////////////////
DECLARE SUB CEPDEMO (MonMode%)
DECLARE SUB PhitPlot (cep)
DECLARE FUNCTION Phit (r, cep)

'*************
'SCREEN 9
'MonMode% = 9
'**************

CEPDEMO MonMode%
LOCATE 12, 20: RANDOMIZE
END

SUB CEPDEMO (MonMode%)
'
DIM RNX1(2), RNX2(2), Sigma(2), RNRadius(2)

LOCATE 1, 6: PRINT "Circular Error Probable"
LOCATE 2, 7: PRINT "(CEP); Demonstration"


' Screen set-up

5

IF (MonMode% = 12) THEN
  VIEW (319, 1)-(639, 241), 3, 4'Coordinates of screen disignated
for graphics
'Axis labels

LOCATE 17, 40: PRINT "-10 ——— Horizontal Error ——— 10"
LOCATE 1, 37: PRINT "10"
LOCATE 2, 38: PRINT "V"
LOCATE 3, 38: PRINT "E"
LOCATE 4, 38: PRINT "R"
LOCATE 5, 38: PRINT "T"
LOCATE 6, 38: PRINT "I"
LOCATE 7, 38: PRINT "C"
LOCATE 8, 38: PRINT "A"
LOCATE 9, 38: PRINT "L"
LOCATE 11, 38: PRINT "E"
LOCATE 12, 38: PRINT "R"
LOCATE 13, 38: PRINT "R"
LOCATE 14, 38: PRINT "O"
LOCATE 15, 38: PRINT "R"
LOCATE 16, 37: PRINT "-10"

ELSE

EGAAdj = 2
VIEW (319, 1)-(639, 165), 3, 4'Coordinates of screen disignated
for graphics
LOCATE 13, 40: PRINT "-10 ——— Horizontal Error ——— 10"
LOCATE 1, 37: PRINT " 10"
LOCATE 2, 38: PRINT "V"
LOCATE 3, 38: PRINT "E"
LOCATE 4, 38: PRINT "R"
LOCATE 5, 38: PRINT "T"
LOCATE 7, 38: PRINT "E"
LOCATE 8, 38: PRINT "R"
LOCATE 9, 38: PRINT "R"
LOCATE 10, 38: PRINT "O"
LOCATE 11, 38: PRINT "R"
LOCATE 12, 37: PRINT "-10"
END IF

WINDOW (-10, 10)-(10, -10)   'Coordinates disignated within the
viewport
LINE (-10, 0)-(10, 0), 0       'Horizontal Axis
LINE (0, 10)-(0, -10), 0       'Vertical Axis

c = 4
LINE (-9, -1)-(-5, 2), c       'Draw AC body
LINE (-5, 2)-(5, 2), c
LINE (5, 2)-(6.5, 5), c
LINE (6.5, 5)-(9, 5), c
LINE (9, 5)-(8, -1), c
LINE (8, -1)-(-9, -1), c
CIRCLE (-4.5, 1.2), .75, c, , , .35  'Cockpit
CIRCLE (1, 0), 3.5, c, , , .09       'Wing
CIRCLE (7.6, 3.5), 1.2, c, , , .08      'Wing

' Future user input data.  Use the data below for auto
demonstration

Kfac = 1                     'K factor for expansion
mean = 1.5                    'Mean point of impact, Bias
Sigma(1) = 1                  'Dispersion in X, circular only
Sigma(2) = 1                  'Dispersion in Y    "     "

'Collect user input

wrong:

LOCATE 4, 2: PRINT "Enter number of shots to"
LOCATE 5, 2: INPUT "fire (5000 Max)"; Num

IF Num ¿ 5000 THEN
 BEEP: LOCATE 6, 2: PRINT "Too Many"
 LOCATE 3, 2: PRINT "                       "
 LOCATE 4, 2: PRINT "                       "
 GOTO wrong
END IF

' Define constants
DistSum = 0
DistSumSq = 0

FOR i = 1 TO Num
FOR j = 1 TO 2

' 1st, generate two IID U(0,1) random variables

10
V1 = 2 * RND(1) - 1   ' On the average, this should generate 50%
V2 = 2 * RND(1) - 1   ' positive and 50% negative values.

W = V1 * V1 + V2 * V2
  IF W ¿ 1! THEN
  GOTO 10
  ELSE
    RNX1(j) = (V1 * (SQR((-1 * 2 * LOG(W)) / W)) * Sigma(j) +
mean) * Kfac
    RNX2(j) = (V1 * (SQR((-1 * 2 * LOG(W)) / W)) * Sigma(j) +
mean) * Kfac
  END IF

'RNX1 and RNX2 are IID N(0,Sigma) random variates
'Need to make to separate passes to create independence

NEXT j

CIRCLE (RNX1(1), RNX1(2)), .1   'Use only one pair
```

```
'CIRCLE (RNX2(1), RNX2(2)), .1
COLOR

Xdist = Xdist + RNX1(1)
Ydist = Ydist + RNX1(2)

dist = SQR(RNX1(1) ^ 2 + RNX1(2) ^ 2)

DistSum = DistSum + dist

DistSumSq = DistSumSq + dist ^ 2

NEXT i

'Calculate the mean impact point

MIP = DistSum / Num
MIPsg = SQR((DistSumSq - (DistSum ^ 2) / Num) / (Num - 1))

Meanx = Xdist / Num
Meany = Ydist / Num

' Calcualte the boundry described by the CEP
'
' Note: At this time, cnly circular error probable calculations
'       are performed.  Sometime in the future, the ellipsoid may
'       be calculated.
'

cep = 1.1774 * MIPsg  'Ref: Hartman 7-5, No Bias, symetrical
dispersion


LOCATE 9 - EGAAdj, 2: PRINT USING "Est. horizontal error is
#.###"; Meanx
LOCATE 11 - EGAAdj, 2: PRINT USING "Est. vertical error is
#.###"; Meany
LOCATE 13 - EGAAdj, 2: PRINT USING "Est. CEP #.###"; cep
"LOCATE 15 - EGAAdj, 2: PRINT USING "Radial standard deviation
is #.###"; MIPsg

LOCATE 19 - EGAAdj, 2: PRINT "To display area enclosed by the
CEP, press the C key now."

DO UNTIL INKEY$ = "C" OR INKEY$ = "c": LOOP

DO WHILE rr ¡ cep
  CIRCLE (Meanx, Meany), rr, 7     'Draw CEP circular
  rr = rr + .01
LOOP
  CIRCLF (Meanx, Meany), cep, 0    'Draw CEP circular

LOCATE 21 - EGAAdj, 2: PRINT "To display the mean impact point
(MIP), press the M key now."

DO UNTIL INKEY$ = "M" OR INKEY$ = "m": LOOP

CIRCLE (Meanx, Meany), .05, 0

LOCATE 23 - EGAAdj, 2: PRINT "Given the MIP, and assumming the
aim point was the origin, it appears our "
LOCATE 24 - EGAAdj, 2: PRINT "weapon has a systematic error.
Press the S key to display the error"

DO UNTIL INKEY$ = "S" OR INKEY$ = "s": LOOP

LINE (0, 0)-(Meanx, Meany), 0
LINE (.05, .05)-(Meanx + .05, Meany + .05), 0'Thicken the line

IF MonMode% = 12 THEN
LOCATE 28, 20: PRINT "Press spacebar to continue"
ELSE
LOCATE 23, 20: PRINT "Press spacebar to continue"
END IF

"pause
DO UNTIL INKEY$ = " ": LOOP
```

```
" Begin PHit graph demonstration

CLS 0

LOCATE 1, 7: PRINT "CEP Demonstration"
LOCATE 3, 1: PRINT USING "Using the calculated CEP of #.###"; cep
LOCATE 4, 1: PRINT "and correcting for bias, a graph "
LOCATE 5, 1: PRINT "of the 'probability of hit' is "
LOCATE 6, 1: PRINT "is generated."

' Establish screen

IF (MonMode% = 12) THEN
  VIEW (325, 1)-(639, 241), 3, 4'Coordinates of screen disignated
for graphics
'Axis labels

LOCATE 17, 41: PRINT "0 ——— TARGET RADIUS ——— 5"
LOCATE 1, 37: PRINT
LOCATE 2, 38: PRINT "1.0"
LOCATE 3, 38: PRINT
LOCATE 4, 38: PRINT "P"
LOCATE 5, 38: PRINT "R"
LOCATE 6, 38: PRINT "O"
LOCATE 7, 38: PRINT "B"
LOCATE 8, 38: PRINT
LOCATE 9, 38: PRINT "O"
LOCATE 10, 38: PRINT "F"
LOCATE 11, 38: PRINT
LOCATE 12, 38: PRINT "H"
LOCATE 13, 38: PRINT "I"
LOCATE 14, 38: PRINT "T"
LOCATE 15, 38: PRINT
LOCATE 16, 37: PRINT " 0"

ELSE

EGAAdj = 2
VIEW (319, 1)-(639, 165), 3, 4'Coordinates of screen disignated
for graphics
LOCATE 13, 41: PRINT "0 ——— TARGET RADIUS ———10"
LOCATE 1, 37: PRINT "1.0"
LOCATE 2, 38: PRINT
LOCATE 3, 38: PRINT "H"
LOCATE 4, 38: PRINT "I"
LOCATE 5, 38: PRINT "T"
LOCATE 6, 38: PRINT
LOCATE 7, 38: PRINT "P"
LOCATE 8, 38: PRINT "R"
LOCATE 9, 38: PRINT "O"
LOCATE 10, 38: PRINT "B"
LOCATE 11, 38: PRINT
LOCATE 12, 38: PRINT "0"
END IF

WINDOW (0, 1.1)-(5, 0)   'Coordinates disignated within the
viewport
LINE (0, 1)-(5, 1)       '1.0 Line

newcep:

' Now plot the function.

PhitPlot cep

' Do the interactive portion

hittst:

LOCATE 10 - EGAAdj, 2
INPUT "Enter a Target Radius "; r

Pk = 0
Pk = Phit(r, cep)
```

```
LOCATE 12 - EGAAdj, 2: PRINT USING "Probabilty of hit is #.####";
Pk

' Locate an tag the point

PSET (r, Pk), 4
CIRCLE (r, Pk), .1, 4

Test = RND(1)

LOCATE 14 - EGAAdj, 2: PRINT USING "Random number generated is
#.####"; Test

LOCATE 16 - EGAAdj, 2
IF Test ¡= Pk THEN
  PRINT "The target has been hit"
ELSE
  PRINT "The target was not hit"
END IF

LOCATE 20 - EGAAdj, 2: PRINT "Why is the probability ˜f hit
increasing as the target size"
LOCATE 21 - EGAAdj, 2: PRINT "approaches and exceeds the CEP?"

' Do it again

Ans$ = " "
LOCATE 23 - EGAAdj, 2: INPUT "Select another target radius (Y/N)
"; Ans$
IF UCASE$(Ans$) = "Y" THEN
LOCATE 23 - EGAAdj, 2: PRINT "
    "
  FOR i = 8 TO 16
   LOCATE i - EGAAdj
   PRINT "                                  "
  NEXT i
  GOTO hittst
END IF

Ans$ = " "
LOCATE 23 - EGAAdj, 2: PRINT "
    "
LOCATE 23 - EGAAdj, 2: INPUT "Select another CEP value (Y/N) ";
Ans$
IF UCASE$(Ans$) = "Y" THEN

LOCATE 25 - EGAAdj, 2: INPUT "Enter new CEP value "; cep
LOCATE 23 - EGAAdj, 2: PRINT "
    "
LOCATE 25 - EGAAdj, 2: PRINT "
    "
LOCATE 3, 1: PRINT USING "Using the calculated CEP of #.###"; cep

  FOR i = 8 TO 16
   LOCATE i - EGAAdj
   PRINT "                                  "
  NEXT i
  GOTO newcep:
END IF

END SUB

FUNCTION Phit (r, cep)

"Reference Przemieniecki's Book Mathematical Methods of
"Defense analysis. Pg 3.25, Eq 3.55

Phit = 1 - EXP(-.693147 * (r ^ 2 / cep ^ 2))

END FUNCTION

SUB PhitPlot (cep)

FOR r = 0 TO 5 STEP .005
  Pk = Phit(r, cep)
  PSET (r, Pk)
NEXT r
```

```
END SUB
```

MODACS

```
' $DYNAMIC
********************************
Air Combat Simulator

Author: Capt R. Moore

Date: 15 Feb 92

********************************
DECLARE SUB title ()
DECLARE SUB Init ()
DECLARE SUB pause ()
DECLARE SUB mainmenu ()
DECLARE SUB MainLoop (MonMode%)
DECLARE SUB missile (I%)
DECLARE SUB Update (I%)
DECLARE SUB Klaxon (Hi%, Lo%)
DECLARE SUB MASTDAT ()
DECLARE SUB TgtSel (Dist)
DECLARE SUB Station ()
DECLARE SUB Refresh ()
DECLARE SUB BLMenu ()
DECLARE SUB Frame (Left%, Right%, Top%, Bottom%)
DECLARE SUB ANLST ()
DECLARE SUB GetData ()
DECLARE SUB USERDAT ()
DECLARE SUB Grid ()
DECLARE SUB BBase ()
DECLARE SUB ACIcon ()
DECLARE SUB ACDraw (AC%)
DECLARE SUB Adv (I%)
DECLARE SUB ZoomOut ()
DECLARE SUB ZoomIn ()
DECLARE SUB Headline ()
DECLARE SUB MSLDraw (AC%, Msl%)
DECLARE SUB MSLIcon ()
DECLARE SUB MSLAdv (AC%, Msl%)
DECLARE SUB Mslterm (AC%)
DECLARE SUB PKill (AC%, Hx, Hy)
DECLARE SUB Destroy (ITgt%)
DECLARE SUB detect ()
DECLARE SUB Explode (Tgt%)
DECLARE SUB ScrnSet (BadScrn%, MonMode%)
DECLARE SUB bing (I%)
DECLARE SUB LoadGo ()
DECLARE SUB msg (MsgNum%)
DECLARE SUB Mmsg (MsgNum%, AC%, Msl%, ACT%)
DECLARE SUB ErrMsg (I%)
DECLARE SUB Sigevnt ()
DECLARE SUB Detlrpt ()
DECLARE SUB ACperf ()

DECLARE FUNCTION tkag! (Blue%, Red%)
DECLARE FUNCTION MHead! (Ax!, Ay!, Bx!, By!)
DECLARE FUNCTION Dist! (dx1!, dx2!, dy1!, dy2!)
DECLARE FUNCTION DMAX! (F1!, F2!)
DECLARE FUNCTION CLDES2! (CLMax!, THAB, SS!, ICL%, IM%, I%)
DECLARE FUNCTION RHO (z)  ' Density of air at alt Z
DECLARE FUNCTION SpdS (z)  ' Speed of sound at alt Z
DECLARE FUNCTION DMIN (F1, F2)  'Find minimum
DECLARE FUNCTION TrackAngle (I%)

COMMON SHARED MonMode%, mennum%
COMMON SHARED /Messages/ Com$, TNOW%
COMMON SHARED /Aircraft/ NumAC%

DIM SHARED Basex, Basey, GCIRg     'Base Attributes
DIM SHARED Format%, AScreen%
DIM SHARED Scale
DIM SHARED Wx, Wy, AR
DIM SHARED endsim%
DIM SHARED StatPtx, StatPty, OEF
DIM SHARED AltArg%, MachArg%, CLArg%, z, Q
DIM SHARED SysStat$, errflg%
DIM SHARED otpt%
' Define Constants
```

```
CONST Pi = 3.14159
CONST TwoPi = 2! * Pi

' Define arrays indices

NumAC% = 4
NumMSL% = 4
AltArg% = 8
MachArg% = 14
CLArg% = 14

' Dimension shared arrays
' Icon dimensions
DIM SHARED X(25), Y(25)
DIM SHARED xxold(NumAC%, 25), yyold(NumAC%, 25)
DIM SHARED OldEndTop(NumAC%), OldEndBot(NumAC%)
DIM SHARED Mode%(4)

' Screen variables
DIM SHARED MaxScrX%, MaxScrY%

'Aircraft locations, headings, dist, track angles, and targets
DIM SHARED ACx(NumAC%), ACy(NumAC%), Tgtx(NumAC%), Tgty(NumAC%)
DIM SHARED ACHead(NumAC%), OldHead(NumAC%)
DIM SHARED TDist((NumAC% + 1), (NumAC% + 1))
DIM SHARED Stat$(NumAC%), IStat%(NumAC%)
DIM SHARED TrkAng(NumAC%)
DIM SHARED Tgt$(4), TgtDetLvl%(NumAC%), ITgt%(NumAC%)

'Senario and AC performance
DIM SHARED Dat(100), Vp(NumAC%)
DIM SHARED V(NumAC%), TV(NumAC%), VelMax(NumAC%)
DIM SHARED AC$(NumAC%), WgArea(NumAC%), MaxWt(NumAC%),
MinWt(NumAC%)
DIM SHARED AC%(NumAC%), MaxAlt(NumAC%), MinAlt(NumAC%),
MaxETA(NumAC%)
DIM SHARED ThrKF(NumAC%), DragKF(NumAC%)
DIM SHARED Alt(AltArg%), MMach(NumAC%, AltArg%)
DIM SHARED MachDat(NumAC%, MachArg%), MaxCL(NumAC%, MachArg%)
DIM SHARED ThrMP(AltArg%, MachArg%), ThrAB(AltArg%, MachArg%)
DIM SHARED Drag(CLArg%, MachArg%)
DIM SHARED THAB(NumAC%), THMP(NumAC%), SusG
DIM SHARED TurnRate(NumAC%), RadTurn(NumAC%, 2), TTurn(NumAC%),
RTmiles(NumAC%, 2)

'Fuel status
DIM SHARED FCMP(AltArg%, MachArg%), FCAB(AltArg%, MachArg%)
DIM SHARED MPFC(NumAC%, 2), ABFC(NumAC%, 2)
DIM SHARED FuelLbs(NumAC%), Fuel(NumAC%), bingo(NumAC%),
FCKF(NumAC%)
DIM SHARED FuelRem(NumAC%), Ftemp(NumAC%), BurnRate(NumAC%)

'Sensors
DIM SHARED OptRng(NumAC%), RadRng(NumAC%)

NumMSL% = 4
'Missiles are unique enities but are identified with an AC
DIM SHARED NumMSL%(NumAC%), MsLV(NumAC%), MsLSenRg(NumAC%)
DIM SHARED MX(10), MY(10)
DIM SHARED mxxold(NumAC%, NumMSL%, 10), myyold(NumAC%, NumMSL%,
10)
DIM SHARED MSLHead(NumAC%, NumMSL%)
DIM SHARED MSLx(NumAC%, NumMSL%), MSLy(NumAC%, NumMSL%)
DIM SHARED MF%(NumAC%), MTNOW%(NumAC%, NumMSL%)
DIM SHARED Pk(NumAC%)
DIM SHARED IMslStat%(NumAC%, 5)
DIM SHARED IMTgt%(NumAC%, 5)

'Station
DIM SHARED DeltaCir(NumAC%)


' Define Keys and initialize
ON KEY(1) GOSUB F1KEY
ON KEY(3) GOSUB F3KEY
ON KEY(4) GOSUB F4KEY
ON KEY(5) GOSUB F5KEY
ON KEY(6) GOSUB F6KEY
ON KEY(7) GOSUB F7KEY

CLS
LOCATE 10, 10: INPUT "Enter Monitor type, VGA or EGA (EGA): ";
MonType$
IF (UCASE$(MonType$) = "VGA") THEN
   MonMode% = 12
ELSE
   MonMode% = 9
END IF
CLS


title
mainmenu
Last: END

' On Key Subs
F1KEY:  'Pause mode
DO
ch$ = RIGHT$(INKEY$, 1)'Breakout the ¡ character
  LOOP UNTIL ch$ = "¡"  '" ¡" is the code for the F2 key
RETURN

F3KEY:  ' Analyst screen mode
  Delay% = ADelay%
  AScreen% = 1
  KEY(4) ON
  KEY(3) OFF
  KEY(5) OFF
  KEY(6) OFF
RETURN

'**** Do not change the order of the F4 Key ****
F4KEY:  ' Graphics screen mode, must keep the order of these
events
  CLS 0
  Format% = 0   'Controls the redrawing of the anal scrn boxes
  AScreen% = 0  'Used to direct passes through main loop
  SCREEN MonMode%
  BLMenu
  Refresh  'Refreshes the screen
  Refs = TNOW%
  Delay% = GDelay%
  KEY(3) ON
  KEY(4) OFF
  KEY(5) ON
  KEY(6) ON
RETURN

F5KEY:
  ZoomIn
RETURN

F6KEY:
  ZoomOut
RETURN

F7KEY:
  msg 10000
  TNOW% = endsim%
RETURN

fileerr:
 BEEP
 ErrMsg 1
 errflg% = 1
 LOCATE 14, 4: PRINT "
      "
RESUME NEXT

SUB ACDraw (AC%)

DIM xx(25), yy(25)
DIM Segments%(NumAC%)
```

```
' Compute radar fan coordinates based on 60 deg sweep ang.
' Radar fan is based on miles and is scaled to the map
' the A/C incons are not. Therefore, the development of the
' fan incon is slightly different and will not be affected
' by the incon scaling factor k.

IF (IStat%(AC%) = 2 OR IStat%(AC%) = 7) GOTO enddraw

IF (RadRng(AC%) = 0) THEN
  Segments%(AC%) = 13
  GOTO Skip1
ELSE
  Segments%(AC%) = 16
END IF


X(15) = X(14) + RadRng(AC%) * COS(.523599)
Y(15) = RadRng(AC%) * SIN(.523599)
X(17) = X(16) + RadRng(AC%) * COS(-.523599)
Y(17) = RadRng(AC%) * SIN(-.523599)

'Erase old image of radar frontier

 CIRCLE (xxold(AC%, 14), yyold(AC%, 14)), RadRng(AC%), 3,
OldHead(AC%), OldEndTop(AC%)
 CIRCLE (xxold(AC%, 14), yyold(AC%, 14)), RadRng(AC%), 3,
OldEndBot(AC%), OldHead(AC%)

Skip1:

FOR I% = 2 TO Segments%(AC%) STEP 2  'Erase old image
  LINE (xxold(AC%, I%), yyold(AC%, I%))-(xxold(AC%, (I% + 1)),
yyold(AC%, (I% + 1))), 3
NEXT I%

'Orientate image to heading

FOR I% = 2 TO Segments%(AC%) + 1

 xx(I%) = (X(I%) * COS(ACHead(AC%)) - Y(I%) * SIN(ACHead(AC%)))
+ ACx(AC%)
 yy(I%) = (X(I%) * SIN(ACHead(AC%)) + Y(I%) * COS(ACHead(AC%)))
+ ACy(AC%)

' Capture current values

 xxold(AC%, I%) = xx(I%): yyold(AC%, I%) = yy(I%)

NEXT I%

' Assign color to side


SELECT CASE AC% 'Establish colors
 CASE 1
  Side% = 1  '1st Blue aircraft
 CASE 2
  IF (IStat%(2) ¡¿ 1) THEN
   Side% = 0
   Mode%(AC%) = 0
  ELSE
   Side% = 4  '1st Red aircraft
   Mode%(AC%) = 4
  END IF
 CASE 3
  Side% = 1  '2nd Blue aircraft
 CASE 4
  IF (IStat%(4) ¡¿ 1) THEN
   Side% = 0
   Mode%(AC%) = 0
  ELSE
   Side% = 4  '2nd Red aircraft
   Mode%(AC%) = 4
  END IF
END SELECT

' Draw Image
```

```
FOR I% = 2 TO Segments%(AC%) STEP 2
  LINE (xx(I%), yy(I%))-(xx(I% + 1), yy(I% + 1)), Side%
NEXT I%

IF RadRng(AC%) = 0 THEN GOTO enddraw

' Draw the radar fan frontier.  60 degree sector of a circle
center on
' the nose of the aircraft.  The circle function is will not
except
' a start or end parameter greater the two pi or less than 0.
Both
' heading and endcirtop and bot have to be checked.  The heading
is
' checked in the mainloop when heading is calculated.

EndCirTop = ACHead(AC%) + .523599
EndCirBot = ACHead(AC%) - .523599

IF (EndCirTop ¿= TwoPi) THEN
  EndCirTop = EndCirTop - TwoPi
END IF

IF (EndCirTop ¡= 0) THEN
  EndCirTop = EndCirTop + TwoPi
END IF

IF (EndCirBot ¿= TwoPi) THEN
  EndCirBot = EndCirBot - TwoPi
END IF

IF (EndCirBot ¡= 0) THEN
  EndCirBot = EndCirBot + TwoPi
END IF

OldEndTop(AC%) = EndCirTop  'Capture old radar fan image
OldEndBot(AC%) = EndCirBot

IF (AC% = 1 OR AC% = 3) THEN
 SELECT CASE TgtDetLvl%(AC%)
   CASE IS = 0            '
     Mode%(AC%) = 1
   CASE IS = 1
     Mode%(AC%) = 1
   CASE IS = 2
     Mode%(AC%) = 14
   CASE IS = 3
     Mode%(AC%) = 10
  END SELECT
END IF

CIRCLE (xx(14), yy(14)), RadRng(AC%), Mode%(AC%), ACHead(AC%),
EndCirTop
CIRCLE (xx(14), yy(14)), RadRng(AC%), Mode%(AC%), EndCirBot,
ACHead(AC%)

enddraw:

END SUB

SUB ACIcon

' Center:             X(1), Y(1)
' Nose:               X(2,4), Y(2,4)
' Right wing tip       X(5), Y(5)
' Back of Wing         X(6,7), Y(6,7)
' Tail Center point    X(8,10), Y(8,10)
' Left Horiz tip       X(9), Y(9)
' Right Horiz tip      X(11), Y(11)
' Back of tail        X(12,13), Y(12,13)
' Radar Fan sides      X(14,15), Y(14,15)

' Setup Icon.  Both aircraft are the same shape, only difference
is color

X(2) = 4:   Y(2) = 0
```

```
X(3) = -1: Y(3) = 3
X(4) = 4:  Y(4) = 0
X(5) = -1: Y(5) = -3
X(6) = -1: Y(6) = 3
X(7) = -1: Y(7) = -3
X(8) = -1: Y(8) = 0
X(9) = -4: Y(9) = 2
X(10) = -1: Y(10) = 0
X(11) = -4: Y(11) = -2
X(12) = -4: Y(12) = 2
X(13) = -4: Y(13) = -2
X(14) = 4: Y(14) = 0
X(16) = 4: Y(16) = 0


' Scale adjustment used for calibration

k = 1   '((1! / Scale) * .7)

FOR I = 2 TO 14
  X(I) = X(I) * k
  Y(I) = Y(I) * k
NEXT I
  X(16) = X(16) * k
  Y(16) = Y(16) * k

' Compute radar fan coordinates based on 60 deg sweep ang.
' Radar fan is based on miles and is scaled to the map
' the A/C incons are not. Therefore, the development of the
' fan incon is slightly different and will not be affected
' by the incon scaling factor k.

RadRng% = RadRng(1) / 5280
X(15) = X(14) + RadRng% * COS(.523599)
Y(15) = RadRng% * SIN(.523599)
X(17) = X(16) + RadRng% * COS(-.523599)
Y(17) = RadRng% * SIN(-.523599)

END SUB

SUB Adv (I%)

IF (IStat%(I%) = 2 OR IStat%(I%) = 7) GOTO endadv

'Cal fuel consumption rates and turn rate based on current vel.

'If Red has been detected or Blue is in pursuit but not within
'firing range cruise at high velocity

IF I% = 2 OR I% = 4 GOTO redcase

' ————————Blue Case————————
IF (IStat%(I%) = 4 AND TgtDetLvl%(I%) ; 3) THEN
   TV(I%) = Vp(I%)
   BurnRate(I%) = ABFC(I%, 2)
   TurnRate(I%) = (TV(I%) * 3600 / RTmiles(I%, 2)) / 3600'Radians
per second
   GOTO advance
ELSE
   TV(I%) = V(I%)
   BurnRate(I%) = MPFC(I%, 1)
   TurnRate(I%) = (TV(I%) * 3600 / RTmiles(I%, 1)) / 3600'Radians
per second
   GOTO advance
END IF

redcase:

' ———————— Red Case ————————-
IF (IStat%(I%) = 1) THEN
   TV(I%) = Vp(I%)
   BurnRate(I%) = ABFC(I%, 2)
   TurnRate(I%) = (TV(I%) * 3600 / RTmiles(I%, 2)) / 3600'Radians
per second
ELSE
   TV(I%) = V(I%)
   BurnRate(I%) = MPFC(I%, 1)
  TurnRate(I%) = (TV(I%) * 3600 / RTmiles(I%, 1)) / 3600'Radians
per second
END IF

advance:

'Find new coordinates after advance
 ACy(I%) = ACy(I%) + TV(I%) * SIN(ACHead(I%))
 ACx(I%) = ACx(I%) + TV(I%) * COS(ACHead(I%))

'Burn fuel compute percentage remaining

Ftemp(I%) = Ftemp(I%) + BurnRate(I%)
FuelRem(I%) = (1 - (Ftemp(I%) / Fuel(I%))) * 100!
 IF ((Fuel(I%) - Ftemp(I%)) ¡= bingo(I%) AND IStat%(I%) ¡¿ 6 AND
TgtDetLvl%(I%) ¡ 2) THEN

'If a target is within radar range hold off bingo.

  bing (I%)
 END IF

 IF (FuelRem(I%) ¡= 0!) THEN
  Destroy I%
 END IF

endadv:
END SUB

SUB ANLST
'////////////////////////////////////////////////////////////////////-
' Program ANALYST SCREEN
' Author: Capt R. Moore
' Date: 26 Oct 92
'
' Source: Maj Garrambone and Head-Space
'
' Description:  This subprogram is displayed when the user
selects
'               it in lieu of the graphic image.  It is call from
'               the main loop.
'
' Called by Sub DisplayMenuBox
'
'////////////////////////////////////////////////////////////////////-
IF AScreen% = 0 GOTO endanlst
DIM ANG(NumAC%)

IF (Format% = 1) GOTO Update
SCREEN 0
BLMenu
' Frame in the data boxes
CLS 1: CLS 2
Frame 1, 28, 2, 21
Frame 28, 41, 2, 21
Frame 41, 54, 2, 21
Frame 54, 67, 2, 21
Frame 67, 80, 2, 21
Bottom% = BLine% - 1
'Frame2 1, 80, 19, Bottom%

' Print column headings
COLOR 14
LOCATE 3, 5: PRINT "Aircraft Attributes"
COLOR 3
LOCATE 3, 31: PRINT "Blue 1"
LOCATE 3, 57: PRINT "Blue 2"
COLOR 4
LOCATE 3, 44: PRINT "Red  1"
LOCATE 3, 70: PRINT "Red 2"
COLOR 7
' Print row headings
'LOCATE 7, 3: PRINT "Altitude (Feet)"

COLOR 15
LOCATE 5, 3: PRINT "Posture/Status"
LOCATE 7, 3: PRINT "X coordinate location"
LOCATE 8, 3: PRINT "Y coordinate location"

LOCATE 9, 3: PRINT "Heading (Deg)"
LOCATE 10, 3: PRINT "Velocity (MPH)"
LOCATE 11, 3: PRINT "Fuel remaining (Percent)"
LOCATE 13, 3: PRINT "Current target"
LOCATE 14, 3: PRINT "Range to target (Miles)"
LOCATE 15, 3: PRINT "Sensor in range of tgt"
LOCATE 17, 3: PRINT "Missile status"
LOCATE 18, 3: PRINT "Missile flight time (Sec)"
LOCATE 19, 3: PRINT "Missiles remaining"

Headline
Format% = 1

' End format printing

Update:
KEY(4) OFF

Pulse% = TNOW% MOD 10
IF (Pulse% = 0!) THEN
' 11 spaces avail on inner frame
Strt% = 30

FOR I% = 0 TO 3  'Can't allow upper end to be a variable
AC% = I% + 1
LOCATE 5, Strt% + (I% * 13): PRINT USING "&"; Stat$(AC%)
LOCATE 7, Strt% + (I% * 13): PRINT USING " ###.#"; ACx(AC%)
LOCATE 8, Strt% + (I% * 13): PRINT USING " ###.#"; ACy(AC%)
IF (IStat%(AC%) = 2 OR IStat%(AC%) = 6 OR IStat%(AC%) = 7) THEN
  LOCATE 9, Strt% + (I% * 13): PRINT "       "
ELSE
  LOCATE 9, Strt% + (I% * 13): PRINT USING " ###.#"; (ACHead(AC%)
* 180 / Pi)
END IF
LOCATE 10, Strt% + (I% * 13): PRINT USING " ###.#"; TV(AC%) *
3600
LOCATE 11, Strt% + (I% * 13): PRINT USING " ###.#"; FuelRem(AC%)
LOCATE 13, Strt% + (I% * 13): PRINT USING "&"; Tgt$(AC%)
IF (IStat%(AC%) = 1 OR IStat%(AC%) = 4 OR IStat%(AC%) = 0) THEN
  LOCATE 14, Strt% + (I% * 13): PRINT USING " ###.#";
Dist(ACx(AC%), ACy(AC%), Tgtx(AC%), Tgty(AC%))
ELSE
  LOCATE 14, Strt% + (I% * 13): PRINT "    0.0"
END IF
SELECT CASE TgtDetLvl%(AC%)
   CASE IS = 0
     Sensor$ = "  None   "
   CASE IS = 1
     Sensor$ = "   GCI   "
   CASE IS = 2
     Sensor$ = "  Radar  "
   CASE IS = 3
     Sensor$ = " Optical  "
END SELECT
LOCATE 15, Strt% + (I% * 13): PRINT USING "&"; Sensor$
j% = MF%(AC%)
SELECT CASE IMslStat%(AC%, j%)
   CASE IS = 0
     Msl$ = " Carried   "
   CASE IS = 1
     Msl$ = " In-Flt    "
   CASE IS = 2
     Msl$ = "  Missed   "
   CASE IS = 3
     Msl$ = " Hit Tgt   "
END SELECT

IF MF%(AC%) ¿ NumMSL%(AC%) THEN
 LOCATE 17, Strt% + (I% * 13): PRINT "   Out     "
 LOCATE 18, Strt% + (I% * 13): PRINT "    0      "
 LOCATE 19. Strt% + (I% * 13): PRINT "    0      "
ELSE
 LOCATE 17, Strt% + (I% * 13): PRINT USING "&"; Msl$
 LOCATE 18, Strt% + (I% * 13): PRINT USING " ####"; MTNOW%(AC%,
MF%(AC%))
```

```
LOCATE 19, Strt% + (I% * 13): PRINT USING "  ###"; (NumMSL%(AC%)
+ 1 - MF%(AC%))
END IF

NEXT I%
END IF
endanlst:
KEY(4) ON
END SUB

SUB BBase

'Draw GCI band

CIRCLE (Basex, Basey), GCIRg

'Draw runway

LINE (Basex - 10, Basey - 10)-(Basex + 10, Basey + 10), 0
LINE (Basex - 5, Basey - 12)-(Basex - 8, Basey + 5), 0
LINE (Basex - 12, Basey)-(Basex + 10, Basey), 0

END SUB

SUB bing (I%)
'Blue case
   Stat$(I%) = "  Bingo    "
   IStat%(I%) = 6
   Tgtx(I%) = Basex
   Tgty(I%) = Basey
   Tgt$(I%) = "   None   "
   TgtDetLvl%(I%) = 0
END SUB

SUB BLMenu

IF AScreen% = 0 THEN
  BLine% = 30
ELSE
  BLine% = 25
END IF

IF MonMode% = 9 THEN BLine% = 25

LOCATE BLine%, 1: PRINT "F1=Halt  F2=Resume  F3=Atrib Scrn
F4=Graphics  F5=Zoom-In  F6=Zoom-Out  F7=Quit":

END SUB

FUNCTION CLDES2 (CLMax, SS, THAB, ICL%, IM%, I%)
'
RR = 1 - SS
'
CDReq = THAB(I%) / (Q * WgArea(I%))'Highest Coefficient of drag
'
CDTST1 = RR * Drag((ICL% - 1), (IM% - 1)) + SS * Drag((ICL% - 1),
IM%)
CDTST2 = RR * Drag(ICL%, (IM% - 1)) + SS * Drag(ICL%, IM%)
'
IF (CDReq ¿= CDTST1) GOTO toohi
CLDES2 = 0!
GOTO 100
'
toohi:
CDHI = RR * Drag(MachArg%, IM%) + SS * Drag(MachArg%, IM%)
'
IF (CDReq ¡ CDHI) GOTO toolow
CLDES2 = CLMax
GOTO 100
toolow:
'
CDLO = RR * Drag((ICL% - 1), (IM% - 1)) + SS * Drag((ICL% - 1),
IM%)
'
TT = (CDReq - CDLO) / (CDHI - CDLO)
CLDES2T = (1! - TT) * MaxCL(I%, (ICL% - 1)) + TT * MaxCL(I%,
ICL%)
```

```
CLDES2 = DMIN(CLDES2T, CLMax)
100 'Continue
END FUNCTION

SUB Destroy (Tgt%)
IF IStat%(Tgt%) = 7 THEN
   Stat$(Tgt%) = "   Landed  "
   ACx(Tgt%) = Basex
   ACy(Tgt%) = Basey
    IF AScreen% = 0 THEN
       BLMenu
       CLS 2
       Refresh
    END IF
ELSE
   FOR a% = 1 TO 4 STEP 2   'Destroy all missile inbound on
target
     FOR m% = 1 TO 4
       IF IMTgt%(a%, m%) = Tgt% THEN
          IMslStat%(a%, m%) = 3
          TgtDetLvl%(a%) = 0
       END IF
     NEXT m%
   NEXT a%
   Explode Tgt%
   Stat$(Tgt%) = "     KIA     " 'Change status
   ACx(Tgt%) = 0
   ACy(Tgt%) = 0
   IStat%(Tgt%) = 2            'Change numerical status
 END IF
 ITgt%(Tgt%) = 0             'Delete target from target list
 Tgtx(Tgt%) = 0             'Zero out locations
 Tgty(Tgt%) = 0
 Tgt$(Tgt%) = "   None    "
 TV(Tgt%) = 0               'Zero out velocity
 FuelRem(Tgt%) = 0           'Zero out fuel
 MF%(Tgt%) = 5              'Max out missiles fired
IF (IStat%(2) = 2 AND IStat%(4) = 2) THEN
    SysStat$ = "Low"        'Return to low posture

  IF IStat%(1) = 4 THEN
    IStat%(1) = 3
  END IF

  IF IStat%(3) = 4 THEN
    IStat%(3) = 3
  END IF

    Tgt$(1) = "   None    "
    Tgt$(3) = "   None    "
    msg 10
END IF
END SUB

SUB detect

'Since ATACS permits the user to establish any initial condition
'and the location of the entities are dynamic with entities
comming in
'and out of the ranges of sensors, constant monitoring of the
ranges
'must be accomplished. The cost is a lot of computer overhead;
however,
'the return is a more beliveable simulation.

'Process:
'  1st compute distances between players and targets
'  2nd determine if GCI has detected intruder
'       make intial target assignment
'  3rd check if the intruder has been pick up by Blue
'       reassign targets if necessary
'  Last, check if within visual range
'
' IStat%(i%) Define
'    0: Red not detected
'    1: Red detected
'    2: Aircraft KIA
'    3: Blue not in pursuit
'    4: Blue in pursuit
'    6: Bingo fuel

'The order of distances is slightly stagered to facilitate
'do loop process below

TDist(2, 0) = Dist(ACx(2), ACy(2), Basex, Basey)'Red 1 to tgt
TDist(4, 0) = Dist(ACx(4), ACy(4), Basex, Basey)'Red 2 to tgt

'Compute distances between interceptors and intruders

FOR I% = 1 TO NumAC% STEP 2
  FOR j% = 2 TO NumAC% STEP 2
    TDist(I%, j%) = Dist(ACx(I%), ACy(I%), ACx(j%), ACy(j%))
  NEXT j%
NEXT I%

'Target detection level is the current mode used by the Blue
forces

'Detection by GCI

FOR I% = 2 TO NumAC% STEP 2
 IF (IStat%(I%) = 2) GOTO d2
 IF TDist(I%, 0) ;= GCIRg AND IStat%(I%) = 0 THEN
     Stat$(I%) = "Detected  "
     IStat%(I%) = 1
     Klaxon 1200, 329
 IF (IStat%(I% - 1) = 2 OR IStat%(I% - 1) = 6) GOTO d2
     Stat$(1) = "Pursuit    "
     Stat$(3) = "Pursuit    "
     IStat%(1) = 4
     IStat%(3) = 4
     SysStat$ = "Hi "
     TgtDetLvl%(1) = 1
     TgtDetLvl%(3) = 1
     TgtSel TDist   'Call targeting sub, Red 1 detected
  END IF
d2:
 NEXT I%

'Check for detection of red 1 or red 2 by blue 1 or blue 2
on-board radar
'and and optical systems

FOR I% = 1 TO NumAC% STEP 2
 IF (IStat%(I%) = 2 OR IStat%(I%) = 6) GOTO d4
  FOR j% = 2 TO NumAC% STEP 2
   IF (IStat%(j%) = 2) THEN GOTO d3

'If you include TgtDetLvl here as an additional constraint, then
'once an interceptor has a target he can't id another

    IF (TDist(I%, j%) ¡ RadRng(I%) AND tkag(I%, j%) ¡ .5236) THEN
      IF IStat%(j%) = 0 THEN
        BEEP
      END IF
      Stat$(j%) = "Detected  "
      IStat%(j%) = 1
      IF IStat%(1) ¡¿ 6 AND IStat%(1) ¡¿ 2 THEN
        IStat%(1) = 4
        Stat$(1) = "Pursuit    "
      END IF
      IF IStat%(3) ¡¿ 6 AND IStat%(3) ¡¿ 2 THEN
        IStat%(3) = 4
        Stat$(3) = "Pursuit    "
      END IF
      SysStat$ = "Hi "
      TgtDetLvl%(I%) = 2
      TgtSel TDist
    END IF
d3:
  NEXT j%
d4:
NEXT I%
```

```
'Optics

FOR I% = 1 TO NumAC% STEP 2
 FOR j% = 2 TO NumAC% STEP 2
  IF (TDist(I%, j%) ¡= OptRng(I%) AND TgtDetLvl%(I%) = 2) THEN
   TgtDetLvl%(I%) = 3
  END IF
 NEXT j%
NEXT I%

'Capture cases where targets move outside of sensor range

'Targets lost by GCI

IF SysStat$ = "Low" GOTO lossgci

' Targets lost by aircraft sensors, 1st radar, 2nd optically
' For each combination of Red and Blue distances, check out of
' bounds for sensor associated with current target.

 FOR I% = 1 TO NumAC% STEP 2
  IF IStat%(I%) = 2 GOTO d6
  IF (TgtDetLvl%(I%) ¿= 2) THEN
  Range = RadRng(I%)
  FOR j% = 2 TO NumAC% STEP 2
'    IF IStat%(j%) = 2 GOTO d5
    IF ((TDist(I%, j%) ¿ Range OR tkag(I%, j%) ¿ .5236) AND
ITgt%(I%) = j%) THEN
      TgtDetLvl%(I%) = 1
    END IF
d5:
  NEXT j%
  END IF
d6:
 NEXT I%

 FOR I% = 1 TO NumAC% STEP 2
  IF IStat%(I%) = 2 GOTO d8
  IF (TgtDetLvl%(I%) = 3) THEN
  Range = 1.01 * OptRng(I%)
  FOR j% = 2 TO NumAC% STEP 2
'   IF IStat%(j%) = 2 GOTO d7
    IF (TDist(I%, j%) ¿ Range AND ITgt%(I%) = j%) THEN
      TgtDetLvl%(I%) = 2
    END IF
d7:
  NEXT j%
  END IF
d8:
 NEXT I%

' Loss by GCI and Blue 1 or Blue 2 Radar

lossgci:

FOR I% = 1 TO NumAC% STEP 2
 IF (TgtDetLvl%(I%) = 1 AND TDist((I% + 1), 0) ¿ GCIRg) THEN
   TgtDetLvl%(I%) = 0
 END IF
NEXT I%

 IF SysStat$ = "Low" THEN
   FOR I% = 1 TO NumAC% STEP 2
     TgtDetLvl%(I%) = 0
   NEXT I%
 END IF

enddetect:

END SUB

FUNCTION Dist (dx1, dy1, dx2, dy2)
Dist = SQR((dx2 - dx1) ^ 2 + (dy2 - dy1) ^ 2)
END FUNCTION

FUNCTION DMAX (F1, F2)
 IF (F1 ¡ F2) THEN
```

```
    DMAX = F2
  ELSE
    DMAX = F1
  END IF
END FUNCTION

FUNCTION DMIN (F1, F2)
  IF (F1 ¿ F2) THEN
    DMIN = F2
  ELSE
    DMIN = F1
  END IF
END FUNCTION

SUB ErrMsg (I%)

SELECT CASE I%

CASE 1
  LOCATE 18, 15: PRINT "File not found, try again"
CASE ELSE

END SELECT

END SUB

SUB Explode (Tgt%) STATIC

X = ACx(Tgt%)
Y = ACy(Tgt%)

IF AScreen% = 1 GOTO endexplode
Radius = 20
  IF MonMode% = 9 THEN Inc# = .5 ELSE Inc# = .41
  FOR C# = 0 TO Radius STEP Inc#
    CIRCLE (X, Y), C#, 14
    NEXT C#
  FOR C# = Radius TO 0 STEP (-1 * Inc#)
    CIRCLE (X, Y), C#, 4
    FOR I = 1 TO 500
    NEXT I
  NEXT C#

endexplode:

END SUB

SUB GetData

' Description:  This program reads various data files
'               File #1 is for Blue one's performance data
'               File #2 is for Red one's performance data
'               File #3 is for Blue two's performance data
'               File #4 is for Red two's performance data
'               File #5 is for the scenario data
'
'               The program is divided into two major section
'               The 1st reads in the data and the 2nd performs
'               a aircraft performance preprocessing function.

OPEN "B1Perf.dat" FOR INPUT AS #1   'Blue One performance data
OPEN "R1Perf.dat" FOR INPUT AS #2   'Red One performance data
OPEN "B2Perf.dat" FOR INPUT AS #3   'Blue Two performance data
OPEN "R2Perf.dat" FOR INPUT AS #4   'Red Two performance data
OPEN "Detail.dat" FOR OUTPUT AS #6   'Detail Summary file
OPEN "ACPerf.dat" FOR OUTPUT AS #7   'AC Performance factors
OPEN "Output.dat" FOR OUTPUT AS #10 'Significant event report

' By having B1 and R1 files first, the convention used to read
' the data allows a one on one scenarios to be generated.

'********* Default system and initial values
***************************
endsim% = 5000
Delay% = 0
OEF = 25 'Orbit expansion factor
Tgt$(1) = "   None   "
```

```
Tgt$(2) = " Blue Base "
Tgt$(3) = "   None    "
Tgt$(4) = " Blue Base "

'**************************************************************

' Assign nomenclatures

AC%(1) = 1
AC%(2) = 2
AC%(3) = 3
AC%(4) = 4

' Read in performance data for the four aircraft

FOR I% = 1 TO NumAC%
INPUT #I%, AC$(I%)     'Side
INPUT #I%, WgArea(I%)  'Wing Area
INPUT #I%, MaxWt(I%)   'Max combat Wgt
INPUT #I%, MinWt(I%)   'Min combat Wgt

'FuelLbs(i%) = MaxWt(i%) - MinWt(i%)     'Find fuel load

INPUT #I%, MaxAlt(I%)  'Max alt
INPUT #I%, MinAlt(I%)  'Min alt
INPUT #I%, MaxETA(I%)  'Max strutual gee force
INPUT #I%, ThrKF(I%)   'Thrust table multiplier (2 for 2 engines)
INPUT #I%, FCKF(I%)    'Fuel consumption multiplier
INPUT #I%, DragKF(I%)  'Drag multiplier
'
' Get altitude arguments
FOR k% = 1 TO AltArg%
  INPUT #I%, Alt(k%)
NEXT k%
'
' Get Maximum mach arguments at altitudes
'
FOR k% = 1 TO AltArg%
  INPUT #I%, MMach(I%, k%)
NEXT k%
'
' Get Mach arguement which will correspond with the max cl
'
FOR k% = 1 TO MachArg%
  INPUT #I%, MachDat(I%, k%)
NEXT k%
'
' Get Maximum coefficient of lift at mach number i,1,k
'
FOR k% = 1 TO MachArg%
    INPUT #I%, MaxCL(I%, k%)'Index 0 is a label
NEXT k%
'
' Get Mil Power thrust values at altitude j% and mach k%
'
FOR j% = 1 TO AltArg%
  FOR k% = 1 TO MachArg%
    INPUT #I%, ThrMP(j%, k%)
    ThrMP(j%, k%) = ThrMP(j%, k%) * ThrKF(I%)
  NEXT k%
NEXT j%
'
' Get After Burner thrust values at altitude j% and mach k%
'
FOR j% = 1 TO AltArg%
  FOR k% = 1 TO MachArg%
    INPUT #I%, ThrAB(j%, k%)
    ThrAB(j%, k%) = ThrAB(j%, k%) * ThrKF(I%)
  NEXT k%
NEXT j%
'
' Get Mil Power fuel consumption values at altitude j% and mach
k%
'
FOR j% = 1 TO AltArg%
  FOR k% = 1 TO MachArg%
    INPUT #I%, FCMP(j%, k%)
  FCMP(j%, k%) = FCMP(j%, k%) * FCKF(I%)
 NEXT k%
NEXT j%
'
' Get After Burner fuel consumption values at altitude j% and
mach k%
'
FOR j% = 1 TO AltArg%
 FOR k% = 1 TO MachArg%
  INPUT #I%, FCAB(j%, k%)
  FCAB(j%, k%) = FCAB(j%, k%) * FCKF(I%)
 NEXT k%
NEXT j%

' Get Drag coefficients for given mach and max CL

FOR j% = 1 TO CLArg%
 FOR k% = 1 TO MachArg%
  INPUT #I%, Drag(j%, k%)
  Drag(j%, k%) = Drag(j%, k%) * DragKF(I%)
 NEXT k%
NEXT j%
CLOSE #I%
NEXT I%

' Program Performance Calculator

' Note: Some of the calculated performance measures are not used
at
' this time; however, they have been codded and debuged for use
' with later model enhancements

' Source:  Mr Robert Mercier and PACAM 8
' Functions and Constants pulled from PACAM documentation

FOR I% = 1 TO NumAC%
FOR k% = 1 TO 2 'Compute performace based on the two velocities

 IF k% = 1 THEN TempVel = V(I%)
 IF k% = 2 THEN TempVel = Vp(I%)

'Compute altitude constants

 Mach = TempVel / SpdS(z)          'Mach Number, A function of
speed
                                   'of sound
 Q = .5 * RHO(z) * TempVel * TempVel     'Dynamic Pressure

' Dynamic pressure is a function of wing area and weight.
' Updates are not performed as the weight of the aircraft
changes.
' This type resolution requires creates a computational load.
' As a compromise, the mean wght will be used.

 WGT = (MaxWt(I%) - MinWt(I%)) / 2

 'Dynamic Pressure at the speed of sound

 U = .5 * RHO(z) * SpdS(z) ^ 2 * (WgArea(I%) / WGT)

 ' Find the low and high index numbers which capture Alt and Mach

FOR j% = 1 TO AltArg%
 IF (z ¡ Alt(j%)) THEN          'Find Argument index
  IAlt% = j%
  GOTO 20
 END IF
NEXT j%

20
FOR j% = 1 TO MachArg%
 IF (Mach ¡ MachDat(I%, j%)) THEN  'Find Argument index
  IM% = j%
  ICL% = j%
  GOTO badmach
 END IF
NEXT j%
```

```
badmach: 'Continue

' Interpolate the Max Coeff of lift

IF (ICL% = 1 AND IM% = 1) THEN
   C1 = 0
   GOTO 40
ELSEIF (ICL% = 1) THEN
   C1 = 0
   GOTO 40
END IF

C1 = (MaxCL(I%, ICL%) - MaxCL(I%, (ICL% - 1))) / (MachDat(I%,
IM%) - MachDat(I%, (IM% - 1)))

40 'Continue

C2 = MaxCL(I%, (ICL% - 1)) - C1 * MachDat(I%, (IM% - 1))'CL
decreases as Mach incs.

SS = (Mach - MachDat(I%, (IM% - 1))) / (MachDat(I%, IM%) -
MachDat(I%, (IM% - 1)))

'Where SS is the slope for Mach arguements

'45 'Continue

TT = (z - Alt(IAlt% - 1)) / (Alt(IAlt%) - Alt(IAlt% - 1))

'Where TT is the interpolation factor for Alt arguements

UU = 1! - TT   'To be used for the double interpolation

'Begin double interpolation for thrust in Mil Pwr and AB
'and fuel consumption in Mil Pwr and AB.  Thrust is in pounds
'and fuel consumption is in Pounds per hour.

E1 = UU * ThrAB((IAlt% - 1), (IM% - 1)) + TT * ThrAB(IAlt%, (IM%
- 1))
E2 = UU * ThrMP((IAlt% - 1), (IM% - 1)) + TT * ThrMP(IAlt%, (IM%
- 1))
E3 = UU * FCAB((IAlt% - 1), (IM% - 1)) + TT * FCAB(IAlt%, (IM% -
1))
E4 = UU * FCMP((IAlt% - 1), (IM% - 1)) + TT * FCMP(IAlt%, (IM% -
1))
F1 = UU * ThrAB((IAlt% - 1), IM%) + TT * ThrAB(IAlt%, IM%)
F2 = UU * ThrMP((IAlt% - 1), IM%) + TT * ThrMP(IAlt%, IM%)
F3 = UU * FCAB((IAlt% - 1), IM%) + TT * FCAB(IAlt%, IM%)
F4 = UU * FCMP((IAlt% - 1), IM%) + TT * FCMP(IAlt%, IM%)

'Find Thrust AB and MP, Fuel AB and MP

THAB(I%) = E1 + SS * (F1 - E1)
THMP(I%) = E2 + SS * (F2 - E2)

ABFC(I%, k%) = (E3 + SS * (F3 - E3)) / 3600!  'fuel consumption
per sec
MPFC(I%, k%) = (E4 + SS * (F4 - E4)) / 3600!  'fuel consumption
per sec
Ftemp(I%) = 0

CLMax = MaxCL(I%, (ICL% - 1)) + SS * MaxCL(I%, ICL%) - MaxCL(I%,
(ICL% - 1))

'Where CLMax is the interpolated maximum coefficient of lift.

VelMax(I%) = SpdS(z) * MMach(I%, (IAlt% - 1)) + TT * (MMach(I%,
IAlt%) - MMach(I%, (IAlt% - 1)))

'Where VelMax is the interpolated maximum vel for given altitude

' Begin determination of the sustained gee force for sweeping
turns
' Set up for another double interpolation of the coefficient of
' drag table. Call Sub CLDES2
```

```
'Can't drop below 1 gee or the aircraft has a bad day

SusG = DMAX(1!, CLDES2(CLMax, SS, THAB, ICL%, IM%, I%) * U *
Mach)

CONST Grav = 32.2    'feet per second square
Convfac = 1.46666    'conversion factor to obtain Rad of turn in
feet

RadTurn(I%, k%) = (TempVel ^ 2 / (Grav * DMIN(3!, SusG))) *
Convfac 'Turning Radius

'Values Retained

    PRINT #7, USING "            Performance Specifications for
Aircraft & "; AC$(I%)
    PRINT #7, USING "Performance based on velocity of #### mph
and altitude of #####"; TempVel; z
    PRINT #7, "         Thrust in AB (Pounds): "; THAB(I%)
    PRINT #7, "         Thrust in Mil Pwr (Pounds): ";
THMP(I%)
    PRINT #7, "         Fuel consumption in AB (Pounds/sec):
"; ABFC(I%, k%)
    PRINT #7, "         Fuel consumption in Mil Pwr
(Pounds/sec): "; MPFC(I%, k%)
    PRINT #7, "         Sustained Gees: "; SusG
    PRINT #7, "         Min Turning Radius (Ft): ";
RadTurn(I%, k%)

 NEXT k%
NEXT I%


'****** Warning! Do not move the following calculation above
this point

'Calc the RTmiles for both vel, initialize TurnRate at the slower
vel.

FOR j% = 1 TO NumAC%
 FOR k% = 1 TO 2 'two velocity levels
  RTmiles(j%, k%) = RadTurn(j%, k%) / 5280! 'Convert to feet
'Ref Shaw 'Fighter Combat', p390
 NEXT k%
  TurnRate(j%) = (V(j%) / RTmiles(j%, 1)) / 3600'Radians per
second
NEXT j%

'The following code is used to establish the race track for
'the cap mission.
'
'Initialize the Blue aircraft on their station
'Place them Pi radians apart
'Operate at mil pwr

ACx(1) = StatPtx + (RTmiles(1, 1) * OEF)
ACy(1) = StatPty
ACx(3) = StatPtx - (RTmiles(3, 1) * OEF)
ACy(3) = StatPty

'Calc station keeper, the target the A/C follow to stay on track

' 1st find circumference
' 2nd find how many time pulses to travel circum, this divides
' the circle up into small traveled segments
' 3rd find the radian equivalent of the small segment

FOR I% = 1 TO NumAC% STEP 2
  circum = (RTmiles(I%, 1) * OEF) * 2 * Pi
  timecircum = circum / (V(I%) / 3600)
  DeltaCir(I%) = 2 * Pi / timecircum
NEXT I%

'**** End Station Keeping Code

'Compute min turn rate in radians per sec other than station
keeping
```

```
'Convert velocity to feet per sec. Must be done after the above
'calculations since the table are base on miles per hour.

'**** Scale Aircraft Velocity
FOR j% = 1 TO NumAC%
  V(j%) = V(j%) / 3600
  Vp(j%) = Vp(j%) / 3600
NEXT j%

'**** Scale Missile velocity
FOR I% = 1 TO NumAC%
   MsLV(I%) = MsLV(I%) / 14400'Missile velocity per quarter sec.
NEXT I%

'**** Scale Radar
FOR j% = 1 TO NumAC%
  RadRng(j%) = RadRng(j%) / 5280!
NEXT j%

'**** Scale Optical
FOR j% = 1 TO NumAC%
  OptRng(j%) = OptRng(j%) / 5280!
NEXT j%

CLOSE #1  'Close files
CLOSE #2
CLOSE #3
CLOSE #4
CLOSE #7
END SUB

SUB Grid
GScale = 1.5
FOR p% = -Wx * GScale TO Wx * GScale STEP (GScale * 10)
FOR s% = -Wy * GScale TO Wy * GScale STEP (GScale * 10)
PSET (p%, s%)
NEXT s%
NEXT p%
END SUB

SUB Headline

LOCATE 1, 1: PRINT "Sec: "
LOCATE 1, 12: PRINT "Posture: "
LOCATE 1, 28: PRINT "Comm: "
LOCATE 1, 35: PRINT Com$
END SUB

REM $STATIC
SUB Init

'If the model is ran more than once, certain variables have to
'be reset back to zero
LOCATE 10, 9: PRINT "Enter"
LOCATE 10, 15: RANDOMIZE ' Seed RNG

'Declare intial values for program variables

CLS
Scale = 1.5     'Step multiple for Zoom
GDelay% = 0     'Graphics Delay
ADelay% = 10    'Analyst Screen Delay
Refs = 1'Screen refresh counter
SysStat$ = "Low"
Com$ = ""

BLMenu ' Set up and print bottom line menu

FOR I% = 1 TO NumAC% STEP 2  'Assign Targets, set detect flags
  Stat$(I%) = "Patrol      "
  IStat%(I%) = 3
  Stat$(I% + 1) = "Undetected "
  IStat%(I% + 1) = 0
  TgtDetLvl%(I%) = 0'Detection currently being used to track
target
  TTurn(I%) = 0  'Station keeping turn
NEXT I%
```

```
FOR I% = 2 TO NumAC% STEP 2 ' Assign Red Targets as the base
location
   Tgtx(I%) = Basex
   Tgty(I%) = Basey
NEXT I%

'Assign Missile initial conditions

FOR I% = 1 TO NumAC%
  MF%(I%) = 1              'Initialize Missiles fired counter
  FOR j% = 1 TO NumMSL%(I%)
    IMslStat%(I%, j%) = 0
    MTNOW%(I%, j%) = 0     'Initialize Missile burn timmer
    MSLx(I%, j%) = 0
    MSLy(I%, j%) = 0
  NEXT j%
NEXT I%

' Load in Icons

ACIcon
MSLIcon

' Initialize Comm

msg 0




END SUB

REM $DYNAMIC
SUB Klaxon (Hi%, Lo%) STATIC
 SOUND Hi%, 5
 SOUND Lo%, 5
END SUB

SUB LoadGo

Frame 1, 80, 1, 20

msg 20

ON ERROR GOTO fileerr

getfile2:
LOCATE 14, 8: INPUT "Use default scenario file Y/N (Y) "; DFAns$

  IF (UCASE$(DFAns$) = "N") THEN      'Get user defined file
    errflg% = 0
    LOCATE 16, 48: PRINT "            "
    LOCATE 16, 8: INPUT "Enter scenario file name with
extension:"; Scnfile$
    OPEN Scnfile$ FOR INPUT AS #5   'Default Scenario file
    IF errflg% = 1 GOTO getfile2:
    otpt% = 0
    LOCATE 18, 8: INPUT "Do you want a detail summary generated
Y/N (N)? "; outans$
    IF UCASE$(outans$) = "Y" THEN otpt% = -1
    MASTDAT
    ELSE
    otpt% = 0
    LOCATE 18, 8: INPUT "Do you want a detail summary generated
Y/N (N)? "; outans$
    IF UCASE$(outans$) = "Y" THEN otpt% = -1
    OPEN "Master.dat" FOR INPUT AS #5  'Default Scenario file
    MASTDAT
  END IF

ON ERROR GOTO 0

END SUB
```

```
SUB MainLoop (MonMode%)

' Main loop for the combat model


DIM RTT(NumAC%)

' Turn Keys On

KEY(1) ON     'F1 Halt Execution
' F2 Resume Function.  Key 2 is used implicitly.
KEY(3) ON     'F3 Switch to analyst screen mode
KEY(4) OFF    'F4 Switch to graphics mode
KEY(5) ON     'F5 Enlarge Scale
KEY(6) ON     'F6 Decrease Scale
KEY(7) ON     'F7 Quit and return to main menu

ScrnSet BadScrn%, MonMode%

IF (BadScrn% = 1) GOTO endmainlp

Init       'Initialize parameters

Refresh    ' Initialize window setting, set Grid, and Base

Station    'Put Blue A/C on station

FOR I% = 1 TO NumAC%  'Compute initial heading
   ACHead(I%) = MHead(ACx(I%), ACy(I%), Tgtx(I%), Tgty(I%))
NEXT I%

Pass = 0  ' Draw first images

FOR I% = 1 TO NumAC%
   ACDraw (I%)
NEXT I%

' Begin first outer loop

FOR TNOW% = 1 TO endsim%

FOR j% = 1 TO Delay% * 1000: NEXT j%' Delay

' Call Detection sub

detect

' Fly on station if no intruder detection

IF (SysStat$ = "Low") THEN   'No intruders, fly on station
  Station
ELSE 'Update Interceptors on target locations
 FOR I% = 1 TO NumAC% STEP 2
  IF IStat%(I%) ¡¿ 6 THEN
   Update (I%)
  END IF
 NEXT I%
END IF

' Call Heading

IF (TNOW% = 1) GOTO Skip

head:

FOR I% = 1 TO NumAC%

' Find new heading

 ACHead(I%) = MHead(ACx(I%), ACy(I%), Tgtx(I%), Tgty(I%))

' Test turn feasibility

TrkAng(I%) = TrackAngle(I%) 'Short cut calculation used for
constrainng
                            'turn.
```

```
'Constrain turn if Trackangle ¿ turnrate

   IF (TrkAng(I%) ¿ TurnRate(I%)) THEN

' Test for best direction to turn

      TstHead = OldHead(I%) + TurnRate(I%) 'Sample a left turn
        TstTrkAng = ABS(ACHead(I%) - TstHead)

        IF (TstTrkAng ¿ Pi) THEN
          TstTrkAng = TwoPi - TstTrkAng
        END IF

        IF (TstTrkAng ¡ TrkAng(I%)) THEN
          ACHead(I%) = OldHead(I%) + TurnRate(I%)     ' Left
tturn
        ELSE
          ACHead(I%) = OldHead(I%) - TurnRate(I%)     ' Right
Turn
        END IF

      test1 = ACHead(I%) - TwoPi   'Increased heading beyond 2
Pi
      test2 = ACHead(I%)           'Set up to capture neg
heading

  IF (test1 ¿ 0) THEN ACHead(I%) = test1   'Reset to zero line
plus overage
  IF (test2 ¡ 0) THEN ACHead(I%) = TwoPi + test2'Reset to 2 Pi -
overage

  END IF

skiphead:

NEXT I%

Skip:

SELECT CASE (AScreen%)
CASE 0
  Pulse% = TNOW% MOD 2
   IF (Pulse% = 0) THEN
    FOR I% = 1 TO NumAC% STEP 2
      ACDraw I%  'Draw the icons
    NEXT I%
    ELSE
    FOR I% = 2 TO NumAC% STEP 2
      ACDraw (I%)  'Draw the icons
    NEXT I%
    END IF
 CASE 1
    ANLST
END SELECT

FOR I% = 1 TO NumAC% STEP 2
      IF (TgtDetLvl%(I%) ¿ 2 OR IMslStat%(I%, MF%(I%)) = 1) THEN
missile (I%)
NEXT I%


' Call Advance

FOR I% = 1 TO NumAC%

Adv (I%)

NEXT I%

IF (AScreen% = 0 AND TNOW% = Refs) THEN
    BBase
    Refs = TNOW% + 30
END IF

FOR I% = 1 TO NumAC%
   OldHead(I%) = ACHead(I%)  'Save last heading
NEXT I%
```

```
' Display Data or Display screen
' Update headline scoreboard

LOCATE 1, 6: PRINT USING "####"; TNOW%   'Update Header line
LOCATE 1, 21: PRINT USING "&"; SysStat$

IF otpt% = 0 GOTO skipdetl

RTT(1) = TDist(1, ITgt%(1))
RTT(2) = TDist(2, 0)
RTT(3) = TDist(3, ITgt%(1))
RTT(4) = TDist(4, 0)
FOR I% = 1 TO NumAC%
'                    Coordinates     Vel  Heading    Tgt
Coord   Range   Det Msl"
'              Sec  AC   X      Y   Stat (Mi/S)  (Rad)   Tgt
X     Y   to Tgt  Lvl Stat              "
PRINT #6, USING "#### #  #### ####  #    #.###  #.##   #
#### ####    ###     #   #"; TNOW%; AC%(I%); ACx(I%); ACy(I%);
IStat%(I%); TV(I%); ACHead(I%); ITgt%(I%); Tgtx(I%); Tgty(I%);
RTT(I%); TgtDetLvl%(I%); IMslStat%(I%, MF%(I%))
NEXT I%

skipdetl:
FOR I% = 1 TO NumAC% STEP 2
 IF (IStat%(I%) = 6 AND Dist(ACx(I%), ACy(I%), Basex, Basey) ¡ 1
AND Landed = 0) THEN
  IStat%(I%) = 7
  Destroy I%
 END IF
NEXT I%
NEXT TNOW%

endmainlp:

CLS
CLOSE #5
CLOSE #6
CLOSE #7
CLOSE #10
SCREEN 0
msg 5000

END SUB

SUB MASTDAT

' Read in default scenario data

'Read in the data

FOR I = 1 TO 100
 INPUT #5, Dat(I)
NEXT I
CLOSE #5

' Base data     Block of 10 data elements 1-10.

Basex = Dat(2)
Basey = Dat(3)
GCIRg = Dat(4)

' Environment:  Block of 10 data elements 11-20

NumAC% = Dat(12)        'Number of aircraft
z = Dat(13)             'Altitude plane
StatPtx = Dat(14)       'CAP patrol station in x
StatPty = Dat(15)       'CAP patrol station in y

' Aircraft data elements, block of 20 for each aircraft.
' Blue 1: 21-40, Red 1: 41- 60, Blue 1: 61-80,  Red 1:81-100

' Blue 1                 Blue 2

Fuel(1) = Dat(22):     Fuel(3) = Dat(62)     'Fuel Load
bingo(1) = Dat(23):    bingo(3) = Dat(63)      'Bingo weight
```

```
V(1) = Dat(24):        V(3) = Dat(64)        'Cruise velocity
Vp(1) = Dat(25):        Vp(3) = Dat(65)        'Pursuit velocity
OptRng(1) = Dat(26):   OptRng(3) = Dat(66)    'Eye sight Range
RadRng(1) = Dat(27):   RadRng(3) = Dat(67)    'Radar Range
NumMSL%(1) = Dat(28):     NumMSL%(3) = Dat(68)      'No. Msls
onboard
MsLV(1) = Dat(29):     MsLV(3) = Dat(69)      'Vel of Missiles
MsLSenRg(1) = Dat(30): MsLSenRg(3) = Dat(70)  'Sensor range of
missiles

' Red 1                Red 2

ACx(2) = Dat(42):       ACx(4) = Dat(82)       'Initial position
ACy(2) = Dat(43):       ACy(4) = Dat(83)       'Initial position
Fuel(2) = Dat(44):     Fuel(4) = Dat(84)     'Fuel weight
bingo(2) = Dat(45):     bingo(4) = Dat(85)     'Bingo weight
V(2) = Dat(46):        V(4) = Dat(86)        'Cruise velocity
Vp(2) = Dat(47):        Vp(4) = Dat(87)        'Pursuit velocity
OptRng(2) = Dat(48):   OptRng(4) = Dat(88)    'Eye sight Range
RadRng(2) = Dat(49):   RadRng(4) = Dat(89)    'Radar Range
NumMSL%(2) = Dat(50):     NumMSL%(4) = Dat(90)      'No. Msls
onboard
MsLV(2) = Dat(51):     MsLV(4) = Dat(91)      'Vel of missiles
MsLSenRg(2) = Dat(52): MsLSenRg(4) = Dat(92)  'Sensor range of
missiles

END SUB

FUNCTION MHead (Ax, Ay, Bx, By)

' Finds Headings for aircraft

' Convention: Compute the heading of A relative to the position
of B

Px = Ax - Bx  'Find x and y range between A and B
Py = Ay - By

' Error Trap for shared longitudal,Latitudal coordinates

IF (Px = 0) THEN
 IF (By ¿= Ay) THEN
    THead = 1.570796
 ELSE
    THead = 4.712389
 END IF
 GOTO 11
END IF

IF (Py = 0) THEN
IF (Bx ¿= Ax) THEN
    THead = 0
   ELSE
    THead = 3.141593
   END IF
   GOTO 11
END IF

IF (By ¿ Ay) THEN
   THead = ATN(Py / Px) + Pi
ELSE
   THead = ATN(Py / Px)
END IF

IF (Bx ¡ Ax) THEN
   THead = ATN(Py / Px) + Pi
ELSE
   THead = ATN(Py / Px)
END IF

' All of the above is used to determine the heading of the
aircraft
' relative to their coordinate system and the their target. The
' transformation below converts negative radians to positive.
The
' ATN function yields negative values for coordinates in the neg
' Y coordinates. Making this transformation permits easier
```

```
checking
' of changes in headings (don't have to worry about their sign).

11  IF (THead ¡ 0) THEN
      MHead = 2 * Pi + THead
    ELSE
      MHead = THead

    END IF


END FUNCTION

SUB missile (AC%)
'Begin embedded missile loop, four missile cycles to one
aircraft.
'
'       Determine missile heading
'       Draw missile missile
'       Advance missile
'       Check for kill
'
'Assumptions: Only one missile is fired at a time per AC
'            Only one target at a time assigned to an AC
'            Missile heading will remain constant while in
embedded loop
'            At the end of missile burn, missile is terminated
'            Once launched, missiles are tied to the launching
AC tgt
'            Missile flight performance will not be  constrained
missle
'            aerodynamics.

'Initialize location of missile at time of launch

IF MF%(AC%) ¿ NumMSL%(AC%) GOTO Mslend

IMslStat%(AC%, MF%(AC%)) = 1

IF (MTNOW%(AC%, MF%(AC%)) = 1) THEN
   MSLx(AC%, MF%(AC%)) = ACx(AC%)
   MSLy(AC%, MF%(AC%)) = ACy(AC%)
   IMTgt%(AC%, MF%(AC%)) = ITgt%(AC%)'Assign target
   Mmsg 10, AC%, MF%(AC%), ITgt%(AC%)
END IF

'Update missile on tgt location

MTx = ACx(IMTgt%(AC%, MF%(AC%)))
MTy = ACy(IMTgt%(AC%, MF%(AC%)))

MSLHead(AC%, MF%(AC%)) = MHead(MSLx(AC%, MF%(AC%)), MSLy(AC%,
MF%(AC%)), MTx, MTy)
Msl% = MF%(AC%)

FOR t% = MTNOW%(AC%, MF%(AC%)) TO MTNOW%(AC%, MF%(AC%)) + 4
     MSLAdv AC%, Msl%       'Advance the missile
     IF AScreen% = 0 THEN   'Not in attrib screen
       MSLDraw AC%, Msl%    'Draw the missile
     END IF
'    IF MTNOW%(AC%, MF%(AC%)) ¿ Burnt(AC%)*4 goto Mslend

' If the missile either hit the target or missed, terminate the
' missile and clean up the screeen.

     IF IMslStat%(AC%, MF%(AC%)) ¿ 1 THEN
       Mslterm AC% 'Advances MF%
       IF AScreen% = 0 THEN
         BLMenu
         CLS 2
         Refresh
       END IF
       GOTO Mslend
     END IF
NEXT t%

'Update missile clock by one second for each pass since each
```

```
'pass is equal to one second. The quarter seconds are not posted
'since MTOW is an integer value.

MTNOW%(AC%, MF%(AC%)) = MTNOW%(AC%, MF%(AC%)) + 1

Mslend:

END SUB

SUB MSLAdv (AC%, Msl%)


'Find new missile coordinates after advance

 MSLy(AC%, Msl%) = MSLy(AC%, Msl%) + MsLV(AC%) * SIN(MSLHead(AC%,
Msl%))
 MSLx(AC%, Msl%) = MSLx(AC%, Msl%) + MsLV(AC%) * COS(MSLHead(AC%,
Msl%))

 Hx = ABS((MSLx(AC%, Msl%) - Tgtx(AC%)))
 Hy = ABS((MSLy(AC%, Msl%) - Tgty(AC%)))

 'if x and y coordinates are out of sync by very much the missile
will miss

 IF (Hx ¡ .5 * MsLV(AC%) AND Hy ¡ .5 * MsLV(AC%)) THEN
   PKill AC%, Hx, Hy
 END IF

END SUB

SUB MSLDraw (AC%, Msl%)

DIM mxx(10), myy(10)

'Orientate missile image to heading

IF MTNOW%(AC%, Msl%) ¡= 2 GOTO endmsldw
FOR I% = 2 TO 7 STEP 2  'Erase old image
  LINE (mxxold(AC%, Msl%, I%), myyold(AC%, Msl%,
I%))-(mxxold(AC%, Msl%, (I% + 1)), myyold(AC%, Msl%, (I% + 1))),
3
NEXT I%

FOR I% = 2 TO 7
  mxx(I%) = (MX(I%) * COS(MSLHead(AC%, Msl%)) - MY(I%) *
SIN(MSLHead(AC%, Msl%))) + MSLx(AC%, Msl%)
  myy(I%) = (MX(I%) * SIN(MSLHead(AC%, Msl%)) + MY(I%) *
COS(MSLHead(AC%, Msl%))) + MSLy(AC%, Msl%)

' Capture current values
  mxxold(AC%, Msl%, I%) = mxx(I%)
  myyold(AC%, Msl%, I%) = myy(I%)

NEXT I%

' Draw Image

FOR I% = 2 TO 7 STEP 2
  LINE (mxx(I%), myy(I%))-(mxx(I% + 1), myy(I% + 1)), 1
NEXT I%
endmsldw:

END SUB

SUB MSLIcon

' Body:             MX(2,3), MY(2,3)
' Left tail tip     MX(4,5), MY(4,5)
' Right tail tip     MX(6,7), MY(6,7)

' Setup missile icon.

MX(2) = 0:  MY(2) = 0
MX(3) = 5:  MY(3) = 0
MX(4) = 0:  MY(4) = 0
MX(5) = -.5:  MY(5) = .5
MX(6) = 0:  MY(6) = 0
MX(7) = -.5:  MY(7) = -.5

' Scale adjustment used for calibration

k = 1   '((1! / Scale) * .7)

FOR I = 2 TO 7
  MX(I) = MX(I) * k
  MY(I) = MY(I) * k
NEXT I

END SUB

SUB Mslterm (AC%) STATIC

  MF%(AC%) = MF%(AC%) + 1
  IMslStat%(AC%, MF%) = 0

END SUB

SUB pause
'pauses the program until the space bar is pressed

DO
    ch$ = INKEY$
  LOOP UNTIL ch$ = " "  'Indefinite Pause

END SUB

SUB PKill (AC%, Hx, Hy)

'Sub PKill determines if the missile actually killed the aircraft
'The user provided Pk is used in this determination.
'The Pk value is compared to a random draw, if the random draw
'is less than the Pk, the target is eliminated.

'***** Temp Val ******
 Pk(AC%) = .85
'***** ******** ******

IF (RND ¡ Pk(AC%)) THEN
    Mmsg 20, AC%, MF%(AC%), ITgt%(AC%)
    Destroy ITgt%(AC%) 'this should yied the target index value

    FOR b% = 1 TO NumAC% STEP 2   'alter sensors for wingman
      FOR r% = 2 TO NumAC% STEP 2
        IF (ITgt%(b%) = r% AND IStat%(r%) = 2) THEN
            TgtDetLvl%(b%) = 0
        END IF
     NEXT r%
    NEXT b%

    IMslStat%(AC%, MF%(AC%)) = 3 'Hit target
    detect
    TgtSel TDist
ELSE
    Mmsg 30, AC%, MF%(AC%), ITgt%(AC%)
    IMslStat%(AC%, MF%(AC%)) = 2 'Missed target
END IF


END SUB

SUB Refresh
      VIEW (1, 17)-(MaxScrX%, MaxScrY%), 3, 4
      WINDOW (-Wx * Scale, Wy * Scale)-(Wx * Scale, -Wy * Scale)
      BBase
      Grid
'LOCATE 1: PRINT "
                                  "
      Headline
END SUB

FUNCTION RHO (z)
'
'Constants from the PACAM User Manual. Used for computing the
```

```
'atmospheric density.
'
a = .0023769199#
b = .000006875347#
C = 4.256155
D = .00070612811#
e = .0000480634#
'
 IF (z ¡ ZBAR) THEN
    RHO = a * (1 - b * z) ^ C
 ELSE
    RHO = D * EXP(-1 * e * (z - ZBAR))
 END IF
'
END FUNCTION

SUB ScrnSet (BadScrn%, MonMode%)

'CLS
'LOCATE 10, 10: INPUT "Enter monitor type VGA or EGA (VGA) ";
MonType$
'CLS

' Find the screen mode that works
' Set screen variables for that screen

'IF (UCASE$(MonType$) = "EGA") THEN GOTO SetEGA
'ON ERROR GOTO SetEGA

IF (MonMode% = 9) GOTO SetEGA

SCREEN 12          ' Hi res VGA
PSET (0, 0)        ' Test Graphics
MonMode% = 12       ' Screen Mode
MonType$ = "VGA"    ' Set graphics type
Wx = 215           ' Range factor for widow setting
Wy = 150           ' and aspect normalization
MaxScrX% = 630      ' Set screen dimension in X(639)
MaxScrY% = 460      ' Set screen dimension in Y - bottom line
pixel
'AR = .75          ' Just in case
                ' requirement
BLine% = 30         ' Set maximum lines per screen

GOTO OK

SetEGA:
'ON ERROR GOTO SetCGA
SCREEN 9:          ' Hi res EGA
PSET (0, 0)
MonMode% = 9
MonType$ = "EGA"
Wx = 215
Wy = 150
MaxScrX% = 630
MaxScrY% = 330
BLine% = 25
'AR = .75

GOTO OK

SetCGA:            ' CGA is not currently permitted, too many
drawbacks.
GOTO PrnErr
'ON ERROR GOTO PrnErr
SCREEN 2
PSET (0, 0)
MonMode% = 2
MonType$ = "CGA"
Wx = 258!
Wy = 150
MaxScrX% = 310
MaxScrY% = 180
BLine% = 25
'AR = .75

GOTO OK
```

```
PrnErr:
SCREEN 0
LOCATE 12, 5: PRINT "Sorry, your monitor is not compatible with
this program."
BadScrn% = 1

OK: ON ERROR GOTO 0

END SUB

FUNCTION SpdS (z)
'
'Constants from the PACAM User Manual. Used for computing the
'speed of sound
'
F = 49.040772#
G = 518.688
H = .00356616#
ZBAR = 36089
'
 IF (z ¡ ZBAR) THEN
  SpdS = F * (SQR(G - H * z))
 ELSE
  SpdS = 968.4652
 END IF
'
END FUNCTION

SUB Station

TTurn(1) = TTurn(1) + DeltaCir(1)
TTurn(3) = TTurn(3) + DeltaCir(3)

'Find new coordinates for tgt centered on the station pt
'The Blue aircraft will follow the tgt, much like dog racing
'where the dog follows the mechanical rabbit.

'Convention: Station point + Radius * sin//cos of current radian
'           Subscript 1 is Blue 1 and 3 is Blue 2.

IF IStat%(1) ¡¿ 6 THEN
 Tgty(1) = StatPty + (RTmiles(1, 1) * OEF) * SIN(TTurn(1) + 0)
 Tgtx(1) = StatPtx + (RTmiles(1, 1) * OEF) * COS(TTurn(1) + 0)
END IF
IF IStat%(3) ¡¿ 6 THEN
 Tgty(3) = StatPty + (RTmiles(3, 1) * OEF) * SIN(TTurn(3) + Pi +
0)
 Tgtx(3) = StatPtx + (RTmiles(3, 1) * OEF) * COS(TTurn(3) + Pi +
0)
END IF
END SUB

SUB TgtSel (TDist)

'Both Red 1 and Red 2 have been detected

IF (IStat%(2) = 1 AND IStat%(4) = 1) THEN GOTO decide:

 IF IStat%(2) ¡¿ 1 AND IStat%(4) = 1 THEN
  IF IStat%(3) ¡¿ 6 AND IStat%(3) ¡¿ 2 THEN
   Tgtx(3) = ACx(4)
   Tgty(3) = ACy(4)
   Tgt$(3) = "  Red 2  "
   ITgt%(3) = 4
   msg 1
  END IF
  IF IStat%(1) ¡¿ 6 AND IStat%(1) ¡¿ 2 THEN
   Tgtx(1) = ACx(4)
   Tgty(1) = ACy(4)
   Tgt$(1) = "  Red 2  "
   ITgt%(1) = 4
  END IF
 END IF

'Red 2 not detect, assign Red 1 to Blue units
```

```
 IF IStat%(4) ¡¿ 1 AND IStat%(2) = 1 THEN
   IF IStat%(3) ¡¿ 6 AND IStat%(3) ¡¿ 2 THEN
     Tgtx(3) = ACx(2)
     Tgty(3) = ACy(2)
     Tgt$(3) = "   Red 1   "
     ITgt%(3) = 2
     msg 2
   END IF
   IF IStat%(1) ¡¿ 6 AND IStat%(1) ¡¿ 2 THEN
     Tgtx(1) = ACx(2)
     Tgty(1) = ACy(2)
     Tgt$(1) = "   Red 1   "
     ITgt%(1) = 2
   END IF
  END IF

'Red 1 is dead, Red 2 not detected

  IF IStat%(2) = 2 AND IStat%(4) = 0 THEN
     Tgt$(1) = "   None    "
     Tgt$(3) = "   None    "
     IStat%(1) = 3
     IStat%(3) = 3
     SysStat$ = "Low"
     msg 3
  END IF

'Red 2 is dead, Red 1 not detected

  IF IStat%(4) = 2 AND IStat%(2) = 0 THEN
     Tgt$(1) = "   None    "
     Tgt$(3) = "   None    "
     IStat%(1) = 3
     IStat%(3) = 3
     SysStat$ = "Low"
     msg 4
  END IF

GOTO endtgt:

decide:

  IF TDist(1, 2) ¡= TDist(3, 2) THEN 'B1 is closer to R1 than B2
is to R1
    IF IStat%(1) ¡¿ 6 AND IStat%(1) ¡¿ 2 THEN
     Tgtx(1) = ACx(2)
     Tgty(1) = ACy(2)
     Tgt$(1) = "   Red 1   "
     ITgt%(1) = 2
     msg 5
    END IF
    IF IStat%(3) ¡¿ 6 AND IStat%(3) ¡¿ 2 THEN
     Tgtx(3) = ACx(4)
     Tgty(3) = ACy(4)
     Tgt$(3) = "   Red 2   "
     ITgt%(3) = 4
     msg 6
    END IF
  ELSE
    IF IStat%(1) ¡¿ 6 AND IStat%(1) ¡¿ 2 THEN
     Tgtx(1) = ACx(4)
     Tgty(1) = ACy(4)
     Tgt$(1) = "   Red 2   "
     ITgt%(1) = 4
     msg 7
    END IF
    IF IStat%(3) ¡¿ 6 AND IStat%(3) ¡¿ 2 THEN
     Tgtx(3) = ACx(2)
     Tgty(3) = ACy(2)
     Tgt$(3) = "   Red 1   "
     ITgt%(3) = 2
     msg 8
    END IF
  END IF

 endtgt:
```

```
END SUB

FUNCTION tkag (Blue%, Red%)

' Find LOS angle

' Convention: Compute the LOS angle relative to Blue position

Px = ACx(Red%) - ACx(Blue%)  'Find x and y range between AC
Py = ACy(Red%) - ACy(Blue%)

' Error Trap for shared longitudal,Latitudal coordinates

IF (Px = 0) THEN
  IF (ACy(Red%) ¿= ACy(Blue%)) THEN
     LOS = 1.570796
  ELSE
     LOS = 4.712389
  END IF
  GOTO tkend
END IF

IF (Py = 0) THEN
IF (ACx(Red%) ¿= ACx(Blue%)) THEN
     LOS = 0
    ELSE
     LOS = 3.141593
    END IF
    GOTO tkend
END IF

IF (ACy(Red%) ¿ ACy(Blue%)) THEN
    LOS = ATN(Py / Px) + Pi
ELSE
    LOS = ATN(Py / Px)
END IF

IF (ACx(Red%) ¡ ACx(Blue%)) THEN
    LOS = ATN(Py / Px) + Pi
ELSE
    LOS = ATN(Py / Px)
END IF

tkend:
   IF (LOS ¡ 0) THEN
     LOS = 2 * Pi + LOS
   ELSE
     LOS = LOS
   END IF

   ttkag = ABS(ACHead(Blue%) - LOS)

'test for trackangle for boundness

   IF ttkag ¿ Pi THEN
     tkag = TwoPi - ttkag
   ELSE
     tkag = ttkag
   END IF


END FUNCTION

FUNCTION TrackAngle (I%)

TA = ABS(ACHead(I%) - OldHead(I%))
  IF (TA ¿ Pi) THEN
    TrackAngle = TwoPi - TA
  ELSE
    TrackAngle = TA
  END IF

END FUNCTION

SUB Update (AC%)
```

```
'Tgtx and Tgty are the Blue aircraft target coordinates
'ITgt%(i%) is the target index for the ith Blue aircraft
'Updates must be performed prior to calculating new heading

   Tgtx(AC%) = ACx(ITgt%(AC%))
   Tgty(AC%) = ACy(ITgt%(AC%))

END SUB

SUB USERDAT

' A subroutine to collect user scenario data.
Frame 1, 80, 4, 20

msg 40

ON ERROR GOTO fileerr:

getfile:
LOCATE 14, 8: INPUT "Use default scenario file Y/N (Y) "; DFAns$

IF (UCASE$(DFAns$) = "N") THEN      'Get user defined file
    errflg% = 0
    LOCATE 16, 48: PRINT "          "
    LOCATE 16, 8: INPUT "Enter scenario file name with
extension:"; Scnfile$
    OPEN Scnfile$ FOR INPUT AS #5     'Modified Scenario file
    IF errflg% = 1 GOTO getfile
  ELSE
    OPEN "Master.dat" FOR INPUT AS #5   'Default Scenario file
END IF

ON ERROR GOTO 0

msg 200

'Read in the data

FOR I = 1 TO 100
 INPUT #5, Dat(I)
NEXT I
CLOSE #5

' print 1st page

msg 30

FOR I = 2 TO 4
 LOCATE (5 + I), 62: PRINT Dat(I)
NEXT I

FOR I = 12 TO 15
 LOCATE (I), 62: PRINT Dat(I)
NEXT I

2 LOCATE 7, 72: INPUT test$      'forward operation x location
  IF (test$ ¡¿ "") THEN  'Assign new value
    Dat(2) = VAL(test$)
  END IF
  Basex = Dat(2)
  LOCATE 7, 72: PRINT USING "####"; Basex

3 LOCATE 8, 72: INPUT test$      'forward operating y location
  IF (test$ ¡¿ "") THEN
    Dat(3) = VAL(test$)
  END IF
  Basey = Dat(3)
  LOCATE 8, 72: PRINT USING "####"; Basey

4 LOCATE 9, 72: INPUT test$      'GCI range
  IF (test$ ¡¿ "") THEN
    Dat(4) = VAL(test$)
  END IF
  GCIRg = Dat(4)
  IF GCIRg ¡ 0 THEN
    BEEP
    LOCATE 9, 72: PRINT "     "
  GOTO 4
 END IF
 LOCATE 9, 72: PRINT USING "####"; GCIRg

12 LOCATE 12, 72: INPUT test$    'Number of aircraft
 IF (test$ ¡¿ "") THEN
  Dat(12) = VAL(test$)
 END IF
 NumAC% = Dat(12)
 IF NumAC% ¡ 0 OR NumAC% ¿ 4 THEN
  BEEP
  LOCATE 12, 72: PRINT "       "
  GOTO 12
 END IF
 LOCATE 12, 72: PRINT USING "   #"; NumAC%


13 LOCATE 13, 72: INPUT test$    'Altitude plane
 IF (test$ ¡¿ "") THEN
  Dat(13) = VAL(test$)
 END IF
 z = Dat(13)
 IF z ¡ 1000 OR z ¿ 60000 THEN
  BEEP
  LOCATE 13, 72: PRINT "       "
  GOTO 13
 END IF
 LOCATE 13, 72: PRINT USING "#####"; z

14 LOCATE 14, 72: INPUT test$    'CAP x position
 IF (test$ ¡¿ "") THEN
  Dat(14) = VAL(test$)
 END IF
 StatPtx = Dat(14)
 LOCATE 14, 72: PRINT USING "####"; StatPtx

15 LOCATE 15, 72: INPUT test$    'CAP y position
 IF (test$ ¡¿ "") THEN
  Dat(15) = VAL(test$)
 END IF
 StatPty = Dat(15)
 LOCATE 15, 72: PRINT USING "####"; StatPty

' Number of data elements per side


BluDat% = 9
RedDat% = 11


FOR k% = 1 TO NumAC%
CLS
SELECT CASE k%
 CASE 1
  LOCATE 2, 10: PRINT "BLUE 1 SCENARIO DATA INPUT SCREEN"
  msg 50
  leep% = 0
 CASE 2
  LOCATE 2, 10: PRINT "RED 1 SCENARIO DATA INPUT SCREEN"
  msg 60
  leep% = 0
  GOTO rdat
 CASE 3
  LOCATE 2, 10: PRINT "BLUE 2 SCENARIO DATA INPUT SCREEN"
  msg 50
  leep% = 40
 CASE 4
  LOCATE 2, 10: PRINT "RED 2 SCENARIO DATA INPUT SCREEN"
  msg 60
  leep% = 40
  GOTO rdat
 CASE ELSE
  'free fighter
  'later enhancement
END SELECT

FOR I = 22 TO (21 + BluDat%)
```

```
 LOCATE (I - 16), 62: PRINT Dat(I + leep%)
NEXT I

'Collect new data

22 LOCATE 6, 72: INPUT test$  'Max combat weight
  IF (test$ ¡¿ "") THEN
    Dat(22 + leep%) = VAL(test$)
  END IF
  Fuel(k%) = Dat(22 + leep%)
  IF Fuel(k%) ¡ 0 THEN
    BEEP
    LOCATE 6, 72: PRINT "        "
    GOTO 22
  END IF
  LOCATE 6, 72: PRINT Fuel(k%)

23 LOCATE 7, 72: INPUT test$  'Bingo fuel weight
  IF (test$ ¡¿ "") THEN
    Dat(23 + leep%) = VAL(test$)
  END IF
  bingo(k%) = Dat(23 + leep%)
  IF (bingo(k%) ¡ 0 OR bingo(k%) ¿ Fuel(k%)) THEN
    BEEP
    LOCATE 7, 72: PRINT "        "
    GOTO 23
  END IF
    LOCATE 7, 72: PRINT bingo(k%)

24 LOCATE 8, 72: INPUT test$   'Patrol velocity
  IF (test$ ¡¿ "") THEN
    Dat(24 + leep%) = VAL(test$)
  END IF
  V(k%) = Dat(24 + leep%)
  IF (V(k%) ¡ 100 OR V(k%) ¿ 1500) THEN
    BEEP
    LOCATE 8, 72: PRINT "        "
    GOTO 24
  END IF
  LOCATE 8, 72: PRINT V(k%)

25 LOCATE 9, 72: INPUT test$   'Pursuit velocity
  IF (test$ ¡¿ "") THEN
    Dat(25 + leep%) = VAL(test$)
  END IF
  Vp(k%) = Dat(25 + leep%)
  IF (Vp(k%) ¡ 100 OR Vp(k%) ¿ 1500) THEN
    BEEP
    LOCATE 9, 72: PRINT "        "
    GOTO 25
  END IF
  LOCATE 9, 72: PRINT Vp(k%)


26 LOCATE 10, 72: INPUT test$   'Visual Range
  IF (test$ ¡¿ "") THEN
    Dat(26 + leep%) = VAL(test$)
  END IF
  OptRng(k%) = Dat(26 + leep%)
  IF (OptRng(k%) ¡ 0) THEN
    BEEP
    LOCATE 10, 72: PRINT "        "
    GOTO 26
  END IF
  LOCATE 10, 72: PRINT OptRng(k%)

27 LOCATE 11, 72: INPUT test$   'Visual Range
  IF (test$ ¡¿ "") THEN
   Dat(27 + leep%) = VAL(test$)
  END IF
  RadRng(k%) = Dat(27 + leep%)
  IF (RadRng(k%) ¡ OptRng(k%)) THEN
    BEEP
    LOCATE 11, 72: PRINT "        "
    GOTO 27
  END IF
  LOCATE 11, 72: PRINT RadRng(k%)
```

```
28 LOCATE 12, 72: INPUT test$  'Number of missiles
   IF (test$ ¡¿ "") THEN
    Dat(28 + leep%) = VAL(test$)
   END IF
   NumMSL%(k%) = Dat(28 + leep%)
   IF (NumMSL%(k%) ¡ 0 OR NumMSL%(k%) ¿ 4) THEN
    BEEP
    LOCATE 12, 72: PRINT "      "
    GOTO 28
   END IF
   LOCATE 12, 72: PRINT NumMSL%(k%)

29 LOCATE 13, 72: INPUT test$  'Missile velocity
  IF (test$ ¡¿ "") THEN
   Dat(29 + leep%) = VAL(test$)
  END IF
  MsLV(k%) = Dat(29 + leep%)
  IF (MsLV(k%) ¡ V(k%)) THEN
   BEEP
   LOCATE 13, 72: PRINT "      "
   GOTO 29
  END IF
  LOCATE 13, 72: PRINT MsLV(k%)

30 LOCATE 14, 72: INPUT test$  'Missile Sen Rng
  IF (test$ ¡¿ "") THEN
   Dat(30 + leep%) = VAL(test$)
  END IF
  MsLSenRg(k%) = Dat(30 + leep%)
  IF (MsLSenRg(k%) ¡ 0) THEN
   BEEP
   LOCATE 14, 72: PRINT "      "
   GOTO 30
  END IF
  LOCATE 14, 72: PRINT MsLSenRg(k%)

GOTO endblue

rdat:

FOR I = 42 TO (41 + RedDat%)
 LOCATE (I - 36), 62: PRINT Dat(I + leep%)
NEXT I

42 LOCATE 6, 72: INPUT test$    'Initial starting X position
  IF (test$ ¡¿ "") THEN
   Dat(42 + leep%) = VAL(test$)
  END IF
  ACx(k%) = Dat(42 + leep%)
  LOCATE 6, 72: PRINT ACx(k%)

LOCATE 7, 72: INPUT test$    'Initial starting Y position
  IF (test$ ¡¿ "") THEN
   Dat(43 + leep%) = VAL(test$)
  END IF
  ACy(k%) = Dat(43 + leep%)
  LOCATE 7, 72: PRINT ACy(k%)

44 LOCATE 8, 72: INPUT test$  'Max combat weight
  IF (test$ ¡¿ "") THEN
   Dat(44 + leep%) = VAL(test$)
  END IF
  Fuel(k%) = Dat(44 + leep%)
  IF Fuel(k%) ¡ 0 THEN
   BEEP
   LOCATE 6, 72: PRINT "      "
   GOTO 44
  END IF
  LOCATE 8, 72: PRINT Fuel(k%)

45 LOCATE 9, 72: INPUT test$  'Bingo fuel weight
  IF (test$ ¡¿ "") THEN
   Dat(45 + leep%) = VAL(test$)
  END IF
  bingo(k%) = Dat(45 + leep%)
  IF (bingo(k%) ¡ 0 OR bingo(k%) ¿ Fuel(k%)) THEN
```

```
    BEEP
    LOCATE 9, 72: PRINT "       "
    GOTO 45
  END IF
    LOCATE 9, 72: PRINT bingo(k%)

46 LOCATE 10, 72: INPUT test$   'Patrol velocity
  IF (test$ ¡¿ "") THEN
    Dat(46 + leep%) = VAL(test$)
  END IF
  V(k%) = Dat(46 + leep%)
  IF (V(k%) ¡ 100 OR V(k%) ¿ 1500) THEN
    BEEP
    LOCATE 10, 72: PRINT "       "
    GOTO 46
  END IF
  LOCATE 10, 72: PRINT V(k%)

47 LOCATE 11, 72: INPUT test$   'Pursuit velocity
  IF (test$ ¡¿ "") THEN
    Dat(47 + leep%) = VAL(test$)
  END IF
  Vp(k%) = Dat(47 + leep%)
  IF (Vp(k%) ¡ 100 OR Vp(k%) ¿ 1500) THEN
    BEEP
    LOCATE 11, 72: PRINT "       "
    GOTO 47
  END IF
  LOCATE 11, 72: PRINT Vp(k%)

48 LOCATE 12, 72: INPUT test$   'Visual Range
  IF (test$ ¡¿ "") THEN
    Dat(48 + leep%) = VAL(test$)
  END IF
  OptRng(k%) = Dat(48 + leep%)
  IF (OptRng(k%) ¡ 0) THEN
    BEEP
    LOCATE 12, 72: PRINT "       "
    GOTO 48
  END IF
  LOCATE 12, 72: PRINT OptRng(k%)

49 LOCATE 13, 72: INPUT test$   'Visual Range
  IF (test$ ¡¿ "") THEN
   Dat(49 + leep%) = VAL(test$)
  END IF
  RadRng(k%) = Dat(49 + leep%)
  IF (RadRng(k%) ¡ OptRng(k%)) THEN
    BEEP
    LOCATE 13, 72: PRINT "       "
    GOTO 49
  END IF
  LOCATE 13, 72: PRINT RadRng(k%)

50 LOCATE 14, 72: INPUT test$  'Number of missiles
   IF (test$ ¡¿ "") THEN
    Dat(50 + leep%) = VAL(test$)
   END IF
   NumMSL%(k%) = Dat(50 + leep%)
   IF (NumMSL%(k%) ¡ 0 OR NumMSL%(k%) ¿ 4) THEN
    BEEP
    LOCATE 14, 72: PRINT "       "
    GOTO 50
   END IF
   LOCATE 14, 72: PRINT NumMSL%(k%)

51 LOCATE 15, 72: INPUT test$  'Missile velocity
  IF (test$ ¡¿ "") THEN
    Dat(51 + leep%) = VAL(test$)
  END IF
  MsLV(k%) = Dat(51 + leep%)
  IF (MsLV(k%) ¡ V(k%)) THEN
    BEEP
    LOCATE 15, 72: PRINT "       "
    GOTO 51
  END IF
  LOCATE 15, 72: PRINT MsLV(k%)
```

```
52 LOCATE 16, 72: INPUT test$  'Missile Sen Rng
 IF (test$ ¡¿ "") THEN
   Dat(52 + leep%) = VAL(test$)
 END IF
 MsLSenRg(k%) = Dat(52 + leep%)
 IF (MsLSenRg(k%) ¡ 0) THEN
   BEEP
   LOCATE 16, 72: PRINT "      "
   GOTO 52
 END IF
 LOCATE 16, 72: PRINT MsLSenRg(k%)

endblue:
NEXT k%


' Begin page 6
CLS
Frame 1, 80, 1, 13
COLOR 3
LOCATE 3, 20: PRINT "FINAL PAGE"
COLOR 7
otpt% = 0
LOCATE 5, 4: INPUT "Do you want a detail summary generated Y/N
(N)? "; outans$
IF UCASE$(outans$) = "Y" THEN otpt% = -1
LOCATE 7, 4: INPUT "Length of simulation (No limit): "; endsim%
LOCATE 9, 4: INPUT "Do you want to save this scenario Y/N (N): ";
ScnAns$
  IF (UCASE$(ScnAns$) = "Y") THEN
    LOCATE 11, 8: INPUT "Enter the new scenario file name: ";
NewFile$
    OPEN NewFile$ FOR OUTPUT AS #5     'Modified Scenario file
      WRITE #5, "Base"
    FOR I = 2 TO 10
      WRITE #5, Dat(I)
    NEXT I
      WRITE #5, "Environment"
    FOR I = 12 TO 20
      WRITE #5, Dat(I)
    NEXT I
      WRITE #5, "Blue1"
    FOR I = 22 TO 40
      WRITE #5, Dat(I)
    NEXT I
      WRITE #5, "Red1"
    FOR I = 42 TO 60
      WRITE #5, Dat(I)
    NEXT I
      WRITE #5, "Blue2"
    FOR I = 62 TO 80
      WRITE #5, Dat(I)
    NEXT I
      WRITE #5, "Red2"
    FOR I = 82 TO 100
      WRITE #5, Dat(I)
    NEXT I
    CLOSE #5
  END IF
END SUB

SUB ZoomIn
      CLS
      Scale = Scale - .2
      IF (Scale ¡ .4) THEN
        Scale = .2
      END IF
      VIEW (1, 17)-(MaxScrX%, MaxScrY%), 3, 4
      WINDOW (-1! * Wx * Scale, Wy * Scale)-(Wx * Scale, -1! * Wy
* Scale)
      BBase
      Grid
END SUB

SUB ZoomOut
```

```
    CLS
    Scale = Scale + .2
    VIEW (1, 17)-(MaxScrX%, MaxScrY%), 3, 4
    WINDOW (-1 * Wx * Scale, Wy * Scale)-(Wx * Scale, -1 * Wy *
Scale)
    BBase
    Grid
END SUB
```

MODOUT

```
'***************************************
'Program: Output Report Generator

'Author: Capt R. Moore

'Date: 15 Feb 91

'***************************************
DECLARE SUB pause ()
' Main program for Output module

DECLARE SUB Frame (Left%, Right%, Top%, Bottom%)
DECLARE SUB Reflect ()
DECLARE SUB Sigevnt ()
DECLARE SUB ACperf ()
DECLARE SUB Detlrpt ()
DECLARE SUB dtlhelp ()

COMMON SHARED /Aircraft/ NumAC%

SUB ACperf

CLOSE #7  'Assurance against file possibly left open

'Reads and displays the aircraft performance report

CLS
Frame 2, 78, 2, 20

COLOR 3: LOCATE 2, 18: PRINT "AIRCRAFT PERFORMANCE FACTORS
LISTING"
COLOR 7
OPEN "acperf.dat" FOR INPUT AS #7
i% = 2
DO WHILE NOT EOF(7)
 i% = i% + 2
 LINE INPUT #7, line$
 LOCATE i%, 5: PRINT line$
 IF i% = 18 THEN
   COLOR 15: LOCATE 20, 25: PRINT "Press any key to continue"
   COLOR 7
   DO WHILE INKEY$ = "": LOOP
   CLS
   Frame 2, 78, 2, 20
   COLOR 3: LOCATE 2, 18: PRINT "AIRCRAFT PERFORMANCE FACTORS
LISTING"
   COLOR 7
   i% = 2
 END IF
LOOP
   COLOR 15: LOCATE 8, 17: PRINT "End of file, press any key to
continue"
   COLOR 7
   DO WHILE INKEY$ = "": LOOP
   CLS

CLOSE #7
END SUB

SUB Detlrpt

'Reads and displays the detail event report

CLS
COLOR 3: LOCATE 1, 25: PRINT "DETAIL EVENT LISTING"
LOCATE 3, 2: PRINT "        Coordinates        Vel   Heading
 Tgt Coord   Range   Det Msl"
LOCATE 4, 2: PRINT " Sec  AC   X      Y   Stat (Mi/S)  (Rad)   Tgt
 X     Y    to Tgt  Lvl Stat"
COLOR 7
CLOSE #6
OPEN "Detail.dat" FOR INPUT AS #6
i% = 4
DO WHILE NOT EOF(6)
 i% = i% + 1
 LINE INPUT #6, line$
 LOCATE i%, 2: PRINT line$
 IF i% = 19 THEN
   COLOR 15: LOCATE 23, 7: PRINT "Press ¡H¿ for help, spacebar to
exit, or any other key to continue"
   COLOR 7
   test$ = ""
   DO WHILE test$ = ""
     test$ = INKEY$
   LOOP
     IF test$ = " " GOTO enddetl
     IF UCASE$(test$) = "H" THEN
       dtlhelp
     END IF
   CLS
    COLOR 3: LOCATE 1, 25: PRINT "DETAIL EVENT LISTING"
    LOCATE 3, 2: PRINT "          Coordinates        Vel   Heading
     Tgt Coord   Range  Det Msl"
    LOCATE 4, 2: PRINT " Sec  AC   X       Y   Stat (Mi/S)  (Rad)
Tgt   X      Y    to Tgt  Lvl Stat"
    COLOR 7
   i% = 4
 END IF
LOOP
  COLOR 15: LOCATE 23, 12: PRINT "End of file, press any key to
continue"
  COLOR 7
  DO WHILE INKEY$ = "": LOOP
  CLS
enddetl:
CLOSE #6
END SUB

SUB dtlhelp
CLS

LOCATE 3, 2: PRINT "Sec: The time of the simulation in seconds"
LOCATE 5, 2: PRINT "AC: The aircraft tail number"
LOCATE 6, 10: PRINT "1 -¿ Blue 1"
LOCATE 7, 10: PRINT "2 -¿ Red 1"
LOCATE 8, 10: PRINT "3 -¿ Blue 2"
LOCATE 9, 10: PRINT "4 -¿ Red 2"
LOCATE 11, 2: PRINT "Coordinates X and Y:  Cartesian coordinates
of the ground"
LOCATE 13, 2: PRINT "Stat: Aircraft status:"
LOCATE 14, 10: PRINT "0 -¿ Undetected"
LOCATE 15, 10: PRINT "1 -¿ Detected"
LOCATE 16, 10: PRINT "2 -¿ KIA"
LOCATE 17, 10: PRINT "3 -¿ Patrol"
LOCATE 18, 10: PRINT "4 -¿ Pursuit"
LOCATE 19, 10: PRINT "6 -¿ Bingo Fuel"
LOCATE 21, 2: PRINT "Vel (Mi/S): Current velocity in miles per
second"
LOCATE 24, 25: PRINT "Press spacebar to continue"
pause
CLS
LOCATE 3, 2: PRINT "Heading: Current heading, in radians, of the
aircraft"
LOCATE 5, 2: PRINT "Tgt: Current target assignment.  Values
correspond with tail numbers above "
LOCATE 7, 2: PRINT "Tgt Coord X and Y: Cartesian coordinates of
the target"
LOCATE 9, 2: PRINT "Rng to Tgt: Range, in miles, to currently
assigned target"
LOCATE 11, 2: PRINT "Det Lvl: Current sensor being used to track
intruder."
LOCATE 12, 10: PRINT "0 -¿ No sensor"
LOCATE 13, 10: PRINT "1 -¿ Tactical surveillance radar"
LOCATE 14, 10: PRINT "2 -¿ Interceptor's on-board radar"
LOCATE 15, 10: PRINT "3 -¿ Interceptor's optical sensor"
LOCATE 17, 2: PRINT "Msl Stat: Missile status"
LOCATE 18, 10: PRINT "0 -¿ Carried"
LOCATE 19, 10: PRINT "1 -¿ In-flight"
LOCATE 20, 10: PRINT "2 -¿ Destroyed"
LOCATE 24, 25: PRINT "Press spacebar to return to report"
pause
CLS

END SUB
```

```
SUB Reflect

'Reads and displays the scenario and performance files

CLS
Frame 2, 78, 2, 23

COLOR 3: LOCATE 2, 20: PRINT "DISPLAY OF DEFAULT SCENARIO INPUT
FILE"
COLOR 7
CLOSE #5
OPEN "MASTER.dat" FOR INPUT AS #5
i% = 3
DO WHILE NOT EOF(5)
 i% = i% + 1
 LINE INPUT #5, line$
 LOCATE i%, 5: PRINT line$
 IF i% = 21 THEN
   COLOR 15: LOCATE 23, 25: PRINT "Press any key to continue"
   COLOR 7
   DO WHILE INKEY$ = "": LOOP
   CLS
   Frame 2, 78, 2, 23
   COLOR 3: LOCATE 2, 20: PRINT "DISPLAY OF DEFAULT SCENARIO
INPUT FILE"
   COLOR 7
   i% = 3
 END IF
LOOP
   COLOR 15: LOCATE 23, 14: PRINT "End of scenario file, press
any key to continue"
   COLOR 7
   DO WHILE INKEY$ = "": LOOP
   CLS


CLOSE #1
CLOSE #2
CLOSE #3
CLOSE #4

OPEN "B1Perf.doc" FOR INPUT AS #1   'Blue One performance data
OPEN "R1Perf.doc" FOR INPUT AS #2   'Red One performance data
OPEN "B2Perf.doc" FOR INPUT AS #3   'Blue Two performance data
OPEN "R2Perf.doc" FOR INPUT AS #4   'Red Two performance data

DIM AC$(NumAC%)

AC$(1) = "Blue 1"
AC$(2) = "Red 1"
AC$(3) = "Blue 2"
AC$(4) = "Red 2"

FOR j% = 1 TO NumAC%
COLOR 3: LOCATE 2, 17: PRINT USING "DISPLAY OF & PERFORMANCE
INPUT FILE"; AC$(j%)
COLOR 7
i% = 3
DO WHILE NOT EOF(j%)
 i% = i% + 1
 INPUT #j%, line$
 LOCATE i%, 5: PRINT line$
 IF i% = 21 THEN
   COLOR 15: LOCATE 23, 25: PRINT "Press any key to continue"
   COLOR 7
   DO WHILE INKEY$ = "": LOOP
   CLS
   COLOR 3: LOCATE 2, 20: PRINT USING "DISPLAY OF & PERFORMANCE
INPUT FILE"; AC$(j%)
   COLOR 7
   i% = 3
 END IF
LOOP
   COLOR 15: LOCATE 23, 14: PRINT "End of  file, press any key to
continue"
   COLOR 7
```

```
  DO WHILE INKEY$ = "": LOOP
  CLS
NEXT j%

CLOSE #1
CLOSE #2
CLOSE #3
CLOSE #4
CLOSE #5



END SUB

SUB Sigevnt

'Reads and displays the significant event report

CLS
Frame 2, 78, 2, 23

COLOR 3: LOCATE 2, 25: PRINT "SIGNIFICANT EVENT REPORT"
COLOR 7
CLOSE #12
OPEN "output.dat" FOR INPUT AS #12
i% = 3
DO WHILE NOT EOF(12)
 i% = i% + 2
 LINE INPUT #12, line$
 LOCATE i%, 5: PRINT line$
 IF i% = 19 THEN
   COLOR 15: LOCATE 23, 25: PRINT "Press any key to continue"
   COLOR 7
   DO WHILE INKEY$ = "": LOOP
   CLS
   Frame 2, 78, 2, 23
   COLOR 3: LOCATE 2, 25: PRINT "SIGNIFICANT EVENT REPORT"
   COLOR 7
   i% = 3
 END IF
LOOP
   COLOR 15: LOCATE 23, 18: PRINT "End of file, press any key to
continue"
   COLOR 7
   DO WHILE INKEY$ = "": LOOP
   CLS

CLOSE #12


END SUB
```

MODMSG

```
'******************************

'Purpose: Select screen messages

'Author: Capt R. Moore

'Date: 15 Feb 91

'******************************


DECLARE SUB Frame (Left%, Right%, Top%, Bottom%)
DECLARE SUB pause ()
COMMON SHARED /messages/ com$, TNOW%

SUB Mmsg (msgnum%, AC%, MSL%, ACT%)
IF AC% = 1 THEN AC$ = "Blue 1"
IF AC% = 2 THEN AC$ = "Red 1"
IF AC% = 3 THEN AC$ = "Blue 2"
IF AC% = 4 THEN AC$ = "Red 2"
IF ACT% = 1 THEN ACT$ = "Blue 1"
IF ACT% = 2 THEN ACT$ = "Red 1"
IF ACT% = 3 THEN ACT$ = "Blue 2"
IF ACT% = 4 THEN ACT$ = "Red 2"



SELECT CASE msgnum%
    CASE 10
     PRINT #10, USING "At #### seconds, & launched missile No. #
at & ."; TNOW%; AC$; MSL%; ACT$
    CASE 20
     PRINT #10, USING "At #### seconds, & missile, No. # ,
destroyed & ."; TNOW%; AC$; MSL%; ACT$
    CASE 30
     PRINT #10, USING "At #### seconds, & missile, No. # ,
missed & "; TNOW%; AC$; MSL%; ACT$
    END SELECT



END SUB

SUB msg (msgnum%) STATIC

'May want to use aircraft numbers instead

a = 35  'Starting position for comm messages

'File 10 is the output file

SELECT CASE msgnum%
 CASE 0 'Initialization case

     Use1% = 0
     Use2% = 0
     Use3% = 0
     Use4% = 0
     Use5% = 0
     Use6% = 0
     Use7% = 0
     Use8% = 0

 CASE 1
  IF Use1% = 0 THEN
     PRINT #10, USING "At #### seconds, target identified as Red
2 was assigned to Blue1 and 2"; TNOW%
     com$ = "Blue 1 and 2 intercept boggie Red 2"
     LOCATE 1, a: PRINT com$
     Use1% = 1
  END IF
 CASE 2
  IF Use2% = 0 THEN
     PRINT #10, USING "At #### seconds, target identified as Red
1 was assigned to Blue1 and 2"; TNOW%
     com$ = "Blue 1 and 2 intercept boggie Red 1          "
     LOCATE 1, a: PRINT com$
    Use2% = 1
  END IF
 CASE 3
  IF Use3% = 0 THEN
     PRINT #10, USING "At #### seconds, System status went Low,
no targets observed"; TNOW%
  END IF
  Use3% = 1
 CASE 4
  IF Use4% = 0 THEN
    PRINT #10, USING "At #### seconds, System status went Low,
no targets observed"; TNOW%
  END IF
  Use4% = 1
 CASE 5
  IF Use5% = 0 THEN
    PRINT #10, USING "At #### seconds, target identified as Red
1 was assigned to Blue1"; TNOW%
    com$ = "Blue 1 intercept boggie Red 1                  "
    LOCATE 1, a: PRINT com$
    Use5% = 1
  END IF
 CASE 6
  IF Use6% = 0 THEN
    PRINT #10, USING "At #### seconds, target identified as Red
2 was assigned to Blue2"; TNOW%
    com$ = "Blue 2 intercept boggie Red 2                  "
    LOCATE 1, a: PRINT com$
    Use6% = 1
  END IF
 CASE 7
  IF Use7% = 0 THEN
    PRINT #10, USING "At #### seconds, target identified as Red
2 was assigned to Blue1"; TNOW%
    com$ = "Blue 1 intercept boggie Red 2                  "
    LOCATE 1, a: PRINT com$
    Use7% = 1
  END IF
 CASE 8
  IF Use8% = 0 THEN
    PRINT #10, USING "At #### seconds, target identified as Red
1 was assigned to Blue2"; TNOW%
    com$ = "Blue 2 intercept boggie Red 1                  "
    LOCATE 1, a: PRINT com$
    Use8% = 1
  END IF
 CASE 10
  com$ = "    Its Miller Time!                    "
  LOCATE 1, a: PRINT com$


' Begin Examples description

 CASE 20  ' Load and Go opening text

COLOR 3
LOCATE 3, 20: PRINT "LOAD SCENARIO AND BEGIN SIMULATION"
COLOR 7
LOCATE 6, 8: PRINT "This option allows you to load an existing "
LOCATE 7, 8: PRINT "scenario file and automatically begin the "
LOCATE 8, 8: PRINT "simulation without returning to the menu. "
LOCATE 10, 8: PRINT "If you want to use a scenario file other
than the "
LOCATE 11, 8: PRINT "default file, enter an 'N' at the next
prompt, "
LOCATE 12, 8: PRINT "otherwise ¡RTN¿."

 CASE 30 '1st page of scenario editor

CLS

Frame 1, 60, 1, 23
Frame 60, 70, 1, 23
Frame 70, 80, 1, 23
COLOR 3
'
12345678901234567890123456789012345678901234567890
```

```
LOCATE 3, 10: PRINT "SCENARIO DATA INPUT SCREEN, PAGE 1"
LOCATE 5, 61: PRINT "DEFAULTS"
LOCATE 5, 74: PRINT "NEW"
LOCATE 6, 4: PRINT "GROUND UNIT DATA"
LOCATE 11, 4: PRINT "GENERAL AIRCRAFT DATA"
COLOR 7
LOCATE 7, 4: PRINT "Forward operating location X coordinate
(-250,250):"
LOCATE 8, 4: PRINT "Forward operating location Y coordinate
(-250,250):"
LOCATE 9, 4: PRINT "Forward operating location GCI max range
(250):"

LOCATE 12, 4: PRINT "Number of aircraft in simulation (Limit 4):"
LOCATE 13, 4: PRINT "Altitude for which simulation takes place
(20000):"
LOCATE 14, 4: PRINT "CAP station location X coordinate
(-250,250):"
LOCATE 15, 4: PRINT "CAP station location Y coordinate
(-250,250):"


 CASE 40  'Edit Scenario text

COLOR 3
LOCATE 2, 20: PRINT "USER DEFINED SCENARIO OPTION"
COLOR 7
LOCATE 6, 8: PRINT "This option permits you the flexibility to
create"
LOCATE 7, 8: PRINT "a new scenario file from data contained
within the"
LOCATE 8, 8: PRINT "default scenario or from a previously
modified scenario file."
LOCATE 10, 8: PRINT "If you want to create a new scenario based
on the default"
LOCATE 11, 8: PRINT "file, press 'Enter' at the next prompt,
otherwise type "
LOCATE 12, 9: PRINT "N and press 'Enter'."

 CASE 50  'Blue labels for scenario editor
COLOR 3
Frame 1, 60, 1, 23
Frame 60, 70, 1, 23
Frame 70, 80, 1, 23

LOCATE 4, 10: PRINT " AIRCRAFT DATA ELEMENTS"
LOCATE 4, 61: PRINT "DEFAULTS"
LOCATE 4, 74: PRINT "NEW"
COLOR 7
LOCATE 6, 4: PRINT "Initial fuel (lbs):"
LOCATE 7, 4: PRINT "Bingo fuel level (greater than initial):"
LOCATE 8, 4: PRINT "Patrol velocity (min: 100, max 1500 mph):"
LOCATE 9, 4: PRINT "Pursuit velocity (min: 100, max 1500 mph):"
LOCATE 10, 4: PRINT "Optical range for 0.5 probability of
detection:"
LOCATE 11, 4: PRINT "Radar range for 0.5 probability of
detection:"
LOCATE 12, 4: PRINT "Initial missile count (min: 0, max: 4):"
LOCATE 13, 4: PRINT "Missile steady state velocity (mph):"
LOCATE 14, 4: PRINT "Missile sensor range for 0.5 probability of
lock-on:"

 CASE 60  'Red labels for scenario editor

COLOR 4

Frame 1, 60, 1, 23
Frame 60, 70, 1, 23
Frame 70, 80, 1, 23

LOCATE 4, 4: PRINT " AIRCRAFT DATA ELEMENTS"
LOCATE 4, 61: PRINT "DEFAULTS"
LOCATE 4, 74: PRINT "NEW"
COLOR 7
LOCATE 6, 4: PRINT "Initial X coordinate (Range: -250, 250):"
LOCATE 7, 4: PRINT "Initial Y coordinate (Range: -250, 250):"
LOCATE 8, 4: PRINT "Initial fuel (min:      max:      lbs):"
LOCATE 9, 4: PRINT "Bingo fuel level (greater than initial):"
LOCATE 10, 4: PRINT "Attack velocity (min: 100, max 1000 mph):"
LOCATE 11, 4: PRINT "Evade velocity (min: 100, max 1000 mph):"
LOCATE 12, 4: PRINT "Optical range for 0.5 probability of
detection:"
LOCATE 13, 4: PRINT "Radar range for 0.5 probability of
detection:"
LOCATE 14, 4: PRINT "Initial missile count (min: 0, max: 4):"
LOCATE 15, 4: PRINT "Missile steady state velocity (mph):"
LOCATE 16, 4: PRINT "Missile sensor range for 0.5 probability of
lock-on:"


 CASE 101  'Main menu text

  LOCATE 17, 3: PRINT "The menu displayed above is ATACS' main
menu.  The first option, Examples"
  LOCATE 18, 3: PRINT "of Processes, introduces a host of textual
and animated examples of common"
  LOCATE 19, 3: PRINT "features found in most combat models.  The
second option, Combat "
  LOCATE 20, 3: PRINT "Demonstration, introduces the model's
scenario and presents an animated "
  LOCATE 21, 3: PRINT "demonstration of simulated air-to-air
combat.  The third option, Display"
  LOCATE 22, 3: PRINT "Output, presents output products commonly
found in a simulation of this type."
  LOCATE 23, 3: PRINT "The final option, Terminate, is
self-explanatory.  To select an option,"
  LOCATE 24, 3: PRINT "enter the highlighted letter of that
option."

CASE 1011
  LOCATE 12, 5: PRINT "You have not ran the combat demonstration
as of yet. You must run the "
  LOCATE 13, 5: PRINT "demonstration first before any output will
be produced.  I'll direct"
  LOCATE 14, 5: PRINT "you to the demonstration menu."
  LOCATE 18, 20: PRINT "Press the spacebar to return."
  pause
  CLS

CASE 102 'Examples menu text
  LOCATE 20, 5: PRINT "The purpose of each example is to
demonstrate some possible feature or "
  LOCATE 21, 5: PRINT "process typically found in combat models.
Several of the examples are"
  LOCATE 22, 5: PRINT "animated and require your participation."

CASE 103  'Demonstration menu text
  LOCATE 17, 5: PRINT "The first option, Express Load and Go,
offers you the opportunity to select"
  LOCATE 18, 5: PRINT "an existing scenario file and begin the
simulation directly.  The second"
  LOCATE 19, 5: PRINT "option, Load and Edit, permits viewing and
editing of any existing "
  LOCATE 20, 5: PRINT "scenario file.  Following an edit session
of a scenario file, select "
  LOCATE 21, 5: PRINT "Run Demonstration to begin the air-to-air
combat simulation."

CASE 1031
  LOCATE 12, 5: PRINT "You have not loaded a scenario file as of
yet. You must select"
  LOCATE 13, 5: PRINT "the second option, Load and Edit Scenario
File, before running the"
  LOCATE 14, 5: PRINT "demonstration or use the express option to
by pass it all."
  LOCATE 18, 20:  PRINT "Press the spacebar to return."
  pause
  CLS

CASE 1032
  LOCATE 12, 5: PRINT "You must reload a scenario before the
demonstration can be ran again."
  LOCATE 18, 20:  PRINT "Press the spacebar to return."
 pause
 CLS

CASE 104

 LOCATE 20, 3: PRINT "This menu lists ATACS' available output
options.  A description of each "
 LOCATE 21, 3: PRINT "output listing is available upon selection
of that option."

CASE 1041  'Reflected input report
CLS
COLOR 3
LOCATE 2, 25: PRINT "REFLECTED INPUT"
COLOR 7
LOCATE 5, 5: PRINT "This feature allows the student to see the
extent of data "
LOCATE 7, 5: PRINT "that may be embedded in the model without the
user's knowledge"
LOCATE 9, 5: PRINT "of its existence.  The point here is to bring
about an awareness"
LOCATE 11, 5: PRINT "of embedded data which is not always
apparent to the"
LOCATE 13, 5: PRINT "user of the combat model."
LOCATE 16, 5: PRINT "The first listing displayed is the scenario
data used as "
LOCATE 18, 5: PRINT "the default scenario file.  The same data
can been seen when"
LOCATE 20, 5: PRINT "editing the scenario file.  The second
listing displayed is"
LOCATE 22, 5: PRINT "the raw aircraft performance data."
LOCATE 24, 20:  PRINT "Press the spacebar to begin."
 pause
 CLS


CASE 1042  'Detailed output report
CLS
COLOR 3
LOCATE 6, 25: PRINT "Detail Summary"
COLOR 7
LOCATE 9, 5: PRINT "A detailed summary is an option found in most
combat models."
LOCATE 11, 5: PRINT "Its purpose is to provided the model user
with a second by "
LOCATE 13, 5: PRINT "second account of events occurring during
the simulation.  "
LOCATE 24, 20: PRINT "Press the spacebar to begin."
 pause
 CLS


CASE 1043  'Aircraft performance
COLOR 3
LOCATE 6, 25: PRINT "Detail Summary"
COLOR 7
LOCATE 9, 5: PRINT "This option displays aircraft parameters
computed by the model"
LOCATE 11, 5: PRINT "during initialization.  For each of the
aircraft's two velocity "
LOCATE 13, 5: PRINT "settings, performance parameters are
calculated.  Those dis-"
LOCATE 15, 5: PRINT "played here were used in modeling such
activities as fuel"
LOCATE 17, 5: PRINT "consumption, sustained G-force for
maintaining level turns "
LOCATE 19, 5: PRINT "and the turning radius given the altitude
and velocity. "
LOCATE 24, 20:  PRINT "Press the spacebar to begin."
 pause
 CLS

CASE 1044  'Sig event report
COLOR 3
LOCATE 3, 25: PRINT "Significant Events Summary"
COLOR 7
LOCATE 6, 5: PRINT "The final report, the Summary of Significant
```

```
Events, provides"
LOCATE 8, 5: PRINT "a condensed listing of significant events,
the participants, and"
LOCATE 10, 5: PRINT "the time of their occurrence. The advantage
of such a report "
LOCATE 12, 5: PRINT "permits examination of the state conditions
of the simulation"
LOCATE 14, 5: PRINT "without having to search through detailed
listing like that "
LOCATE 16, 5: PRINT "provided by the second option, Detail
Summary. This report is"
LOCATE 18, 5: PRINT "is typical of reports used by modelers in
debugging and "
LOCATE 20, 5: PRINT "validating models."
LOCATE 24, 20: PRINT "Press the spacebar to begin."
 pause
 CLS

 CASE 110 'Random number example text

CLS
ans$ = " "
LOCATE 10, 20: INPUT "Display Random Number Generation
Introduction (Y/N)"; ans$
IF UCASE$(ans$) = "N" GOTO endrngx

CLS
COLOR 3
LOCATE 3, 25: PRINT "Random Number Generation"
COLOR 7


LOCATE 6, 5: PRINT "A random number generator (RNG) lies at the
core of any Monte-Carlo"
LOCATE 8, 5: PRINT "simulation. Its purpose is to simulate the
random chance or"
LOCATE 10, 5: PRINT "uncertainty that occurs in real world
systems. Since the RNG plays"
LOCATE 12, 5: PRINT "such an important role in simulation, as a
minimum, the RNG should"
LOCATE 14, 5: PRINT "be graphically validated. The purpose of
this example is to"
LOCATE 16, 5: PRINT "provide that visual validation and
interaction with the RNG."

LOCATE 19, 5: PRINT "The uniform RNG generates independent,
identically distributed"
LOCATE 21, 5: PRINT "random variates from a uniform (0,1)
distribution. Typically, a RNG"

LOCATE 24, 20: PRINT "Press the spacebar to continue"
pause
CLS

LOCATE 3, 5: PRINT "produces random variates ranging between 0 to
1. The"
LOCATE 5, 5: PRINT "probability of any value occurring in the
generator's range"
LOCATE 7, 5: PRINT "is the same for each value."

LOCATE 10, 5: PRINT "You begin the demonstration by entering the
number of samples to be"
LOCATE 12, 5: PRINT "taken. The number of samples should be 500
or more. You will then"
LOCATE 14, 5: PRINT "need to enter a seed value (a starting
point) for the generator."
LOCATE 16, 5: PRINT "When the test begins, an animated bar chart
displays the progress of."
LOCATE 18, 5: PRINT "the test. Once the test is complete, the
mean and variance of the "
LOCATE 20, 5: PRINT "samples are computed and displayed. "

LOCATE 24, 20: PRINT "Press the spacebar to continue"
pause
CLS

LOCATE 3, 5: PRINT "The demonstration allows you to perform the
```

```
test over and over."
LOCATE 5, 5: PRINT "You may want to investigate what impact seed
values have on the"
LOCATE 7, 5: PRINT "RNG. "

LOCATE 24, 20: PRINT "Press the spacebar to continue"
pause

endrngx:
CLS

CASE 111


CLS
COLOR 3
LOCATE 3, 25: PRINT "The Process of Search"
COLOR 7

LOCATE 5, 5: PRINT "To search implies a process of physically
looking or possibly"
LOCATE 7, 5: PRINT "electronically scanning over an area for a
specified period of"
LOCATE 9, 5: PRINT "time.  In simulating this process, the
looking may be an attempt to"
LOCATE 11, 5: PRINT "detect, and the frequency at which the
simulation attempts to"
LOCATE 13, 5: PRINT "detect may be a recursive loop slaved to a
timing mechanism.  In"
LOCATE 15, 5: PRINT "such a case, the process of searching is
more implicitly modeled"
LOCATE 17, 5: PRINT "than explicitly.  That is, search may exists
as a function of some"
LOCATE 19, 5: PRINT "other explicitly simulated process.  Such is
the case in this"
LOCATE 21, 5: PRINT "combat demonstrator."

LOCATE 24, 20: PRINT "Press the spacebar to continue"
pause
CLS

CASE 112

CLS
ans$ = " "
LOCATE 10, 15: INPUT "Display Process of Detection Introduction
(Y/N)"; ans$
IF UCASE$(ans$) = "N" GOTO enddetx

CLS
COLOR 3
LOCATE 3, 25: PRINT "The Process of Detection"
COLOR 7
LOCATE 5, 5: PRINT "The purpose of this example is to illustrate
and contrast"
LOCATE 7, 5: PRINT "two simple methods used in simulating target
detection. "
LOCATE 9, 5: PRINT "These two are the cookie cutter and
continuous detection methods."

LOCATE 12, 5: PRINT "The first example begins by plotting the
cookie cutter step"
LOCATE 14, 5: PRINT "function.  From the plot, you may observe
that this function"
LOCATE 16, 5: PRINT "offers two probabilities of detection.
These are 0 and 1."
LOCATE 18, 5: PRINT "The transition from 0 to 1 takes place at
the maximum range. "
LOCATE 20, 5: PRINT "For the cookie cutter, this value is that
range at which the"
LOCATE 22, 5: PRINT "probability of detection is 0.5. "
LOCATE 24, 20: PRINT "Press the spacebar to continue"
pause
CLS

LOCATE 3, 5: PRINT "Once the step function has been plotted, you
are prompted to"
```

```
LOCATE 5, 5: PRINT "enter a range-to-target.  The program
evaluates this"
LOCATE 7, 5: PRINT "range, places a maker on the plot
corresponding to this"
LOCATE 9, 5: PRINT "range, and displays the outcome of the
detection event. "

LOCATE 12, 5: PRINT "The second example introduces the continuous
detection method. A"
LOCATE 14, 5: PRINT "plot generated by the continuous detection
exponential model is"
LOCATE 16, 5: PRINT "over laid on the previous step function to
contrast these two"
LOCATE 18, 5: PRINT "methods.  From the plot of the output, you
may observe there exists"
LOCATE 20, 5: PRINT "an infinite number of probability values.
Pay particular attention"
LOCATE 22, 5: PRINT "at the point where the two functions cross.
"
LOCATE 24, 20: PRINT "Press the spacebar to continue"
pause
CLS

LOCATE 3, 5: PRINT "As before, you are prompted for a target
range.  Once entered, the"
LOCATE 5, 5: PRINT "probability of detection is determined and a
marker is placed on"
LOCATE 7, 5: PRINT "the curve.  The program then generates a
random number.  The random"
LOCATE 9, 5: PRINT "number is compared to the probability of
detection.  If the random"
LOCATE 11, 5: PRINT "number is lower, then the target was
detected; otherwise, the"
LOCATE 13, 5: PRINT "target was not detected."
LOCATE 24, 20: PRINT "Press the spacebar to continue"
pause

enddetx:
CLS

CASE 113

CLS
COLOR 3
LOCATE 3, 20: PRINT "The Process of Target Selection"
COLOR 7
LOCATE 5, 5: PRINT "The simulation of target selection is very
much simulation specific."
LOCATE 7, 5: PRINT "There are no target selection models readily
available to the combat"
LOCATE 9, 5: PRINT "modeler.  This is, because quite often target
selection is rule based."
LOCATE 11, 5: PRINT "A rule of engagement (ROE) is but one such
example.  The ROE may"
LOCATE 13, 5: PRINT "dictate what, how, and when a target is
selected and engaged."

LOCATE 16, 5: PRINT "In some cases, target selection is a process
of prioritization. An"
LOCATE 18, 5: PRINT "example where the simulated environment may
be rich with targets,"
LOCATE 20, 5: PRINT "or the targets present different threat
levels, a rule of"
LOCATE 22, 5: PRINT "prioritization must dictate the selection
order. "

LOCATE 24, 20: PRINT "Press the spacebar to continue"
pause
CLS

LOCATE 3, 5: PRINT "These rules may be represented as a set of
conditional statements"
LOCATE 5, 5: PRINT "embedded in the computer code.  The ATACS
combat demonstration has"
LOCATE 7, 5: PRINT "a target selection rule based on proximity.
Whichever interceptor"
LOCATE 9, 5: PRINT "is closer to a newly detected target is
```

```
assigned that target."
LOCATE 11, 5: PRINT "Additional interceptor's, possibly wingman,
are then free to engage"
LOCATE 13, 5: PRINT "additional targets as they are discovered. "

LOCATE 24, 20: PRINT "Press the spacebar to continue"
pause
CLS

CASE 114

CLS
ans$ = " "
LOCATE 10, 22: INPUT "Display CEP Introduction (Y/N)"; ans$
IF UCASE$(ans$) = "N" GOTO endcepx


CLS
COLOR 3
LOCATE 2, 19: PRINT "An Example of Circular Erorr Probable"
COLOR 7
LOCATE 4, 5: PRINT "The circular error probable (CEP) example is
designed"
LOCATE 6, 5: PRINT "to illustrate the concept and application of
the CEP factor. "

LOCATE 9, 5: PRINT "You begin the demonstration by test firing a
simulated weapon."
LOCATE 11, 5: PRINT "The data generated by the impact points are
used to determine"
LOCATE 13, 5: PRINT "dispersion characteristics of the weapon.
The mean impact point"
LOCATE 15, 5: PRINT "(MIP), the systematic dependent error,
referred as bias, and the"
LOCATE 17, 5: PRINT "CEP are computed and used to describe our
simulated weapon."

LOCATE 20, 5: PRINT "The MIP is the arithmetic mean of all of the
impact point"
LOCATE 22, 5: PRINT "coordinates. If the calculated mean impact
point does not"
LOCATE 24, 20: PRINT "Press the spacebar to continue"
pause
CLS


LOCATE 3, 5: PRINT "correspond to the aim point, then an
unfortunate systematic error"
LOCATE 5, 5: PRINT "is present.  Fortunately, this error, once
known, can be eliminated"
LOCATE 7, 5: PRINT "with appropriate offset of the weapon. "

LOCATE 10, 5: PRINT "The final weapon characteristic is the CEP.
The CEP denotes a"
LOCATE 12, 5: PRINT "radius.  The meaning of this radius is that
50% of all the impact"
LOCATE 14, 5: PRINT "points will fall within it.  The CEP, once
determined, is then used"
LOCATE 16, 5: PRINT "to plot the probability of hit cumulative
distribution curve. Once"
LOCATE 18, 5: PRINT "the curve has been plotted, you can select
various target"
LOCATE 20, 5: PRINT "radii and observe their impact on the hit
probabilities."
LOCATE 24, 20: PRINT "Press the spacebar to continue"
pause
CLS


LOCATE 3, 5: PRINT "To contrast the impact of CEP on the
probability of hit, you may enter"
LOCATE 5, 5: PRINT "additional CEP values to generate new plots.
Again you can select"
LOCATE 7, 5: PRINT "target radii and compare the outcome between
plots."

LOCATE 24, 20: PRINT "Press the spacebar to continue"
```

```
pause

endcepx:
CLS

CASE 115

CLS
COLOR 3
LOCATE 3, 17: PRINT "The Concept and Application of Track Angle"
COLOR 7
LOCATE 5, 5: PRINT "A common term found in high resolution
aircraft combat models is"
LOCATE 7, 5: PRINT "the term track angle. This term is commonly
defined as the angle"
LOCATE 9, 5: PRINT "between the aircraft's velocity vector and
the line of site (LOS)"
LOCATE 11, 5: PRINT "vector. The velocity vector generally
extends in the same"
LOCATE 13, 5: PRINT "direction as the heading. The LOS vector
originates from the nose"
LOCATE 15, 5: PRINT "of the aircraft and terminates at the
target. For example, an"
LOCATE 17, 5: PRINT "aircraft that has a target directly off its
beam has a track angle"
LOCATE 19, 5: PRINT "of 90 degrees. If the target is directly
behind the aircraft,"
LOCATE 21, 5: PRINT "then the track angle is 180 degrees."

LOCATE 24, 20: PRINT "Press the spacebar to continue"
pause
CLS

LOCATE 3, 5: PRINT "ATACS combat demonstrator makes use of the
track angle concept in"
LOCATE 5, 5: PRINT "two sperate applications. In one case, the
track angle is used in"
LOCATE 7, 5: PRINT "limiting an aircraft's turn towards a target.
If the magnitude of"
LOCATE 9, 5: PRINT "the track angle is greater than the
aircraft's turn rate per"
LOCATE 11, 5: PRINT "second, the turn is then limited to the turn
rate. In the second"
LOCATE 13, 5: PRINT "application track angle is used to determine
if a target is within"
LOCATE 15, 5: PRINT "the sweep angle of the aircraft's radar. "

LOCATE 18, 5: PRINT "Additional applications of the track angle
include prioritization"
LOCATE 20, 5: PRINT "of threats. If the track angle between, say
A/C 1 and A/C 2, is"
LOCATE 22, 5: PRINT "high (120-180 degrees) and the track angle
between A/C 2 and A/C 1"

LOCATE 24, 20: PRINT "Press the spacebar to continue"
pause
CLS

LOCATE 3, 5: PRINT "is low (0-30 degrees), then A/C 2 is behind
A/C 1 and pointing"
LOCATE 5, 5: PRINT "toward the tail of A/C 1. Given this
condition, A/C 2 becomes a"
LOCATE 7, 5: PRINT "high priority threat for A/C 1. The action
taken by A/C 1 is then"
LOCATE 9, 5: PRINT "dependent on the embedded target selection
rules."

LOCATE 24, 20: PRINT "Press the spacebar to continue"
pause

endtkx:
CLS

CASE 200 'Introduction to the Model

CLS
ans$ = " "
```

```
LOCATE 10, 22: INPUT "Present Scenario Synopsis (Y/N)"; ans$
IF UCASE$(ans$) = "N" GOTO endscen

CLS
COLOR 3
LOCATE 3, 25: PRINT "A Brief Scenario Synopsis"
COLOR 7


LOCATE 6, 5: PRINT "Three principal types of entities exist in
this model.  They are"
LOCATE 8, 5: PRINT "the Red forces, Blue forces, and Blue's
airbase. The objective of"
LOCATE 10, 5: PRINT "the Red force is to attack Blue's airbase.
Blue's objective is"
LOCATE 12, 5: PRINT "of course to defend the airbase.  The
airbase provides long"
LOCATE 14, 5: PRINT "range, early warning detection with a
tactical surveillance radar."

LOCATE 17, 5: PRINT "Red forces begin at the location specified
by the scenario with a"
LOCATE 19, 5: PRINT "heading directed towards the airbase.  Red
is non-reactive and will"
LOCATE 21, 5: PRINT "continue on this initial heading throughout
the simulation."
LOCATE 24, 20: PRINT "Press the spacebar to continue"
pause
CLS


LOCATE 3, 5: PRINT "Blue aircraft are initially on airborne alert
flying a combat air"
LOCATE 5, 5: PRINT "patrol (CAP) mission.  Blue will continue to
fly the CAP until a"
LOCATE 7, 5: PRINT "threat has been detected.  The location of
the CAP is provided by"
LOCATE 9, 5: PRINT "the scenario file."

LOCATE 12, 5: PRINT "Until the Red intruders are detected their
existence is unknown to"
LOCATE 14, 5: PRINT "Blue.  Once detected, either by the
surveillance radar or aircraft"
LOCATE 16, 5: PRINT "radar, Blue forces are vectored to Red's
location by ground"
LOCATE 18, 5: PRINT "controllers until airborne radar contact is
made.  Radar contact of"
LOCATE 20, 5: PRINT "an intruder by a Blue interceptor in
signified by a color change of"
LOCATE 22, 5: PRINT "the interceptors radar frontier."

LOCATE 24, 20. PRINT "Press the spacebar to continue"
pause
CLS

LOCATE 4, 5: PRINT "An embedded rule of engagement requires Blue
to make visual"
LOCATE 6, 5: PRINT "identification of the Red intruder before
defensive action can be"
LOCATE 8, 5: PRINT "taken.  Visual identification is also
signified by a change in"
LOCATE 10, 5: PRINT "color of the radar frontier.  Once the
intruder is visually "
LOCATE 12, 5: PRINT "identified as a threat, Blue will then
attempt to destroy it."
LOCATE 24, 20: PRINT "Press the spacebar to continue"
pause
endscen:

COLOR 7

CASE 210
CLS
COLOR 3
LOCATE 2, 17: PRINT "Features of the Air Combat Simulator   "
COLOR 7
```

```
LOCATE 4, 2: PRINT "These keys are active only during the
simulation "

LOCATE 6, 5: PRINT "F1 Halt:        Temporarily suspends the
simulation."

LOCATE 8, 5: PRINT "F2 Resume:      Causes simulation to resume."

LOCATE 10, 5: PRINT "F3 Atrib Scrn:  Displays the status of the
simulation and"
LOCATE 11, 5: PRINT "                entities.  Permits you to
look inside"
LOCATE 12, 5: PRINT "                the simulation and see the
changing states."
LOCATE 13, 5: PRINT "                May be used to accelerate
simulation.          "

LOCATE 15, 5: PRINT "F4 Graphics:    Returns you to the animation
screen."

LOCATE 17, 5: PRINT "F6 Zoom In:     Moves the point of view
further in."

LOCATE 19, 5: PRINT "F5 Zoom Out:    Moves the point of view
farther out."

LOCATE 21, 5: PRINT "F7 Quit:        Ends the simulation."


LOCATE 24, 20: PRINT "Press the spacebar to continue"
pause
CLS

CASE 5000
CLS
LOCATE 12, 15: INPUT "Display ATACS Review Questions Y/N (Y)";
QAns$
IF UCASE$(QAns$) = "N" GOTO endquest
CLS
COLOR 3: LOCATE 3, 30: PRINT "POINTS TO PONDER"

COLOR 15: LOCATE 6, 2: PRINT "MODEL CLASSIFICATION"

COLOR 7
LOCATE 8, 2: PRINT "Was the purpose of the combat demonstration
for education and training"
LOCATE 9, 2: PRINT "or analysis ?"
COLOR 15: LOCATE 11, 2: PRINT "CLASSIFICATION BY QUALITIES"
COLOR 7
LOCATE 13, 2: PRINT "What was the model's domain; was land in the
domain?"
LOCATE 15, 2: PRINT "What was the model's span?"
LOCATE 17, 2: PRINT "What was the model's environment?"
LOCATE 19, 2: PRINT "Was darkness modeled, and if not, would it
make a difference in the outcome?"
LOCATE 21, 2: PRINT "Was weather modeled, and if not, would it
make a difference in the outcome?"

LOCATE 24, 15: PRINT "Hit space bar to continue"
pause
CLS

LOCATE 3, 2: PRINT "What was the model's force composition?"
LOCATE 5, 2: PRINT "What was the model's scope of conflict?"
LOCATE 7, 2: PRINT "What was the model's mission area?"
LOCATE 9, 2: PRINT "What was the model's level of detail of
processes and entities?"
COLOR 15: LOCATE 11, 2: PRINT "CLASSIFICATION BY CONSTRUCTION"
COLOR 7
LOCATE 13, 2: PRINT "Was your involvement required during the
combat demonstration?"
LOCATE 15, 2: PRINT "What time advanced mechanism (fixed step or
event step) was"
LOCATE 16, 2: PRINT "used in the combat demonstration?"
LOCATE 18, 2: PRINT "If fixed step, what was the size of the
step, and was there more than one size?"
LOCATE 20, 2: PRINT "If event step, which events advanced the
```

```
simulation time?"
LOCATE 22, 2: PRINT "What time advance mechanism is best for this
type of simulation and why?"

LOCATE 24, 15: PRINT "Hit space bar to continue"
pause
CLS

LOCATE 3, 2: PRINT "Is the combat demonstration a deterministic
or stochastic model?"
LOCATE 5, 2: PRINT "Is the combat demonstration a Monte-Carlo
simulation?"
LOCATE 7, 2: PRINT "How many sides were represented in the
demonstration?"
LOCATE 9, 2: PRINT "Was treatment of the sides symmetric or
asymmetric?"
LOCATE 11, 2: PRINT "If asymmetric, was only one or both sides
reactive?"
COLOR 15: LOCATE 13, 2: PRINT "ENTITIES AND ATTRIBUTES"
COLOR 7
LOCATE 15, 2: PRINT "What entities were simulated by the combat
demonstration?"
LOCATE 17, 2: PRINT "What attributes did these entities posses?"
LOCATE 19, 2: PRINT "Was the existence of these entities
explicitly defined in the simulation"
LOCATE 20, 2: PRINT "or did they exist as a collection of
attributes?"

LOCATE 24, 15: PRINT "Hit space bar to continue"
pause
CLS

LOCATE 3, 2: PRINT "How were these entities destroyed?"
LOCATE 5, 2: PRINT "What happens to entities when they are
destroyed?"
COLOR 15: LOCATE 7, 2: PRINT "AIRCRAFT"
COLOR 7
LOCATE 9, 2: PRINT "Was altitude a factor in the simulation of
the aircraft performance?"
LOCATE 11, 2: PRINT "Was the aircraft velocity a factor in the
simulation?"
LOCATE 13, 2: PRINT "What aircraft performance measures were used
in the simulation of"
LOCATE 14, 2: PRINT "the aircraft's flight and maneuverability?"
LOCATE 16, 2: PRINT "What sensors did the aircraft posses?"
LOCATE 18, 2: PRINT "Were these sensors active?"
LOCATE 20, 2: PRINT "What aircraft resources were represented?"

LOCATE 24, 15: PRINT "Hit space bar to continue"
pause
CLS

LOCATE 3, 2: PRINT "Were these resources consumed?"
LOCATE 5, 2: PRINT "What was the outcome if these resources were
expended?"
COLOR 15: LOCATE 7, 2: PRINT "MISSILES"
COLOR 7
LOCATE 9, 2: PRINT "What missile performance measures were used
in the simulation?"
LOCATE 11, 2: PRINT "Did missiles share the same time mechanism
as the aircraft?"
LOCATE 13, 2: PRINT "If so, did the missile share the same time
step?"
LOCATE 15, 2: PRINT "Did missiles actively track their targets?"
LOCATE 17, 2: PRINT "How was the missile effectiveness (killing
ability) modeled?"
LOCATE 19, 2: PRINT "Could a missile miss a target?"

LOCATE 24, 15: PRINT "Hit space bar to continue"
pause
CLS

COLOR 15: LOCATE 3, 2: PRINT "SEARCH"
COLOR 7
LOCATE 5, 2: PRINT "What methods were used to simulate the
process of search?"
LOCATE 7, 2: PRINT "Was the process of search explicit or
```

```
implied?"
LOCATE 9, 2: PRINT "Which simulated sensors, either explicitly or
implicitly"
LOCATE 10, 2: PRINT "simulated, were actively searching for
intruders?"
LOCATE 12, 2: PRINT "Were sensors modeled as entities or
attributes of a higher entity?"
COLOR 15: LOCATE 14, 2: PRINT "DETECTION"
COLOR 7
LOCATE 16, 2: PRINT "What method was used to simulate the process
of detection?"
LOCATE 18, 2: PRINT "Was the method deterministic or stochastic?"
LOCATE 20, 2: PRINT "Could the method be classified as
Monte-Carlo?"

LOCATE 24, 15: PRINT "Hit space bar to continue"
pause
CLS

COLOR 15: LOCATE 3, 2: PRINT "TARGET ASSIGMENT"
COLOR 7
LOCATE 5, 2: PRINT "Was the target assignment random or was there
an underlying rule?"
LOCATE 7, 2: PRINT "Would the simulation permit the same target
to be assigned to more"
LOCATE 8, 2: PRINT "than one shooter?"
LOCATE 10, 2: PRINT "How many simultaneous targets could be
assigned to one shooter?"
COLOR 15: LOCATE 12, 2: PRINT "COMMUNICATION, COMMAND, AND
CONTROL"
COLOR 7
LOCATE 14, 2: PRINT "Was C3 present?"
LOCATE 16, 2: PRINT "If so, who was in command; what did they
command; and was C3"
LOCATE 17, 2: PRINT "actively used to control entity activities?"
LOCATE 19, 2: PRINT "Could shooters coordinate or concentrated
fire (more than one"
LOCATE 20, 2: PRINT "shooter shooting at a target)?"

LOCATE 24, 15: PRINT "Hit space bar to continue"
pause
CLS

COLOR 15: LOCATE 3, 2: PRINT "ADVANCE"
COLOR 7
LOCATE 5, 2: PRINT "Did the entity advance mechanism for aircraft
and missiles use a fixed"
LOCATE 6, 2: PRINT "rate throughout the simulation or was the
rate altered by changes"
LOCATE 7, 2: PRINT "in the entity's status?"
COLOR 15: LOCATE 9, 2: PRINT "TARGET DESTRUCTION"
COLOR 7
LOCATE 11, 2: PRINT "To destroy a target, was the missile
required to impact the target"
LOCATE 12, 2: PRINT "(Hint: think of the time advance and entity
advance mechanism)?"

LOCATE 17, 10: PRINT "That's it.  Hit space bar to exit."
pause

CLS



endquest:
CLS



CASE 10000
PRINT #10, ""
PRINT #10, USING "Simulation was terminated at ##### seconds.";
TNOW%

END SELECT
```

```
' ac launch msl # at tnow. Rng to tgt was ####
' msl # hit/missed ac at tnow, tgt destroyed
' ac went bingo fuel at tnow


END SUB
```

Bibliography

1. Aeronautical Systems Division, Air Force Systems Command. *Piloted Air Commat Analysis Model User's Manual.* Technical Report No./ 83-5018, Vol 1. Wright-Patterson AFB OH, November 1983.
2. Aeronautical Systems Division, Air Force Systems Command. *Piloted Air Commat Analysis Model Analyst Manual.* Technical Report No./ 83-5018, Vol 3, Part 1. Wright-Patterson AFB OH, May 1984.
3. Aeronautical Systems Division, Air Force Systems Command. *Piloted Air Commat Analysis Model Analyst Manual.* Technical Report No./ 83-5018, Vol 3, Part 2A. Wright-Patterson AFB OH, November 1983.
4. Aeronautical Systems Division, Air Force Systems Command. *Piloted Air Commat Analysis Model Analyst Manual.* Technical Report No./ 83-5018, Vol 3, Part 2B. Wright-Patterson AFB OH, November 1983.
5. Aeronautical Systems Division, Air Force Systems Command. *Piloted Air Commat Analysis Model Analyst Manual.* Technical Report No./ 83-5018, Vol 3, Part 2C. Wright-Patterson AFB OH, November 1983.
6. Brewer, G.D. and Hall, O. *Policy Analysis by Computer Simulation: The Need for Appraisal,* P-4893, The Rand Corporation, Santa Monica, California, August 1972.
7. Cushman, John et al. "On Representing Warfare," Article prepared for the Joint Analysis Directorate, OJCS, 28 February 86, rev 9 April 1986.
8. DeArmon, James. "Improving Random Number Generators on Micro Computers," *Computers and Operations Research,* 17:283-290 (March 1990).
9. Department of the Air Force. *US Air Force Basic Doctrine.* AFM 1-1. Air University, Maxwell Air Force Base, Alabama, March 1984
10. Department of Defense.*Catalog of Wargaming and Military Simulation Models* Joint Analysis Directorate, Organization of the Joint Chiefs of Staff J8, Tecnical Support Division, 11th ed., 1989.
11. Dimas, Chris. "Strategy For Developing Computer Aided Instructions," *Education and Technology,* 28:26-28 (April 1987).
12. Duisberg, Robert A. "Animation Using Temporal Constraints: An Overview of the Animas System," *Human- Computer Interactions,* 3:275-307, 1987-88.
13. Dunningan, James F. *The Complete Wargames Handbook.* Willam Morrow, NY, 1980.
14. Dupuy, Trevor, Curt Johnsom, and Grace P. Hays. *Dictionary of Military Terms.* New York: The H.W. Wilson Company, 1986
15. Edmunds, Robert A. *The Prentice-Hall Encyclopedia of Information Technology.* Englewood Cliffs NJ: Prentice-Hall, 1987.
16. Etter D. M. *Structured FORTRAN for Engineers and Scientists* (Second Edition). MA: Benjamin/Cummings Publishing Company Inc., 1987.
17. *Foxbat and Phantom Tactical Aerial Combat in the 1970's.* NY: Simulations Publications Inc. 1973.

18. Garrambone, Major Michael W. Personal conversations, class lectures, and handouts. OPER 775, Land Combat Modeling I. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH. July-September 1991.

19. Hartman, James K. *Lecture Notes in High Resolution Combat Modeling.* James K. Hartman, 1985.

20. Hathaway, Michael D. "Variables of Computer Screen Display and How They Affect Learning,"*Education Technology*, 14:7-11, (January 1984).

21. Hoeke, Carol M. "CAI: A Guideline for Effective Use, " *Interface*, 10:103-104 (Winter 1988/89).

22. Hughes, Wayne P. Jr. *Military Modeling.* Washington DC: Government Printing Office, 1984.

23. Knuth, D. *The Art of Computer Programming*, Maryland: Addison-Westley, 1973.

24. Law, Averill M. and Kelton, David W.*Simulation Modeling and Analysis.* New York: McGraw-Hill Book Company, 1982.

25. Military Operations Research. *SIMTAX: A Taxonomy for Warfare Simulation*, 1987.

26. McLeod, John. "But Mr. President—Is It Ethical?," *Procedings of the 1987 Winter Simulation Conference*, 86:69-71, (1986).

27. Mercier, Robert, Piloted Air Combat Analysis Model Developer. Personal interview. Wright-Patterson AFB, 10 October 1991.

28. Perla, Peter P. *Wargame Design, Development, and Play.* Research Memorandum, Center for Naval Analysis, Washington DC: Government Printing Office, February 1986.

29. Perla, Peter P.*A Guide to Navy Wargaming.* Research Memorandum, Center for Naval Analysis, Washington DC: Government Printing Office, May 1986.

30. Perla, Peter P. *The Art of Wargaming: A Guide for Professionals and Hobbyists.* Annapolis, Maryland: United States Naval Institute, 1990.

31. Pesapane, Capt John B. and Irvine, Maj Robert B Jr. *Derivation of CEP Formula to Approximate RAND-234 Tables.* Research Memorandum, Ballistic Missile Evaluation, Headquarters Strategic Air Command, Washington DC: Government Printing Office, February 1977.

32. Pritsker, A. A. *Introduction to Simulation and SLAM II* (Second Edition). NY: A Halsted Press Book, 1984.

33. Przemieniecki, J. S. Introduction to Mathematical Methods of Defense Analysis (Draft). Air Force Institute of Technology, Wright-Patterson AFB, 1989.

34. Roskem, Jan *Flight Dynamics of Ridgid and Elastic Airplanes.* KN: Roskem Aviation and Engineering Corporation, 1976.

35. Sargent, Robert G. "An Overview of Verification and Validation of Simulation Models",*Procedings of the 1987 Winter Simulation Conference*, 87:33-39, (1987).

36. Savanaes, Dag. "Simulation Models + User Interfaces = Interactive Application,"*Computers and Education*, 14:363-370 (April 1990).

37. Schruben, Lee. "Using Simulation To Solve Problems: A Tutorial on the Analysis of Simulation Output," *Procedings of the 1987 Winter Simulation Conference*, 87:40-41, (1987).

38. Seidel, Robert J. et al. "Tip for Managing Computer Aided Instruction,"*Education and Technology*, 18:33-36 (April 1987).

39. Shammas, Namir S. "The BASIC Revival," *Byte*, 13:295-300 (September 1988).

40. Shannon, Robert E. "Models and Artifical Intellengence", *Procedings of the 1987 Winter Simulation Conference*, 87:19-20, (1987).

41. Shaw, Robert L. *Fighter Combat Tactics and Maneuvering.* Maryland: United States Navel Institute Press, 1985.

42. Siebert, Karl, Piloted Air Combat Analysis Model Manager. Personal interview. Booz-Allen INC., Fairborn OH, 26 July 1991.

43. CACI Inc. *SIMSCRIPT II.5 Programing Language.* CA: CACI Inc, 1983.

44. United States General Accounting Office. *Models, Data, And War: A Critique Of The Foundation For Defense Analysis.* Washington DC: Government Printing Press, 1980.

45. Hergerrt, Douglas *Microsoft Quick Basic 4.5.* Washington: Microsoft Press, 1988.

Vita

Captain Richard S. Moore was born on 25 November 1955 in Lakeland, Florida. Following his graduation from Riverview High School in Sarasota, Florida he entered the Air Force. During his 12 years of enlisted service he performed maintenance on RF-4C Phantom reconnaissance sensor systems and managed databases in support of developmental and operational testing. During this period, he completed his Bachelor of Science degree. Upon commissioning, Captain Moore served as a logistics analyst for the Global Positioning System (GPS). His assignments include 67th Avionics Maintenance Squadron (AMS) at Bergstrom AFB, Texas, 26th AMS at Zweibrucken AB, FRG, 4243 AMS and Armament Division Engineering Directorate at Eglin AFB, Florida, Headquarters Air Force Operational Test and Evaluation Center at Albuquerque, New Mexico, and the Air Force Institute of Technology where he earned the Degree of Master of Science in operations research. In March of 1992, he was assigned to Headquarters Air Force Logistics Command Wright–Patterson AFB at Dayton, Ohio.

Permanent address: 2252 Mill Terrace
Sarasota, Florida 33581