ADAM
Acquisition Deployment
And Maneuvering
The Space Game

THESIS

Jeffrey E. Beier
Captain, USAF

AFIT/GSO/ENS/87D-7

Best Available Copy
ADAM

Acquisition Deployment
And Maneuvering
The Space Game

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
Master of Science in Space Operations

Jeffrey E. Heier, B.S.
Captain, USAF

December 1987
Preface

The purpose of this study was to provide a model which would enhance middle to senior level officers' understanding of how space forces are acquired, deployed, and maneuvered in response to the ground situation.

While there are some simplifications used in this study, the model provides a useful representation of the acquisition and deployment of space assets. This study should provide the baseline for several follow-on efforts. These efforts will add more realism into the model.

In performing this study, I have had a great deal of help from others. I am indebted to my faculty advisor, Lt Col Jim Robinson, for his patience and assistance in times of need. A word of thanks is also owed to Lt David Hunninghake at Space Command for providing the database for the world outline. I would also like to thank Major Bruce Thieman for providing the baseline for this effort.

Jeffrey E. Heier
**Table of Contents**

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td>ii</td>
</tr>
<tr>
<td>List of Figures</td>
<td>v</td>
</tr>
<tr>
<td>List of Tables</td>
<td>vi</td>
</tr>
<tr>
<td>Abstract</td>
<td>vii</td>
</tr>
<tr>
<td>I. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>Background</td>
<td>1</td>
</tr>
<tr>
<td>Purpose</td>
<td>3</td>
</tr>
<tr>
<td>Scope</td>
<td>4</td>
</tr>
<tr>
<td>Assumptions</td>
<td>4</td>
</tr>
<tr>
<td>II. Review of the Literature</td>
<td>5</td>
</tr>
<tr>
<td>Introduction</td>
<td>5</td>
</tr>
<tr>
<td>Definition of a Wargame</td>
<td>5</td>
</tr>
<tr>
<td>Description of Wargames</td>
<td>5</td>
</tr>
<tr>
<td>Design of Wargames</td>
<td>7</td>
</tr>
<tr>
<td>Wargames Currently in Use at ACSC and AWC</td>
<td>11</td>
</tr>
<tr>
<td>Current Efforts</td>
<td>12</td>
</tr>
<tr>
<td>Contractor Models</td>
<td>13</td>
</tr>
<tr>
<td>Software Used in this Effort</td>
<td>15</td>
</tr>
<tr>
<td>Summary</td>
<td>16</td>
</tr>
<tr>
<td>III. Concept Development</td>
<td>17</td>
</tr>
<tr>
<td>Introduction</td>
<td>17</td>
</tr>
<tr>
<td>Similarities Between Air and Space Forces</td>
<td>17</td>
</tr>
<tr>
<td>Learning Objectives at ACSC and AWC</td>
<td>18</td>
</tr>
<tr>
<td>Research Objectives</td>
<td>21</td>
</tr>
<tr>
<td>SATRAK Interface</td>
<td>22</td>
</tr>
<tr>
<td>Players</td>
<td>22</td>
</tr>
<tr>
<td>Summary</td>
<td>23</td>
</tr>
<tr>
<td>IV. Game Design</td>
<td>25</td>
</tr>
<tr>
<td>Introduction</td>
<td>25</td>
</tr>
<tr>
<td>Game Design</td>
<td>25</td>
</tr>
<tr>
<td>Game Play</td>
<td>27</td>
</tr>
<tr>
<td>Summary</td>
<td>44</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>V. Results, Conclusions and Recommendations</td>
<td>45</td>
</tr>
<tr>
<td>Results</td>
<td>45</td>
</tr>
<tr>
<td>Conclusions</td>
<td>45</td>
</tr>
<tr>
<td>Potential Uses for ADAM</td>
<td>46</td>
</tr>
<tr>
<td>Recommendations for Further Study</td>
<td>47</td>
</tr>
<tr>
<td>Summary</td>
<td>53</td>
</tr>
<tr>
<td>Appendix A: Resource Catalog</td>
<td>54</td>
</tr>
<tr>
<td>Appendix B: Requirements Catalog</td>
<td>66</td>
</tr>
<tr>
<td>Appendix C: Scenario</td>
<td>86</td>
</tr>
<tr>
<td>Appendix D: Users Guide</td>
<td>90</td>
</tr>
<tr>
<td>Appendix E: Source Code</td>
<td>124</td>
</tr>
<tr>
<td>Bibliography</td>
<td>220</td>
</tr>
<tr>
<td>Vita</td>
<td>222</td>
</tr>
</tbody>
</table>
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Acquisition Phase Main Menu</td>
<td>101</td>
</tr>
<tr>
<td>2. Satellite/Reusable Launcher ACB Screen</td>
<td>102</td>
</tr>
<tr>
<td>3. Survivability Features Screen</td>
<td>105</td>
</tr>
<tr>
<td>4. Constellation Age Screen</td>
<td>106</td>
</tr>
<tr>
<td>5. Display Orbits/Maneuver Sats Screen</td>
<td>107</td>
</tr>
<tr>
<td>6. Ground Swath Screen</td>
<td>109</td>
</tr>
<tr>
<td>7. Assign Ground Support Screen</td>
<td>110</td>
</tr>
<tr>
<td>8. Deployment Phase Main Menu</td>
<td>111</td>
</tr>
<tr>
<td>9. Satellites Ready for Launch Screen</td>
<td>113</td>
</tr>
<tr>
<td>10. Choose Launch Site Screen</td>
<td>113</td>
</tr>
<tr>
<td>11. Choose Launcher Screen</td>
<td>114</td>
</tr>
<tr>
<td>12. Choose Launch Pad Screen</td>
<td>114</td>
</tr>
<tr>
<td>13. Multiple Payload Launch Screen</td>
<td>118</td>
</tr>
<tr>
<td>14. NASA Check Screen</td>
<td>118</td>
</tr>
</tbody>
</table>
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I.</td>
<td>List of ACSC Space Curriculum Objectives Which Can be Supported by Wargames or Computer-Aided Instruction</td>
<td>19</td>
</tr>
<tr>
<td>II.</td>
<td>List of ANC Space Curriculum Objectives Which Can be Supported by Wargames or Computer-Aided Instruction</td>
<td>20</td>
</tr>
<tr>
<td>III.</td>
<td>Player Positions</td>
<td>23</td>
</tr>
<tr>
<td>IV.</td>
<td>Survivability Features</td>
<td>34</td>
</tr>
<tr>
<td>V.</td>
<td>Mission FOVs</td>
<td>36</td>
</tr>
<tr>
<td>VI.</td>
<td>Unit Status Definitions</td>
<td>94</td>
</tr>
<tr>
<td>VII.</td>
<td>Orbit Descriptions</td>
<td>95</td>
</tr>
<tr>
<td>VIII.</td>
<td>Player Positions</td>
<td>96</td>
</tr>
<tr>
<td>IX.</td>
<td>Launch Pad Fire Rates</td>
<td>115</td>
</tr>
<tr>
<td>X.</td>
<td>System Commands</td>
<td>121</td>
</tr>
</tbody>
</table>
Abstract

The purpose of this effort was to provide middle to senior level managers a tool which would enhance their knowledge of how space assets are acquired, deployed, and maneuvered in response to the ground situation.

This effort had two basic objectives: (1) Provide a user-friendly model which effectively simulates the acquisition and deployment process of space assets. (2) Develop a model which could be used as a baseline for follow-on efforts.

The model was developed on a micro-computer because it was perceived this media would provide a more user-friendly and expandable environment.
Space has been a vital part of our national defense for over two decades. Currently, the US relies on space-based assets to perform many critical functions including surveillance, intelligence collection, communication and navigation. The Soviets also view space as an integral part of their national defense. While the Soviets do not rely on their space-based assets as much as the US, the number, complexity, and value of their satellites are increasing rapidly. Accordingly, the Soviets are relying more on these assets.

As these space assets become more critical to the US and USSR, the possibility of a space conflict becomes more likely. There may come a time when it becomes important to deny the Soviet's use of space for military purposes, and at the same time, preserve US space assets. A space conflict would probably precede, or possibly accompany, the outbreak of hostilities on earth. The space conflict could potentially begin with a preemptive attack against enemy satellites using one or more of the proposed anti-satellite weapons. It is also probable that such a conflict would involve a wide range of defensive counter-measures to protect the satellites under attack (5:2).
Should there be a general superpower conflict at any point from the mid-1980s to the end of the century, there will be warlike action in, and relating to, space. In the event of general war, conflict in space will occur. (9:5)

There are many models which currently simulate certain aspects of a general space conflict. These models typically simulate only one element of a highly complex system. For example, Boeing Aerospace has developed the Intercept Opportunity Deterministic Model (IODM) which simulates an air-launched ASAT directed at a single Soviet satellite (11:--). While these models are useful to develop detailed performance requirements for particular elements, as single-element simulators, they are inadequate to model the overall space environment.

Wargames are often limited to the direct effects of engaged forces, without consideration of the interrelated effects of other deployed forces or supporting elements that might be applied to or bear indirectly upon the conflict. (10:2)

Even though space has been critical for a long time, there is a definite lack of understanding of how space assets interact with the terrestrial forces. Commanders make daily use of information gathered from space, such as weather reports or intelligence briefings, without considering how the information was collected. Unless commanders "have an appreciation of space assets, their capabilities cannot be fully exploited and the impact of their loss or degradation understood" (2:7).

Because both the US and USSR rely heavily on their space forces, a future conflict in space is a very real possibility. Therefore, it is vital that we develop a space-war doctrine, just as we have developed doctrines for land, sea, and air combat.
A key step in developing a space-war doctrine is developing an understanding of the dynamics of space-based conflicts. To enhance this understanding, a model is needed which represents how the space forces are acquired, launched, and maneuvered in regards to demand of the ground situation.

In response to this requirement, a manual board game called the **Space Resource Allocation Exercise** (SRAE) was developed for use at the Air Command and Staff College (ACSC) and the Air War College (AWC) (16:—). This game was "developed as a tool for teaching basic space terminology, limitations on the use of space, and how space systems affect combat forces" (20:iii).

To enhance this learning experience, it was decided to computerize and improve the SRAE. This would enable satellite orbits and ground traces to be projected graphically, to enable the player to develop an understanding of these "fuzzy" concepts. Computerizing SRAE would also free the players to concentrate on the concepts of the game, instead of the normal bookkeeping associated with manual games.

This study is an initial step towards developing a wargame which incorporates the effect of the space conflict on the ground combat situation. This effort would then be used as a front end for the next logical step in the process of defining a space-war doctrine. This future game is part of the charter of the Air War College Space Command Chair (20:iii).

**Purpose**

The purpose of this thesis is to provide a model which would enhance middle to senior level officers' understanding of how space
forces are acquired, deployed, and maneuvered in response to the ground situation.

Scope

The scope of this research is to:

1. Provide the Air War College (AWC) and the Air Command and Staff College (ACSC) an effective tool to show the importance of our space assets with respect to the ground environment.

2. Develop a model which is useful for the space "novice" as well as experienced space personnel.

3. Develop a model which is user-friendly and expandable.

4. Show the effect orbit altitude and inclination have on satellite missions.

5. Show the importance of careful long-range planning of our space force structure.

6. Develop a game which can model different employment strategies, such as the acquisition and deployment of SDI-type constellations.

Assumptions

The following assumptions were made in this study:

1. The game was designed for the Z-158 micro-computer because of the availability of this system at ACSC.

2. Actual data were used for all satellites except the surveillance and reconnaissance systems.

3. Fictional data were used for the surveillance and reconnaissance systems to keep this effort unclassified.
II. Review of the Literature

Introduction

This chapter contains a review of a definition of a wargame, a description of wargames, and the design of wargames. Also covered in this chapter is a brief description of current efforts aimed at integrating wargames into the Space Curriculum at the AMC and ACSC. Additionally, several models currently being used at these schools will be described. Also covered in this chapter are several low-level space models which have been developed by various contractors. Finally, a brief description of the software used in this thesis effort is included.

Definition of a Wargame

A wargame is

...any type of warfare model or simulation, not involving actual military forces, in which the flow of events is affected by decisions made during the course of the game by players representing the opposing sides. (13:2)

A simulation is the approximation of a real system by the use of mathematical models.

Description of Wargames

Wargames are used when it is impractical or impossible to use the actual elements modeled in the game. Wargames are also used when it would be too costly, either in time or dollars, to use the real elements. Finally, wargames are used because it is easier to model
changes in policy, strategy, force levels, or tactics in a game than in an actual situation (6:3).

**Historical References.** Wargaming in the modern context was introduced during the Napoleonic era by George Heinrich Rudolph Johann von Resswitz, a Prussian artillery officer (6:5). Wargaming in the U.S. began in the 1890's with the use of wargames at the Naval War College (6:5).

**Strengths of Wargames.** Wargames can help explore questions of strategy, human behavior, and war-fighting trends.

If one group represents an assumed potential enemy, the War Game not only provides training, but affords some measure of the capability of the group to cope with the enemy under the conditions simulated. (21:1)

Wargames enable users to look at reality in a different light and can lead to fundamental changes in how they see that reality. The human factor is key in designing and playing a wargame.

By explicitly allowing human decisions, made under the press of time and on the basis of imperfect or incomplete information, and by incorporating randomness or "luck," wargaming comes closer than any other form of intellectual exercise to illustrating the dynamics of warfare. (13:10)

Perla goes on to say "as an explanatory device, wargames can be very effective at communicating analytical insights to appropriate decisionmakers" (14:10). Wargames can show these findings and the reasons behind them very clearly. "In addition, decisionmakers involved in such games can provide new insights that can be explored in subsequent analyses. In this sense, wargaming completes the cycle" (13:11).

**Weaknesses of Wargames.** Wargames are of little use in providing rigorous, quantitative measures to "objectively" prove or disprove
technical or tactical theories. Instead they can often provide the
kernel of new theories which can be tested with other analytical tools
(13:10).

The purpose of wargaming is not to teach students how to react to
specific situations, but to make them aware of the factors that
influence the outcome in conflict situations. By learning what is
important, they will be better prepared to develop the real
operational plans needed to secure strategic and military
objectives. (12:25)

Design of Wargames

There are two reasons for designing a wargame: education and
research. Educational games focus on providing an active learning
experience of their own, reinforcing lessons taught in a more
traditional academic setting, or evaluating the extent to which
students have learned such lessons (14:3). Research objectives may
also be divided into three classes: developing strategies,
identifying issues, or building a consensus (14:3). This paper is
concerned with educational games.

There are many different wargame design steps which have been
described as the "key" steps in designing wargames. For the purpose of
this effort, these steps have been broken up into three broad
categories: defining the game's objective and how the players will
interact, defining the audience of the game, and deciding how
information is relayed to the players. Also included is a brief
analysis on the importance of game validation.

Objective. The first step in designing a wargame is specifying
the objectives or goals of the game (6:236). The end product should be
defined before continuing on to the other steps. After identifying the
objectives, it must be decided how the game design will meet these objectives. The next step is defining the infrastructure of the game: collecting, refining, and distilling the information required to play and control the game (6:236). The rules (mechanics) for playing the game are also developed here.

Audience. Along with the rules, the audience the game is being designed for, and at what level it will be played, must be determined. The audience for a game can range from the Joint Chiefs of Staff to a platoon leader. The amount of detail the game provides will be very different depending on the level of the game. The level of the game can be classified as one of three types: strategic, tactical, or operational. An example of a strategic game would be a game in which the objective was defending the United States against a Soviet ICBM attack. A tactical game would include such scenarios as defending a divisional front against a Soviet Tank Corps. An operational game could model the effectiveness of an F-15 against a Soviet MIG-23.

Information Provided to Players. Once the audience the game is being designed for has been determined, the amount and type of information made available to the players must be addressed. The designer must ensure that the game provides a clear understanding of whom the player is emulating and what the player is supposed to do. A successful game design will provide the user with the "kind of information appropriate for carrying out their assigned functions" (14:8). This information is broken up into two categories: scenario and data base (14:10).
The scenario is a critical element of a wargame. The scenario furnishes the necessary background for the players and provides for specific updates in the situation during play. The scenario can also influence the developing situation and elicit player response to specific items of interest (14:9). "By defining the setting and scope of player decisions, scenarios can direct the course of a game into very narrow or fairly broad channels, depending on the game's goals" (14:10). However, scenarios must allow for sufficient freedom of action within which the game's objectives can be met. The scenario should not necessarily prompt the user into only one direction of action. "A scenario should include all essential information about the game's setting and subsequent planned modifications to it, and should contain no superfluous information" (14:10). For example, if a game is modeling infantry tactics in the jungle, the inclusion of submarine-based cruise missiles is probably not needed. Scenarios must also provide for specific objectives and missions which must be met by the players. A scenario must be coherent and consider all elements of the game, from its objectives, to its mechanics, to its analysis (14:10). A scenario helps the pieces to fit together.

Scenarios must be credible. Players should be able to believe the possibility of the occurrences set out in the scenario. The scenario does not have to provide the only course of action or even the most likely, just a plausible one. Scenarios must have "a plausible and consistent set of conditions carefully researched, without unexplained or radical alteration from the present environment" (3:12).
In summary, a scenario bounds the problem as specified in the objective phase of the design. The scenario dictates how the game will progress. The scenario contains mostly qualitative information (14:10).

In contrast to the scenario, the database contains quantitative information. This information includes data concerned with such things as "capabilities of forces, levels of logistics, and relative likelihoods of the occurrence and outcome of interaction between forces" (14:14).

Once the information concerning the game has been determined (the scenario and database), it must be decided how much of this data, and in what manner, is to be revealed to the players. There should always be a valid reason for a player to be given any data, and the capability for that player to have received the data must exist (14:22). If there was no logical means for the information to be gathered or transmitted, that information should not be provided. Depending on the information available to the players, games may be considered to be either open or closed.

In an open game, each side knows everything about the other, including the position, status, and strength of the opposing units. The only thing not known is the particular course of action the opponent will follow. Most good wargames attempt to style the "fog of war" into the design in which not all the information (correct or otherwise) is known by the players.

This style of game, called a closed wargame, allows for only a portion of the information to be known by the players. This
information does not have to be completely accurate, and often this data can even be misleading or incorrect. Clausewitz said, "many intelligence reports in war are contradictory; even more are false, and most are uncertain" (6:236).

Validation. Even if a game contains all the elements discussed above, it is useless if nobody accepts the game results. If there is not a logical procession of events with sound data, the game will not be considered valid.

A major problem confronting the wargames analyst is the compilation of data and assessing it in terms of validity to portray the performance characteristics expected in a real situation (21:14).

A wargame's results may be unexpected, but the game must have followed a series of logical decisions.

The compilation of sound data is a critical task for the designer. "Data is most valid when the conditions under which it is obtained most nearly portray the real world environment in which the systems are intended to operate" (21:15). However, the collection of this type of data is often very difficult or impossible.

Wargames Currently in Use at ACSC and AWC

There are several wargames currently used by ACSC and the AWC. The first game, called Big Stick, is a simulation of a strategic war (2:12). The game includes war planning and war simulation phases which allow players to procure, deploy, and target selected weapon systems (2:12).

Fast Stick is a theater air war simulation used at ACSC. Fast Stick provides opportunities for the player to plan and conduct
"missions based on their concept of operations and target priorities" (2:12). It is basically a tactical game which allows students to "apply techniques to determine weapon effectiveness and select strike flight composition to accomplish a desired damage expectancy" (2:12).

The Theater War Exercise (TWX) model is used at AWC. It is a two-sided theater-level game which uses "computerized air and land battle simulations to assess the impact of player resource allocation decisions in a NATO central region conflict scenario" (2:12). TWX provides the student with "insights into decision processes which relate principles of war, warfighting systems, and force employment decision to military objectives of war" (2:12).

The Space Resource Allocation Exercise (SRAE) was developed at ACSC. It is a board game which models the acquisition and deployment of U.S. space systems. It was developed as a tool for teaching basic space concepts (20:1). SRAE was designed as a baseline for a "full scale theater or strategic wargame with emphasis on space" (20:1). SRAE was used as the baseline in the development of ADAM.

Current Efforts

There is an on-going effort at ACSC to improve the quality of wargames currently used at ACSC and the AWC. Specifically the goal of this project is to determine how to integrate space into wargames. A Space Wargaming elective course was given at ACSC in the Fall of 1986 to develop methods to reach this goal (22:iii). Efforts developed in this course included a study of information and displays required for a space wargame, a study of the use of wargames to facilitate the instruction of the Space Curriculum Phase at ACSC, and a study of
possible Space Rules of Engagement (22:34). The Space Resource Allocation Exercise was also developed in this course.

The Air Force Wargaming Center (AFWC) is also involved in improving the quality of wargames at the War Colleges. The AFWC was the result of the Command Readiness Exercise System (CRES) program, and was developed to put "more war in the war colleges" (19:—).

The Air Force Wargaming Center has the BDM Corporation under contract to design several new computer wargames (2:14). These new games will replace the wargames currently used at the AWC and ACSC. The games are being designed so that they can be modified by the AFWC (2:14). This will ensure the games are kept current and updated as required.

**Contractor Models**

There are hundreds of wargames currently in use which model different aspects of ground, air, or naval tactics. There is a much smaller set which models space forces or space tactics. However, there is not a game currently in use which effectively incorporates the effect of space forces on the ground force. What follows is a brief description of three models which deal exclusively with space assets.

General Electric Company has developed a model called SSSA (System Survivability Simulation). It simulates threats against satellite constellations from ASATs or GBLs (Ground-Based Lasers) (11:—). The model provides a means of developing the appropriate countermeasures including detailed mission and threat analysis, system design, evaluation of system performance, and risk assessment in the projected threat environment (11:—). SSSA simulates the engagement between
multiple satellites and direct ascent and orbital ASATs. The model also simulates the prompt radiation effects caused by the detonation of a nuclear device in space and the engagement between an enemy space-based laser and a satellite (11:—).

The Space Applications International Corporation has developed the Space Defense Simulator (SPASIM) (17:1). SPASIM models how Cheyenne Mountain personnel perform some of their space related tasks. These tasks include: assessing foreign launches, detecting and processing satellite maneuvers, and detecting and processing a satellite breakup. SPASIM also includes procedures which model space launches, satellites, Red co-orbital ASATs, Blue sensors, and Blue ASATs (17:4).

The Space Forces Engagement Model (SFEM) is currently under development by Ford Aerospace and Communications Corporation. SFEM is scheduled for delivery to Air Force Space Command in April 1988 (5:2). The objective is to develop an accurate simulation of a two-sided space conflict (5:3). SFEM will model the major components of the US and Soviet space control networks, including the space-based assets and the ground-based support assets (5:3).

The model is being developed in two phases. Phase one models a limited space conflict in which the Red force launches a preemptive strike against Blue satellites (5:3). The Blue force has a defensive capability, but no ASAT (Anti-Satellite) weapons (5:3). This defensive capability is represented by DSATs (Defensive Satellites). The role of the DSAT is to protect high value mission satellites against an ASAT attack. Phase one should provide for an optimized attack strategy and a corresponding optimized defensive response (5:8).
Phase two will model a full two-sided attack and defense scenario (5:11). Either side may initiate the conflict in Phase two. However, in this phase, the defender may retaliate against the aggressor with ASAT weapons of their own (5:11). Both sides will also have the capability to reconstitute their affected constellations. Phase two should accommodate a variety of offensive and defensive objectives, strategies, and tactics (5:11).

**Software Used in this Effort**

ADAM consists of approximately 5,000 lines of Turbo Pascal code. Turbo Pascal provides a simple, user-friendly programming environment based on the original Pascal language (8:1). Turbo Pascal is a trademark of Borland International, Inc.

SAITRAK (Satellite Tracking) software was developed by Teledyne Brown Engineering (TBE) as an integrated toolbox in the area of astrodynamics. SAITRAK contains many different routines, but the ones used for this effort were the ORBIT and TRAK procedures. ORBIT produces a three-dimensional image of desired orbit types, and TRAK generates the data points for a ground track for a given satellite.

SAITRAK is designed to run on an IBM-PC compatible system operating under MS-DOS 2.0 or higher (18:3). SAITRAK was developed under contract for Air Force Space Command, directorate of Space Systems and Activities (DOS). The initial version 1.0, was delivered in September 1986. The original contract was renewed, and TBE continued working on the package. They delivered version 2.5 in June 1987.
Summary

In the past, wargames have proved to be a valuable tool in developing doctrine and strategy. The ability of wargames to explore future strategies and doctrines is especially important for space systems.

Strategies and doctrines for space defense and future space engagements are constrained by both the dearth of practical experience and the tremendous costs involved with exercising such capabilities (22:2).

By definition, doctrine is gathered from either experience or theory. We do not have experience with space systems in a conflict situation; therefore, we must rely primarily on theory provided by wargaming if we are to develop an effective space doctrine.
III. Concept Development

Introduction

This chapter contains a discussion of the similarities between the current space force and the early development of the airplane. The applicable learning objectives of the Space Curriculum Phase at ACSC and AMC are also presented. The Space Phase at ACSC and AMC was developed to enhance middle to senior level officers' knowledge of space assets and how they interact with ground forces (1:1). Also covered in this chapter are research objectives this effort incorporated, and the interface between this model and the SAIRAK software. Finally, a brief description of the players and their roles in this game is included.

Similarities Between Air and Space Forces

The current force structure in space closely resembles the airplane and its missions around 1914. In the beginning of World War I, planes were used to direct artillery fire and for troop spotting, missions which balloons had performed for about a century. Initially there was no thought of arming the airplane. The airplane was also used to deliver messages to and from the battlefield commander.

As the war progressed, so did the uses of the airplane. Soon the early pilots began carrying pistols and rifles to engage enemy pilots. Later, the weapons turned into machine guns and bombs. Then came the need for specialized planes designed for specific missions. Fighter
planes, reconnaissance planes, and bombers were developed as air power slowly matured.

As World War I ended, a new form of warfare was evolving from these glorified balloons and messengers. Strategic bombing was in its infancy by the end of the war and developed during the following years. Even though the airplane had proved itself during the war, many people did not think the airplane would become very important. It took men of vision like Mitchell and Douhet to further the cause of the airplane.

"There are many parallels between airpower in its infancy and spacepower today" (2:21). Like the airplane, two of the first missions performed by space forces were reconnaissance and communication. "Airpower was initially subservient to ground and naval forces just as spacepower is now subordinated to air and naval forces ..." (2:21).

Just as airpower began with airplanes shooting other airplanes, it appears likely that the first offensive capability developed by satellites will be the capability to kill other satellites (2:21).

These similarities between the early airplane and today's space force should not be overlooked. Careful studying of these parallels may provide critical insights into developing an effective space doctrine.

Learning Objectives at ACSC and AWC

The space phase managers for ACSC and AWC were interviewed and asked to identify the objectives which could be supported by the use of a wargame or computer-aided instruction (2:3). The ACSC and AWC objectives are prioritized and listed in Tables I and II, respectively.
Table I

List of ACSC Space Curriculum Objectives Which can be Supported by Wargames or Computer-Aided Instruction (15—)

1. Comprehend how current and planned (near-term) US military space systems can enhance the effectiveness of our military forces.

2. Comprehend how characteristics of the space medium influence the characteristics of space forces.

3. Comprehend how application of the principles of war must be tailored to the space environment.

4. Comprehend that Soviet space systems threaten US terrestrial forces and our ability to operate freely in space.

5. Comprehend specific strategies to achieve space systems survivability.

6. Comprehend the potential role and employment of the US anti-satellite.

7. Know the relationship of Soviet military space doctrine to Soviet military doctrine.

8. Comprehend the combat functions of space control and force application.

9. Comprehend how US space policy and international space law and treaties constrain military activities in space.

10. Comprehend existing and planned (near-term) space support capabilities for launch and on-orbit control of DOD spacecraft.

11. Comprehend general techniques which can be employed to make space systems survivable.

12. Know the basic tenets of US Space policy and international space law and treaties.
Table II

List of ANC Space Curriculum Objectives Which can be Supported by Wargames or Computer-Aided Instruction (16:—)

1. To comprehend the role of critical space systems in support of operational military forces and how commanders obtain operational support from these systems.

2. To comprehend the value and utility of space systems to enhance the effectiveness of the US Army, Navy, Marine Corps, and Air Force.

3. To comprehend the capabilities and utility of naval space-based systems and the operational support available from these systems to military commanders.

4. To comprehend existing and planned space support capabilities for launch and on-orbit spacecraft command and control operations.

5. To comprehend the operational concept for achieving military superiority and the role and employment of the US anti-satellite program.

6. To comprehend the Soviet Union's current and projected military capabilities and their implications for the global balance of power.

7. To know the organizational structure of the new DOD Unified Space Command and to comprehend how the Unified Space Command assigns priorities and allocates space assets to meet operational requirement of US military forces.

8. To comprehend US space policy and international space law and some of the potential moral and legal constraints on space activities.
Research Objectives

"Unless the future Air Force leaders have an appreciation of space assets, their capabilities cannot be fully exploited and the impact of their loss or degradation understood" (2:7). The controllers of space assets must understand the concepts involved in space operations.

Since Sputnik I was launched 30 years ago, the number and complexity of the satellite systems have grown; however, their primary mission has remained the same. That mission is support of the terrestrial forces. Therefore, a space war game is not complete without considering the effect the space assets have on the terrestrial forces.

The research objectives for this study are listed below.

Characteristics of Space. Determine the satellite type and orbital characteristics necessary to perform a given mission. The players should discover basic orbital mechanics to include regression of the ascending node, the effect inclination has on the ground coverage, and the cost involved in maneuvering satellites.

Interaction with Ground Forces. Recognize how various space systems can influence the ground forces. Determine the critical space systems in a given conflict situation.

Ground Support. Identify the required ground support necessary to launch and maintain a satellite. Recognize the importance of launch location (latitude) with regard to orbit inclination.

Survivability. Recognize the added capability offered by survivability enhancements. Recognize the additional cost and weight penalties incurred by incorporating survivability enhancements into
satellites. Determine the best combination of survivability enhancements required to defend against a possible enemy threat.

**Acquisition Cycle.** Recognize the long lead times associated with space systems. Recognize the importance of defining system requirements at the beginning of the acquisition cycle. Recognize the delays caused by late requirements, longer development time and more dollars.

**Force Structure.** Given a budget, a requirements list, and a list of available resources, the players should be able to design an effective force structure. Determine a proper mix of low and high technologically advanced systems. Recognize the importance of a balanced fleet of launchers.

**SATRAK Interface**

To enable players to better visualize the satellite orbits, the SATRAK software will be used to graphically display the satellite orbit traces. The seminar rooms at the AWC have multiple Z-158 computers available; therefore, one system will be used to play this game and another will run the SATRAK software. The game controller will perform the necessary commands to enable SATRAK to display the orbits.

**Players**

The game is designed for multiple players in a seminar type environment. The exact number and responsibilities can be modified as necessary; however, there is a required subset of players which should be present to play the game. This subset is a modification of the
player positions used in the SRAE game (20:14). The players and their responsibilities are listed in Table III.

Table III
Player Positions

<table>
<thead>
<tr>
<th>Player</th>
<th>Responsibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chairman, Joint Chiefs of Staff</td>
<td>Overall Commander</td>
</tr>
<tr>
<td>CINCSpace</td>
<td>Determines operational requirements and orbital requirements</td>
</tr>
<tr>
<td>Space Division</td>
<td>Responsible for system R&amp;D and production phases</td>
</tr>
<tr>
<td>NASA</td>
<td>Ensures the necessary Commercial requirements as dictated in the scenario are included in operational decisions</td>
</tr>
<tr>
<td>NCINC</td>
<td>Strategic Warfare Requirements</td>
</tr>
<tr>
<td>TCINC</td>
<td>Theater Warfare Requirements</td>
</tr>
<tr>
<td>Controller</td>
<td>Reads scenario, interprets rules, and controls the SAMRAK interface</td>
</tr>
</tbody>
</table>

Summary

"A lack of education about space and its potential has hindered the development of military space systems." (2:8) Because of the great potential for a conflict in space, the US must be prepared for it. Therefore, the US must develop a space-warfighting doctrine. Schools like ACSC and NWC are obvious places for this development. Space has long been recognized as an integral part of the national defense; however, very little time is allotted for teaching space and
its concepts at these two schools (16:--). Therefore, it is imperative
that the students get as much out of the available time as possible.
Tools must be designed that teach space concepts quickly and
effectively. ADAM is such a tool.
ADAM is a simulation of how US military space systems are acquired, deployed and maneuvered to support the land, sea, and air based forces. This chapter will cover the game design and the rationale behind certain design concepts. This chapter also discusses the input required from the players during the game. The SRAE model designed at ACSC was used as the baseline for this effort.

**Game Design**

ADAM is designed to be played with the current (1987) US Space Force or a user generated database. There are ten one-year turns in the game. The game objective is for the players to meet and maintain as many mission requirements as possible. It is not possible to attain all the requirements; therefore, the players must determine which of the requirements are most important and concentrate on meeting these demands.

Before the game begins, the players are provided with a Resource Catalog and a Requirements Catalog. The Requirements Catalog was developed at ACSC for use in the SRAE game. The Resource Catalog developed for the SRAE game was modified for this effort. These catalogs are listed in appendices A and B, respectively.

The Resource Catalog contains information on the different systems in the game. The different system types are: satellites, ground
support elements, launch pads, and launch vehicles. The following satellite missions are modeled:

1) Communication  
2) Meteorology  
3) Navigation  
4) Reconnaissance  
5) Surveillance  
6) Offensive Weapons  
7) Defensive Weapons  

The information in the Resource Catalog includes details such as system weight, lifetime, research and development cost, acquisition cost, and other system pertinent data. The systems are organized by mission type (COMM, NAV, etc.) and are arranged from low to high technology. The database is developed from the Resource Catalog. The database may be modified if specific learning objectives are to be taught. It would be beneficial to the user to develop a library of databases. Instructions on how to create a database are in the User's Guide.

The Requirements Catalog contains lists of prioritized requirements. Each list is divided into the mission area of satellite systems, ground support, and launch systems. The use of the Requirements Catalog is optional, and is designed for use by players unfamiliar with space system requirements. Civilian/NASA requirements of DOD systems are also included. The Requirements Catalog in Appendix B was developed at ACSC, and was designed for use with a database which reflected the space structure of the 1960's. If this catalog is used
with the current space structure, the players will notice some requirements have already been met. The players should ignore these requirements and concentrate on meeting the unfilled requirements. The Requirements Catalog may be replaced by a user generated catalog if specific scenarios are to be studied.

**Game Play**

ADAM contains ten one-year turns. Each turn contains the following phases:

**Turn Sequence**

1) Read Scenario Phase (done by Controller)
2) Database Update Phase (done by Computer)
3) Acquisition Phase
4) Assign Ground Support Phase
5) Deployment Phase
6) Scoring Phase (done by Controller)

The players repeat these steps through turn 10 or the Game Controller decides to stop the game.

**Read Scenario Phase.** At the beginning of each turn, the Game Controller gives the players the scenario for that turn. The scenario describes conditions the players must abide by for that turn. The scenario includes the space budget provided by Congress for the turn, a US Space Policy Update, and an Intelligence Report of probable Soviet space activities. The Policy Update may restrict deployment of particular systems or may place additional demands on the military launch system to meet certain Civilian/NASA demands. The Intelligence Report provides information on space activities by other countries such
as the possible deployment of new weapons. The Scenario developed for the SRAE game at ACSC was modified for this effort. The Scenario is listed in Appendix C.

**Database Update Phase.** In this phase, the computer advances the status of units currently being upgraded. Each unit in the database contains an update field that maintains the time remaining to complete the next phase in the acquisition cycle. If a unit is currently being updated, the update field is decreased by one in this phase. If the update requires two years, the update field will be changed from a "2" to a "1". This indicates that the update will occur the following year. If the update requires only one year, the update field will be changed from a "1" to a "0", and the unit will advance to the next stage. For example, during the Acquisition Phase of year 1988, the player decides to begin Research on the MILSTAR 1, tail number 1, Communication Satellite. The current status of MILSTAR 1 is Feasible, so Research may begin on the system. The Research cost for MILSTAR 1 ($200 Million) is deducted from the budget, and the computer sets the update field of that system to "2". This indicates the unit will complete the research phase in two years. During the Update Database Phase of year 1989, the computer changes the update field to a "1", indicating the unit has one year of research remaining. Finally, the research for MILSTAR 1, tail number 1, is completed in the Database Update Phase of year 1990. The unit is moved to the Research column, and the update field is set to "0", indicating the upgrade is completed.
The computer also updates the budget with the current year figures in the Database Update Phase. No amount of the budget may be carried over from one turn to the next, so players should spend the entire budget every turn.

Next, the computer performs a system failure check for all active satellites. The probability of failure for a satellite is a constant 0.3% per year until the satellite is two years from its expected lifetime. At this time, the probability of a satellite failure is 8%. There is a 58% probability of system failure at the end of its expected lifetime (20:40). The program automatically "kills" a satellite if it survives one year past its expected lifetime. These figures are approximations and a more realistic approach to satellite reliability is a topic for further improvement.

During this phase, the computer also determines the current Technology (Tech) Level of the Defense Industry in each mission area (Communication, Navigation, etc.). The current Tech Level greatly impacts the space force structure. It determines the feasibility of a satellite system. A feasible system indicates technology has reached a sufficient level to begin research on that system. For example, the MILSTAR 2 satellite system is not feasible until the necessary technology has been developed by the MILSTAR 1 satellite system. At least one MILSTAR 1 satellite must complete its development phase, before the MILSTAR 2 system is feasible.

With a few exceptions, the Research and Development for systems listed higher up in the Resource Catalog must be accomplished before the more advanced systems may be acquired.
The last step the computer performs in this phase is assessing the Operational and Maintenance (O&M) costs for the active ground support and launch pad facilities.

**Acquisition Phase.** In this phase the players give orders for units to begin the next Milestone of the acquisition cycle. The Satellite Acquisition Cycle Board (ACB) Screen lists all the satellites, support elements, launch pads, and reusable/recyclable launcher units currently in the acquisition pipeline. The status headings on the ACB Screen (Feasible, Research, Development, and Production) roughly correspond to the Acquisition Milestones 0, 1, 2, and 3 as defined by DOD Directive 5000.1 (4:5).

Once the first copy of a particular system has completed the Development phase, additional copies skip the Research phase and go directly from Feasible to Development. When a unit is marked with a particular status, it is considered to be at the end of that phase. For example, Comstar, tail number 1, has a current status of Research. This indicates that the Research has been completed, and the satellite is waiting to proceed into the Development phase.

The player indicates which unit to update by entering its system name and tail number. The Check Budget procedure is then called to determine if the purchase is within the remaining budget. There are two costs associated with each unit on the ACB. They are the Research and Development (R&D) costs and the Production costs. The R&D cost is split between the two R&D stages. For example, the R&D cost for a MILSTAR 1 Satellite is $400 Million. This means Research for a MILSTAR 1 Satellite costs $200 Million, and the price of Development is $200.
Million. The Production cost is the cost involved with producing the
unit. Check Budget determines the cost for the updated status and
compares it against the remaining budget. If the upgrade is within the
remaining budget, the associated upgrade cost is subtracted from the
current year's budget. An asterisk is put next to the unit's name on
the ACS Screen to indicate that the unit is currently being upgraded.
If the upgrade exceeds the budget, the system returns a message to that
effect, and the unit is not updated. The player may update as many
units as the budget allows for.

The ACS Screen will only show units currently in the acquisition
pipeline. It will not show satellites ready for launch or ones that
have already been deployed. The player may add copies of a unit which
is on the ACS Screen or one that has already been deployed. This copy
will be given the next available tail number of that system type. For
example, four DSCS II satellites have been deployed and there are no
more in the acquisition pipeline. To add another DSCS II satellite,
the players enter the name of the satellite ("DSCS II") and then a tail
number of "0". The program will then put a DSCS II, tail number 5, on
the ACS screen. The new unit is placed in the Development stage
because research has been completed for the DSCS II system.

When development begins on a Launch Pad, the program will ask the
player where to construct the Pad. The player has two choices:
Vandenberg or Cape Canaveral. Once the location is set, it cannot be
changed.
The expendable launcher units are handled differently. Each expendable launcher is not uniquely identified with a name and a tail number. Instead, each expendable launcher type (DELTA, TITAN, etc.) has one name and tail number associated with it. For example, all the Delta launchers are identified by Delta, tail number 1. This entry tracks the number of launchers in each stage (Development, Production, etc.). There are two reasons for this design: the large number of launchers and the "sameness" within each launcher type. If they were uniquely identified, the large number of launchers would require too much space to be easily displayed on the ACB Screen. The second reason launchers are uniquely handled is that within each launcher class, (DELTA, TITAN, etc.) the launcher characteristics are identical. For example, each Atlas launcher can lift the same amount to orbit.

The reusable launcher units are handled like all the other systems because the computer must keep track of the remaining launches of each unit.

The expendable launchers have a different acquisition cycle than the other units in the game. The necessary technology has already been developed for all the expendable launchers; therefore, they skip the Feasible and Research phases. The Launcher ACB Screen displays the name and number of all the expendable launchers currently in the Development and Production Phase. It also shows the number which are currently available to launch. As an aid to the player, the screen also shows the number of launchers currently in transition between phases.
When the player brings up the Launcher ACB Screen, the computer asks for the type of missile (ATLAS, NIMFA etc.) to be upgraded from the Development stage. The computer also asks for the number of missiles to be upgraded. The computer ensures the upgrade is affordable. The Development column is the only phase the player can modify. Each year, the computer automatically updates the other columns (Dev-Prod to Production to Prod-Active to Active). At the player's request, the Game Controller may add additional launchers into the acquisition pipeline as the game progresses.

In the Acquisition Phase, the players can modify satellites to incorporate survivability features. The player indicates which unit to modify, and the Add Survivability procedure returns a list of survivability modifications. These survivability features and associated costs are listed in Table IV. The costs are a function of the unit being modified. This screen shows the name of the satellite being modified, the current mass, the remaining year's budget, and the seven different survivability factors which may be added to the satellite. The cost of each modification is also listed on the screen.

Some modifications (5, 5, and 7) increase the mass of the satellite. Laser/Nuclear hardening with 15 Day autonomy increases the satellite mass by 20%. Laser/Nuclear hardening with 180 Day autonomy increases the satellite mass by 30%. Also, each maneuver added to a satellite increases its mass by 30%. If the player picks one of these modifications, the computer displays the modified mass and requests confirmation.
Table IV
Survivability Features

<table>
<thead>
<tr>
<th>Method</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anti-Jam/nuclear protection on data/control links</td>
<td>.3 * R&amp;D</td>
</tr>
<tr>
<td>Low Anti-Jam/nuclear protection on COM links</td>
<td>.1 * R&amp;D</td>
</tr>
<tr>
<td>Medium Anti-Jam/nuclear protection on COM links</td>
<td>.2 * R&amp;D</td>
</tr>
<tr>
<td>High Anti-Jam/nuclear protection on COM links</td>
<td>.3 * R&amp;D</td>
</tr>
<tr>
<td>Laser/nuclear hardening and 15 Day autonomy</td>
<td>.3 * Prod</td>
</tr>
<tr>
<td>Laser/nuclear hardening and 180 Day autonomy</td>
<td>2 * Prod</td>
</tr>
<tr>
<td>Satellite maneuverability (Price is per maneuver)</td>
<td>.3 * Prod</td>
</tr>
</tbody>
</table>

The players are allowed to change their minds about the modification. By entering the same modification number, the computer will remove that survivability aspect. It will also update the budget and satellite mass accordingly. There are some satellites which cannot be modified because of the satellite system design. These units have an "X" in their first survivability attribute. The program will not allow these satellites to be modified. There are also some satellites which have survivability features built into the design.

The cost of these added features must also be within the remaining budget. It is important to recognize the added weight these measures can add to the system. If too many features are added, the system may become too heavy to launch.
There are several other procedures the player may call during the Acquisition Phase. Find Unit will show the current attributes of a unit. The player may also check the health-status of a mission constellation (COMM, WEATHER, etc.) by calling the Constellation Age Screen procedure. This screen displays the name and tail number of all deployed (active and spare) systems performing that mission. It also shows the expected lifetime and the number of maneuvers remaining for each satellite. The screen also displays the orbit parameters of each satellite.

The player may maneuver satellites by calling the Display Orbits/Maneuver Sats Procedure. The player enters the orbit they are interested in, and the screen shows all the satellites currently in that orbit. At this time, the player may put a satellite into a Spare status, re-activate a satellite, or move a satellite. The players may change any orbit parameter if the satellite has sufficient fuel on board. Each orbit modification decrements the remaining maneuvers by one. When the remaining maneuvers equal zero, the satellite cannot perform any more maneuvers.

Finally, the player may see the global coverage of a given mission. The player indicates the mission, and the computer generates a world map with the ground swaths of the active satellites performing that mission. The coverage is shown for a single orbit of each active satellite. The ground swath for each satellite begins its coverage at its ascending node. Each satellite has a unique ground swath which is determined by the system Field Of View (FOV) and orbit altitude. The FOVs for each mission and their appropriate orbits are summarized in
Table V. The satellites are most effective at the orbit listed. The FOV for the systems will be less if another orbit is used.

<table>
<thead>
<tr>
<th>Mission</th>
<th>Orbit</th>
<th>FOV (Degrees)</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMM</td>
<td>GEO</td>
<td>60</td>
</tr>
<tr>
<td>IMINT</td>
<td>VLEO</td>
<td>5</td>
</tr>
<tr>
<td>NAV</td>
<td>MEO</td>
<td>20</td>
</tr>
<tr>
<td>SIGINT</td>
<td>VLEO</td>
<td>15</td>
</tr>
<tr>
<td>WARNING</td>
<td>GEO</td>
<td>60</td>
</tr>
<tr>
<td>WEATHER</td>
<td>LEO</td>
<td>12</td>
</tr>
</tbody>
</table>

**Assign Ground Support Phase.** In this phase the players assign ground support to the satellites. This is a key factor of the game. Each satellite must have its required ground support before it can be launched. A satellite can only be upgraded from Production status to Ready status when its required ground support has been assigned. Ground support can only be assigned to a satellite in the Production status. The ground support units must be active to be assigned to a satellite.

There are three types of ground support units: ground control terminals, data processing centers, and user terminals. The ground support units vary in the number required to accomplish their mission. Usually, the higher Tech Level units require fewer numbers to perform the mission. The player must determine the trade off between many-less-capable systems or fewer-more-capable systems. As expected, the higher Tech Level systems are more expensive to acquire.
Some ground units may be used for more than one satellite. For example, one System Unique LD Processor can provide the necessary data processing for all the LSCS II satellites on orbit. Each satellite must have all three of the ground support categories assigned before they can be launched.

At the beginning of this phase, the computer calls the Ground Support procedure. It lists all the satellites currently in the Production phase and the ground support elements which have already been assigned to them. On the bottom half of the screen, the computer lists the ground units which have not been assigned yet. The player indicates which ground unit he wishes to assign and then the satellite he wishes to assign it to. This phase continues as long as there are available ground units or until the player wishes to go on to the next phase.

Deployment Phase. In this phase, players match the desired payloads with launchers. The computer calls the Display Launchable Sats procedure which presents the list of launchable satellites. A launchable satellite is a satellite which has been produced and has the necessary ground support (a status of Ready). Then, the players choose which satellite they want launched.

There are two types of satellites: composite and regular. A composite satellite contains systems which have been added to the original satellite. A regular satellite does not have any add-on systems. Survivability factors are not considered add-on systems.

If the player wants to create a Composite Satellite, they access the Composite Satellite Screen. The screen shows all the satellites
currently in the "Ready" status (system has completed the production phase and has its ground support assigned). The player first enters the "host" satellite. The "host" satellite must have a larger mass than any of the "parasites" added on. The screen displays the "host" system and then asks for the "parasite" system. Because the "parasite" systems can use many of the sub-units of the "host", the "parasite" systems add only 50% of their mass to the overall composite satellite's mass. There is a maximum of five "parasites" on a composite satellite (20:39). The players should watch the overall system weight as they create composite satellites. They may create a satellite which is too heavy to lift. The "host" system and all its "parasites" are considered as one system once the satellite has been launched. When a composite satellite is maneuvered or is put into a "spare" status, all the systems on board are affected.

**Composite Satellite Example.** The player wishes to add a Single Channel Transponder (SCT) to a GPS-Al Navigation Satellite. The GPS is considered the host system because it has the larger mass. The mass of the GPS satellite is 780 kg, and the mass of the SCT is 20 kg. "Parasite" systems add only 50% of their mass to the overall system mass. Therefore, 10 kg is added, resulting in an overall system mass of 790 kg.

After selecting the satellite to be launched, the next step is determining the satellite orbital parameters. The parameters modeled in this game are the orbit altitude (VLEO, LEO, MEO, MOL, and GEO), inclination (0 to 905), and ascending node (-180 to +180). The orbital parameters are a function of mission requirements. These requirements
include the necessary ground coverage, the required system accuracy, and how often coverage is required.

**Orbit Parameter Description.** The orbit altitude plays a major role in the satellite mission. In the game, all orbits (except Molniya) are considered to be circular. Obviously, the greater the altitude, the more area of the earth can be observed; consequently, the number of systems required for global coverage decreases. However, as the altitude increases, the resolution capability of the satellite decreases. This is especially important for the Reconnaissance and Weather satellites. Therefore, these satellites normally populate the lower orbits (VIOO and LEO respectively). On the other hand, Communication and Surveillance satellites are often placed high above the earth in Geosynchronous Orbit. This placement allows the satellite to appear to remain stationary in space. Actually the satellite rotates at the same rate as the earth. This allows the satellite to remain over the same spot on the ground at all times. A disadvantage of a high orbit is the increased weight penalty. Launchers can lift less weight to higher orbits than they can to lower orbits. The player must consider these factors when determining the orbit altitude for a satellite.

The satellite inclination determines the amount of the globe which will be overflown. For example, the ground path of an orbit with an inclination of 20 degrees follows a sinusoidal curve between 20 degrees North latitude and 20 degrees South latitude. The amount of the globe covered is proportional to the satellite's inclination. The global coverage reaches a maximum at an inclination of 90 degrees. This type
of inclination is called a polar orbit. The ground track of a polar orbit consists of vertical lines from the north pole to the south pole. Therefore, satellites in this type of orbit will overfly the entire globe. This inclination is of particular interest to reconnaissance and weather satellites. Another inclination of special interest is that used by geosynchronous satellites. These satellites need to stay over the same ground point; therefore, these satellites use an orbital inclination of zero.

The ascending node of the satellite is basically the longitude at which the satellite crosses from the southern hemisphere to the northern hemisphere. In the game, it is used to show the relative position of satellites with the same orbit altitude and inclination. The ascending node is also used as the starting point for a requested ground trace. For a satellite at GEO, the ascending node marks the ground point which the satellite hovers over.

After the orbit parameters are set, the player must choose a launch site. Two launch sites are modeled in the game: the Western Test Range (WIR) at Vandenberg, and the Eastern Test Range (EIR) at Cape Canaveral. Each site begins the game with the following pads: two Atlas, one Titan, one Delta, and one STS. Additional pads may be acquired as the game progresses. Each pad can support only a limited number of launches each year, so the player must take care when assigning launch sites.

The launch sites have their own orbit inclination window. The inclination window represents the range of inclinations which payloads may be injected into via a direct ascent. A launch within the launch
window is more fuel efficient than a launch outside the window. The window for launches from Vandenberg is 62 - 122 degrees of launch inclination. The window for launches from Cape Canaveral is 29 - 48 degrees of launch inclination. A launch designed to place a payload within the inclination window uses the standard orbit-weight correction factor (function of inclination and booster type). Any launch designed to place a payload in an orbital inclination outside this window is penalized, and its orbit correction factor is assigned as one-half. This represents the additional fuel required to put the satellite into the proper inclination. The orbit correction factor is multiplied by the launcher lift capacity to get the total to-orbit-lift-capacity for the launch. Because of the weight penalty imposed on an out-of-window launch, the player should carefully choose the launch site.

Once the player has designated a launch site, the computer will display a list of launchers which have the lift capability to place the payload into the desired orbit. This list includes the launcher name and number available, the total lift capacity, the final payload mass, and the remaining lift capacity of the launcher. The remaining lift capacity is used if the player is using a multiple payload launch. The computer tells the players if there are no active launchers which can lift the satellite. The player must also designate the launch pad to be used. If the appropriate pad is not available (each pad can support only a limited number of launches each year), the player must choose a different launcher type.

At this point the player has the option of creating a Multiple Launch. If there is sufficient lift capacity remaining, the player may
place additional payloads on board the launcher. Multiple Payloads take advantage of common sub-systems required for launching all satellites. Therefore, extra payloads add only 75% of their mass to the overall mass to be lifted (20:41). The Multiple Launch Screen shows the remaining satellites ready for launch and asks for the additional payloads. The computer will not allow the launcher to be over-loaded. There can be no more than four separate payloads, and the payloads must be launched into the same orbit (VLEO, MED, etc.) (20:41).

Once the payloads have been finalized, the computer asks the NASA player if the launch violates the civilian constraint. This restriction states at least 80% of the civilian launch requirements listed in the scenario must be met during the turn. If NASA says the launch would violate these constraints, the launch is scrubbed and the launch allocation process begins over again (20:36).

If NASA says the launch is okay, then the computer checks the launch reliability factor of the launcher and calls the random number generator to determine if the launch is successful. The random number generator returns a number between 0 and 1. If this number is greater than the reliability factor of the launcher, then the launch fails, and the launcher and all of its payload are considered destroyed. If the launcher was a reusable type, then considerable damage could be done to the overall lift capability. If the launch is unsuccessful, the launch pad is damaged. The number of remaining launches the pad can support that year is halved, and a repair fee of $10 Million is assessed.
If it is a successful launch, then the launcher and payloads are considered to have achieved orbit. At this point the launcher must successfully deploy each of its payloads. There is a uniform 97% chance per payload of a successful deployment (20:40). Again, the computer calls the random number generator. If the number is greater than .97, then the deployment has failed and the payload is considered destroyed. If the number is .97 or less, then the deployment was successful, and the payload is considered active. The payloads of a multiple payload launch must be deployed separately. If one payload is not successfully deployed, it does not affect the other systems.

Launch Example. The player wishes to launch an AFSATCOM Communication satellite. The desired orbit is GEO at an inclination of zero degrees. The player chooses to launch the satellite from Vandenberg. The AFSATCOM satellite is to be launched by itself without any add-on systems, so its mass is 1410 kg. The computer determines that the Titan 34D is the only active launcher which can lift the satellite. The lift capacity of the Titan 34D to GEO is 4500 kg. However, the inclination is outside the Vandenberg launch window; therefore, the orbit correction factor is one-half. Multiplying this factor by the lift capacity (.5 * 4500) gives a lift capacity of 2250 kg. Therefore, the Titan can launch the system. Loading the AFSATCOM leaves an excess lift capacity of 840 kg (2250 - 1410). The player also wants to launch a DSCS II Satellite; therefore, he decides to make a multiple launch. The original mass of the DSCS II Satellite (620 kg) is multiplied by .75 since it is a multiple launch. This mass (465
kg) is added to the overall launch weight resulting in a final launch weight of 1875 kg (1410 + 465).

Once the launch process has been completed, the computer decreases the active launchers of that type by one. If the launcher was reusable or recyclable, then its total missions are reduced by one.

The Find Unit, Constellation Age Screen, Display Orbits/Maneuver Sats, and Show Mission Ground Traces procedures may also be called from the Deployment Phase. These procedures were described in the Acquisition Phase.

This phase continues until all the launchable satellites have been launched, or the launch capability is exhausted.

Scoring Phase. After every launch phase, the game controller will assign a point for each mission requirement which has been met during the turn. These points are used to evaluate the effectiveness of the force structure designed by the players.

The game continues repeating these phases until the end of turn ten or at the discretion of the game controller.

Summary

The game is designed to be as user-friendly as possible. It is playable by space "novices" with minimal instructions. The game is also designed to be expandable. Additional mission types or improved units can be added to the game with relative ease.
V. Results, Conclusions and Recommendations

Results

A full scale validation of ADAM has not been performed. However, the game has been reviewed by Col Schroeder, Space Command Chair, Air War College, and by faculty and students of the Graduate Space Operations (GSO) Program at the Air Force Institute of Technology. They have concluded that ADAM is a playable and useful representation of the acquisition and deployment process of space assets. Their comments for modifications to ADAM are included in this chapter.

A full scale validation of ADAM will be performed at ACSC during the space elective course in January. The game is scheduled for initial use in May during the AWC Space Series Course. This course is designed to provide a

...working knowledge of terminology, concepts, and the environment for space operations; to identify key mission areas for space systems; and to appreciate fundamental space issues affecting space operations (1:1).

Conclusions

It is essential for managers of US space assets to know the concepts involved in space operations. Due to the strategic importance of space and space assets to both the US and Soviet Union, there is a great potential for space conflict. Consequently, the US must be prepared to fight, and win, a war in space. To succeed in a space war, the US must develop two essential ingredients of warfare: doctrine and technology (7:216).
There are two ways to develop doctrine: through experience and through theory. Since there is no experience in space conflict, the necessary data must be developed through theory. Schools like ACSC and AMC are obvious places to develop a Space Doctrine; however, very little time is allotted for teaching space and its concepts at these two schools. The amount of time allotted in the curriculum for space is only 40 hours at ACSC and 26 hours at AMC (16:—). ADAM will quickly and effectively teach concepts necessary to develop an effective Space Doctrine.

Potential Uses for ADAM

There are potential uses for ADAM besides those outlined in Chapter 1. The model can be used to show the difficulty associated with deploying SDI-type constellations. The game can reveal the tremendous costs associated with acquiring the large numbers of satellites in a typical SDI configuration. It can show how the rest of the space structure would suffer as a result of deploying the SDI satellites. The game can reveal the tremendous costs associated with acquiring the large numbers of satellites in a typical SDI configuration. The game can reveal many of the barriers to developing a ground-support network capable of handling all the satellites. Finally, the game can demonstrate the inability of the present launch facilities to deploy SDI.

ADAM can also reveal the steps needed to develop different types of space forces. The game can be used as an "electronic sketch pad" to help the user develop methods to create alternative space forces. The users would be able to determine which technologies must be
investigated and when. The model would also show the most cost effective method to reach this technology; when to make certain research decisions, or what type of survivability factors to add to the space force. The game can be used as a long range planning tool to develop a space order of battle.

Finally, the game can be used as an educational tool. It can show space neophytes the intricacies involved in the development of a space force. It can show the requirements of the ground support element, the difficulties encountered when maneuvering space assets, and the importance of incorporating survivability factors into satellites.

Recommendations for Further Study

ADAM can serve as the baseline for several follow-on efforts. These efforts would provide even greater uses of the current model. Some of these efforts are converting the game to a continuous simulation, adding a Soviet player, and developing a conflict capability.

Continuous Simulation. The game was developed as a discrete simulation to simplify the effort. The design divides the game into distinct phases. The player can only perform the missions included in that phase. For example, ground support elements cannot be assigned in the launch phase.

Converting the game to a continuous simulation would add more realism to the game. It would allow the acquisition, deployment and maneuvering of space assets to be accomplished simultaneously, just as is done in the real world. It would also allow for an immediate response to a launch failure or a satellite malfunction. Incorporating
A continuous simulation consists of three elements: developing a "game clock," developing an event scheduler, and modifying the use of the Update field currently in the database structure.

The game clock keeps track of the passage of time during play. It would allow the player to enter commands at any time during the turn (such as purchasing support elements or launching a satellite). The event scheduler accumulates a list of actions performed by the player. There is a "time-to-completion" associated with each action. This list would be sorted by the shortest time-to-completion. Once the previous action has been completed, the event scheduler takes the action to be performed next, and calls the appropriate procedures. Examples of these actions include updating the acquisition status of a ground support unit, updating the position of a satellite, or checking for system failure. The update field can be used to store the time-to-next action for each record in the database. This field would be continuously updated as the game clock progressed.

A continuous simulation allows each unit to have its own update time. Incorporating a more accurate passage of time into the game would inject more realism into the simulation.

Along with incorporating a continuous simulation, there should be a "history" file created so the players can review the commands that were given throughout the game. This "history" file should include the command given, the result of the command, and the turn in which the command was given.

Soviet Player. Another possible enhancement, is adding a Soviet Player to the game and playing the two sides against one another. The
game can be used with a Soviet database with no modifications to the program. The teams would play on separate computers without any direct interaction. However, after every turn, the database for each side would be written to a disk and the disks swapped. The Soviet database would be given to the US player and vice versa.

The database would then run through a filter program. The filter would provide some information on the other's force structure. The accuracy of this information would be a function of the satellite mission and available intelligence. The information provided by the filter program would vary according to the mission of the satellite. In the real world, satellite missions can sometimes be identified by monitoring the signals sent from ground stations and by observing the satellite maneuvers. Some satellite missions are easier to determine than others. The missions of satellites which often send and receive signals from ground stations are normally easier to identify than satellites which have little ground signal interaction. Therefore, a communication satellite would be easier to distinguish than an early warning satellite. The accuracy of this information increases with the on-orbit time of the satellite. The greater on-orbit time allows for more signal exchanges and more opportunities to observe the satellite maneuvering.

As an example, in turn 2 the filter reports a possible reconnaissance satellite located at X. By turn 4, however, there has been sufficient ground interaction and satellite maneuvering to correctly identify the unit as a weather satellite.
The accuracy of these predictions could also be modified by purchasing intelligence. The player could allocate some of his budget to intelligence gathering, and the filter program would provide more accurate and detailed information on the enemy's space force.

This information would be displayed on a "Strategic Alert Screen." This screen would show possible Soviet space deployments and probable constellation statuses. The filter could also reveal some portion of the enemy's ACB screen. Instead of waiting for a system to be deployed to determine its mission, the system could be identified while still in the acquisition process. This would provide a quicker reaction to enemy deployments.

By allowing this cross flow of information, the players can react to the other's actions. It provides the players with an active adversary to game against.

Conflict Capability. ADAM can be used as the front end for a space conflict model. The output from ADAM can be used as the starting point for a space conflict game. ADAM can be used to develop and deploy the space forces until some event occurs to trigger a conflict. At that time, all the deployed units would be saved to a data file and used as the initial input to the conflict simulation.

The development of a conflict simulation should begin by modeling a few one-on-one space engagements and eventually model a full all-out attack by both sides. The initial model should include weapon systems which are currently available or those which can be developed within the next five to ten years. The initial version should not include
exotic weapons like space-based electro-magnetic rail guns, but should include things like ground-based lasers and ground-based interceptors.

The model must develop a plausible transition scenario between peace-time and the beginning of the conflict. In the real world, it might be very difficult to distinguish between a covert anti-satellite attack and a normal system outage. The game could be designed to have several such pivotal events in the scenario. The game flow could be changed depending how the players react to the situations.

This transition phase should mark a change in the length of real time each game turn represents. Once the conflict begins, units in the multi-year acquisition pipeline will not play a role in the space situation. The war will have to be fought with the elements on hand, plus whatever systems which can be quickly deployed. The time steps should go from months and years to minutes and hours. When a tactical alert is given that a possible ASAT attack is underway, the player will have only a short time (in real terms) to react.

Once a conflict has begun, the program should provide some tactical data on enemy actions. Intelligence information such as an impending SS-11 ASAT launch or a possible laser attack from Sary Shagan should be available to the players. Once the program has alerted the player of a possible attack, the program should provide a list of possible counter-actions. Examples of these actions are maneuvering the target out of the way, shutting down the target satellite, or deploying decoys. Any maneuvering done in space should reflect the actual limitations of orbital mechanics.
The program should also provide an expected probability of success of an ASAT attack. The accuracy of this prediction could be a function of the type and amount of the intelligence purchased. Also, the program should provide a list of counter-attacks available to the player. Possible actions include attacking an enemy’s satellite or activating a spare to fill the gap left by the attack. The game should provide an estimated probability of success for each action. Finally, the game should display the result the space conflict has on the terrestrial demands of the space asset. To model this effectively, the space conflict game should be integrated with a ground combat game which accurately models the use of space assets. For example, if a weather satellite is destroyed, there should be a possibility of unexpected weather delaying or foiling an attack.

A space conflict scenario should also reflect the quick-launch replacement capability of the Soviets and the long lead times associated with US launches. It should model the current vulnerability of the US ground control network to direct or covert attacks. It should show the effect a degradation of the ground network would have on the space force.

The integration of a graphics package which provides a three dimensional view of orbits would allow the players to better visualize the various orbit types. The graphics package should be incorporated into the game code to allow easy access to the desired screens. This graphics package should display satellite launches and maneuvers, orbit traces, system failures, and ASAT attacks.
These modifications and follow-on efforts can be developed in the micro-computer environment. However, Turbo Pascal, Version 3, has a memory limitation which restricts expansion of the current code. Boreland International, Inc. has since developed Version 4, which does not have this memory limitation. Therefore, Version 4 should be used in any follow-on efforts.

Summary

Micro-computers offer an exceptional learning environment. Games developed on Micros provide an alternative method to traditional board games to develop strategies and tactics. These computer models are more user-friendly, and less manpower intensive, than the typical board game. Therefore, many more scenarios can be examined in the same amount of time: more viewpoints can be developed. The media of micro-computers allows wide dissemination of the game. This allows more people to benefit from the game, and it allows for more feedback for future modifications.

Because space is critical to the defense of the US, the managers of our space assets must be aware of the concepts involved with space operations. ADAM provides an effective environment to teach these concepts.
Appendix A: Resource Catalog

This appendix contains the units which can be acquired during the game. It is a modified version of the Resource Catalog developed for the SRAE. The users may develop their own catalog if specific objectives are to be taught.
## Communication Systems

<table>
<thead>
<tr>
<th>System</th>
<th>Tech Level</th>
<th># Channels</th>
<th>Mass (kg)</th>
<th>Lifetime (YR)</th>
<th>Cost ($M) R&amp;D/Prod</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFSATCOM</td>
<td>1</td>
<td>25,000</td>
<td>1410</td>
<td>6</td>
<td>200/80</td>
<td></td>
</tr>
<tr>
<td>DSCS II</td>
<td>2</td>
<td>35,000</td>
<td>620</td>
<td>8</td>
<td>200/50</td>
<td></td>
</tr>
<tr>
<td>DSCS III</td>
<td>3</td>
<td>40,000</td>
<td>1150</td>
<td>8</td>
<td>300/85</td>
<td>Includes Med Anti-Jam</td>
</tr>
<tr>
<td>Intelsat V</td>
<td>4</td>
<td>58,000</td>
<td>1000</td>
<td>8</td>
<td>100/80</td>
<td>Cannot be made Survivable</td>
</tr>
<tr>
<td>MILSTAR</td>
<td>5</td>
<td>30,000</td>
<td>3000</td>
<td>8</td>
<td>800/120</td>
<td>Includes 180 day autonomy &amp; Hi Anti-Jam</td>
</tr>
<tr>
<td>MILSTAR II</td>
<td>6</td>
<td>40,000</td>
<td>2500</td>
<td>8</td>
<td>1000/120</td>
<td>Same as MILSTAR</td>
</tr>
<tr>
<td>CrossLink</td>
<td>1</td>
<td>5,000</td>
<td>100</td>
<td>8</td>
<td>50/20</td>
<td>Use as Parasite Does not need R&amp;D from above systems</td>
</tr>
</tbody>
</table>
COMMUNICATION SYSTEMS (cont.)

<table>
<thead>
<tr>
<th>SYSTEM</th>
<th>TECH LEVEL</th>
<th># CHANNELS</th>
<th>MASS (kg)</th>
<th>LIFETIME (YR)</th>
<th>COST ($M)</th>
<th>R&amp;D/PROD</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRANSPODERS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCT - Single Channel Transponder</td>
<td>1</td>
<td>20</td>
<td>4</td>
<td></td>
<td>--/5</td>
<td></td>
<td>Low Anti-Jam only</td>
</tr>
<tr>
<td>SCT-1</td>
<td>1</td>
<td>30</td>
<td>6</td>
<td></td>
<td>60/8</td>
<td></td>
<td>Med Anti-Jam only</td>
</tr>
<tr>
<td>SCT-M</td>
<td>1</td>
<td>40</td>
<td>10</td>
<td></td>
<td>50/15</td>
<td></td>
<td>Hi Anti-Jam only</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TERMINALS</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Manpack</td>
<td>1</td>
<td>1</td>
<td>--</td>
<td>6</td>
<td>10/2</td>
<td></td>
<td>O&amp;M Costs are 10% of Prod Cost</td>
</tr>
<tr>
<td>C3-small</td>
<td>2</td>
<td>5</td>
<td>--</td>
<td>6</td>
<td>20/5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C3-medium</td>
<td>3</td>
<td>10</td>
<td>--</td>
<td>8</td>
<td>50/7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C3-large</td>
<td>4</td>
<td>50</td>
<td>--</td>
<td>10</td>
<td>50/25</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## WEATHER SYSTEMS

<table>
<thead>
<tr>
<th>SYSTEM</th>
<th>TECH LEVEL</th>
<th>DOWN LEVEL</th>
<th>VIDEO D/N</th>
<th>IMAGE D/N</th>
<th>TEMP WIND</th>
<th>SOLAR ACT</th>
<th>MASS (kg)</th>
<th>LIFETIME (YR)</th>
<th>COST (R&amp;D/PROD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIMBUS</td>
<td>1</td>
<td>Daily</td>
<td>M M</td>
<td>L L</td>
<td>Y</td>
<td>-</td>
<td>950</td>
<td>4</td>
<td>200/65</td>
</tr>
<tr>
<td>TIROS</td>
<td>2</td>
<td>Daily</td>
<td>M M</td>
<td>M M</td>
<td>Y</td>
<td>Y</td>
<td>1420</td>
<td>6</td>
<td>400/65</td>
</tr>
<tr>
<td>ATTROS</td>
<td>3</td>
<td>Real</td>
<td>H M</td>
<td>H M</td>
<td>Y</td>
<td>Y</td>
<td>1725</td>
<td>8</td>
<td>300/85</td>
</tr>
<tr>
<td>SMS/GOES</td>
<td>4</td>
<td>Real</td>
<td>--</td>
<td>M M</td>
<td>--</td>
<td>Y</td>
<td>1130</td>
<td>6</td>
<td>500/85</td>
</tr>
<tr>
<td>GOES-D</td>
<td>5</td>
<td>Real</td>
<td>--</td>
<td>H M</td>
<td>--</td>
<td>Y</td>
<td>1800</td>
<td>8</td>
<td>500/70</td>
</tr>
<tr>
<td>DMSP</td>
<td>5</td>
<td>Real</td>
<td>H H</td>
<td>H H</td>
<td>Y</td>
<td>-</td>
<td>1650</td>
<td>6</td>
<td>100/100</td>
</tr>
</tbody>
</table>

DMSP satellite includes 15 Day Autonomy, does not require R&D from GOES-D

L - low resolution, M - medium resolution, H - high resolution
drop - deorbits data package on ground command
daily - radio down-link on regular schedule
real - real time down-link

## NAVIGATION SYSTEMS

<table>
<thead>
<tr>
<th>SYSTEM</th>
<th>TECH LEVEL</th>
<th>NAV ACCURACY (M)</th>
<th>INTEGRATED POSITION (M)</th>
<th>REQUIRED REDUNDANCY (#)</th>
<th>MASS (kg)</th>
<th>LIFETIME (YR)</th>
<th>COST ($M) R&amp;D/PROD</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOVA</td>
<td>1</td>
<td>160</td>
<td>1.0</td>
<td>4</td>
<td>136</td>
<td>4</td>
<td>400/35</td>
</tr>
<tr>
<td>GPS-A1</td>
<td>2</td>
<td>50</td>
<td>0.1</td>
<td>3</td>
<td>780</td>
<td>6</td>
<td>500/60</td>
</tr>
<tr>
<td>GPS-A2</td>
<td>3</td>
<td>15</td>
<td>0.01</td>
<td>4</td>
<td>780</td>
<td>6</td>
<td>500/60</td>
</tr>
<tr>
<td>GPS-B1</td>
<td>4</td>
<td>12</td>
<td>0.01</td>
<td>3</td>
<td>1500</td>
<td>6</td>
<td>500/80</td>
</tr>
<tr>
<td>GPS-B2</td>
<td>5</td>
<td>6</td>
<td>0.005</td>
<td>4</td>
<td>1500</td>
<td>6</td>
<td>500/80</td>
</tr>
</tbody>
</table>

57
### RECONNAISSANCE SYSTEMS

<table>
<thead>
<tr>
<th>SYSTEMS</th>
<th>TECH LEVEL</th>
<th>DOWN LINK</th>
<th>VIDEO D/N</th>
<th>IMAGERY D/N</th>
<th>MASS (kg)</th>
<th>LIFETIME (YR)</th>
<th>COST ($M) R&amp;D/PROD</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IMINT</strong> - Imagery/Photo Intelligence</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eagle</td>
<td>1</td>
<td>Daily</td>
<td>L - M -</td>
<td>11000</td>
<td>1</td>
<td>800/100</td>
<td></td>
</tr>
<tr>
<td>Image</td>
<td>2</td>
<td>Daily</td>
<td>M M - M M</td>
<td>3000</td>
<td>1</td>
<td>400/80</td>
<td></td>
</tr>
<tr>
<td>Scene</td>
<td>3</td>
<td>Real</td>
<td>H H H H H</td>
<td>5000</td>
<td>1</td>
<td>500/50</td>
<td></td>
</tr>
<tr>
<td>Snoop</td>
<td>4</td>
<td>Real</td>
<td>H H M M M</td>
<td>6500</td>
<td>2</td>
<td>750/150</td>
<td></td>
</tr>
</tbody>
</table>

**SIGINT** - Signal Intelligence

<table>
<thead>
<tr>
<th>SIGNAL</th>
<th>ACCURACY</th>
<th>SIGNAL IDENTITY</th>
<th>ACCURACY</th>
<th>SIGNAL IDENTITY</th>
<th>SIGNAL IDENTITY</th>
<th>SIGNAL IDENTITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ferret-2</td>
<td>1</td>
<td>M Good</td>
<td>200</td>
<td>6</td>
<td>100/50</td>
<td></td>
</tr>
<tr>
<td>Ferret-3</td>
<td>2</td>
<td>M Good</td>
<td>1500</td>
<td>6</td>
<td>250/60</td>
<td></td>
</tr>
<tr>
<td>Ferret-4</td>
<td>3</td>
<td>H VGood</td>
<td>700</td>
<td>8</td>
<td>100/80</td>
<td></td>
</tr>
</tbody>
</table>

Ferret 2 and Ferret 3 deorbit data packages
Ferret 4 has a real time down-link

L - Low accuracy, M - Medium accuracy, H - High accuracy
<table>
<thead>
<tr>
<th>SYSTEM</th>
<th>TRACK</th>
<th>IDENTIFY</th>
<th>MASS (kg)</th>
<th>LIFE TIME (YR)</th>
<th>COST ($M) R&amp;D/ACQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>SENS</td>
<td>Y</td>
<td>Y country</td>
<td>1100</td>
<td>6</td>
<td>700/150</td>
</tr>
<tr>
<td>SENS-2</td>
<td>Y</td>
<td>Y target</td>
<td>1400</td>
<td>8</td>
<td>400/200</td>
</tr>
<tr>
<td>ATS-1</td>
<td>HAIA</td>
<td>Y target</td>
<td>315</td>
<td>.5</td>
<td>700/200</td>
</tr>
<tr>
<td>ATS-2</td>
<td>HA</td>
<td>Y target</td>
<td>435</td>
<td>2</td>
<td>500/200</td>
</tr>
<tr>
<td>ATS-3</td>
<td>AIA</td>
<td>Y target</td>
<td>1400</td>
<td>4</td>
<td>500/150</td>
</tr>
</tbody>
</table>

HAIA - High altitude in afterburner
HA - High altitude
AIA - All altitudes in afterburner

Identifying the threat systems target (TARGET IDENTIFY) is not applicable to aircraft threats.
## OFFENSIVE SYSTEMS

<table>
<thead>
<tr>
<th>SYSTEM TECH :</th>
<th>SPACE :</th>
<th>TARGETS</th>
<th>LEO</th>
<th>MEO</th>
<th>GEO</th>
<th>GROUND</th>
<th>A/C</th>
<th>SHIPS</th>
<th>MASS (kg)</th>
<th>LIFETIME (YR)</th>
<th>COST ($M)</th>
<th>R&amp;D/P&amp;D</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEVEL</td>
<td>LEO</td>
<td>MEO</td>
<td>GEO</td>
<td>GROUND</td>
<td>A/C</td>
<td>SHIPS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Space Rocks</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>500</td>
<td>10</td>
<td>100/30</td>
<td></td>
</tr>
<tr>
<td>Orbit Bomb</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>500</td>
<td>N/A</td>
<td>100/100</td>
<td></td>
</tr>
<tr>
<td>Space Mine</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>500</td>
<td>20</td>
<td>-/20</td>
<td></td>
</tr>
<tr>
<td>F-15 MHV</td>
<td>1</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>N/A</td>
<td>20</td>
<td>-/20</td>
<td></td>
</tr>
<tr>
<td>Space Plane</td>
<td>1</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>N/A</td>
<td>N/A</td>
<td>300/100</td>
<td></td>
</tr>
<tr>
<td>G/Laser (Ground Based)</td>
<td>1</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>N/A</td>
<td>6</td>
<td>500/100</td>
<td></td>
</tr>
<tr>
<td>X-Ray Laser</td>
<td>2</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>-</td>
<td>H</td>
<td>-</td>
<td>4000</td>
<td>6</td>
<td>200/200</td>
<td></td>
</tr>
<tr>
<td>P-BEAM</td>
<td>3</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>-</td>
<td>H</td>
<td>Y</td>
<td>7500</td>
<td>6</td>
<td>700/125</td>
<td></td>
</tr>
</tbody>
</table>

Orbit Bomb and Space Plane are launched on demand

H - high altitude
<table>
<thead>
<tr>
<th>SYSTEM TECH :</th>
<th>ICBM</th>
<th>CRUISE</th>
<th>Mass Time Cost ($M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEVEL :BOOST MIDCOURSE REENTRY: A/C MISSILE: (kg) (yr) R&amp;D/PROD</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**DEFENSIVE SYSTEMS**

<table>
<thead>
<tr>
<th>TARGETS</th>
<th>LIFE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rocket- Int</td>
<td>4000</td>
</tr>
<tr>
<td>Rocket</td>
<td>N/A</td>
</tr>
<tr>
<td>G/Laser1 (Ground Based)</td>
<td>N/A</td>
</tr>
<tr>
<td>G/Laser2 (Ground Based)</td>
<td>N/A</td>
</tr>
<tr>
<td>S/Laser (Space Based)</td>
<td>9500</td>
</tr>
<tr>
<td>S/P-BEAM (Space Based)</td>
<td>7500</td>
</tr>
</tbody>
</table>

H - high altitude, L - low altitude, H/L - all altitudes
## GROUND CONTROL SYSTEMS

<table>
<thead>
<tr>
<th>SYSTEM</th>
<th>LIFETIME (YR)</th>
<th>COST ($M)</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ground Control Terminals</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Unique Ground Terminal (SUGCT)</td>
<td>15</td>
<td>20 5 5</td>
<td>One terminal required for each system type (DSCS II, GPS, etc.) and one required for each GEO satellite</td>
</tr>
<tr>
<td>System Unique Mobile Ground Terminal (SUMGCT)</td>
<td>10</td>
<td>20 10 10</td>
<td>Same as SUGCT</td>
</tr>
<tr>
<td>Common Ground Control Terminal (CGCT)</td>
<td>15</td>
<td>160 20 15</td>
<td>Four required to control entire Architecture</td>
</tr>
<tr>
<td>Consolidated Space Ops Center (CSOC)</td>
<td>25</td>
<td>20 100 5</td>
<td>One required plus three CGCTs to control entire Architecture</td>
</tr>
<tr>
<td>Common Mobile Ground Terminal (CMGCT)</td>
<td>10</td>
<td>60 30 20</td>
<td>Same as CGCT</td>
</tr>
<tr>
<td><strong>User Terminals</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Unique User Terminal (SUUT)</td>
<td>15</td>
<td>20 2 1</td>
<td>One required for each system (DSCS II, GPS, etc.) user</td>
</tr>
<tr>
<td>Mission Unique User Terminal (MUT)</td>
<td>15</td>
<td>100 2 1</td>
<td>One required for each user of a mission area (Comm, Nav, etc.)</td>
</tr>
<tr>
<td>SYSTEM</td>
<td>LIFETIME</td>
<td>COST ($M)</td>
<td>NOTES</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>----------</td>
<td>-----------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>Data Processing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Unique Data Processor</td>
<td>15</td>
<td>100 10</td>
<td>One required for each system type (DSCS II, SEWS, etc.)</td>
</tr>
<tr>
<td>(SUDP)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Unique Mobile Data</td>
<td>10</td>
<td>100 30</td>
<td>Same as SUDP</td>
</tr>
<tr>
<td>Processor (SUMDP)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mission Unique Data Processor</td>
<td>15</td>
<td>200 20</td>
<td>One required for each Mission area (Comm, Nav, etc.)</td>
</tr>
<tr>
<td>(MUDP)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>consolidated Space</td>
<td>25</td>
<td>20 100</td>
<td>Used for all systems</td>
</tr>
<tr>
<td>Ops Center (CSOC)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mission Unique Mobile Data</td>
<td>10</td>
<td>150 60</td>
<td>Same as SUDP</td>
</tr>
<tr>
<td>Processor (MIMDP)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# LAUNCH VEHICLES

<table>
<thead>
<tr>
<th>LAUNCHER</th>
<th>RELIABILITY (%)</th>
<th>MASS-TO ORBIT (KG)</th>
<th>COSTS ($M)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>V&amp;EO</td>
<td>LEO</td>
<td>MEO</td>
</tr>
<tr>
<td>Delta</td>
<td>92</td>
<td>5170</td>
<td>4800</td>
</tr>
<tr>
<td>Atlas/Cent</td>
<td>88</td>
<td>9575</td>
<td>8895</td>
</tr>
<tr>
<td>Titan 34D</td>
<td>94</td>
<td>13900</td>
<td>12915</td>
</tr>
<tr>
<td>Titan/IUS</td>
<td>92</td>
<td>11120</td>
<td>10975</td>
</tr>
<tr>
<td>Titan/Cent</td>
<td>89</td>
<td>30000</td>
<td>27000</td>
</tr>
<tr>
<td>Shuttle-1 R</td>
<td>92</td>
<td>29275</td>
<td>26200</td>
</tr>
<tr>
<td>Shuttle-2 R</td>
<td>89</td>
<td>14375</td>
<td>13355</td>
</tr>
<tr>
<td>STS R</td>
<td>97</td>
<td>63530</td>
<td>59245</td>
</tr>
<tr>
<td>Space Plane R</td>
<td>97</td>
<td>1600</td>
<td>1550</td>
</tr>
<tr>
<td>Space Taxi R</td>
<td>97</td>
<td>120000</td>
<td>120000</td>
</tr>
<tr>
<td>LLV C</td>
<td>97</td>
<td>7945</td>
<td>7380</td>
</tr>
<tr>
<td>MLV C</td>
<td>97</td>
<td>14191</td>
<td>13184</td>
</tr>
<tr>
<td>HLV C</td>
<td>97</td>
<td>19060</td>
<td>17710</td>
</tr>
<tr>
<td>SLV C</td>
<td>94</td>
<td>100000</td>
<td>93255</td>
</tr>
<tr>
<td>STS-PO R</td>
<td>97</td>
<td>38120</td>
<td>35415</td>
</tr>
</tbody>
</table>

R - reusable launcher, C - recyclable launcher
<table>
<thead>
<tr>
<th>SYSTEM</th>
<th>COST ($M)</th>
<th>R&amp;D/PROD/O&amp;M</th>
<th>LAUNCH RATE PER YEAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delta LP</td>
<td>—</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Atlas LP</td>
<td>—</td>
<td>25</td>
<td>10</td>
</tr>
<tr>
<td>Titan LP</td>
<td>—</td>
<td>50</td>
<td>10</td>
</tr>
<tr>
<td>STS LP</td>
<td>—</td>
<td>250</td>
<td>10</td>
</tr>
<tr>
<td>LHV LP</td>
<td>100</td>
<td>12</td>
<td>5</td>
</tr>
<tr>
<td>MLV LP</td>
<td>150</td>
<td>25</td>
<td>5</td>
</tr>
<tr>
<td>HLV LP</td>
<td>200</td>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>SLV LP</td>
<td>200</td>
<td>70</td>
<td>5</td>
</tr>
<tr>
<td>STS-PO-LP</td>
<td>160</td>
<td>350</td>
<td>10</td>
</tr>
</tbody>
</table>
Appendix B: Requirements Catalog

This appendix contains requirements the players must meet to be awarded points. The requirements are listed by player responsibility. Each list is divided into the mission areas of satellite systems, ground support, and launch systems. The team receives one point for each level of requirements which is filled. The use of the Requirements Catalog is optional and is designed for players unfamiliar with space system requirements. The Requirements Catalog was developed at ACSC for use in the SRAE game.
Space Command Requirements
NORAD, USCINCSpace

Communications

Level I
- 100 narrowband channels for COM within CONUS
- 20 color video (high-speed digital) for COM within CONUS
- 10 narrowband, 4 color video, 2 voice to Western Pacific
- 4 narrowband, 1 color video, 1 voice to Indian Ocean
- 20 narrowband, 8 color video, 2 voice to Europe
- 22 wideband voice within CONUS

Level II
- 22 narrowband channels with medium anti-jam within CONUS
- 4 narrowband channels with high anti-jam to Europe, 4 to Pacific, and 1 to Indian Ocean
- Satellite autonomy for 15 days

Level III
- 11 wideband voice channels with high anti-jam within CONUS, 1 to Europe, 1 to Pacific, and 1 to Indian Ocean
- 20 narrowband low speed digital with high anti-jam within CONUS, 4 to Europe, 4 to Pacific, and 1 to Indian Ocean

Level IV
- All level I requirements with high anti-jam
- Satellite autonomy for 180 days

Weather Meteorology

Level I
- Low resolution, day video for next-day launch prediction

Level II
- Low resolution, day/night video and image for launch window prediction
- Track solar storm activity for manned flight safe-launch periods
- Anti-jam data down-link
Level III
- Atmospheric temperature profile and medium resolution, day/night video and imaging for weekly launch schedule prediction and satellite control link monitoring
- Search-and-rescue beacon tracking
- Laser/nuclear hardening for 15-days autonomy

Level IV
- Defensive maneuverability for satellites
- Real-time medium resolution day/night video or sequenced images for shuttle launch and re-entry
- Determination of cloud height for shuttle re-entry
- Laser/nuclear hardening for 180-days autonomy

Navigation/Position

Level I
- Position accuracy for space systems to 100 meters (integrated)

Level II
- Position accuracy for space systems to 50 meters integrated: 100 Meters real-time
- Laser/nuclear hardening for 15-days autonomy
- Anti-jam/encryption on down-link
- Aircraft (interceptor) navigation accuracy to 100 meters

Level III
- Position accuracy for space systems to 1.5 meters (integrated); 50-meter real-time accuracy for satellite maneuvering
- Laser/nuclear hardening for 180-days autonomy
- Aircraft (interceptor) navigation accuracy to 15 meters

Level IV
- Position accuracy for space systems to .001 meters (integrated); 6 meter real-time
- Satellite defensive maneuverability

Reconnaissance—the act of obtaining the position, strength, and movement of enemy forces.

Level 0
- Determine orbit parameters of Soviet spacecraft.

Level I
Level II

- Identify by IMINT or SIGINT, the function of Soviet systems
- Identify possible nuclear weapons in space

Level III

- Track continuously, in real-time, Soviet spacecraft

Level IV

- No requirements

Surveillance & Warning—the act of watching for activity

Level 0

- Detect ICBM launches against the US

Level I

- Detect ICBM/SLBM launches against the US

Level II

- Provide attack assessment of areas under ICBM/SLBM attack to aid response selection
- Provide general count of weapons per target area
- Provide anti-jam data links
- Provide nuclear/laser hardening & autonomy for 15 days

Level III

- Provide specific target assessment and accurate count
- Provide detection and tracking of high-altitude aircraft
- Provide mobile data processing for satellite data
- Provide nuclear/laser hardening & autonomy for 180 days

Level IV

- Provide detection & tracking of low-altitude aircraft
- Provide detection of cruise missile launches
- Provide for satellite defensive maneuverability

Offense

Level I

- Deploy space satellite tracking system
- Provide low earth orbit interceptor rocket, laser)
Level II

- Deploy battle management control station
- Provide anti-jam command links and decoy detection in the space tracking system
- Provide weapons to engage mid earth orbit space systems (SLBM launched nuclear X-ray lasers, kinetic energy weapons, rockets)
- Provide weapons for attacking large fixed targets on the ground

Level III

- Provide mobile battle management backup control station
- Provide weapons to engage high earth orbit space systems
- Provide weapons to engage time-urgent ground and naval targets

Level IV

- Provide weapons to engage large ships and high-flying aircraft

Level V

- Provide weapons to engage all ground, air, and sea targets

Defense

Level I

- Deploy space satellite tracking system, and battle management control station
- Provide weapon to engage ICBM/SLBM within 10 minutes of launch (ground-based laser, directed energy, or space-based kinetic energy weapons)

Level II

- Backup battle management control station
- Anti-jam on command links and decoy detection in tracking system
- Provide weapons to engage ICBM/SLBM re-entry vehicles in mid-course (space-based lasers, particle beam accelerators, SLBM-launched nuclear X-ray laser, kinetic energy weapons)
- Provide weapon to engage high-altitude space planes and aircraft (space-based lasers, particle beams)
- Deploy tracking system for high-altitude aircraft

Level III

- Mobile battle management backup control station
- Provide weapons to engage re-entry vehicles in re-entry phase (ground- & air-launched interceptor rockets)
- Provide survivable high- and low-altitude sensor systems
Level IV
- Deploy weapon to engage low-altitude targets (laser, particle beam)

Ground Control/Tracking

Level 0
- Thin-line ground based tracking radar/optical network for determining orbit parameters (requires multi-orbit to determine track)

Level I
- Central ground control center for each satellite system
- Four equidistant equatorial ground relay centers for each geosynchronous satellite system to relay data to CONUS, unless crosslinks are employed

Level II
- Single consolidated space operations center to control all satellite systems
- Single orbit satellite track determination
- Eliminate overseas ground control stations
- Anti-jam control links

Level III
- Backup for consolidated space operations center
- Continuous global tracking of space objects
- Survivable, mobile ground stations for satellite control—Nuclear/laser hardening to provide 15-days autonomy for tracking network

Level IV
- Nuclear hardening to provide 180-days autonomy for tracking network
- Provide satellite maneuverability
Air Force Requirements

Communications

Level I
- 9 color video (high-speed digital) and 18 wideband voice channels within CONUS
- 4 narrowband, 4 color video, 20 wideband voice channels from Eastern CONUS to the Pacific
- 4 narrowband, 4 color video, 8 wideband voice channels from Eastern CONUS to Europe

Level II
- 96 narrowband and 96 wideband voice channels for secure communications within CONUS
- 27 narrowband and 27 wideband voice channels for secure communications from Eastern CONUS to Europe
- 10 narrowband and 10 wideband voice channels for secure communications from Eastern CONUS to the Pacific
- 2 narrowband and 2 wideband voice channels for secure communications from Eastern CONUS to South America

Level III

Level IV

Launch

Level I
- 1 heavy-lift, every 6 months
- 1 medium-lift, every 1.5 months
- 1 light-lift, every month

Level II
- 1 heavy-lift, every 3 months
- 1 medium-lift, every month
- 1 light-lift, every month

Level III
- 1 super-heavy-lift, every 6 months
- 1 heavy-lift, every month
- 1 medium-lift, every 3 weeks
- 1 light-lift, every week
Level IV

- 1 super-heavy-lift, every month
- 1 heavy-lift, every month
- 1 medium-lift, every 7 days
- 2 light-lift, every 7 days
Civilian/NASA Requirements
for Space Systems

Annual Launch Requirements

Level I
- 9 light-lift launches
- 4 medium-lift launches
- 1 heavy-lift launch
- 3 shuttle launches

Level II

Level III
- 1 super-heavy-lift launch

Level IV
- 2 super-heavy-lift launches

Civilian Use of Military Satellites

Communications

Level I
- 27 narrowband and 27 wideband voice channels for secure embassy communications from Wash DC to Europe
- 2 narrowband and 2 wideband voice channels for secure embassy communications from Wash DC to Canada
- 5 narrowband and 5 wideband voice channels for secure embassy communications from Wash DC to Northern Europe and the Soviet Union

Level II
- 18 narrowband and 18 wideband voice channels for secure embassy communications from Wash DC to East Asia
- 14 narrowband and 14 wideband voice channels for secure embassy communications from Wash DC to Southwest Asia
- 22 narrowband and 22 wideband voice channels for secure embassy communications from Wash DC to Australia and the South Pacific
Level III

- 12 narrowband and 12 wideband voice channels for secure embassy communications from Wash DC to the Caribbean and Central America
- 13 narrowband and 13 wideband voice channels for secure embassy communications from Wash DC to South America
- 39 narrowband and 39 wideband voice channels for secure embassy communications from Wash DC to Africa

Level IV

- 5 narrowband and 5 wideband voice channels for secure embassy communications from Wash DC to Southern Africa and Southern South America

Weather

Level I

- Low resolution, day video and imagery of the world for weather forecasting, every 48 hours
- Real-time, low resolution day imagery for storm/earthquake tracking around the world

Level II

- Medium resolution day and low resolution night imagery, every 24 hours, over the US, Atlantic, and Pacific
- Radiometric data of water vapor and carbon dioxide to make it possible to determine the 3-dimensional structure of the atmosphere, including its temperature and humidity

Navigation

Level I

- 100-meter continuous resolution for ship navigation in bays and harbors
- 50-meter resolution for tracking trains
- 1000-meter resolution for aircraft navigation and semi-tractor traffic tracking
- 30-centimeter resolution (integrated) for surveying

Level II

- 15-meter resolution for non-precision aircraft landing approach

Level III

- 10-meter resolution for automatic automobile navigation
NCINC Requirements

Communications

Level 0

- One-way, teletype broadcast to Pacific, US, Atlantic, and Europe from the Central US

Level I

- 16 wideband voice channels within the US for National Command Authority (NCA) conferencing
- 1 wideband voice channel to Europe and to Pacific
- 20 color video (high-speed digital) channels within CONUS, 2 to Europe, and 2 to Pacific for WHECS
- 40 narrowband channels within CONUS, 4 to Pacific, and 4 to Europe for force control
- 1 broadcast teletype to Pacific, US, Atlantic, and Europe from the central US with medium anti-jam for force execution

Level II

- 20 narrowband channels with medium anti-jam within CONUS
- 20 narrowband channels with medium anti-jam from CONUS over the North Pole
- 2 narrowband channels with high anti-jam to Europe, 2 to Pacific, and 1 to Indian Ocean
- Satellite autonomy for 15 days

Level III

- 5 wideband voice channels with high anti-jam within CONUS for force planning, 1 back to Europe, Pacific, and Indian Ocean
- 1 narrowband teletype broadcast to US, Europe, Pacific, and Indian Ocean, with high anti-jam
- 20 narrowband teletype channels with high anti-jam for force reportback over the CONUS and North Pole

Level IV

- All level I requirements with high anti-jam
- Laser/nuclear hardening for 180 days of autonomy

Weather

Level I

- Low resolution day video for 24-hour weather prediction
Level II

- Track solar storm activity for high frequency (HF) communication frequency selection and satellite communication link noise problem assessment
- Anti-jam down-link
- Low resolution day/night video and imaging every 24 hours for 48 hour weather prediction

Level III

- Temperature profile and medium resolution day/night video and image, every 12 hours, for 5-day weather prediction
- Search-and-rescue beacon tracking
- Laser/nuclear hardening for 15-day autonomy
- Measure ocean currents and temperature

Level IV

- Cloud height and high resolution day/night video/image of Northern Hemisphere for storm tracking and low-level aircraft flight planning, every 6 hours

Navigation

Level 0

- Open sea and transcontinental navigation (1000-meters accuracy)

Level I

- Air traffic control and harbor navigation (100-meters accuracy). Aid improved bombing accuracy

Level II

- Coastal inland waterway navigation (50-meters accuracy)
- Improved bombing accuracy combined with inertial navigation system (INS) (80-meters accuracy)
- Laser/nuclear hardening for 15-days satellite autonomy

Level III

- Small craft/air and sea navigation in high density areas (15-meters accuracy)
- Day/night blind bombing (25-meters accuracy)
- Non-precision landing approach (12-meters accuracy)
- 180-days satellite autonomy and endurability
Level IV
- High accuracy pinpoint blind bombing (5-meters accuracy)
- Aircraft low-level penetration aid (6-meters accuracy)
- Encrypted down-link
- Proliferation or defensive maneuverability for survivability

Reconnaissance

Level I
- Low resolution day/night imagery to determine ICBM and SLBM compliance with treaties. Update required every 2 weeks
- Ability to determine that a nuclear detonation has occurred within a given country

Level II
- Low resolution, day/night imagery, every 7 days, for tracking troop movements, mobile ICBMs, and for strategic warning
- Provide location of large ships at sea, every 7 days
- Identify general location of naval activity of smaller naval ships, every 7 days
- Provide real-time reporting of nuclear detonations with medium location and yield accuracy

Level III
- Low resolution, day video and medium resolution, day imagery, every 7 days
- Tracking of large ships, every 24 hours
- Identify SAM radar sites and missile frigate location with medium accuracy, every 48 hours
- Laser/nuclear hardening for 15 days of satellite autonomy
- Mobile ground processing system

Level IV
- Medium resolution day/night video for tracking troop formations, every 48 hours
- Tracking of ships every 24 hours with short periods (1 hour) of real-time tracking
- Real-time tracking of large, high-altitude aircraft
- Real-time high location nuclear detonation detection accuracy with medium accuracy yield and weapon type identification
- Track mobile SAMs and AAA, every 6 hours
- Anti-jam on control links
- Laser/nuclear hardening for 180-days autonomy
Level V

- High resolution day/night imagery for target analysis with real-time down-link
- Real-time tracking of low-altitude aircraft movements
- High accuracy location, yield and weapon typing of nuclear detonations over the Northern Hemisphere
- Real-time tracking of mobile SAMs, AAA aircraft, and ships
- Anti-jam on all data links
- Defense maneuverability for the sensor satellite

Offense

Level 0

Level I

Level II

- Attack time-urgent, strategic targets as part of executing the Single Integrated Operations Plan (SIOP)

Level III

- Non-nuclear attack on hard industrial targets. Require non-nuclear to minimize collateral damage and casualties

Level IV

- Attack airborne interceptors
TCINNC Requirements

**Communications** (4 Theaters—CENTRAL AMERICA/US, PACIFIC, ATLANTIC/EUROPE, INDIAN/ASIA. Identical requirements for each theater.)

**Level I**
- 1000 narrowband, intra-theater channels
- 10 high-speed digital, intra-theater channels
- 50 wideband, intra-theater voice channels
- 20 narrowband, intra-theater computer channels

Inter-theater requirements to CONUS are:
- 20 narrowband channels
- 2 high-speed digital channels
- 4 narrowband computer channels
- 4 wideband voice channels

**Level II**
- 50 medium anti-jam, intra-theater channels
- 1 medium anti-jam, intra-theater voice channel
- 1 high anti-jam, intra-theater broadcast teletype

**Level III**
- 4 high anti-jam, intra-theater wideband voice channels
- 1 high anti-jam, inter-theater wideband voice channel
- Satellite autonomy of 15 days

**Navigation**

**Level 0**
- Open sea and transcontinental navigation (1000-meter accuracy)

**Level I**
- Air traffic control and harbor navigation (100-meter accuracy)
- Improved bombing accuracy (150-meter accuracy)

**Level II**
- Improved artillery and rocket accuracy (50-meter accuracy)
- Improved bombing accuracy with inertial navigation system (INS) (80-meter accuracy)
Level III
- Air and sea navigation in high density areas (15-meter accuracy)
- Day/night blind bombing (25-meter accuracy)
- Non-precision landing approach (12-meter accuracy)

Level IV
- Low-level aircraft penetration aid (6-meter accuracy)
- Encrypted down-link

Weather/Meteorology

Level I
- Low resolution, day video for 24-hour weather prediction

Level II
- Medium resolution, day/night video and imaging, every 12 hours, for daily mission planning

Level III
- Anti-jam and encryption of data links
- Radiometer (temperature profile) of ocean currents, surface winds, and cloud height for naval engagements and low-altitude missions, every 12 hours
- Continuous search-and-rescue beacon tracking

Level IV
- High resolution, day/night video imagery for storm tracking and mission planning, every 6 hours
- Medium resolution imaging on demand from the equator to 60 degrees North

Reconnaissance

Level I
- Low resolution, day/night imagery every 7 days for troop movement tracking and strategic warning
- Provide location of large ships at sea, every 7 days
Level II

- Medium resolution day imagery for target identification, every 14 days
- Tracking of large ships, every 24 hours
- Identify general location of naval activity of smaller naval ships, every 7 days
- Identify SAM, radar site, and missile frigate location with medium accuracy, including target identification, every 7 days

Level III

- Real-time reporting of nuclear detonations
- Medium resolution, day/night video for crisis management
- Track mobile SAMs, AAA, and radars, every 6 hours
- Anti-jam on data links

Level IV

- High resolution, day/night imaging for target analysis using real-time down-link
- Track, in real-time, mobile radio and communications emitters

Surveillance

Level 0
Level I
Level II
- Map accuracy of 1:1,000,000
- Detect short-range and medium-range rocket launches

Level III
- Provide anti-jam data links

Level IV
- Provide detection/tracking of high-altitude aircraft and large ships

Offense

Level I

- Provide detection/tracking of all aircraft and combatant ships
- Map accuracy of 1:100,000
Level II
- Provide non-nuclear weapon for attacking large fixed ground targets
- Anti-jam data links
- Weapon to engage high-altitude space plane and aircraft

Level III
- Weapon to engage time-urgent ground and naval targets

Level IV
- Weapon to engage ships and low-flying aircraft

**Defense**

Level 0
Level I
Level II
- Weapon to engage high-altitude space plane and aircraft

Level III
Level IV
- Weapon to engage low-altitude targets
Communications—Responsible for inter-theater national level communications through Defense Communications Agency

Supports all inter-theater requirements included in TCINC plus:

**Level 0**
- 4 wideband voice and 100 narrowband channels to Europe

**Level I**
- 9 wideband voice and 18 narrowband channels for CINC to CINC secure communications within CONUS with low anti-jam capability

**Level II**
- 9 color video channels for conferencing between the CINCs within CONUS
- Medium anti-jam for the channels in I above

**Level III**
- High anti-jam for the channels in I above
- Common communications terminal for all communications systems

**Level IV**
- High anti-jam for all inter-theater narrowband communications

**Reconnaissance**

**Level I**
- Low resolution, daytime imagery for ICBM launcher for SALT verification
- Low location accuracy, nuclear detonation detection for nuclear test monitoring

**Level II**
- Medium resolution, daytime video and imagery for SALT verification
- Medium location and yield accuracy of nuclear detonations for nuclear test monitoring
Level III

- High resolution, daytime imagery for SALT verification
- High accuracy location and yield of nuclear detonations for nuclear test monitoring

Level IV

- Provide nuclear/laser hardening and autonomy for systems in III above for 15 days
- Provide mobile data processing for systems in I, II, or III above

Ground Control & Tracking

Level I

- 1 central ground control center for each space system plus three for geosynchronous systems unless crosslinks are used

Level II

- Single consolidated space operations center for all satellite system control
- Three relay centers for geosynchronous satellite crosslinks are not used

Level III

- Backup for consolidated space operation center
- Mobile, common ground station for nuclear operations
- Crosslink communications on all space systems—to remove dependence on overseas ground sites

Level IV

- Provide satellite maneuverability for high anti-jam communications and warning systems
Appendix C: Scenario

This appendix contains the information listed in the Scenario.

This information consists of three reports: a US Space Policy Update, an Intelligence Report, and the Space Budget provided by Congress for that turn. The Space Policy may restrict deployment of particular systems or may place additional demands on the military launch system to meet certain civilian/NASA demands. The Intelligence Report provides information on space activities by the Soviet Union. The scenario provided in the SRAE game was modified for this effort. This scenario is generic and should be replaced by a user generated scenario if specific objectives are to be taught.
TURN 1

US POLICY: Abide by all current space treaties and conventions.

INTEL REPORT: Soviet activity on space lab and shuttle increasing.

BUDGET: $4.5 Billion

TURN 2

US POLICY: Abide by all space treaties and conventions except Development of land-based or space-based ballistic missile defense. Ballistic missile defense baseline consists of three levels: boost, midcourse and terminal. Systems required for boost are: 5 warning satellites and 160 laser or particle beam battle stations in high earth orbit. Systems for midcourse are: 28,000 kinetic energy kill weapons or 150 particle beam weapons and 20 surveillance satellites. Systems for terminal defense are: 140,000 ground-to-space kinetic energy/rocket systems or 1000 ground-based lasers. Defense is intended to be 90% effective.

INTEL REPORT: Soviets begin space station construction. Two surveillance systems were temporarily inoperative while over Soviet Union — unknown cause.

BUDGET: $4.5 Billion

TURN 3

US POLICY: Continue development of ballistic missile defense systems.

INTEL REPORT: Apparent debris in low earth orbit first detected over Soviet Union.

BUDGET: $5 Billion

TURN 4

US POLICY: Deployment of sensors and C3 for ballistic missile defense is authorized.

INTEL REPORT: Construction of Soviet space station continuing. Space plane and shuttle activity increasing.

BUDGET: $6 Billion
TURN 5

US POLICY: Push for space arms control. Begin production of ballistic missile defense weapon systems. Push for nuclear survivable space system ground control and processing. NCA directs minimum survivable ground control systems consisting of the Consolidated Space Operations Center (CSOC) and three common mobile ground control terminals (CMGCT).

INTEL REPORT: Soviet space station reaches initial operational capability. Shuttle and space plane flights increasing. Debris cleared from low earth orbit by the Soviet Union.

BUDGET: $7 Billion

TURN 6

US POLICY: Continue production of ballistic missile defense systems.

INTEL REPORT: Soviets begin moon launches from the space station. Large ground-based laser has been used to destroy target in low earth orbit. Direct ascent ASAT missile can now reach middle earth orbit. Several unidentified objects now in geosynchronous orbit.

BUDGET: $8 Billion

TURN 7

US POLICY: Deployment of space defensive systems authorized. No system with offensive capability against air and surface targets is authorized.

INTEL REPORT: Soviets continue working on direct ascent ASAT.

BUDGET: $7.5 Billion

TURN 8

US POLICY: Continue deployment of defensive systems.

INTEL REPORT: Soviets execute a ground system breakout of ABM systems. Six ground lasers, 300 new ABM missile sites, and a new missile are identified as under construction. Four new unidentified systems now in geosynchronous orbit.

BUDGET: $8.5 Billion
TURN 9

US POLICY: Deployment of offensive sensor systems authorized.

INTEL REPORT: Four unidentified systems in geosynchronous orbit are probably space mines. Soviet defensive/offensive space sensors are deployed.

BUDGET: $10 Billion

TURN 10

US POLICY: Begin acquisition of offensive space systems. Push arms agreement on space.

INTEL REPORT: Soviets test a particle beam weapon in space. Soviets propose elimination of all beam weapons in space.

BUDGET: $10 Billion
Appendix D: Users Guide

Description

ADAM allows space assets to be acquired, deployed, and maneuvered in response to ground requirements. The different missions modeled in the game include: Communication, Navigation, Weather, Reconnaissance, Surveillance (Attack Warning), Launchers, Launch Pads, and Ground-Support elements. The game models two different types of Reconnaissance: ELINT (Electronic Intelligence), and SIGINT (Signal Intelligence). The objective of ADAM is to design a force structure which meets as many requirements as possible while operating within a limited budget. The game begins with the current (1987) US space force and continues through 10 turns (each turn is one year).

The database may be modified if specific learning objectives are to be taught. It would be beneficial to the user to keep a library of different databases.

This appendix contains a rule book for players, a section for the game controller, and hints for future programmers. The player rule book describes how to play ADAM. The game controller section describes the special procedures available only to the controller. It also includes instructions on how to develop a new database. The hints for future programmers contain information like the design of the data base and how to add new mission types.
## Table of Contents

1. General Comments .............................................. 92
2. Glossary of Game Terms ..................................... 92
3. Player Descriptions .......................................... 95
4. The 6-Phase turn sequence ................................. 96
   What you do in a turn
5. Game Set-Up .................................................. 98
6. Phase 1 ....................................................... 98
   Read Scenario Phase
7. Phase 2 ....................................................... 99
   Database Update Phase
8. Phase 3 ....................................................... 101
   Acquisition Phase
9. Phase 4 ....................................................... 108
   Assign Ground Support Phase
10. Phase 5 ...................................................... 110
    Deployment Phase
11. Phase 6 ...................................................... 120
    Scoring Phase
12. Game Controller ........................................... 120
    Some helpful hints
13. Future Programmers ................................. 122
    Some more helpful hints

91
1.0 General Comments

Each unit in the game is identified by a unique name and tail number. When asked to input a unit, the player must type the name and number exactly as displayed on the screen. For example, if the name is "FERRET-1", the player should not enter "Ferret-1". All the units in the database are in capital letters, so the players should keep the "Caps Lock" key on. The program will tell the player if it cannot find a unit. All cost figures are in terms of one million dollars. Therefore, a budget of 4,500 represents $4.5 Billion.

2.0 Glossary of Game Terms

Ascending Node - Point at which satellite ground trace crosses from Southern Hemisphere to Northern Hemisphere. Used as the point over which geosynchronous satellites hover.

Altitude - Height above the Earth's surface of satellite's orbit. Height is defined as either VLBO, LEO, MEO, MOL, or GEO.

Composite Satellite - A satellite which has a piggyback system which performs its own mission. Can contain up to 5 add-on systems. Each add-on system contributed 50% of its mass to total system mass.

GEO - Geosynchronous Orbit (37,000 km).

Game Controller - Person who controls game. Reads scenario, interprets rules, and controls SAI-RAK interface.

Ground Swath - Area of the ground which the satellite can view. Function of mission and altitude.

HLV - Heavy Lift Vehicle.

Inclination - determines amount of globe overflown. Defines a sinusoidal curve which the satellite ground track follows. Two types of particular interest:

- 90 degrees for recon and weather satellites and
- 0 degrees for GEO satellites.

LEO - Low Earth Orbit (2,000 km).

LLV - Light Lift Vehicle.
Glossary of Game Terms (cont.)

Lift Capacity - Amount of mass a launcher can lift to a particular orbit.

MEO - Medium Earth Orbit (20,000 km).

MLV - Medium Lift Vehicle.

Milestone - Represents different stages of acquisition cycle. Milestones 0, 1, 2, 3 correlate to Feasible, Research, Development, and Production.


Molnyina - Highly elliptical orbit. Apogee of 40,000 km and Perigee of 600 km.

Multiple Launch - Placing more than one payload on a launcher. Extra payloads add 75% of their mass to overall payload mass. Maximum of 4 payloads. Must be launched to same altitude.

Orbit - defines location of satellite space. Composed of three parameters: Altitude, Inclination, and Ascending Node.

SIW - Super Heavy Lift Vehicle.

Status - Defines condition of unit at any particular time. Total of 9: None, Feasible, Research, Development, Production, Ready, Active, Spare, and Dead.

Survivability Features - Enhancements added to satellites to allow performance of mission in a hostile environment. Can add to mass of system.

Tech Level - Defines current level of technology within each mission area. Research may not begin on advanced unit before development has completed on simpler unit.

VLEO - Very Low Earth Orbit (200 km).

2.1 Game Terminology. A game unit may be in one of 9 different statuses. The statuses and an explanation of each are listed in Table VI.
Table VI.

Unit Status Definitions

<table>
<thead>
<tr>
<th>Status</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>The technology level has not advanced far enough to allow research to have begun on unit</td>
</tr>
<tr>
<td>Feasible</td>
<td>The technology level has advanced far enough to allow research to begin on the unit</td>
</tr>
<tr>
<td>Research</td>
<td>The research required for the unit has been completed</td>
</tr>
<tr>
<td>Development</td>
<td>The development phase for the unit has been completed</td>
</tr>
<tr>
<td>Production</td>
<td>The unit has completed production and is waiting for the ground support element to be completed</td>
</tr>
<tr>
<td>Ready</td>
<td>The unit has its ground support elements and is ready for launch</td>
</tr>
<tr>
<td>Active</td>
<td>The unit is operational</td>
</tr>
<tr>
<td>Spare</td>
<td>The unit is dormant, it does not burn fuel and does not age</td>
</tr>
<tr>
<td>Dead</td>
<td>The unit has failed due to natural causes or by a launch accident</td>
</tr>
</tbody>
</table>

There are five different orbit types used in the game. The type, altitude, and usefulness are listed in Table VII. The altitude is given in km above the earth's surface. All the orbits, except the Molniya, are considered circular.
Table VII.
Orbit Descriptions

<table>
<thead>
<tr>
<th>Type</th>
<th>Altitude (km)</th>
<th>Suggested Missions</th>
</tr>
</thead>
<tbody>
<tr>
<td>VLEO (Very Low Earth Orbit)</td>
<td>200</td>
<td>Reconnaissance</td>
</tr>
<tr>
<td>LEO (Low Earth Orbit)</td>
<td>2000</td>
<td>Weather</td>
</tr>
<tr>
<td>MEO (Medium Earth Orbit)</td>
<td>20000</td>
<td>Navigation</td>
</tr>
<tr>
<td>MOL (Molynia Orbit)</td>
<td>* 40000 x 600</td>
<td>??</td>
</tr>
<tr>
<td>GEO (Geosynchronous Orbit)</td>
<td>37000</td>
<td>Communication</td>
</tr>
</tbody>
</table>

*entry indicates an apogee of 40,000 km and a perigee of 600 km

3.0 Player Descriptions

ADAM is designed for multiple players in a seminar type environment. The exact number and responsibilities can be modified as necessary; however, there is a required subset of players which should be present to play the game. This subset is a modification to the players in the SRAE game (20:14). The players and their responsibilities are listed in Table VIII.
Table VIII.

Player Positions

<table>
<thead>
<tr>
<th>Player</th>
<th>Responsibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chairman, Joint Chief</td>
<td>Overall Commander</td>
</tr>
<tr>
<td>of Staff</td>
<td></td>
</tr>
<tr>
<td>CINCSpace</td>
<td>Determines operational requirements and orbital parameters</td>
</tr>
<tr>
<td>Space Division</td>
<td>Responsible for system R&amp;D and production phases</td>
</tr>
<tr>
<td>NASA</td>
<td>Ensures the necessary commercial requirements as dictated in scenario are</td>
</tr>
<tr>
<td></td>
<td>included in operational decisions</td>
</tr>
<tr>
<td>NCINC</td>
<td>Strategic Warfare Requirements</td>
</tr>
<tr>
<td>TCINC</td>
<td>Tactical Warfare Requirements</td>
</tr>
<tr>
<td>Controller</td>
<td>Reads Scenario, interprets rules, and controls the SATRAK interface</td>
</tr>
</tbody>
</table>

4.0 The 6-Phase Turn Sequence

ADAM consists of ten one-year turns. Each turn contains the following phases:

1. Read Scenario Phase (done by controller)
2. Database Update Phase (done by computer)
3. Acquisition Phase
4. Assign Ground Support Phase
5. Deployment Phase
6. Scoring Phase (done by controller)
The players continue through these phases until the end of turn ten or at the discretion of the game controller. Each of the above phases is explained briefly below and in detail in pages 98 through 123.

4.1 Read Scenario Phase. In this phase, the Game Controller gives the players the scenario. The scenario describes the conditions the players must abide by for that turn.

4.2 Database Update Phase. In this phase, the computer advances the status of units upgraded last turn. A satellite system failure check is also done in this phase.

4.3 Acquisition Phase. In this phase, the players give orders for units to begin the next Milestone.

4.4 Assign Ground Support Phase. In this phase, the players assign the ground support element to satellites about to be launched. A satellite may not be launched until it has a completed ground support system.

4.5 Deployment Phase. In this phase, the players match satellites to be deployed with a launch vehicle. The player may create composite satellites or initiate a multiple payload launch. The player may also maneuver active satellites or put them in a spare status.

4.6 Scoring Phase. (optional) In this phase, the Game Controller evaluates the active space force and measures how effectively it meets the ground requirements.
5.0 Game Set Up.

The Game Controller should ensure the system has been booted up and the proper database loaded. When the system has warmed up and prompts the user with ">", enter "ADAM", to start the game. After the game has loaded, it will prompt the controller to load the database. The controller should enter the name of the database designed to teach that day's lessons.

If the SATRAK program is used, the controller enters "SATRAK25" to start the program.

6.0 Read Scenario Phase

In this phase, the Game Controller gives the players the scenario. The scenario describes conditions the players must abide by for that turn. The scenario includes the space budget provided by Congress for the turn, a US Space Policy Update, and an Intelligence Report of probable Soviet space activities. The Policy Update may restrict deployment of particular systems or may place additional demand on the military launch system to meet certain civilian/NASA demands. The Intelligence Report provides information on space activities of other countries. This report contains information like the possible deployment of new weapons or the activation of a space station. Scenarios can be modified as needed to reflect the current learning objectives.

Scenario Example. The following is an example of a one-turn environment provided by a Scenario.

US Policy: Begin deployment of ballistic missile defense system.
7.0 Database Update Phase

This phase is accomplished by the computer and is transparent to the players; however, they should understand what the program does in this phase. In the Database Update Phase, the computer updates the status of units, performs a satellite system failure check, assesses unit O&M costs, and determines the current Tech Level of each mission. Included in this section is a description of the above functions and examples of their applications.

Upgrade Units. In this phase, the computer advances the status of units currently being upgraded. Each unit in the database contains an update field that maintains the time remaining to complete the next phase in the acquisition cycle. If the upgrade required two years, the update field will be changed from a "2" to a "1". This indicates the unit will complete its upgrade the following year.

Database Update Example. In year 1988 the DSCS III, tail number 6, satellite has completed its development phase and awaits an upgrade to begin production. The order to begin production is given in the Acquisition Phase of year 1988. The update field of DSCS II #6 is set to "2" to indicate the upgrade (production) will take two years. In the Database Update Phase of year 1989, the update field of the satellite is changed to "1" to indicate there is one year left in the Production Phase. Finally, in the Database Update Phase of year 1990,
the update field of DSCS II #6 is set to "0" to indicate that production has been completed.

System Failure Check. The computer performs a system failure check for all active satellites every turn. The probability of failure for a satellite is .3% per year for its first active year, and increases as the satellite ages. When the satellite is two years from its expected lifetime, the probability of failure is 8%. When the satellite reaches its expected lifetime, the probability of failure is 58%. The program automatically "kills" a satellite if it survives one year past its expected lifetime.

Assess O&M Costs. During this phase, the Operations and Maintenance (O&M) costs for the active ground support units and launch pad facilities are assessed.

Determine Tech Level. During this phase, the computer also determines the current Technology (Tech) Level of the Defense Industry in each mission area (Communication, Navigation, etc.). The current Tech Level greatly impacts the space force structure. It determines the feasibility of a system. A feasible system indicates technology has reached a sufficient level to begin research on that system. For example, research on the MILSTAR 2 satellite system may not begin until the necessary technology has been developed by the MILSTAR 1 satellite system. At least one MILSTAR 1 satellite must complete its development phase before research may begin on the MILSTAR 2 system.

Tech Level Example. An example using DOD milestones may better define this important concept. Milestone 2 (Development) for MILSTAR 1, tail number 1, began in the Acquisition Phase of year 1989. It will
take two years for this upgrade, so Milestone 2 for this unit is scheduled for completion in the Database Update Phase of year 1991. Research cannot begin on MILSTAR 2 until the required technology is developed by the MILSTAR 1 system; therefore, MILSTAR 2 will not complete Milestone 0 (Feasible) until the Acquisition Phase of year 1991. Finally, in the Acquisition Phase of 1991, the player may then begin work on Milestone 1 for the MILSTAR 2 system.

8.0 Acquisition Phase

This is the first of three phases in which the player directly interacts with the computer. The Main Menu for the Acquisition Phase is displayed in Figure 1.

![ Acquisition Phase Main Menu

**Figure 1.** Acquisition Phase Main Menu
Display Satellite/Reusable Launcher ACB Screen. Enter a "1" from the Acquisition Phase Main Menu to access the Satellite/Reusable Launcher Acquisition Cycle Board (ACB) Screen. The ACB Screen lists all the satellites, support elements, and reusable/recyclable launcher units currently in the acquisition pipeline. The units are listed by name and tail number. The status headings on the screen (FEASIBLE, RESEARCH DEVELOPMENT, and PRODUCTION) roughly correspond to the Milestones 0,1,2, and 3 respectively as defined in DOD 5000.1. An example of the Satellite ACB Screen is displayed in Figure 2.

```
SATELLITE ACQUISITION BOARD
YEAR: 1987
* Indicates Unit has been Upgraded
BUDGET: 10000

MISSION FEASIBLE RESEARCH DEVELOPMENT PRODUCTION
COMM MILSTAR 1 *DSCS III 1 DSCS II 2 APSATCOM 4
SIGINT FERRET-4 1 FERRET-3 3 - FERRET-3 2
LAUNCHER MLV 1 IIV 1 *STS-1 5 -
WARNING SEWS-2 2 SEWS-2 1 *SEWS 4 -
WEATHER - - *DMSP 5 DMSP 4

Enter Name of unit to be updated (DSCS II, SEWS-2, etc.): SEWS-2
Enter tail number ("0" will create new copy): 2
```

Figure 2. Satellite/Reusable Launcher ACB Screen

To update a unit on the ACB Screen, enter the name and tail number exactly as they appear on the screen. If the unit name is "EAGLE", do not enter "Eagle". Once the name and tail number are entered the screen will show the current state of all the unit's attributes. These attributes contain information such as the unit's mass, its expected
lifetime, and a list of any modifications done to the unit. The attributes also contain the current status of the unit. The screen will then show how much the upgrade will cost and ask for a go-ahead. If the player types "Y", the unit will begin the next stage in the acquisition cycle. The player should notice the budget has been decremented to reflect this upgrade. They should also notice an asterisk is placed next to the unit name indicating the unit is being upgraded.

The ACB Screen will only show units currently in the acquisition pipeline. It will not show satellites ready for launch or ones that have already been deployed. The player may add copies of a unit which is on the ACB Screen or one that has already been deployed. This copy will be given the next available tail number of that system type. For example, four DSCS II satellites have been deployed and there are no more in the acquisition pipeline. To add another DSCS II satellite, the players enter the name of the satellite ("DSCS II") and then a tail number of "0". The program will then put a DSCS II, tail number 5, on the ACB screen. The new unit is placed in the Research stage and is then processed just like the units which began the game on the ACB screen. When development begins on a Launch Pad, the program will ask the player where to construct the Pad. The player has two choices: Vandenberg or Canaveral. Once the location is set, it cannot be changed.

**Display Launcher ACB Screen.** Enter a "2" from the Acquisition Phase Main Menu to access the Launcher ACB Screen. This screen displays all the expendable launchers in the acquisition pipeline. The
expendable launchers have a different acquisition cycle than the other units in the game. The necessary technology has already been developed for the launchers; therefore, they skip the Feasible and Research phase. The Launcher ACS Screen displays the name and number of all the expendable launchers currently in the Development and Production Phase. It also shows the number which are currently available to launch. As an aid to the player, the screen also shows the number of launchers currently in transition between the phases.

Add Survivability Features. Enter a "3" from the Acquisition Phase Main Menu to access the Add Survivability Features Screen. The players must enter the name and tail number of the unit they want modified. This screen lists the seven enhancements which may be added to satellites to allow them to perform better in a hostile environment. It also displays the cost of the features and which features are currently incorporated into the satellite. An example of the Survivability Features Screen is listed in Figure 3. This screen shows the name of the satellite being modified, the current mass, the remaining year's budget, and the seven different survivability factors which may be added to the satellite. The cost of each modification is also listed on the screen. Some modifications (5, 6, and 7) also add mass to the satellite. If the player picks one of these modifications, the computer displays the modified mass and requests confirmation. The player is allowed to change his mind about the modification. By entering the same modification number, the computer will remove that survivability aspect. It will also update the budget and satellite mass accordingly. There are some satellites
ADD SURVIVABILITY FEATURES

"X" Indicates unit already has that feature

SATELLITE: DESCS III 2  MASS: 1150  BUDGET: 10000

<table>
<thead>
<tr>
<th>Method</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1] Anti-Jam/nuclear protection on data/control links</td>
<td>45</td>
</tr>
<tr>
<td>[2] Low Anti-Jam nuclear protection on COM links</td>
<td>15</td>
</tr>
<tr>
<td>[3] Medium Anti-Jam/nuclear protection on COM links</td>
<td>+ 30</td>
</tr>
<tr>
<td>[4] High Anti-Jam/nuclear protection on COM links</td>
<td>45</td>
</tr>
<tr>
<td>[5] Laser/nuclear hardening and 15 Day autonomy</td>
<td>28</td>
</tr>
<tr>
<td>[6] Laser/nuclear hardening and 180 Day autonomy</td>
<td>160</td>
</tr>
<tr>
<td>[7] Satellite maneuverability (Cost is per maneuver)</td>
<td>28</td>
</tr>
</tbody>
</table>

Enter 1-7 or "Q" to quit

Figure 3. Survivability Features Screen

which cannot be modified because of the satellite system design.
These units have an "X" in their first survivability attribute. The program will not allow these satellites to be modified. There are also some satellites which have survivability features built into the system design. The players must recognize the added weight these measures add to the system. If too many features are incorporated, the system may become too heavy to launch. Enter "Q" to go back to the Main Acquisition Phase Menu.

Display Unit Attributes. Enter a "4" from the Acquisition Phase Main Menu to access the Display Unit Attribute Screen. The players must enter the name and tail number of the unit they wish to see. This screen lists the current state of all the unit's attributes. These attributes differ somewhat between mission types (COMM, NAV, etc.).
The attributes contain information such as the mass of the unit, when it was deployed, and what ground support elements have been assigned. If the program cannot find the requested unit, it generates a message which says the unit was not found. Hit any key to return to the Acquisition Phase Main Menu.

**Constellation Age Screen.** Enter a "5" from the Acquisition Phase Main Menu to access the Constellation Age Screen. The players indicate which mission they want to view. This screen gives a health-status check of a mission's constellation. An example of the screen is displayed in Figure 4. The screen displays the name and tail number of all deployed systems (active and spare) of that mission. It also shows the expected lifetime and the number of maneuvers remaining for each satellite. The screen also displays the orbit parameters of each satellite. Hit any key to go back to the main menu.

<table>
<thead>
<tr>
<th>SYSTEM</th>
<th>YEARS # REMAINING</th>
<th>MANEUVERS # REMAINING</th>
<th>ORBIT</th>
<th>INCLINATION</th>
<th>ASCENDING NODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSOC II</td>
<td>4 2</td>
<td>1</td>
<td>GEO</td>
<td>0</td>
<td>120</td>
</tr>
<tr>
<td>DSOC II</td>
<td>2 1</td>
<td>0</td>
<td>GEO</td>
<td>0</td>
<td>65</td>
</tr>
<tr>
<td>DSOC III</td>
<td>1 4</td>
<td>2</td>
<td>GEO</td>
<td>0</td>
<td>-65</td>
</tr>
<tr>
<td>*DSOC III</td>
<td>2 6</td>
<td>2</td>
<td>GEO</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

* Indicates spare status

**Figure 4. Constellation Age Screen**

**Display Orbits/maneuver Sats.** Enter a "6" from the Acquisition Phase Main Menu to access the Display Orbits/Maneuver Sats Screen. The
players enter the orbit they are interested in. The screen shows all
the satellites currently in the requested orbit. An example of the
screen is shown in Figure 5. The screen displays the name and tail
number of all deployed systems (Active and Spare) in that orbit. The
computer will ask the players if they want to place a satellite in a
spare status. The lifetime of a satellite is extended one year for
each year it is in a spare status. The computer will then ask if the
players wish to move a satellite. The players may change any orbit
parameters they desire if the satellite has sufficient fuel on board.

**VLEO**
**ORBIT SCREEN**
* Indicates Spare Status

<table>
<thead>
<tr>
<th>SYSTEM</th>
<th># MANEUVERS</th>
<th>INCLINATION</th>
<th>ASCENDING NODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>EAGLE</td>
<td>2</td>
<td>95</td>
<td>120</td>
</tr>
<tr>
<td>*EAGLE</td>
<td>3</td>
<td>95</td>
<td>0</td>
</tr>
<tr>
<td>SENIRY</td>
<td>4</td>
<td>95</td>
<td>30</td>
</tr>
</tbody>
</table>

Moving SENIRY 4
Enter Desired Orbit (VLEO, LEO, etc.)  VLEO
Enter Desired Inclination (0-95)  95
Enter Desired Ascending Node  (-180..+180)
(−180..+180)
Are these correct (Y/N)?

Figure 5. Display Orbits/Maneuver Sats Screen

Each orbit modification decrements the remaining maneuvers by one.
When the remaining maneuvers reach zero, the satellite cannot perform
any more maneuvers. To return to the main menu, enter "N" when the
program asks if a satellite is to be moved.
Show Mission Ground Swaths. Enter a "7" from the Acquisition Phase Main Menu to access the Mission Ground Swath Display. The players enter the orbit they are interested in. The display shows the ground swath for all active satellites in that mission area. An example of this screen is listed in Figure 6. The screen displays the ground swaths for all the active satellites in the requested mission areas. The ground track represents one orbit of the Earth. The players have an option of adding hypothetical satellites to determine needed coverage. The players hit any key to go back to the main menu.

Quit. Enter a "Q" from Acquisition Phase Main Menu to continue to the Assign Ground Support Phase.

9.0 Assign Ground Support Phase

In this phase, the players assign the necessary ground support elements to the satellites about to be launched. The ground support consists of three different elements: Ground Control, Data Processing, and User Terminals. An example of the Assign Ground Support Screen is displayed in Figure 7. A satellite must have its ground support assigned before that satellite may be launched. The Assign Ground Support Screen displays the satellites which have completed the production phase along with the ground units already assigned to them. The bottom half of the screen shows the active ground units which are available to assign to satellites. The players enter the name and tail number of the ground unit they want to assign. The players then enter the name and tail number of the desired satellite. The players may continue to assign ground units until there are no more available or all the satellites have their ground support elements filled. To
continue on to the Deployment Phase, the players should enter "N" when
the program asks if the player wants to assign more ground units.

The ground units vary in the number of satellites each can
support. Units like the SUDP can provide the Data Processing for all
the satellites of a particular system (DSCS II, Ferret, etc.). Units
like the MUDP can provide the Data Processing for all the satellites of
a particular mission (Comm, NAV, etc.). The CSOC is unique in that it
provides Ground Control and Data Processing.
**SYSTEM CHECK LIST**

For Satellites in Production Status

<table>
<thead>
<tr>
<th>Satellite #</th>
<th>Ground Control System #</th>
<th>Data Processing System #</th>
<th>User Terminal System #</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSCS II 2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DSCS II 3</td>
<td>0</td>
<td>0</td>
<td>SUIT 1</td>
</tr>
</tbody>
</table>

Assign to Satellite: DSCS II
Tail Number: 3

**ASSIGN GROUND SYSTEMS TO SATELLITES**

<table>
<thead>
<tr>
<th>Ground Control</th>
<th>Data Processing</th>
<th>User Terminal</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUGC 2</td>
<td>SUDP 2</td>
<td>SUIT 2</td>
</tr>
<tr>
<td>SUGC 1</td>
<td>SUDP 1</td>
<td></td>
</tr>
</tbody>
</table>

Enter name of ground unit to assign: SUDP
Enter tail number: 2

Figure 7. Assign Ground Support Screen

The players should reference the Resource Catalog for a more detailed description of number and type of units needed to complete the Ground Support Network. A satellite must have its Ground Support Network completed before launch.

10.0 **Deployment Phase**

During this phase, the players match systems designated for launch with launchers and launch pads. The target orbit is set, and the systems are launched. The options available in this phase are listed in Figure 8.

There are two types of launches: single payload and multiple payload. A single payload launch has only one satellite loaded onto
Remaining Budget 4500

YEAR: 1987

[1] Create a Composite Satellite
[2] Start the Launch Process
[3] Display Unit Attributes
[4] Constellation Age Screen
[5] Display Orbit/Maneuver Sats
[6] Show Mission Ground Swaths
[7] Quit to Next Phase

Figure 8. Deployment Phase Main Menu

the lift vehicle. A multiple payload launch consists of a single
launcher with up to four separate payloads.

There are also two types of satellites: composite and regular. A
composite satellite contains systems which have been added on to the
original satellite. A regular satellite does not have any add-on
systems. Survivability factors are not considered add-on systems.

Create a Composite Satellite. Enter a "1" from the Deployment
Phase Main Menu to access the Composite Satellite Screen. The screen
shows all the satellites currently in the "Ready" status (system has
completed the production phase and has its necessary ground support).
The player first enters the "host" satellite. The "host" satellite
must have a larger mass than any of the "parasites" added on. The
screen displays the "host" system and then asks for the "parasite"
system. The "parasite" systems can use many of the sub-units of the
"host"; therefore, the "parasite" systems add only 50% of their mass to
the overall composite satellite's mass. There is a maximum of five "parasites" on a composite satellite. The players should watch the overall system weight as they create composite satellites. They may create a satellite which is too heavy to lift. The "host" system and all its "parasites" are considered as one system once the satellite has been launched. When a composite satellite is maneuvered or is put into a "spare" status, all the systems on board are affected. To get back to the Deployment Phase Main Menu, enter "N" when the program asks if you want to add another system.

**Composite Satellite Example.** The player wishes to add a Single Channel Transponder (SCT) to a GPS-A1 Navigation Satellite. The GPS satellite is considered the host because it has the larger mass. The mass of the GPS satellite is 780 kg, and the mass of the SCT is 20 kg. Creating a composite satellite, adds 50% of the "parasite" mass to the overall system weight. Therefore, 10 kg is added, resulting in an overall system weight of 790 kg.

**Start the Launch Process.** Enter a "2" from the Deployment Phase Main Menu to begin the launch process. The launch process consists of the following actions by the player:

1) Choose Satellite to be launched
2) Choose launch site
3) Choose launcher type
4) Choose launch pad

Examples of these screens are listed in Figures 9, 10, 11, and 12, respectively.
SATELLITES READY FOR LAUNCH

<table>
<thead>
<tr>
<th>MISSION</th>
<th>NAME</th>
<th>NUMBER</th>
<th>ORBIT</th>
<th>INCLINATION</th>
<th>MASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMM</td>
<td>DSCS II</td>
<td>6</td>
<td>GEO</td>
<td>0</td>
<td>780</td>
</tr>
<tr>
<td>WEATHER</td>
<td>DMSP</td>
<td>5</td>
<td>LEO</td>
<td>90</td>
<td>1650</td>
</tr>
<tr>
<td>WEATHER</td>
<td>GOES-D</td>
<td>4</td>
<td>LEO</td>
<td>90</td>
<td>1800</td>
</tr>
<tr>
<td>SIGINT</td>
<td>FERRET-3</td>
<td>4</td>
<td>VLEO</td>
<td>95</td>
<td>1500</td>
</tr>
<tr>
<td>WARNING</td>
<td>SENS</td>
<td>2</td>
<td>GEO</td>
<td>0</td>
<td>1100</td>
</tr>
</tbody>
</table>

Enter the Name of the Satellite to be Launched: DMSP
Tail Number: 5.

Figure 9. Satellites Ready for Launch Screen

LAUNCH SITE SELECTION SCREEN

(N) Indicates remaining launches pad can support this year

<table>
<thead>
<tr>
<th>LAUNCH WINDOW</th>
<th>VANDENBERG</th>
<th>CANAVERAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>68 - 122</td>
<td>TITAN LP # 4 (3)</td>
<td>DELTA LP # 2 (1)</td>
</tr>
<tr>
<td>P</td>
<td>TITAN LP # 5 (2)</td>
<td>STS LP # 2 (3)</td>
</tr>
<tr>
<td>A</td>
<td>ATLAS LP # 1 (4)</td>
<td>LLV LP # 1 (7)</td>
</tr>
<tr>
<td>D</td>
<td>DELTA LP # 1 (1)</td>
<td>ATLAS LP # 2 (0)</td>
</tr>
<tr>
<td>S</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Enter V for Vandenberg or C for Canaveral: V

Figure 10. Choose Launch Site Screen
CHOOSE LAUNCHER FROM VANDENBERG

DMSP # 5  ORBIT = LEO  MASS = 1650

<table>
<thead>
<tr>
<th></th>
<th>Capacity</th>
<th>Satellite</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number</td>
<td>Life</td>
</tr>
<tr>
<td></td>
<td>Launcher</td>
<td>Available</td>
</tr>
<tr>
<td>Atlas</td>
<td>4</td>
<td>8100</td>
</tr>
<tr>
<td>Delta</td>
<td>2</td>
<td>4800</td>
</tr>
<tr>
<td>STS-1</td>
<td>1</td>
<td>26200</td>
</tr>
</tbody>
</table>

Choose a launcher by Name (Atlas, Titan 34D etc.): Atlas
Enter Tail Number: 1

Figure 11. Choose Launcher Screen

CHOOSE LAUNCH PAD TO LAUNCH ATLAS FROM

(N) Indicates Remaining Launches Pad can Support this Year

DMSP # 5  Orbit = LEO  Mass = 1650

Pad Name #

ATLAS LP # 4 (3)
ATLAS LP # 1 (4)

Enter Name of Launch Pad: ATLAS LP
Enter Tail Number of Pad: 1

Figure 12. Choose Launch Pad Screen
The player enters the name and tail number of the satellite to be launched. Once the players have designated a system for launch, they may change the targeted orbital parameters. After the satellite has been launched, it must use its own fuel to maneuver. Therefore, it is important that the correct orbital parameters are set before the system is launched.

After the orbital parameters are set, the player must choose a launch site. Two launch sites are modeled in the game: the Western Test Range (WTR) at Vandenberg, and the Eastern Test Range (ETR) at Cape Canaveral. Each site begins the game with the following pads: one Atlas, two Titan, one Delta, and one STS. Additional pads may be acquired as the game progresses. Each pad can support only a limited number of launches each year, so the player must take care when assigning launch sites. The number of launches per year each pad can support is listed in Table IX.

Table IX
Launch Pad Fire Rates

<table>
<thead>
<tr>
<th>NAME</th>
<th>LAUNCH RATE/YEAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delta</td>
<td>3</td>
</tr>
<tr>
<td>Atlas</td>
<td>2</td>
</tr>
<tr>
<td>Titan</td>
<td>3</td>
</tr>
<tr>
<td>STS</td>
<td>5</td>
</tr>
<tr>
<td>LLV</td>
<td>12</td>
</tr>
<tr>
<td>MLV</td>
<td>9</td>
</tr>
<tr>
<td>HLV</td>
<td>8</td>
</tr>
<tr>
<td>SLV</td>
<td>5</td>
</tr>
</tbody>
</table>

115
Each launch site has an orbit inclination window. The inclination window represents the range of inclinations which payloads may be injected into via a direct ascent. A launch within the launch window is more fuel efficient than a launch outside the window. The window for launches from Vandenberg is 68 - 122 degrees of launch inclination. The window for launches from Cape Canaveral is 29 - 48 degrees of launch inclination. A launch designed to place a payload within the inclination window uses the standard orbit-weight correction factor (function of inclination and booster type). Any launch designed to place a payload in an orbital inclination outside this window is penalized, and its orbit correction factor is assigned as one-half. This represents the additional fuel required to put the satellite into the proper inclination. The orbit correction factor is multiplied by the launcher lift capacity to get the total to-orbit-lift-capacity for the launch. Because of the weight penalty imposed on an out-of-window launch, the player should carefully choose the launch site. See Figure 10 for an example Choose Launch Site Screen.

Once the player has designated a launch site, the computer will display a list of launchers which have the lift capability to place the payload into the desired orbit. This list includes the launcher name and number available, the total lift capacity, the final payload mass, and the remaining lift capacity of the launcher. The remaining lift capacity is used if the player is using a multiple payload launch. The computer tells the players if there are no active launchers which can lift the satellite. See Figure 11 for an example Choose Launcher Screen.
After choosing a launcher, the player must select a launch pad to launch the lift vehicle from. The computer lists all the available launch pads at the launch site which are compatible with the launcher. It also displays the number of remaining launches each pad can support that year. See Figure 12 for an example Choose Launch Pad Screen.

At this point the player has the option of creating a Multiple Launch. If there is sufficient lift capacity remaining, the player may place additional payloads on board the launcher. Multiple Payloads take advantage of common sub-systems required for launching all satellites. Therefore, these extra payloads add only 75% of their mass to the overall mass to be lifted. The Multiple Launch Screen shows the remaining satellites ready for launch and asks for the additional payloads. The computer will not allow the launcher to be overloaded. There can be no more than four separate payloads, and the payloads must be launched into the same orbit (VLEO, MEO, etc.). See Figure 13 for an example Multiple Payload Launch Screen.

Once the payloads have been finalized, the computer asks the NASA player if the launch violates the civilian constraint. This restriction states at least 80% of the civilian launch requirements listed in the scenario must be met during the turn. If NASA says the launch would violate these constraints, the launch is scrubbed and the launch process begins over again. See Figure 14 for an example NASA Check Screen.

If NASA says the launch is okay, then the computer checks the launch reliability factor of the launcher and calls the random number
MULTIPLE PAYLOAD DESIGNATED FOR ATLAS
Multiple Payloads add 75% of their mass to overall system

Target Orbit Remaining Lift Weight Current Payload #

LEO 6450 DMSP 5

REMAINING LAUNCHABLE SATELLITES

MISSION NAME # MASS ORBIT INC ASCENDING NODE

COMM DSCS II 6 780 GEO 0 45
WEATHER GOES-D 4 1800 LEO 90 120
SIGINT FERRET-3 4 1500 VLEO 95 0
WARNING SENS 2 1100 GEO 0 -45

Enter next payload: GOES-D
Tail number: 4

Figure 13. Multiple Payload Launch Screen

NASA, Please Confirm Launch of

DMSP 3

Add-on Packages Additional Payloads
None GOES-D 4

Y/N

Figure 14. NASA Check Screen

generator returns a number between 0 and 1. If this number is greater than the reliability factor of the launcher, then the launch fails and the launcher and all of its payload are considered destroyed. If the launcher was a reusable type, then considerable damage could be done to the overall lift capability. If the launch fails the launch pad is
damaged. The number of remaining launches the pad can support that year is halved, and a repair fee of $10 Million is assessed.

If it is a successful launch, then the launcher and payload are considered to have achieved orbit. At this point the launcher must successfully deploy each of its payloads. There is a uniform 97% chance per payload of a successful deployment (20:40). Again, the computer calls the random number generator. If the number is greater than .97, then the deployment has failed and the payload is considered destroyed. If the number is .97 or less, then the deployment was successful and the payload is considered active. The payloads of a multiple payload launch must be deployed separately. If one payload is not successfully deployed, it does not affect the other systems.

Once the launch process has been completed, the computer decreases the active launchers of that type by one. If the launcher was reusable or recyclable, then its total missions is reduced by one.

Deployment Phase Example. The player wishes to launch a Comstar Communication satellite. The desired orbit is GEO at an inclination of zero degrees. The player chooses to launch the satellite from Vandenberg. The Comstar satellite is to launch by itself without any add-on systems, so its mass is 1410 kg. The computer determines that the Titan 34D is the only active launcher which can lift the satellite. The lift capacity of the Titan 34D to GEO is 4500 kg. However, the inclination is outside the Vandenberg launch window; therefore, the orbit correction factor is one-half. Multiplying this factor by the lift capacity (.5 * 4500) gives a lift capacity of 2250 kg. Therefore, the Titan can launch the system. Loading the Comstar leaves an excess
lift capacity of 840 kg (2250 - 1410). The player also wants to launch a DSCS II Satellite; therefore, he decides to make a multiple launch. The original mass of the DSCS II Satellite (620 kg) is multiplied by .75 since it is a multiple launch. This mass (465 kg) is added to the overall launch weight resulting in a final launch weight of 1875 kg (1410 + 465).

The remaining procedures available in the Deployment Phase have been explained in the Acquisition Phase.

11.0 Scoring Phase (optional)

After every deployment phase, the game controller will assign a point for each mission requirement which has been met during the turn. The mission requirements are defined in the Requirements Catalog. These points are used to evaluate the effectiveness of the force structure designed by the players. The points may be used for comparison purposes when evaluating results from different teams.

12.0 Game Controller

The game controller has access to some extra routines. These procedures are accessed through a secret password. They may be used to create a new database or fix any potential problems with the database. These procedures and a description are listed in Table X. The controller may access these procedures during the Acquisition and Deployment Phase by striking the shift key and the alternate key simultaneously. The password may also be used at the end of every turn when the program asks if the game is to continue.
Table X

System Commands

<table>
<thead>
<tr>
<th>PROCEDURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add_Unit</td>
<td>Used to add units to the database</td>
</tr>
<tr>
<td>Load_List</td>
<td>Used to load a database</td>
</tr>
<tr>
<td>View_List</td>
<td>Used to view the entire database entry by entry</td>
</tr>
<tr>
<td>Delete_Unit</td>
<td>Used to delete a unit from the database</td>
</tr>
<tr>
<td>Change_Status</td>
<td>Used to change the status of a unit</td>
</tr>
<tr>
<td>Add_Copy</td>
<td>Used to add duplicates of a unit (duplicate has different tail number)</td>
</tr>
<tr>
<td>Add_Expendable</td>
<td>Used to add expendable launchers onto the launcher ACB (adds them to the Research stage)</td>
</tr>
<tr>
<td>Modify_Budget</td>
<td>Used to modify the Budget</td>
</tr>
</tbody>
</table>

Follow these steps when creating a new database.

1. Use Add_Unit to create one copy of every unit in the database.
2. Use Add_Copy to create as many of each unit as desired.
3. Use Change_Status to set the desired status for each unit.

Using Add_Copy will greatly reduce the amount of typing necessary to create a new database.

Satrak Interface

The use of the SATRAK program is optional. It is used to display a three-dimensional picture of mission constellations. It can simultaneously display up to ten different satellite orbit graces. The
Game Controller must input the necessary commands into the SATRAK program to generate the display.

This User's Guide does not include instructions on running SATRAK. Refer to the SATRAK User Manual if there are questions.

13.0 Future Programmers

The database was set up as a forward-linked list. The database was designed with variant records because each mission area has many unique fields. This conserves memory space.

The Search procedure was called every time a field had to be changed. The procedure used the global variable "Where" as a pointer to the desired record. For example, the status field of AFSATCOM, tail number 2, needs to be changed to "Development". The program would call Search with the name and tail number as part of the parameter list. The program would then assign the new status as follows:

```
Where^.Status := 'Development'
```

If another mission type is to be added, the following four changes must be made.

1) Add the mission type to the global variable Task in the main program
2) Add the required fields to the data base structure in the main program
3) Add the desired fields to the Add_Unit procedure in the library routine
4) Add the desired fields to the Show_Unit procedure in the library routine
The procedures are not in a logical order because of the memory limitation in Turbo Pascal, Version 3. This ordering allowed a more efficient use of the Overlay command.
APPENDIX E: SOURCE CODE

This appendix contains the source code for ADAM. It consists of approximately 5,000 lines of Turbo Pascal code. Due to the memory limitation of Turbo Pascal, Version 3, modifications to this program should be done in Version 4.
PROGRAM ADAM;
(Code is current as of 5 Dec 87)

TYPE

WorldArray = array [0..16387] of byte;

(different missions modeled in game)
Task = (COMM, WEATHER, NAV, IMINT, SIGINT, WARNING,
OFFENSE, DEFENSE, GROUND, LAUNCHER, PAD);

string10 = string [10];
string5 = string [5];
string80 = string [80];
string11 = string [11];
Window = SET OF 0..180;

UnitPtr = "Unit ;
Unit = Record:
    Mission : string10;
    Name : string10;
    TailNumber : integer;
    Mass : real;
    LifeTime : integer;
    Deployed : integer;
    RDCost : integer;
    ACQCost : integer;
    OMCost : integer;
    Status : string11;
    ControlName : string5;
    ControlNumber : integer;
    ProcessorName : string5;
    ProcessorNumber : integer;
    UserName : string5;
    UserNumber : integer;
    Orbit : string5;
    Inclination : integer;
    AscendingNode : integer;
    AddOnName1 : string10;
    AddOnNum1 : integer;
    AddOnName2 : string10;
    AddOnNum2 : integer;
    AddOnName3 : string10;
    AddOnNum3 : integer;
    AddOnName4 : string10;
    AddOnNum4 : integer;
    AddOnName5 : string10;
    AddOnNum5 : integer;
    AntiJamDataLink : string [1];
    LowAntiJamComLink : string [1];
    MedAntiJamComLink : string [1];
    HighAntiJamComLink : string [1];
Autonomy[15] : string [1];
Autonomy[180] : string [1];
Manuvers : real ;
Update : integer ;
TechLevel : integer ;
Next : UnitPtr ;

CASE Purpose : Task OF

COMM : (Channels : real) ;

WEATHER : (Downlink : string [5] ;
DayVideo : string [1] ;
NightVideo : string [1] ;
DayImage : string [1] ;
NightImage : string [1] ;
Temp : string [1] ;
Solar : string [1] ) ;

NAV : (Accuracy : integer ;
Position : real ;
Redundancy : integer ) ;

IMINT : (DLink : string [5] ;
DVideo : string [1] ;
NVideo : string [1] ;
DImage : string [1] ;
NImage : string [1] ) ;

SIGINT : (DataLink : string [1] ;
Signal : string [5] ) ;

WARNING : (Aircraft : string [5] ;
ICBM : string [1] ;
SLBM : string [1] ;
Target : string [10] ;
Count : string [5] ) ;

OFFENSE : (LowOrbit : string [1] ;
MedOrbit : string [1] ;
HighOrbit : string [1] ;
GroundTarget : string [1] ;
Ships : string [10] ;
TimeUrgent : string [1] ) ;

DEFENSE : (Boost : string [1] ;
MidCourse : string [1] ;
Reentry : string [1] ;
CruiseMissile : string [1] ) ;

GROUND : (Kind : string[10] ;
SupportName : string[10] ;

126
SupportNumber : integer )

LAUNCHER : ( NumActive : integer;
ProdActive : integer;
NumProd : integer;
DevProd : integer;
Hold : integer;
NumDev : integer;
ReuseField : string10;
Sorties : integer;
RemUses : integer;
Reliability : real;
VLEO : real;
LEO : real;
MOL : real;
MEO : real;
GEO : real )

PAD : ( LaunchRate : integer;
NumRemaining : integer;
Location : string10 )

END;

UnitFile = FILE OF Unit ;

CONST
FiveBlanks = ' ';
TenBlanks = ' ' ' ';
FifteenBlanks = ' ' ' ' ' ';

(change this array if different budget figures are to be used)
(4600, 4500, 5000, 6000, 7000, 8000, 7500, 8500, 10000, 10000);

VAR
Ch : Char;
Root : UnitPtr;
Head : UnitPtr ; (points to head of list)
Found : boolean;
Year : integer;
SearchName : string10;
BUDGET : integer;
SearchNumber : integer;
Buyable : boolean;
Copy : boolean;
ThisMission : string10;
Where : UnitPtr;
LaunchSite : string10;
Quit : boolean;
NASAOX: boolean;
LiftVehicle: string10;
LiftVehicleNum: integer;
LiftWeight: real;
RemWeight: real;
Payload1, Payload2, Payload3, Payload4: string10;
Num1, Num2, Num3, Num4: integer;
MultipleLaunch: boolean;
IRText: Text;
World: WorldArray;
LaunchPad: string10;
LaunchPadNum: integer;
Help: boolean;
StringNumber: string10;

{#I graph.p)
{#I Library3.Pas)
{#I Library.Pas)
{#I Library2.pas)

{called from multiple_launch routine. displays all 'ready' sats)

PROCEDURE Display_Remaining_Sats (Var Root: UnitPtr);

VAR
  Current: UnitPtr;
  CountLines: Integer;

BEgin
  Window (1,11,80,25);
  TextBackground (0);
  Clrscr;
  CenterLine (2, 'REMAINING LAUNCHABLE SATELLITES');
  Gotoxy (4,4); Write ('Mission');
  Gotoxy (15,4); Write ('Name *');
  Gotoxy (35,4); Write ('Mass'); Gotoxy (45,4); Write ('Orbit');
  Gotoxy (55,4); Write ('Inc');
  Gotoxy (65,4); Write ('Ascending Node');
  Gotoxy (1,6);
  Current := Root;
  CountLines := 0;
  While (Current <> Nil) AND (CountLines < 8) DO
    BEGIN
      IF (Current^.Mission <> 'PAD')
      AND (Current^.Mission <> 'LAUNCHER') AND
      (Current^.Mission <> 'GROUND') AND
      (Current^.Status = 'READY') THEN
      BEGIN
        Writeln (Current^.Mission:10, Current^.Name:13,
                  Current^.TailNumber:3, 
                  Current^.Mass:6:0,
                  Current^.Orbit:10, Current^.Inclination:9,
      

128

OVERLAY PROCEDURE Multiple_Launch (Root : UnitPtr ; WorkUnit : Unit);
GotoXY (60,4); Write ('Current');
GotoXY (20,5); Write ('Orbit');
GotoXY (40,5); Write ('Lift Weight');
GotoXY (60,5); Write ('Payload *');
TextColor (Yellow);
GotoXY (20,7); Write (WorkUnit.Orbit);
GotoXY (40,7); Write (RemWeight:6:0);

IF Loads = 1 THEN (only done on first pass)
BEGIN
  Keep := Where ;
  HoldMass := WorkUnit.Mass ;
  PayloadI := WorkUnit.Name ;
  Numl := WorkUnit.TailNumber ;
END ;

GotoXY (57,7); Write (Payload1:10, Num1:4);  

IF Loads > 1 THEN
BEGIN
  GotoXY (57,8);  
  Write (Payload2:10, Num2:4);  
END ;

IF Loads > 2 THEN
BEGIN
  GotoXY (57,9);  
  Write (Payload3:10, Num3:4);  
END ;

IF Loads > 3 THEN
BEGIN
  GotoXY (57,10);  
  Write (Payload4:10, Num4:4);  
END ;

IF Loads > 4 THEN
BEGIN
  Center-Line (23,
  'Can only have 4 systems on a multiple launch');
  Center-Line (24, 'Please hit <CR> to continue'); read ;
  TooManyLoads := True ;
END ;

IF NOT TooManyLoads THEN
BEGIN
  Display_Remaining_Sats (Root);
  ClearLines (23,25);
  Center-Line (23, 'enter next payload ');
  Center-Line (24, 'tail number ');
  GotoXY (55,23); Read (SearchName);
  SearchNumber := GetInt (StringNumber, 55, 24);
  Search (Root, SearchName, SearchNumber, 'NO');

130
IF Found THEN
BEGIN
IF (Where^,Orbit = Keep^,Orbit) THEN
BEGIN
ThisPayload := Where^,Mass;
ELSE
BEGIN
ClearLines (23,25);
GotoXY (30,23);
Write (Where^,Name, 'is too heavy');
textcolor (White);
Center_Line (25, 'Please hit (CR) to continue'); read;
TextColor (Yellow);
END;
IF OKLoad THEN
BEGIN
MultipleLaunch := True;
RemWeight := RemWeight - ThisPayload * 0.75;
Loads := Loads + 1;
IF Loads = 2 THEN
BEGIN
PayLoad2 := Where^,Name;
Num2 := Where^,TailNumber;
Where^,Status := 'PENDING';
END;
IF Loads = 3 THEN
BEGIN
PayLoad3 := Where^,Name;
Num3 := Where^,TailNumber;
Where^,Status := 'PENDING';
END;
IF Loads = 4 THEN
BEGIN
PayLoad4 := Where^,Name;
Num4 := Where^,TailNumber;
Where^,Status := 'PENDING';
END;
END; (OKLoad)
END (orbit)
ELSE
BEGIN
ClearLines (23,25);
Center_Line (24, 'Multiple launch orbit altitudes must be identical');
textcolor (white);
Center_Line (25, 'Please hit (CR) to continue'); Read;
textColor (Yellow);
END; (orbit check)
END ; (Found)
END ; (TooManyLoads)
ClearLines (23,25) ; GotoXY (15,24);
Write ('Would you like to place another load on ',
   LiftVehicle, ' (Y/N) ');
Read (kbd,Ch);
IF (Ch = 'N') OR (Ch = 'n') OR (TooManyLoads) THEN Stop := True;
UNTIL Stop;
END;

PROCEDURE DUMMY2 ; (put in because multiple launch and choose launcher)
BEGIN
   (can't be in same overlay)
END;

(player chooses which launcher they want to use)
(player also chooses which launch pad to use)

OVERLAY PROCEDURE Choose_Launcher (Root : UnitPtr ;
VAR WorkUnit : Unit) ;

VAR
   Stop : boolean;
   Keep : UnitPtr;
   Current : UnitPtr;
   Affordable : boolean;
   Check : string [3];
   HavePad : boolean;

BEGIN
   HavePad := False;
   Affordable := True;
   Current := Root;
   Textbackground (1);
   ClrScr;
   Stop := False;
   PayLoad1 := ''; PayLoad2 := ''; PayLoad3 := ''; PayLoad4 := '';
   WITH WorkUnit DO
      BEGIN
         TextColor (White);
         GotoXY (25,2) ; Write ('CHOOSE LAUNCHER FROM ', LaunchSite);
         Textcolor (Yellow);
         GotoXY (19,4) ; Write (Name:10,' #', TailNumber:3,
         ' Orbit =', Orbit:5,
         ' Mass =', Mass:6:0);
         TextColor (White);
         Textbackground (1);
         GotoXY (21,6) ; Write ('Number');
         GotoXY (34,6) ; Write ('Capacity');
         GotoXY (51,6) ; Write ('Satellite');
         GotoXY (67,6) ; Write ('Remaining');
         GotoXY (1,7) ; Write (' Launcher #');
GotoXY (17,7); Write (' Available');
GotoXY (34,7); Write ('Lift Weight');
GotoXY (51,7); Write ('Final mass');
GotoXY (67,7); Write ('Lift Capacity');
TextColor (Yellow);
Display_Launchers (Root, Where);
ClearLines (23,25);
Center_Line (24, 'Enter 'Q' to to stop the launch process');
Read (kbd,Ch);
IF (Ch = 'Q') OR (Ch = 'q') THEN
BEGIN
Stop := True;
NASAOK := False;
Where^.Status := 'READY';
Payload1 := ''; Payload2 := ''; Payload3 := '';
Payload4 := '';
AddOnName1 := ''; AddOnName2 := ''; AddOnName3 := '';
AddOnName4 := ''; AddOnName5 := '';
END;

IF NOT Stop THEN
BEGIN
ClearLines (23,25);
Center_Line (23,
  'Choose a launcher by name (Atlas, Titan 34D etc) ');
Center_Line (24, 'Enter Tail Number ');
GotoXY (65,23); Read (LiftVehicle);
LiftVehicleNum := GetInt (StringNumber, 65, 24);

While (Current <> Nil) AND ((Current^.Name <> LiftVehicle) OR
  (Current^.TailNumber <> LiftVehicleNum)) DO
  Current := Current^.Next;

IF (Current <> Nil) THEN
BEGIN
IF ((Current^.ReuseField <> '')
  AND (Current^.ReuseField <> 'NO') AND
  (Current^.Status = 'ACTIVE') OR
  (Current^.ReuseField = '') OR
  (Current^.ReuseField = 'NO') AND
  (Current^.NumActive > 0)) THEN
BEGIN
IF (Current^.ReuseField = 'REUSEABLE') OR
  (Current^.ReuseField = 'RECYCLABLE') OR
  (Current^.ReuseField = 'R') OR
  (Current^.ReuseField = 'C') THEN
BEGIN
  IF (BUDGET - Current^.OMCost) < 0 THEN
BEGIN
    ClearLines (23,25);
    Center_Line (23, 'Sorry, you cant afford to launch !');
    Center_Line (25, 'Hit any key to continue');
    Read (kbd,Ch);
  END;
END;

133
Status := 'READY';
Affordable := False;
END; (exceeds budget)

END; (resuable)

(Pick launch pad)

IF Affordable THEN  (player designates pad)
BEGIN
ClearLines (1,25); GotoXY (24,2); TextColor (White);
WRITE ('CHOOSE PAD TO LAUNCH ', LiftVehicle, ' FROM');
GotoXY (11,4); TextColor (Yellow);
WRITE ('(N) Indicates Remaining Launches Pad');
WRITE ('can Support This Year');
TextColor (White); GotoXY (19,6);
WRITE (Name:10, ' ', TailNumber:3,
' Orbit = ', Orbit:5, ' Mass = ', Mass:6:0);

Current := Root;
Center_Line (9, 'Pad Name ');
GotoXY (1,11);
Check := LiftVehicle;  (get first three letters)
(to check if launcher is same type)
(as pad)

While Current <> Nil DO
BEGIN
IF (Current^.Mission = 'PAD') AND
(Current^.Status = 'ACTIVE') AND
(Current^.Location = LaunchSite) AND
(Pos (Check, Current^.Name ) <> 0) AND
(Current^.NumRemaining > 0) THEN
WRITE (Current^.Name:41, Current^.TailNumber:3,
' ', Current^.NumRemaining:2, ')');

Current := Current^.Next;
END;

Center_Line (24 , 'Enter Name of Launch Pad ');
Center_Line (25, 'Enter Tail Number of Pad ');
GotoXY (55,24); Read (LaunchPad);
LaunchPadNum := GetInt (StringNumber, 55, 25);
Current := Root; (check if its in database)

While (Current <> Nil) AND ((Current^.Name <> LaunchPad) OR
(Current^.TailNumber <> LaunchPadNum)) DO
Current := Current^.Next;

IF Current = Nil THEN
BEGIN
ClearLines (24,25);
Center_Line (24, 'Pad is not in the database');
Center_Line (25, 'Hit any key to continue');
Read (kbd,Ch);

134
Where^.Status := 'READY';
END
ELSE (found)
BEGIN  (check LP compatibility)
  IF (Current^.NumRemaining > 0) AND
  (Pos (Check, LaunchPad) <> 0)
  THEN HavePad := True;
END;

IF (NOT HavePad) AND (Current <> Nil) THEN
BEGIN
ClearLines (24,25);
Center_Line (24, 'Sorry Pad is out of launches or not compatible');
Center_Line (25, 'Hit any key to continue'); Read.(kbd,Ch);
Where^.Status := 'READY';
END;

IF HavePad THEN
BEGIN
ClearLines (23,25);
Keep := Where;
Center_Line (24, 'Will this be a multiple payload launch (Y/N) ? ');
 IF Yes THEN Multiple_Launch (Root, Where');
NASA_Check (Keep');
END; (have pad)

END; (affordable)
END (Reusable not active or expendable numactive = 0)
ELSE Where^.Status := 'READY';
END (not nil. check if Launcher in Database)
ELSE Where^.Status := 'READY';

END; (not stop)

END ; (with)

END ;

(shows status of pads at both sites and players inputs which site)
(to launch from)

OVERLAY PROCEDURE Choose_Launch_Site (Root : UnitPtr) ;

Var
  Current : UnitPtr;
  CapeLine, VandLine : integer;

BEGIN
  ClrScr;
  Center_Line (1, 'LAUNCH SITE SELECTION SCREEN');

135
GotoXY (11,2); Write ('(N) Indicates remaining launches pad can support this year'); TextColor (White); GotoXY (25,4); Write ('VANDENBERG'); GotoXY (56,4); Write ('CANAVERAL'); TextColor (Yellow); GotoXY (10,5); Write ('LAUNCH'); GotoXY (10,6); Write ('WINDOW'); GotoXY (26,6); Write ('68 - 122'); GotoXY (57,6); Write ('29 - 48'); TextColor (White); GotoXY (12,9); Write ('P'); GotoXY (12,11); Write ('A'); GotoXY (12,13); Write ('D'); GotoXY (12,15); Write ('S'); TextColor (Yellow); GotoXY (1,9); VandLine := 9; CapeLine := 9; Current := Root; While (Current <> Nil) AND (Current^.Mission <> 'PAD') DO Current := Current^.Next; While Current^.Mission = 'PAD' DO BEGIN IF (Current^.Location = 'VANDENBERG') AND (Current^.Status = 'ACTIVE') THEN BEGIN GotoXY (20, VandLine); Write (Current^.Name:10, ' #', Current^.TailNumber:3, (' ', Current^.NumRemaining:2, ')'); VandLine := VandLine + 1; END; IF (Current^.Location = 'CANAVERAL') AND (Current^.Status = 'ACTIVE') THEN BEGIN GotoXY (50, CapeLine); Write (Current^.Name:10, ' #', Current^.TailNumber:3, (' ', Current^.NumRemaining:2, ')'); CapeLine := CapeLine + 1; END; Current := Current^.Next; End; (mission = pad) REPEAT CenterLine (25, 'enter V for Vandenberg or C for Cape Canaveral '); Read (kbd, Ch); CASE Ch OF 'V', 'v' : LaunchSite := 'VANDENBERG'; 'C', 'c' : LaunchSite := 'CANAVERAL'; END;
UNTIL Ch IN ['V', 'V', 'C', 'c'] ;

END ;

(player chooses which satellite to launch from list of all Ready sats)
(allow player to change target orbit parameters before launch)

OVERLAY PROCEDURE Choose_Launch_Sat (VAR Root : UnitPtr) ;

VAR
  Current : UnitPtr ;
  CountLines : integer ;
  Stop : boolean ;

BEGIN
  ClrScr ;
  Stop := False ;
  Current := Root ;
  GotoXY (24,2) ; Write ('SATELLITES READY FOR LAUNCH') ;
  GotoXY (1,4) ; Write ('Mission') ; GotoXY (20,4) ; Write ('Name') ;
  GotoXY (30,4) ; Write ('Tail Number') ;
  GotoXY (45,4) ; Write ('Orbit') ;
  GotoXY (55,4) ; Write ('Inclination') ;
  GotoXY (72,4) ; Write ('Mass') ;
  GotoXY (1,6) ;
  CountLines := 0 ;

  WHILE (Current <> Nil) AND (CountLines < 17) DO
      BEGIN
        IF (Current^.Mission <> 'GROUND') AND
            (Current^.Mission <> 'LAUNCHER')
            AND (Current^.Mission <> 'PAD') AND
            (Current^.Status = 'READY') THEN
          BEGIN
            Writeln (Current^.Mission:10, Current^.Name:15,
                      Current^.TailNumber:11,
                      Current^.Orbit:12, Current^.Inclination:12,
                      TenBlanks, Current^.Mass:5:0) ;

            CountLines := CountLines + 1 ;
          END ;

        Current := Current^.Next ;
      END ; (not nil and lines < 17)

      (IF CountLines >= 17 THEN
        BEGIN
          ClearLines (23,25) ;
          Center_Line (24, 'Hit 'C' to continue listing') ;
          IF Continue THEN Choose_Launch_Sat (Current) ;
        END ;)

137
Enter the name of the satellite to be launched:

SearchName := GetString (StringNumber, 65, 24);
Search (Root, SearchName, SearchNumber, 'YES')
IF Found THEN
BEGIN

IF Where^.Status = 'READY' THEN  {Verify Orbit Parameters}
BEGIN

WHERE^.Status := 'PENDING';
ClsScr;
TextBackground (1);
ClrScr;
Border1;
TextBackground (1);
GotoXY (20,6);
TextColor (white);
Write ('Projected Orbit Parameters for ', Where^.Name, ':
', Where^.TailNumber);
TextColor (Yellow);
GotoXY (26,10);
Write ('Orbit: ', Where^.Orbit);
GotoXY (26,12);
Write ('Inclination: ', Where^.Inclination);
GotoXY (26,14);
Write ('Ascending Node: ', Where^.AscendingNode);
REPEAT
CenterLine (20, 'Do you want to change these parameters (Y/N) ? ');
IF Yes THEN:
BEGIN

GotoXY (41,10); Write ('');
GotoXY (41,12); Write ('');
GotoXY (41,14); Write ('');
GotoXY (42,10); Read (Where^.Orbit);
Where^.Inclination := GetInt (StringNumber, 42,12);
Where^.AscendingNode := GetInt (StringNumber, 42,14);
END
ELSE Stop := True;
UNTIL Stop;
END {status is LAU}
ELSE  {Status is not LAU}
BEGIN

ClearLines (23,25);
Gotoxy (20,23);
Write (Where^.Name, ' is not Ready For Launch, enter another');
CenterLine (25, 'Please hit <CB> to continue'); Read;
END

END; (Found)
END.
OVERLAY PROCEDURE Composite_Satellite (Root : UnitPtr);

VAR
ThisMass : real;
HoldMass : real;
Parasite : integer;
Keep : UnitPtr;

BEGIN
Parasite := 0;
ClearLines (23,25);
Center_Line (23, 'Enter name of the Host Satellite ');
Center_line (24, 'Tail Number ');
GotoXY (60,23); Read (SearchName);
SearchNumber := getInt (StringNumber, 60,24);
Search (Root, SearchName, SearchNumber, 'NO');
IF Found THEN
begin
Keep := Where;
HoldMass := Where'.Mass;
ThisMass := Where'.Mass;
REPEAT
GotoXY (32,4); Write ('Total Mass ', ThisMass:5:0);
ClearLines (23,25);
Center_line (23, 'Enter The name of the Add-On System ');
Center_line (24, 'Tail Number ');
GotoXY (60,23); Read (SearchName);
SearchNumber := getInt (StringNumber, 60,24);
Search (Root, SearchName, SearchNumber, 'NO');
IF Found THEN
begin
IF Where'.Mass > HoldMass THEN (host system must be largest)
BEGIN
ClearLines (23,25);
Center_Line (23, 'Add-On System is heavier than host satellite!! REDO !!');
Center_Line (24, 'Please hit <CR> to continue'); Read;
END
ELSE
BEGIN (ok to add on)
Parasite := Parasite + 1;
ThisMass := ThisMass + 0.5 * Where'.Mass;
Where'.Status := 'Parasite';
Keep'.Mass := ThisMass;
Composite_Satellite_Header (Root);
GotoXY (32,4); Write ('Total Mass ', ThisMass:5:0);
END

139
IF Parasite = 1 THEN
BEGIN
  Keep^.AddOnName1 := Where^.Name ;
  Keep^.AddOnNum1 := Where^.TailNumber ;
END ;

IF Parasite = 2 THEN
BEGIN
  Keep^.AddOnName2 := Where^.Name ;
  Keep^.AddOnNum2 := Where^.TailNumber ;
END ;

IF Parasite = 3 THEN
BEGIN
  Keep^.AddOnName3 := Where^.Name ;
  Keep^.AddOnNum3 := Where^.TailNumber ;
END ;

IF Parasite = 4 THEN
BEGIN
  Keep^.AddOnName4 := Where^.Name ;
  Keep^.AddOnNum4 := Where^.TailNumber ;
END ;

IF Parasite = 5 THEN
BEGIN
  Keep^.AddOnName5 := Where^.Name ;
  Keep^.AddOnNum5 := Where^.TailNumber ;
END ;

IF Parasite > 5 THEN
BEGIN
  ClearLines (23,25) ;
  Center_Line (24, 'Sorry, a max of five sub-systems on a satellite ') ;
  Center_Line (25, ' hit any key to continue') ; Read (kbd,Ch) ;
  TextColor (Yellow) ;
END ;

END ; (weight comparision check)

END ; (2nd Found)

ClearLines (23,25) ;
GotoXY (15,24) ;
Write ('Do you want to place another system on ', Keep^.Name; ' (Y/N) ? ') ;

UNTIL NOT Yes ;

END ; (1st Found)

END ;
(shows all satellites currently in designated orbit (active and spare))
(lets player put sat into spare status or maneuver sat, must have fuel)
(to maneuver, each maneuver burns one fuel load)

OVERLAY PROCEDURE Orbit_Screen (VAR Root : UnitPtr;
            ThisOrbit : string5);

VAR
  CountLines : integer;
  Current : UnitPtr;
  HaveFuel : boolean;
  ChangeOrbit : boolean;
  NewOrbit : string5;
  NewInclination : integer;
  NewNode : integer;
  Spare : string[2];
  ThisName : string[12];

BEGIN
  Spare := '*';
  HaveFuel := False;
  ChangeOrbit := False;
  Current := Root;
  Textbackground (1);
  ClrScr;
  CenterLine (2, ThisOrbit);
  CenterLine (3, 'ORBIT SCREEN');
  CenterLine (4, '* Indicates Spare Status');
  GotoXY (7,8); Write ('SYSTEM');
  GotoXY (28,8); Write ('FUEL');
  GotoXY (38,8); Write ('INCLINATION');
  GotoXY (60,0); Write ('ASCENDING NODE');
  GotoXY (7,8);
  CountLines := 8;

  While (Current <> Nil) AND (CountLines < 23) DO
  BEGIN
    IF (Current^.Orbit = ThisOrbit) AND
        (Current^.Mission <> 'LAUNCHER') AND
        (Current^.Mission <> 'GROUND') AND
        (Current^.Mission <> 'PAD') AND
        ((Current^.Status = 'ACTIVE') OR
         (Current^.Status = 'SPARE')) THEN
      BEGIN
        IF Current^.Status = 'ACTIVE'
          THEN ThisName := Current^.Name;
        IF Current^.Status = 'SPARE'
          THEN ThisName := Spare + Current^.Name;

        WriteLn (ThisName:12, Current^.TailNumber:6, TenBlanks,
Current^Manuvers:2:0, Current^Inclination:14,
Current^AscendingNode:24);
CountLines := CountLines + 1;
END;
Current := Current^Next;
END;

(IF CountLines >= 23 THEN
BEGIN
ClearLines (23, 25);
Center_Line (24, 'Hit 'C' to continue listing ');
IF Continue THEN Orbit_Screen (Current, ThisOrbit);
END);
ClearLines (23, 25);
Center_Line (23, 'Activate a spare or put a system into spare status (Y/N)' );
IF Yes THEN
BEGIN
ClearLines (23, 25);
Center_Line (23, 'Enter Satellite Name ');
Center_Line (24, 'Enter Satellite Number ');
Gotoxy (55, 23); Read (SearchName);
SearchNumber := GetInt (StringNumber, 55, 24);
Search (Root, SearchName, SearchNumber, 'NO');
IF Found THEN
BEGIN
IF Where^.Status = 'ACTIVE' THEN
BEGIN
Where^.Status := 'SPARE';
END;
ELSE Where^.Status := 'ACTIVE';
END; (Found)
END; (yes)
ClearLines (23, 25);
Center_Line (23, 'DO you wish to change the orbit of a satellite (Y/N) ? ');
IF Yes THEN ChangeOrbit := True;

IF ChangeOrbit THEN
BEGIN
ClearLines (23, 25);
Center_Line (23, 'Enter name of satellite ');
Center_Line (24, 'which tail number ');
Gotoxy (55, 23); Read (SearchName);
SearchNumber := GetInt (StringNumber, 55, 24);
Search (Root, SearchName, SearchNumber, 'NO');
IF Found THEN
BEGIN
IF Where^.Manuvers >= 1 THEN HaveFuel := True;
IF (HaveFuel) AND (Where^.Status = 'ACTIVE') THEN
BEGIN
REPEAT
ClearLines (21,25); GotoXY (32,21);
Write ('Moving ', Where^.Name, Where^.TailNumber:3);
Center_Line (22, 'Enter desired orbit (VLEO, LEO, MOL etc) ');
Center_Line (23, 'Enter desired inclination (0 - 95) ');
Center_Line (24, 'Enter desired ascending node (-180 .. +180)');
GotoXY (65,22); Read (NewOrbit);
NewInclination := GetInt (StringNumber, 65,23);
NewNode := GetInt (StringNumber, 65,24);
Center_Line (25, 'Are these correct (Y/N) ?? ');
UNTIL Yes ;
Where^.Orbit := NewOrbit ;
Where^.Inclination := NewInclination ;
Where^.AscendingNode := NewNode ;
END
ELSE
BEGIN
ClearLines (23,25);
Center_Line (24, 'NOT ENOUGH FUEL OR Satellite not active');
TextColor (White);
Center_Line (25, 'hit any key to continue'); Read (kbd, Ch);
TextColor (Yellow);
END ; (fuel check/ active status)
END ; (Found)
END ; (change orbit)
END ;

{gives health status check of a constellation. shows orbit parameters}
(and the expected lifetime left in each sat)

OVERLAY PROCEDURE System_Age (VAR Root : UnitPtr ;
ThisMission : string10) ;

VAR
Age, Remaining : integer ;
Current : UnitPtr ;
ThisName : string [12] ;
CountLines : integer ;
BEGIN:
textbackground (1) ;
ClrScr ;

143
Spare := '*';
Center_Line (2, ThisMission);
GotoXY (25,3); Write ('CONSTITUTION AGE SCREEN ', Year:4);
GotoXY (28,4); Write ('* indicates spare status');
TextColor (White);
GotoXY (20,6); Write ('YEARS');
GotoXY (32,6); Write ('MANEUVERS');
GotoXY (70,6); Write ('ASCENDING');
GotoXY (5,7); Write ('SYSTEM *');
GotoXY (20,7); Write ('REMAINING');
GotoXY (32,7); Write ('REMAINING');
GotoXY (46,7); Write ('ORBIT');
GotoXY (55,7); Write ('INCLINATION');
GotoXY (73,7); Write ('NODE');
TextColor (Yellow);
GotoXY (1,9);
CountLines := 9;
Current := Root;
While (Current <> Nil) AND (Current^.Mission <> ThisMission) DO
  Current := Current^.Next;
While (Current <> Nil) AND (Current^.Mission = ThisMission) AND
  (CountLines < 23) DO
BEGIN
  IF (Current^.Status = 'ACTIVE') OR
     (Current^.Status = 'SPARE') THEN
    BEGIN
      IF Current^.Status = 'ACTIVE'
      THEN ThisName := Current^.Name;
      IF Current^.Status = 'SPARE'
      THEN ThisName := Spare + Current^.Name;
      Age := Year - Current^.Deployed;
      Remaining := Current^.LifeTime - Age;
      Writeln (ThisName:12, Current^.TailNumber:3, Remaining:9, TenBlanks, ', ', Current^.Manuvers:2:0, Current^.Orbit:11, Current^.Inclination:12, Current^.AscendingNode:14);
      CountLines := CountLines + 1;
    END; (status is active or spare)
  END; (thismission)
END; (thismission)

{ IF CountLines >= 23 THEN
BEGIN
  ClearLines (23,25);
  Center_Line (24, 'Hit 'C' to continue listing');
  IF Continue THEN System_Age (Current, ThisMission);
END; }

144
allows players to see ground swaths for a mission

OVERLAY PROCEDURE Ground_Track (Root : UnitPtr);

VAR
Current : UnitPtr;
ThisMission : string10;
Quit : boolean;
Orbit : string5;
Inc : integer;
Ascending : integer;
Stop : boolean;

BEGIN
Stop := False;
Quit := False;
TextBackground (1);
ClrScr;
Border1;
TextBackground (1);
TextColor (White);
Center_line (3, 'GROUND SWATH SCREEN'):
TextBackground (1);
Center_line (8, 'NAV'):
Center_line (10, 'WEATHER'):
Center_line (12, 'IMINT'):
Center_line (14, 'SIGINT'):
Center_line (16, 'ALARM'):
Center_line (18, 'QUIT'):
Center_line (23,
  'Enter which mission you want to see ground swaths of'):

Gotoxy (40,24); Read (kbd,Ch);

CASE Ch OF
  'C', 'c' : ThisMission := 'COMM';
  'N', 'n' : ThisMission := 'NAV';
  'W', 'w' : ThisMission := 'WEATHER';
  'I', 'i' : ThisMission := 'IMINT';
'S', 's': ThisMission := 'SIGINT';

'A', 'a': ThisMission := 'WARNING';

'Q', 'q': Quit := True;

End; (Case)

If Not Quit Then
  Begin
    Current := Root;
    GraphColorMode;
    PutPic (World, 0, 200);
    GotoXY (10, 1); Write (ThisMission, ' Ground Swaths');

    While (Current <> Nil) Do
      Begin
        If (Current^.Mission = ThisMission) AND
            (Current^.Status = 'ACTIVE')
          Then Trace (Current^.Orbit, Current^.Inclination,
                          Current^.AscendingNode, Current^.Mission);
        Current := Current^.Next
      End; (While)
  End; (Not Quit)
End; (Not Quit)

Repeat
  GotoXY (1, 25); Write ('Add a Hypothetical Satellite (Y/N) ? '); If Yes Then
    Begin
      GotoXY (1, 25);
      Write ('Enter Orbit (VLEO, MEO etc)');
      GotoXY (30, 25); Read (Orbit);
      GotoXY (1, 25);
      Write ('Enter Inclination (0-95)');
      GotoXY (30, 25);
      Inc := GetInt (StringNumber, 30, 25);
      GotoXY (1, 25);
      Write ('Enter Ascending Node (-180..+180)');
      GotoXY (35, 25);
      Ascending := GetInt (StringNumber, 35, 25);
      Trace (Orbit, Inc, Ascending, ThisMission);
    End
Else Stop := True;

Until Stop;

TextColor (White);
GotoXY (1, 25); Write ('hit any key to continue');
read (kbd, Ch);
TextColor;
End;

(checks ground units)
(for multiple satellite capability and automatically assigns ground units to the appropriate satellites)

OVERLAY PROCEDURE Assign_Support (Root : UnitPtr) ;

VAR
  CountLines : integer ;
  Assign : boolean ;
  Current, Keep : UnitPtr ;
  ThisName : string10 ;
  ThisNumber : integer ;
  TempName : string10 ;
  TempNumber : integer ;
  Catagory : string10 ;

BEGIN
  CountLines := 0 ;
  ClrScr ;
  REPEAT
    Display_System_CheckList (Root) ;
    Window (1,14,80,25) ;
    Textbackground (0) ;
    ClrScr ;
    Assign := True ;
    Current := Root ;
    Textcolor (white)' ;
    Center Line (1, 'ASSIGN GROUND SYSTEMS TO SATELLITES') ;
    GotoXY (5,3) ; Write ('Ground Control') ;
    GotoXY (30,3) ; Write ('Data Processor') ;
    GotoXY (55,3) ; Write ('User Terminal') ;
    GotoXY (1,5) ;
    Catagory := 'CONTROL' ;
    Count_Ground (Root, Catagory) ;
    Catagory := 'PROCESSOR' ;
    Count_Ground (Root, Catagory) ;
    Catagory := 'USER' ;
    Count_Ground (Root, Catagory) ;

    Center Line (12,
      'Enter "Q" to stop assigning ground units or <CR> to continue') ;
    Read (kbd,Ch) ;
    IF (Ch = 'Q') OR (Ch = 'q') THEN Assign := False ;
    IF Assign THEN
      BEGIN
        Window (1,1,80,25) ;
        ClearLines (24,25) ;
        Center Line (24, 'Enter name of the ground unit to assign ') ;
        Center Line (25, 'enter tail number ') ;
        GotoXY (60,24) ; Read (SearchName) ;
        SearchNumber := GetInt (StringNumber, 60,25) ;
        Search (Root, SearchName, SearchNumber, 'NO') ;
        IF Found THEN

147
begin
Keep := Where :

Textbackground (1) ;
Center_Line (12, 'assigned to which satellite') ;
Center_Line (15, 'which tail number') ;
Gotoxy (55,12) ; Read (SearchName) ;
SearchNumber := GetInt (StringNumber, 55,13) ;
ClearLines (12,13) ;
Search (Root, SearchName, SearchNumber, 'NO') ;
IF Found THEN
begin
TempName := Keep^.Name ;
TempNumber := Keep^.TailNumber ;

IF Keep^.Kind = 'CONTROL' THEN
BEGIN
IF (Keep^.Name = 'SUGCT') OR (Keep^.Name = 'SUMGCT') THEN
BEGIN
Current := Root ;
While (Current <> Nil) DO
BEGIN
IF (Current^.Name = Where^.Name) AND
(Where^.Orbit <> 'GEO') THEN
BEGIN
Current^.ControlName := Keep^.Name ;
Current^.ControlNumber := Keep^.TailNumber ;
END ;
Current := Current^.Next ;
END ;
END ;
Where^.ControlName := TempName ;
Where^.ControlNumber := TempNumber ;
Keep^.SupportName := Where^.Name ;
Keep^.SupportNumber := Where^.TailNumber ;
END ;

IF Keep^.Kind = 'PROCESSOR' THEN
BEGIN
IF (Keep^.Name = 'SUDP') OR (Keep^.Name = 'SUMDP') THEN
BEGIN
Current := Root ;
While (Current <> Nil) DO
BEGIN
IF (Current^.Name = Where^.Name) AND
(Where^.Orbit <> 'GEO') THEN
BEGIN
Current^.ProcessorName := Keep^.Name ;
Current^.ProcessorNumber := Keep^.TailNumber ;
END ;
Current := Current^.Next ;
END ;
END;

IF (Keep^.Name = 'MUDP') OR (Keep^.Name = 'MUMDP') THEN
BEGIN
  Current := Root;
  While Current <> Nil DO
  BEGIN
      BEGIN
        Current^.ProcessorName := Keep^.Name;
        Current^.ProcessorNumber := Keep^.TailNumber;
      END;
      Current := Current^.Next;
  END;
END;

Where^.ProcessorName := TempName;
Where^.ProcessorNumber := TempNumber;
Keep^.SupportName := Where^.Name;
Keep^.SupportNumber := Where^.TailNumber;
END;

IF Keep^.Kind = 'USER' THEN
BEGIN
  IF Keep^.Name = 'SUUT' THEN
  BEGIN
    Current := Root;
    While Current <> Nil DO
    BEGIN
      IF Current^.Name = Where^.Name THEN
        BEGIN
          Current^.UserName := Keep^.Name;
          Current^.UserNumber := Keep^.TailNumber;
        END;
        Current := Current^.Next;
    END;
  END;(SUUT)

  IF Keep^.Name = 'MUUT' THEN
  BEGIN
    Current := Root;
    While Current <> Nil DO
    BEGIN
        BEGIN
          Current^.UserName := Keep^.Name;
          Current^.UserNumber := Keep^.TailNumber;
        END;
        Current := Current^.Next;
    END;(name = MUUT)

  END;
END;

Where^.UserName := TempName;
Where^.UserNumber := TempNumber;
Keep^.SupportName := Where^.Name;
Keep^.SupportNumber := Where^.TailNumber;
END;

END ; (round)
END ; (foundl)
END ; (assign)

UNTIL NOT Assign;

Window (1, 1, 80, 25);
END;

PROCEDURE Constellation_Age_Screen (Root : UnitPtr);
VAR
Quit : boolean;
BEGIN
Quit := False;
REPEAT
TextBackground (1);
ClrScr;
Border1;
textbackground (1);
textcolor (white);
Center_LINE (3, 'CONSTELLATION AGE SCREEN');
TextBackground (1);
Center_LINE (6, '[COMM CONSTELLATION']);
Center_LINE (8, '[W]EATHER CONSTELLATION']);
Center_LINE (10, '[N]AV CONSTELLATION');
Center_LINE (12, '[I]MINT CONSTELLATION');
Center_LINE (14, '[SIGINT CONSTELLATION']);
Center_LINE (16, '[WARNING CONSTELLATION']);
Center_LINE (18, '[QUIT TO MAIN MENU']);
TextColor (White);
Center_LINE (23, 'Choose the constellation you want to see');
TextColor (Yellow);
GotoXY (40, 24);
Read (kbd, Ch);
Case Ch OF
'C', 'c' : System_Age (Root, 'COMM');
'W', 'w' : System_Age (Root, 'WEATHER');
'N', 'n' : System_Age (Root, 'NAV');
'I', 'i' : System_Age (Root, 'IMINT')
'S', 's' : System_Age (Root, 'SIGINT')
'A', 'a' : System_Age (Root, 'WARNING')
'Q', 'q' : Quit := True

END ; (Case)
Until Quit ;

END ;

PROCEDURE Choose_Orbit_Screen (Root : UnitPtr);

VAR
ThisOrbit : string5 ;
Quit : boolean;

BEGIN

REPEAT
Quit := False;
Textbackground (1);
ClrScr;
Border1;
Textbackground (1);
TextColor (White);
Center_Line (3, 'TABULAR ORBIT SCREEN') ;
GotoXY (35, 5) ; Write ('YEAR ', Year:4);
TextBackground (1);
CenterLine (8, '[V]L EO ORBIT ') ;
CenterLine (10, '[L]EO ORBIT ') ;
CenterLine (12, '[M]EO ORBIT ') ;
CenterLine (14, '[O]L ORBIT ') ;
CenterLine (16, '[G]EO ORBIT ') ;
CenterLine (18, '[Q]uit to MAIN MENU') ;
CenterLine (21, 'Enter which orbit screen you want to see ') ;
GotoXY (40, 23) ;
Read (kbd, Ch) ;
CASE Ch OF
'V', 'v' : Orbit_Screen (Root, 'VLEO') ;
'L', 'l' : Orbit_Screen (Root, 'LEO') ;
'M', 'm' : Orbit_Screen (Root, 'MEO') ;
'O', 'o' : Orbit_Screen (Root, 'MOL') ;
'G', 'g' : Orbit_Screen (Root, 'GEO') ;
'Q', 'q' : Quit := True ;

END ; (CASE)

UNTIL Quit ;

END ;

(main deployment procedure)

PROCEDURE Match_Payloads_With_Launchers (Root : UnitPtr) ;

VAR
  Quit : boolean ;

BEGIN
  Quit := False ;
  ClrScr ;
  TextBackground (1) ;
  Border1 ;
  TextBackground (1) ;
  Center_Line (12, 'DEPLOYMENT PHASE') ;
  Delay (2000) ;

  REPEAT
    TextBackground (1) ;
    ClrScr ;
    Border1 ;
    TextBackground (1) ;
    Textcolor (White) ;
    GotoXY (29,3) ; Write ('Remaining Budget', BUDGET:S) ;
    GotoXY (35,4) ; Write ('Year', Year:S) ;
  textBackground(1) ;
  Center_Line (7, '[1] Create a Composite Satellite') ;
  Center_Line (9, '[2] Start the Launch Process') ;
  Center_Line (11, '[3] Display Unit Attributes') ;
  Center_Line (13, '[4] Display Orbits/Maneuver Satellites') ;
  Center_Line (15, '[5] Constellation Age Screen') ;
  Center_Line (17, '[6] Show Ground Traces') ;
  Center_Line (19, '[Q] Quit to Next Phase') ;
  GotoXY (40,20) ;
  Read (kbd, Ch) ;

  CASE Ch OF
    '1' : BEGIN
      IF Help THEN Help_Screen (4) ;
      Composite_Satellite_Header (Root) ;
      Composite_Satellite (Root) ;
      END ;
'2' : BEGIN
  IF Help THEN Help_Screen (5) ;
  Choose_Launch_Site (Root) ;
  IF (Found) AND ((Where^.Status = 'READY') OR
  (Where^.Status = 'PENDING')) THEN
    BEGIN
      Choose_Launch_Site (Root) ;
      Choose_Launcher (Root, Where^) ;
      IF WASAOK THEN Launch (Root, Where^) ;
      END ;
    END ;
  END ;

'3' : BEGIN
  ClrScr ;
  Center_line (12, 'enter name of unit '){
  'Center_line (14, 'enter tail number '){
  GotoXY (50, 12) ; Read (SearchName) ;
  SearchNumber := GetInt (StringNumber, 50, 14) ;
  Search (Root, SearchName, SearchNumber, 'YES') ;
  END ;

'4' : Choose_Orbit_Screen (Root) ;

'5' : Constellation_Age_Screen (Root) ;

'6' : Ground_Track (Root) ;

#27 : BEGIN
  IF KeyPressed THEN
    BEGIN
      Read (kbd, Ch) ;
      IF Ch = #15 THEN System_Level_Commands (Root) ;
      END
    END ;

'Q', 'q' : Quit := True ;

END ;
UNTIL Quit ;
END ;

(main/acq proc)
PROCEDURE Update_Units (VAR Root : UnitPtr) ;
VAR
Quit : boolean;

BEGIN
  Quit := False;
  Textbackground (1);
  ClrScr;
  Border1;
  TextBackground (1);
  Center_Line (12, 'ACQUISITION PHASE');
  Delay (2000);
REPEAT
  Textbackground (1);
  clrscr;
  Border1;
  TextBackground (1);
  TextColor (White);
  GotoXY (29,3); Write ('Remaining Budget ', BUDGET:6);
  GotoXY (35,4); Write ('Year ', Year:6);
  textbackground (1);
  Center_Line (7, '[1] Display Satellite ACB Screen');
  Center_Line (8, ' AND Resuable Launchers.');
  Center_Line (10, '[2] Display Launcher ACB Screen');
  Center_Line (12, '[3] Add Survivability');
  Center_Line (14, '[4] Display Unit Attributes');
  Center_Line (16, '[5] Constellation Age Screen');
  Center_Line (18, '[6] Display Orbits/Maneuver Sats');
  Center_Line (20, '[7] Show Ground Traces');
  Center_Line (22, '[Q] Quit to Next Phase');
  GotoXY (40,24);
  Read (kbd,Ch);
CASE Ch OF
  '1' : BEGIN
    IF Help THEN Help_Screen (1);
    Display_Satellite_ACX_Screen (Root);
    END;

  '2' : BEGIN
    IF Help THEN Help_Screen (2);
    Display_ExpendsableLauncher_ACX_Screen (Root);
    END;

  '3' : BEGIN
    IF Help THEN Help_Screen (3);
    Add_Survivability (Root);
    END;

  '4' : BEGIN
    ClrScr;
    Center_Line (12, 'enter name of unit ');
    Center_Line (14, 'enter tail number ');

END.
GotoXY (50,12) ; Read (SearchName) ;
SearchNumber := GetInt (StringNumber, 50,14) ;
Search (Root, SearchName, SearchNumber, 'YES') ;
END ;

'5' : Constellation_Age_Screen (Root) ;
'6' : Choose_Orbit_Screen (Root) ;
'7' : Ground_Track (Root) ;

#27 : BEGIN,
  IFKeyPressed THEN
    BEGIN
      Read (kbd.Ch) ;
      IF Ch = #15 THEN System_Level_Commands (Root) ;
      END ;
    'END ;
  'Q', 'q' : Quit := True ;
END ;
UNTIL Quit ;
END ;

{called at year start, charges all active pads and ground units}

PROCEDURE Assess_OMCosts (Root : UnitPtr) ;

VAR
  Current : UnitPtr ;

BEGIN
  Textbackground (1) ;
  C1rScr ;
  Border1 ;
  Textbackground (1) ;
  Center_Line (12, 'ASSESSING O&M COSTS FOR GROUND UNITS AND PADS') ;
  GotoXY (33,14) ; Write ('FOR YEAR: ', Year:5) ;
  Delay (2000) ;
  Current := Root ;
  WHILE (Current <> Nil) DO
    BEGIN
      IF (Current^.Mission = 'PAD') OR (Current^.Mission = 'GROUND')
        AND (Current^.Status = 'ACTIVE') THEN
        BUDGET := BUDGET - Current^.OMCost ;
      Current := Current^.Next ;
    END ; (while)
PROCEDURE Assign_Ground_Node (Root : UnitPtr; Node : string10; Name : string10; Number : integer);

VAR
  Current : UnitPtr;

BEGIN
  Current := Root;
  WHILE Current <> Nil DO
    BEGIN
      IF (Current^.Mission <> 'LAUNCHER') AND
          (Current^.Mission <> 'PAD') AND,
          (Current^.Mission <> 'GROUND') THEN
        BEGIN
          IF Node = 'CONTROL' THEN
            BEGIN
              Current^.ControlName := Name;
              Current^.ControlNumber := 1;
              END;
          END;
      END;
    END;
    Current := Current^.Next;
END;

PROCEDURE Check_Support (VAR Root : UnitPtr);

VAR
  Current : UnitPtr;
  NumCGCT, NumCMGCT : integer;
  CompleteCGCT, CompleteCMGCT, HaveCSOC : boolean;

BEGIN
  NumCGCT := 0; NumCMGCT := 0;
  CompleteCGCT := False; CompleteCMGCT := False; HaveCSOC := False;
  Current := Root;
  WHILE Current <> Nil DO
    BEGIN
      IF Current^.Mission = 'GROUND' THEN
        BEGIN
          IF (Current^.Name = 'CGCT') AND
              (Current^.Status = 'ACTIVE') THEN
            NumCGCT := NumCGCT + 1;
          IF (Current^.Name = 'CMGCT') AND
              (Current^.Status = 'ACTIVE') THEN
            NumCMGCT := NumCMGCT + 1;
          IF (Current^.Name = 'CSOC') AND
              (Current^.Status = 'ACTIVE') THEN
            HaveCSOC := True;
        END;
    END;
END.
HaveCSOC := True;

END ; (mission = Ground)
Current := Current^.Next ;
END ; (not nil)

IF (NumCGCT >= 4) OR ((NumCGCT >= 3) AND HaveCSOC) THEN
  CompleteCGCT := True ELSE CompleteCGCT := False ;

IF (NumCMGCT >= 4) OR ((NumCMGCT >= 3) AND HaveCSOC) THEN
  CompleteCMGCT := True ELSE CompleteCMGCT := False ;

IF CompleteCMGCT
  THEN Assign_Ground_Node (Root, 'CONTROL', 'CMGCT', 1)
ELSE
  IF CompleteCGCT
     THEN Assign_Ground_Node (Root, 'CONTROL', 'CGCT', 1) ;

(***************************

Current := Root ;
While Current <> Nil DO
  BEGIN
    IF (Current^.Mission <> 'GROUND') AND:
      (Current^.Mission <> 'LAUNCHER')
     AND (Current^.Mission <> 'PAD') THEN
       BEGIN
         IF HaveCSOC THEN
           BEGIN
             Current^.ProcessorName := 'CSOC' ;
             Current^.ProcessorNumber := 1 ;
           END ;
         IF (Current^.ControlName <> '') AND
           (Current^.ControlName <> 'NONE') AND
           (Current^.ProcessorName <> '') AND
           (Current^.ProcessorName <> 'NONE') AND
           (Current^.UserName <> '') AND
           (Current^.UserName <> 'NONE') AND
           (Current^.Status = 'PRODUCTION')
           THEN Current^.Status := 'READY' ;
           END ;
    Current := 'Current^.Next ;
  END ; (not ground/launcher/pad)

END ; (not nil)

END ; (checks database for new feasible units)

PROCEDURE Update_Buyable (Root : UnitPtr ; ThisMission : string10 ;
                         HighTechLevel : integer ) ;

VAR
Current : UnitPtr;

BEGIN
  Current := Root;
  WHILE (Current <> Nil) AND (Current^.Mission <> ThisMission) DO
    Current := Current^.Next;

  WHILE (Current <> Nil) AND (Current^.Mission = ThisMission) DO
    BEGIN
      IF (Current^.TechLevel = HighTechLevel + 1) AND
          ((Current^.Status = 'NONE') OR (Current^.Status = ''))
        THEN Current^.Status := 'FEASIBLE';
    END;
  END;
END;

OVERLAY PROCEDURE Assign_Ground_Support (Root : UnitPtr);

BEGIN
  Textbackground (1);
  ClrScr;
  Border1;
  Textbackground (1);
  Center_Line (12, 'ASSIGN GROUND SUPPORT ELEMENTS TO SATELLITES');
  GotoXY (33,14); Write ('FOR YEAR ', Year:5);
  Delay (2000);
  IF Help THEN HelpScreen (7);
END;

REPEAT
  Check_Support (Root);
  Assign_Support (Root);
  Window (1,1,80,25);
  Check_Support (Root);
  ClearLines (25,25);
  Center_Line (25, 'Do you want to assign more ground units (Y/N) ? ');
  Read (kbd,Ch);
  IF (Ch- #27) AND KeyPressed THEN
    BEGIN
      Read (kbd,Ch);
      IF Ch = #15 THEN System_Level_Commands (Root);
      END;
  UNTIL Ch IN ['N', 'n'];
END;
OVERLAY PROCEDURE Get_TechLevel (Root : UnitPtr);

VAR
Current : UnitPtr;
ThisMission : string10;
HighTechLevel : integer;

BEGIN
Textbackground (1);
Center_line (12, 'UPDATING *FEASIBLE* SYSTEMS');
Delay (2000);

Current := Root;
While (Current <> nil) DO
BEGIN
ThisMission := Current^.Mission;
HighTechLevel := 1;
While (Current <> nil) AND (Current^.Mission = ThisMission) DO
BEGIN
IF ((Current^.Status = 'PRODUCTION') OR
    (Current^.Status = 'READY') OR
    (Current^.Status = 'ACTIVE') OR
    (Current^.Status = 'PARASITE') OR
    (Current^.Status = 'SPARE') OR
    (Current^.Status = 'DEAD')) AND
    (Current^.TechLevel > HighTechLevel)
THEN HighTechLevel := Current^.TechLevel;

Current := Current^.Next;
END; (while thismission)

Update_Buyable (Root, ThisMission, HighTechLevel);
END; (while not nil)

END;

OVERLAY PROCEDURE Advance (Root : UnitPtr; VAR WorkUnit : Unit);

VAR
ThisStatus : string11;
ReusableLY : boolean;

BEGIN

WITH WorkUnit DO
BEGIN
IF (WorkUnit.Mission = 'LAUNCHER') AND
   ((WorkUnit.ReuseField = 'REUSEABLE') OR


159
(WorkUnit.ReuseField = 'RECYCLABLE') OR
(WorkUnit.ReuseField = 'R') OR (WorkUnit.ReuseField = 'C'))
THEN ReusableLV := True
ELSE ReusableLV := False;
ThisStatus := Status;

IF (Mission <> 'LAUNCHER') OR (ReusableLV) THEN
(expendables different)
BEGIN
   IF (Mission = 'GROUND') OR (Mission = 'PAD') THEN
      BEGIN
         IF (ThisStatus = 'PRODUCTION') OR
             (ThisStatus = 'DEVELOPMENT')
            THEN Status := 'ACTIVE';
END;
      (no LAU phase for ground or pads)
      END
   IF ReusableLV THEN
      BEGIN
         IF ThisStatus = 'FEASIBLE' THEN Status := 'DEVELOPMENT';
         IF ThisStatus = 'DEVELOPMENT' THEN
            BEGIN
               Status := 'ACTIVE';
               NumActive := 1;
            END;
         Update := -1;
      END
   ELSE (not reusableLV)
   BEGIN
   IF ThisStatus = 'FEASIBLE'
      THEN Check_If_Second_Copy (Root, WorkUnit);
   IF ThisStatus = 'RESEARCH'
      THEN Status := 'DEVELOPMENT';
   IF ThisStatus = 'DEVELOPMENT'
      THEN Status := 'PRODUCTION';
   Update := -1;
   END: (not reusable)
   END
   (mission not launcher or is reusable)
ELSE (expendable launchers)
BEGIN
   NumActive := NumActive + ProdActive;
   ProdActive := NumProd;
   NumProd := DevProd;
   DevProd := Hold;
   Hold := 0;
END: (launcher)
END: (With)
PROCEDURE Update_Systems (Root : UnitPtr) ;

VAR
  Current : UnitPtr ;

BEGIN
  Where := Root ;
  Current := Root ;
  While Current <> Nil DO
    BEGIN
      IF Current^.Status = 'SPARE' THEN
        Current^.LifeTime := Current^.LifeTime + 1 ;
      IF Current^.Update > 0 THEN
        Current^.Update := Current^.Update - 1 ;
      IF Current^.Update = 0 THEN Advance (Root, Current^) ;
      Current := Current^.Next ;
      Where := Where^.Next ;
    END;

END ;

PROCEDURE Upgrade_Units (Root : UnitPtr) ;

BEGIN
  Border1 ;
  Textbackground (1) ;
  Center_Line (12, 'UPDATING UNITS PURCHASED/UPGRADED LAST TURN') ;
  Delay (2000) ;
  Update_Systems (Root) ;
  Get_TechLevel (Root) ;

END ;

{called at year start}

PROCEDURE Update_Global_Variabies (Root : UnitPtr) ;

VAR
  Current : UnitPtr ;

BEGIN
  Textbackground (1) ;

Year := Year + 1;
Center_Line (12, '');
Center_Line (12, 'UPDATING GLOBAL VARIABLES');
GotoXY (33,14); Write ('FOR YEAR ', Year:5);
Delay (2000);
Current := Root;
While Current <> Nil DO
BEGIN
  IF Current^.Mission = 'PAD' THEN
    Current^.NumRemaining := Current^.LaunchRate;
    Current := Current^.Next;
  END;
BUDGET := XBUDGET [Year];
Check_Fail (Root);
END;

( main program )
BEGIN
ClrScr;
Root := Nil; Head := Nil; Quit := False; Found := False;
MultipleLaunch := False;
Copy := False; NASAOK := False;
Where := Nil; Year := 1986;
RemWeight := 0; LiftWeight := 0;
Help := False;
Border1; Border2;
TextBox (1); Window (20,6,59,19);
crsr;
Gotoxy (18,2); Write ('ADAM');
GotoXY (9,4); Write ('Acquisition Deployment');
GotoXY (12,5); Write ('And Maneuvering');
GotoXY (13,7); Write ('THE SPACE GAME');
TextColor (White);
GotoXY (14,9); Write ('Developed By');
GotoXY (12,10); Write ('CAPT. JEFF HEIER');
Gotoxy (8,12); Write ('Graduate Space Operations');
GotoXY (4,13); Write ('Air Force Institute of Technology');
Delay (6000);
Window (1,1,80,25);
crsr;

Draw_World;
TextBox (1); crs;
Help_Screen (0);
Borderl ; Textbackground (1) ;
Center_line (12, 'Enter 'L' to load the database ');
Read (kbd,Ch) ;
IF (Ch = 'L') OR (Ch = '1') THEN LoadList (Root) ;

Center_line (16, 'Enter 'H' to activate the-Help Screens') ;
Read (kbd,Ch) ;
IF Ch IN ['H', 'h'] THEN Help := True ELSE Help := False ;
IF Help THEN Help_Screen (6) ; (game explanation)

REPEAT
Randomize ;
Textbackground (1) ;
ClrScr ;
Borderl ;
textbackground (1) ;
Gotoxy (30,12) ; Write ('BEGINNING YEAR ', (Year + 1):5) ;
Delay (2000) ;
Update_Gloabal_Variables (Root) ;
Upgrade_Units (Root) ; (purchased last turn)
Update_Units (Root) ; (will take effect next turn)
Assign_Ground_Support (Root) ;
Assess_OMCosts (Root) ;
Match_Payloads_With_Launchers (Root) ;
Textbackground (1) ;
ClrScr ;
Borderl ;
textbackground (1) ;
IF Year = 1996 THEN Center_line (12, 'GAME OVER') ELSE
Center_line (12, 'DO YOU WISH TO CONTINUE TO NEXT YEAR ? ') ;
Read (kbd,Ch) ;
IF (Ch = '#27') AND KeyPressed THEN
BEGIN
Read (kbd,Ch) ;
IF Ch = '15' THEN System_Level_Commands (Root) ;
END ;
UNTIL (Ch = 'N') OR (Ch = 'n') OR (Year = 1996) ;

END .

(Yes is used on all yes/no questions to the user)
FUNCTION Yes : boolean;
begin
  Read (kbd, Ch);
  IF Ch IN ['Y', 'y'] THEN Yes := True ELSE Yes := False;
END;

(Continue is used on all continue questions to the user)

FUNCTION Continue : boolean;
begin
  Read (kbd, Ch);
  IF Ch IN ['C', 'c'] THEN Continue := True ELSE Continue := False;
END;

(Center_Line centers the line on the screen)

PROCEDURE Center_Line (Row : integer; Line : string80);
VAR
  I, J : integer;
BEGIN
  TextColor (White);
  I := Length (Line);
  J := Round ((80 - I) / 2);
  GotoXY (J, Row); Write (Line);
  TextColor (Yellow);
END;

(Border1 and Border2 draw a border around the screen)

OVERLAY Procedure Border1;
var i : integer;
begin
  clrscr;
  textbackground (white);
  for i := 1 to 25 do
    begin
      gotoxy (79, i);
      write (' ');
    end;
  for i := 1 to 25 do
    begin
      gotoxy (1, i); write (' ');
    end;
  gotoxy (1, 1); clrscr;
end;

OVERLAY Procedure Border2;
var i : integer;
begin
  textbackground (3);
  for i := 2 to 24 do
    begin
      // Code continues
    end;
end;
gotoxy (3,i) ; write (' ') ;
gotoxy (77,i) ; write (' ') ;
end;
for 'i := 3 to 78 do
begin
  gotoxy (i,2) ; write (' ') ;
gotoxy (i,24) ; write (' ') ;
end;
end ;

(ClearLines clears the screen between the line numbers passed to it)

PROCEDURE ClearLines (First,Last : integer) ;

VAR
  I : integer ;
BEGIN
  For I := First TO Last DO
  BEGIN
    GotoXY (1,I) ;
    ClrEOL
  END
END ;

(GetInt is used to ensure the user does not enter a character for an integer value, else the program bombs)

FUNCTION GetInt (SST : string10 ; X,Y : Integer) : Integer ;

VAR
  R : integer ;
  Result : integer ;
BEGIN
  REPEAT
    GotoXY (X,Y) ; Write (' ') ;
    GotoXY (X,Y) ;
    Read (SST) ;
    Val (SST, R, Result) ;
    IF Result = 0 THEN GetInt := R
    ELSE
      BEGIN
        GotoXY (X,Y) ; Write ('Enter *') ; Write (Chr(7)) ;
        Delay (2000) ;
      END ;
UNTIL Result = 0 ;
END ;

(LoadList loads the database from the disk. List must have been created)
PROCEDURE LoadList (VAR Root : UnitPtr)

VAR
WorkName : String[30];
WorkFile : UnitFile;
Current : UnitPtr;
I : integer;
OK : boolean;
Quit : boolean;

BEGIN
Quit := False;
REPEAT
TextBackground (1);
ClrScr;
Border1;
TextBackground (1);
CenterLine (12, 'Please enter the file name as *.dat ');
GotoXY (35,14);
Read "(WorkName);
IF Length (WorkName) = 0 THEN  (hit <CR> only to abort load )
BEGIN
ClearLines (10,12);
Quit := True
END
ELSE
BEGIN
Assign (WorkFile,WorkName);
(*I-) Reset (WorkFile); (*I+)
IF IOResult <> 0 THEN  (0 = OK; 255 = File Not Found )
BEGIN
CenterLine (12, 'That file does not exist. Check Spelling')
OK := False
END
ELSE OK := True
END;
UNTIL OK OR Quit;
IF NOT.Quit THEN
BEGIN
Current := Root;
IF Root = Nil THEN  (if list is currently empty )
BEGIN
NEW (Root);  (load first unit to Root"
Read (WorkFile,Root"
Current := Root; Read := Current
END  (if list is not empty, find the end: )
ELSE WHILE Current <> Nil DO Current := Current".Next;
IF Root".Next <> Nil THEN  (if file contains more than 1 record)
REPEAT
NEW (Current".Next);  (read and units to list )
Current := Current".Next;  (until a unit's next field )
Read (WorkFile,Current"
UNTIL Current".Next = Nil;
Close (WorkFile)
END

{ShowUnit is used to display the current status of a unit's attributes}
{Each mission type has its own fields}.

OVERLAY PROCEDURE ShowUnit (VAR WorkUnit: Unit);

VAR
  I : integer;
  Current: UnitPtr;
  Last: UnitPtr;

BEGIN
  textbackground (1);
  ClrScr;
  GotoXY (1,1);

  With WorkUnit DO
  BEGIN
    Writeln ('>> Mission: ', Mission);
    Writeln ('>> Name: ', Name);

    IF Mission = 'LAUNCHER' THEN
    BEGIN
      Writeln ('>> Tail Number', TailNumber);
      Writeln ('>> Reusable Type', ReuseField);
      IF (ReuseField = 'REUSABLE') OR (ReuseField = 'RECYCLABLE') OR
      (ReuseField = 'R') OR (ReuseField = 'C') THEN
      BEGIN
        Writeln ('>> Annual Sorties Left', Sorties);
        Writeln ('>> System Launches Left', NumUses);
        Writeln ('>> Launch/Refurbish Cost', OMCost);
        Writeln ('>> Status', Status);
      END
    ELSE
    BEGIN
      Writeln ('>> # Active', NumActive);
      Writeln ('>> # Prod-Active', ProdActive);
      Writeln ('>> # Production', NumProd);
      Writeln ('>> # Dev-Prod', DevProd);
      Writeln ('>> # Development', NumDev);
    END
    Writeln ('>> Reliability', Reliability);
    Writeln ('>> R&D Cost', RDCost);
    Writeln ('>> ACQ Cost', ACQCost);
    Writeln ('>> Mass to VLEO', VLEO);
    Writeln ('>> Mass to LEO', LEO);
    Writeln ('>> Mass to MOL', MOL);
    Writeln ('>> Mass to MEO', MEO);
    Writeln ('>> Mass to GEO', GEO);
    Writeln ('>> Update', Update);
  END;
}

167
IF Mission = 'GROUND' THEN
BEGIN
  Writeln ('>> TailNumber: '', TailNumber);
  Writeln ('>> Type: '', Kind);
  Writeln ('>> LifeTime: '', LifeTime);
  Writeln ('>> Deployed: '', Deployed);
  Writeln ('>> R&D Cost: '', RDCost);
  Writeln ('>> ACQ Cost: '', ACQCost);
  Writeln ('>> Status: '', Status);
  Writeln ('>> O&M Cost: '', OMCost);
  Writeln ('>> Supporting: '', SupportName);
  Writeln ('>> TailNumber: '', SupportNumber);
  Writeln ('>> Update
  Writeln ('>> Tech Level
END

IF Mission = 'PAD' THEN
BEGIN
  Writeln ('>> TailNumber: '', TailNumber);
  Writeln ('>> Location: '', Location);
  Writeln ('>> Deployed: '', Deployed);
  Writeln ('>> Status: '', Status);
  Writeln ('>> Update
  Writeln ('>> R&D Cost: '', RDCost);
  Writeln ('>> ACQ Cost: '', ACQCost);
  Writeln ('>> O&M Cost: '', OMCost);
  Writeln ('>> Launches/Year
  Writeln ('>> Launches/Rem
END

IF Mission <> 'LAUNCHER'
  AND Mission <> 'GROUND'
  AND Mission <> 'PAD') THEN
BEGIN
  Writeln ('>> TailNumber: '', TailNumber);
  Writeln ('>> Mass: '', Mass:5.0);
  Writeln ('>> LifeTime: '', LifeTime);
  Writeln ('>> Deployed: '', Deployed);
  Writeln ('>> R&D Cost: '', RDCost);
  Writeln ('>> ACQ Cost: '', ACQCost);
  Writeln ('>> Status: '', Status);
  Writeln ('>> Orbit
  Writeln ('>> Inclination: '', Inclination);
  Writeln ('>> Ascending Node
  Writeln ('>> Update
  Writeln ('>> Tech Level: '', TechLevel);
  Writeln ('>> Controller
  Writeln ('>> ControlNumber)
  Writeln ('>> Processor
  Writeln ('>> ProcessorNumber)
  Writeln ('>> User
  Writeln ('>> UserName)

168
WriteN ("'>> Add-On : 1" ', AddOnName1); 
WriteN ("'>> Add-On : 2" ', AddOnName2); 
WriteN ("'>> Add-On : 3" ', AddOnName3); 
WriteN ("'>> Add-On : 4" ', AddOnName4); 
WriteN ("'>> Add-On : 5" ', AddOnName5); 
WriteN ("'>> Add-On : 6" ', AddOnName6); 
WriteN ("'>> Add-On : 7" ', AddOnName7); 
WriteN ("'>> Add-On : 8" ', AddOnName8); 

GotoXY (40,1); 
Write ("'>> Add-On : 3" ', AddOnName3); GotoXY (40,2); 
Write ("'>> Add-On : 4" ', AddOnName4); GotoXY (40,3); 
Write ("'>> Add-On : 5" ', AddOnName5); GotoXY (40,4); 
Write ("'>> Add-On : 6" ', AddOnName6); GotoXY (40,5); 
Write ("'>> Add-On : 7" ', AddOnName7); GotoXY (40,6); 
Write ("'>> Add-On : 8" ', AddOnName8); GotoXY (40,7); 
Write ("'>> Anti-jam Data" ', AntiJamDataLink); GotoXY (40,8); 
Write ("'>> Low Anti-Jam Comm" ', LowAntiJamComLink); GotoXY (40,9); 
Write ("'>> Med Anti-Jam Comm" ', MedAntiJamComLink); GotoXY (40,10); 
Write ("'>> Hi Anti-Jam Comm" ', HighAntiJamComLink); GotoXY (40,11); 
Write ("'>> 15 Day Autonomy" ', Autonomy15); GotoXY (40,12); 
Write ("'>> 180 Day Autonomy" ', Autonomy180); GotoXY (40,13); 
Write ("'>> # of Munovers" ', Manuvers:2:0); GotoXY (40,14); 

END ; 

IF Mission = 'COMM' THEN 
  Write ("'>> Channels:" ', Channels:6:0); 

IF Mission = 'WEATHER' THEN 
BEGIN 
  Write ("'>> DownLink" ', DownLink); 
  GotoXY(40,15); 
  Write ("'>> DayVideo" ', DayVideo); 
  GotoXY(40,16); 
  Write ("'>> NightVideo" ', NightVideo); 
  GotoXY (40,17); 
  Write ("'>> DayImage" ', DayImage); 
  GotoXY (40,18); 
  Write ("'>> NightImage" ', NightImage); 
  GotoXY (40,19); 
  Write ("'>> Temp" ', Temp); 
  GotoXY (40,20); 
  Write ("'>> Solar" ', Solar); 
  GotoXY (40,21); 

END ; 

IF Mission = 'NAV' THEN 
BEGIN 
  Write ("'>> Accuracy" ', Accuracy); 
  GotoXY (40,15); 
  Write ("'>> Position" ', Position:5:2); 
  GotoXY (40,16); 
  Write ("'>> Redundancy" ', Redundancy); 
  GotoXY (40,17); 

END ; 

IF Mission = 'IMINT' THEN 

169
BEGIN
   Write ('>> DownLink' , DLink) ;
   GotoXY (40,15) ;
   Write ('>> DayVideo' , DVideo) ;
   GotoXY (40,16) ;
   Write ('>> NightVideo' , NVideo) ;
   GotoXY (40,17) ;
   Write ('>> DayImage' , DImage) ;
   GotoXY (40,18) ;
   Write ('>> NightImage' , NImage) ;
   GotoXY (40,19) ;
END ;

IF Mission = 'SIGINT' THEN 
BEGIN
   Write ('>> Data Link' , DataLink) ;
   GotoXY (40,15) ;
   Write ('>> Signal Identity' , Signal) ;
   GotoXY (40,16) ;
END ;

IF Mission = 'WARNING' THEN 
BEGIN
   Write ('>> Aircraft' , Aircraft) ;
   GotoXY (40,15) ;
   Write ('>> ICBM' , ICBM) ;
   GotoXY (40,16) ;
   Write ('>> SLBM' , SLBM) ;
   GotoXY (40,17) ;
   Write ('>> Target' , Target) ;
   GotoXY (40,18) ;
   Write ('>> Count' , Count) ;
   GotoXY (40,19) ;
END ;

IF Mission = 'OFFENSE' THEN:
BEGIN
   WRITE ('>> Affect LEO' , LowOrbit) ;
   GotoXY (40,15) ;
   WRITE ('>> Affect MEO' , MedOrbit) ;
   GotoXY (40,16) ;
   WRITE ('>> Affect GEO/HEO' , HighOrbit) ;
   GotoXY (40,17) ;
   WRITE ('>> Ground Targets' , GroundTarget) ;
   GotoXY (40,18) ;
   WRITE ('>> Aircraft' , Planes) ;
   GotoXY (40,19) ;
   WRITE ('>> Time Urgent' , TimeUrgent) ;
   GotoXY (40,20) ;
END ;

IF Mission = 'DEFENSE' THEN 
BEGIN
   Write ('>> Affect Boost Phase' , Boost) ;
GotoXY (40,15);
Write ('>') Affect MidCourse Phase ', MidCourse);
GotoXY (40,16);
Write ('>') Affect Reentry Phase ', Reentry);
GotoXY (40,17);
Write ('>') Affect Aircraft ', AC);
GotoXY (40,18);
Write ('>') Affect Cruise Missiles ', CruiseMissile);
GotoXY (40,19);
END;

ClearLines (25,25); Center_Line (25, 'Hit any key to continue');
read (kbd,Ch);
END;
END;

{Search is called every time an assignment is made. It sets the)
(global variable 'Where' which points to the desired unit. If the)
(unit is not found, a message is flashed to the user)

PROCEDURE Search (VAR Root : UnitPtr ; SearchName : string10 ;
SearchNumber : integer ; Display : string5);

VAR
Current : UnitPtr;
Last : UnitPtr;
Show : boolean;

BEGIN
Found := False;
Current := Root;
Where := Root;
IF Display = 'YES' THEN Show := True
ELSE Show := False;

WHILE (Current <> Nil) AND ((Current^.Name <> SearchName)
OR (Current^.TailNumber <> SearchNumber)) DO
BEGIN
Last := Current;
Where := Where^.Next;
Current := Current^.Next;
END;

IF (Current^.Name = SearchName) AND (Current^.TailNumber = SearchNumber)
THEN Found := True;
IF Found AND Show THEN ShowUnit (Current);
IF NOT Found Then
BEGIN
ClearLines (23,25);
GotoXY (28,23); Write (SearchName:10, SearchNumber:3, ' not found');
Textcolor (White);
Center_Line (25, 'Hit any key to continue'); Read (kbd,Ch);
(Add_Survivability modifies the survivability fields of units.)
(It checks the current acquisition status of the unit and determines)
(if it can be updated. Only units in feasible, research, and)
(development stage can be modified. Units in development stage must)
(either pay 50% more for mods or delay an extra 2 years in acq. cycle)

OVERLAY PROCEDURE Add_Survivability (VAR Root : UnitPtr);

VAR
Quit : boolean;
Amount : real;  Diff : real;
FuelLoader : integer;
WorkUnit : Unit;
OriginalMass : real;
Got1, Got2, Got3, Got4, Got5, Got6, Got7 : string[2];
Autonomy15Mass : integer;
Autonomy180Mass : integer;
FuelMass : integer;
Modify : boolean;
LateCost : real;

BEGIN
Got1 := ''; Got2 := ''; Got3 := ''; Got4 := ''; 
Got5 := ''; Got6 := ''; Got7 := ';
Quit := False;
LateCost := 1.0;
Modify := True;
TextBackground := (1);
ClrScr;
Border1;
TextBackground := (1);
CenterLine (4, 'ADD SURVIVABILITY FEATURES');
CenterLine (12, 'Enter name of unit to be made survivable');
CenterLine (14, 'Tail Number');
GotoXY (65,12); Read (SearchName); 
SearchNumber := GetInt (StringNumber, 65,14); 
Search (Root, SearchName, SearchNumber, 'YES');

IF Found THEN
begin
IF (Where'.AntiJamDataLink <> 'X') AND ((Where'.Status = 'RESEARCH') OR 
(Where'.Status = 'DEVELOPMENT') AND (Where'.Status = 'FEASIBLE'))
THEN (can upgrade)
BEGIN
IF Where'.Status = 'DEVELOPMENT' THEN
BEGIN
ClearLines (22,25);
CenterLine (22, 'Modifying satellite in DEVELOPMENT Stage is costly');
CenterLine (23, 'Enter '1' for a two year program delay ');
CenterLine (24, 'Enter '2' for a 50% cost overrun on mods');
CenterLine (25, 'Enter 'Q' to not modify the satellite ');
Read (kbd, Ch);
CASE Ch OF
        ELSE Where^.Update := Where^.Update + 2;
        {if not being updated, flag for check budget}
        {in check budget, update=99 --> update = 4}
  '2' : LateCost := 1.5;
  'Q','q': Modify := False;
END;
END;

IF Modify THEN
BEGIN
  IF Where^.AntiJamDataLink = 'Y' THEN Got1 := '+';
  IF Where^.LowAntiJamComLink = 'Y' THEN Got2 := '+';
  IF Where^.MedAntiJamComLink = 'Y' THEN Got3 := '+';
  IF Where^.HighAntiJamComLink = 'Y' THEN Got4 := '+';
  IF Where^.Autonomy15 = 'Y' THEN Got5 := '+';
  IF Where^.Autonomy180 = 'Y' THEN Got6 := '+';
  IF Where^.Manuevers > 0 THEN Got7 := '+';

  Autonomy15Mass := Round (Where^.Mass * 0.2);
  Autonomy180Mass := Round (Where^.Mass * 0.3);
  FuelMass := Round (Where^.Mass * 0.3);

  REPEAT
    ClrScr;
    TextBackground (1);
    CenterLine (1, 'ADD SURVIVABILITY FEATURES');
    TextColor (Yellow);
    CenterLine (2, '+' Indicates unit already has that method');
    GotoXY (10,4) ; Write ('SATELLITE ', Where^.Name:10, Where^.TailNumber:3);
    GotoXY (40,4) ; Write ('MASS: ', Where^.Mass:6:0);
    GotoXY (60,4) ; Write ('BUDGET: ', BUDGET);
    TextColor (White);
    GotoXY (20,6) ; Write ('Method' ); GotoXY (68,6) ; Write ('Cost');
    GotoXY (19,7) ; Write ('-------') ; GotoXY (67,7) ; Write ('-------');
    TextColor (Yellow);
    GotoXY (3,9) ; Write ('[1] Anti-jam/nuclear protection on');
    Write ('data/control links');
    Gotoxy (60,9) ; Write (Got1);
    GotoXY (69,9) ; Write (Round (0.3 * Where^.RDCost));
    GotoXY (3,11) ; Write ('[2] Low anti-jam/nuclear protection');
    Write ('on COM links');

  173
GotoXY (60,11); Write (Got2);
GotoXY (69,11); Write (Round (0.1 * Where ^.RDCost));
GotoXY (3,13); Write ('[3] Medium anti-jam/nuclear protection');
Write ('on COM links');
GotoXY (60,13); Write (Got3);
GotoXY (69,13); Write (Round (0.2 * Where ^.RDCost));
GotoXY (3,15); Write ('[4] High anti-jam/nuclear protection on');
Write ('COM links');
GotoXY (60,15); Write (Got4);
GotoXY (69,15); Write (Round (0.3 * Where ^.RDCost));
GotoXY (3,17); Write ('[5] Laser/nuclear hardening and 15 day autonomy');
GotoXY (60,17); Write (Got5);
GotoXY (69,17); Write (Round (0.3 * Where ^.ACQCost));
GotoXY (3,19); Write ('[6] Laser/nuclear hardening and 180 day autonomy');
GotoXY (60,19); Write (Got6);
GotoXY (69,19); Write (Round (2.0 * Where ^.ACQCost));
GotoXY (3,21); Write ('[7] Satellite manuverability (cost/maneuver)');
GotoXY (60,21); Write (Got7);
GotoXY (69,21); Write (Round (0.3 * Where ^.ACQCost));

Center_Line (24, 'Enter 1-7 or 'Q' to quit ');
Read (kbd, Ch);

OriginalMass := Where ^.Mass;

CASE Ch OF
  '1'  : BEGIN
      Amount := (Where ^.RDCost div 2) * 0.3 * LateCost;
      IF Where ^.AntiJamDataLink = 'Y' THEN
          BEGIN
              Where ^.AntiJamDataLink := 'N';
              BUDGET := BUDGET + Round (Amount);
              Got1 := '';
          END
      ELSE
          BEGIN
              IF (BUDGET - Amount) >= 0 THEN
                  BEGIN
                      Where ^.AntiJamDataLink := 'Y';
                      Got1 := ' + ';
                      BUDGET := BUDGET - Round (Amount);
                  END
              ELSE IF (BUDGET - Amount) < 0 THEN
                  BEGIN
                      ClearLines (24,25);
                      Center_Line (24, 'CANT AFFORD IT LOOK AT BUDGET ');
                      Center_Line (25, 'hit any key to continue');
                      Read (kbd, Ch);
                  END
          END
      END ; (doesnt allready have)
END ;

174
'2' : BEGIN
  Amount := (Where^.RDCost div 2) * 0.1 * LateCost;
  IF Where^.LowAntiJamComLink = 'Y' THEN
    BEGIN
      Where^.LowAntiJamComLink := 'N';
      BUDGET := BUDGET + Round (Amount);
      Got2 := '';
    END
  ELSE IF (BUDGET - Amount) >= 0 THEN
    BEGIN
      Where^.LowAntiJamComLink := 'Y';
      Got2 := '+';
      BUDGET := BUDGET - Round (Amount);
    END
  ELSE IF (BUDGET - Amount) < 0 THEN
    BEGIN
      ClearLines (24,25);
      Center_Line (24, 'can afford it look at budget');
      Center_Line (25, 'hit any key to continue');
      Read (kbd,Ch);
    END;
  END;
END;

'3' : BEGIN
  Amount := (Where^.RDCost div 2) * 0.2 * LateCost;
  IF Where^.MedAntiJamComLink = 'Y' THEN
    BEGIN
      Where^.MedAntiJamComLink := 'N';
      BUDGET := BUDGET + Round (Amount);
      Got3 := '';
    END
  ELSE
    BEGIN
      Where^.MedAntiJamComLink := 'Y';
      Got3 := '+';
      BUDGET := BUDGET - Round (Amount);
    END
  ELSE IF (BUDGET - Amount) < 0 THEN
    BEGIN
      ClearLines (24,25);
      Center_Line (24, 'can afford it look at budget');
      Center_Line (25, 'hit any key to continue');
      Read (kbd,Ch);
    END;
END;
'4' : BEGIN
    Amount := (Where^.RDLCost div 2) * 0.3 * LateCost;
    IF Where^.HighAntiJamComLink = 'Y' THEN
        BEGIN
            Where^.HighAntiJamComLink := 'N';
            BUDGET := BUDGET + Round (Amount);
            Got4 := '';
        END
    ELSE
        BEGIN
            IF (BUDGET - Amount) = 0 THEN
                BEGIN
                    Where^.HighAntiJamComLink := 'Y';
                    Got4 := '+';
                    BUDGET := BUDGET - Round (Amount);
                END
            ELSE IF (BUDGET - Amount) < 0 THEN
                BEGIN
                    ClearLines (24,25);
                    CenterLine (24, 'cant afford it look at budget');
                    CenterLine (25, 'hit any key to continue');
                END
            END;
    END;

'5' : BEGIN
    Amount := Where^.ACQCost * 0.3 * LateCost;
    IF Where^.Autonomy15 = 'Y' THEN
        BEGIN
            Where^.Autonomy15 := 'N';
            BUDGET := BUDGET + Round (Amount);
            Got5 := '';
        END
    ELSE
        BEGIN
            IF (BUDGET - Amount) >= 0 THEN
                BEGIN
                    ClearLines (23,25); GotoXY (24,23);
                   TextColor (White);
                    Write ('Adding 15 Day Autonomy increases sat');
                    Write ('mass to ', (Where^.Mass + Autonomy15Mass):5:0);
                    CenterLine (25, 'Is this OK (Y/N) ? ');
                    IF Yes THEN
                        BEGIN
                            Where^.Autonomy15 := 'Y'; Got5 := '+';
                        END
                    END;
        END;

176
BUDGET := BUDGET - Round (Amount);  
Where^\text{.Mass} := Where^\text{.Mass} +  
\text{Autonomy180Mass};  
END ; (yes)  
END  

ELSE IF (BUDGET - Amount) < 0 THEN  
BEGIN  
ClearLines (24,25);  
Center_Line (24,  
'cant afford it look at budget');  
Center_Line (25, 'hit any key to continue');  
Read (kbd,Ch);  
END ;  
END ; (doesnt already have)  
END ;  

'6' : BEGIN  
Amount := Where^\text{.ACQCost} * 2 * LateCost;  
IF Where^\text{.Autonomy180} = 'Y' THEN  
BEGIN  
Where^\text{.Autonomy180} := 'N';  
BUDGET := BUDGET + Round (Amount);  
Got6 := ' ';  
Where^\text{.Mass} := Where^\text{.Mass} - Autonomy180Mass;  
END  
ELSE  
BEGIN  
IF (BUDGET - Amount) >= 0 THEN  
BEGIN  
ClearLines (23,25); GotoXY (21,23);  
TextColor (White);  
Write ('Adding 180 day Autonomy increases mass');  
Write ('to ',  
(Where^\text{.Mass} + Autonomy180Mass):5:0);  
TextColor (Yellow);  
Center_Line (25, 'Is this OK (Y/N) ? ');  
IF Yes THEN  
BEGIN  
Where^\text{.Autonomy180} := 'Y'; Got6 := '+';  
BUDGET := BUDGET - Round (Amount);  
Where^\text{.Mass} := Where^\text{.Mass} + Autonomy180Mass;  
END ; (yes)  
END  
ELSE IF (BUDGET - Amount) < 0 THEN  
BEGIN  
ClearLines (24,25);  
Center_Line (24,  
'cant afford it look at budget');  
Center_Line (25, 'hit any key to continue');  
Read (kbd,Ch);  
END ;
ClearLines (23, 25);
CenterLine (24, 'How many maneuvers do you want? ');
FuelLoads := GetInt (StringNumber, 58, 24);
IF FuelLoads <> Where^.Manuvers THEN
BEGIN
IF FuelLoads < Where^.Manuvers THEN
BEGIN (removing fuel)
  Diff := Where^.Manuvers - FuelLoads;
  Amount := Where^.ACQCost *
           Diff * 0.3 * LateCost;
  BUDGET := BUDGET + Round (Amount);
  Where^.Manuvers := FuelLoads;
  IF Where^.Manuvers = 0 THEN Got7 := ' ';
END
ELSE (adding fuel)
BEGIN
  Amount := Where^.ACQCost *
            Diff * 0.3 * LateCost;
  IF (BUDGET - Amount) >= 0 THEN
BEGIN
  ClearLines (23, 25); GotoXY (23, 23);
  Write ('adding this fuel increases mass to ',
          (Where^.Mass + FuelMass * Diff):5:0);
  CenterLine (25, 'Is this OK (Y/N)? ');
  IF Yes THEN
BEGIN
  Where^.Manuvers := Fuel Loads;
  BUDGET := BUDGET - Round (Amount);
  Got7 := '+';
END ; (yes)
END
END
ELSE IF (BUDGET - Amount) < 0 THEN
BEGIN
  ClearLines (24, 25);
  CenterLine (24, 'can't afford it look at budget');
  CenterLine (25, 'hit any key to continue');
  Read (kbd, Ch);
END;
END ; (off loading fuel)
END ; (not changing fuel loads)
END;

'Q', 'q' : Quit := True;
UNTIL Quit;
END ; (modify unit)
END (unit can be modified)
ELSE
BEGIN  (Where^.AntiJamDataLink = 'X')
  ClearLines (24,25) ; GotoXY (25,24);
  Write (SearchName,'Cannot be made survivable');
  Center-Line (25, 'Hit any key to continue'); Read (kbd,Ch);
END ; (Unit cant be modified)
END ; (if found)
END ;

(Help_Screen is used to help the players understand the different)
(procedures and the necessary input to them. Each procedure has a)
(different number associated with it. This number is sent to Help_Screen)
(and that determines which block is sent to the screen.)

OVERLAY PROCEDURE Help_Screen (Screen : Integer) ;

BEGIN
  TextBackGround (1);
  ClrScr;
  Window (13,2,75,24) ; ClrScr;
  CASE Screen OF
    0 : BEGIN
      GotoXY (23,1) ; TextColor (White);
      Write ('WELCOME TO ADAM');
      GotoXY (1,3) ; TextColor (Yellow);
      Writeln ('ADAM was designed for use in the Space Series Course at the');
      Writeln ('AWC and ACSC. It models how space assets are acquired, ');
      Writeln ('deployed, and maneuvered in response to ground requirements.');
      Writeln ('The different missions modeled in the game include:');
      Writeln ('Communication, Navigation, Weather, Reconnaissance, ');
      Writeln ('Surveillance (Attack Warning), Launchers, Launch Pads,');
      Writeln ('and Ground-Support Elements. The game models two types');
      Writeln ('of Renaissance: ELINT (Electronic Intelligence) and ');
      Writeln ('SIGINT Signal Intelligence). The objective of ADAM is to');
      Writeln ('design a space force which meets as many requirements as');
      Writeln ('possible while operating within a limited budget. The');
      Writeln ('game begins with the current (1987) US space force and');
      Writeln ('continues through 10 turns (each turn is one year).');
      Writeln ('The database may be modified if specific learning');
      Writeln ('objectives are to be taught. Reference the 'ADAM');
      Writeln ('Users Guide for directions in developing a new database.');
      Writeln ('ADAM used the Space Resource Allocation Exercise (SRAE)');
      Writeln ('designed at ACSC by Major Bruce Thieman as a baseline.');
      GotoXY (10,23) ; TextColor (white);
      Write ('Hit any key to begin the game'); Read (kbd,Ch);
      TextColor (yellow);
    END
1: BEGIN (SAT ACB Screen)
Gotoxy (9,1); TextColor (white);
Write ('SATELLITE/REUSABLE LAUNCHER ACB HELP SCREEN');
TextColor (Yellow);
GotoXY (1,3);
Write ('Enter a '1' from the Acquisition Phase Main Menu to ');
Write ('access the Satellite/Reusuable Launcher ACB Screen.');
Write ('The ACB Screen lists all the satellites, support elements,');
Write ('and reusable/recyclable launcher units currently in the');
Write ('acquisition pipeline. The units are listed by Name and');
Write ('Tail Number. The status headings on the screen');
Write ('(FEASIBLE, RESEARCH, DEVELOPMENT, and PRODUCTION)');
Write ('roughly correspond to the DOD Milestones 0,1,2, and 3.');
Write ('To update a unit, enter its name and tail number as they');
Write ('appear on the screen. All units in the database are in');
Write ('capital letters; so, put the 'Caps Lock' key on!');
Write ('The screen will show the current status of the units');
Write ('attributes (mass, when deployed etc), and will then give');
Write ('the update cost. Enter 'Y' if you want the unit to begin');
Write ('the next stage of the acquisition cycle. Each update takes');
Write ('two years to complete. An asterik is placed next to the');
Write ('entry, indicating the unit is being updated. To inject a');
Write ('copy of a satellite into the acquisition pipeline, enter the');
Write ('satellite name and a tail number of "0". The program will');
Write ('generate a copy of the unit and put it in the research phase');
GotoXY (19,23); TextColor (white);
Write ('Hit any key to continue');
Read (kbd, Ch);
ClrScr;
GotoXY (9,1); TextColor (white);
Write ('SATELLITE/REUSABLE LAUNCHER ACB HELP SCREEN');
GotoXY (1,3); TextColor (Yellow);
Write ('generate a copy of the unit and put it in the research phase.');
Write ('A unit is considered finished with the phase its listed under');
Write ('For example, DSCS II # 2 is listed in the Development Phase');
Write ('This means the unit has completed its development, and is');
Write ('waiting for authority to proceed onto the Production Phase');
Write ('The Screen displays the budget remaining in the current year');
Write ('The program will not allow you to spend more than you have');
Write ('A word of caution !! Do not spend your entire budget in the');
Write ('Acquisition Phase, It costs money to refurbish and re-launch');
Write ('the reusable launchers (ie. Shuttle)');
GotoXY (1,23); textColor (White);
Write ('Hit any key to continue to the Satellite/Reusable ACB Screen');
Read (kbd, Ch); END

2: Begin
GotoXY (15,1); TextColor (White);
Write ('EXPENDABLE LAUNCHER ACB SCREEN');
TextColor (Yellow); GotoXY (1,3);

180
Writeln ('Enter a "2" from Acquisition Phase Main Menu to access');
Writeln ('the Expendable Launcher ACB Screen. An Expendable Launcher');
Writeln ('(DELTA, ATLAS, etc) can only be used once. This screen displays');
WRiteln ('all the expendable launchers in the acquisition pipeline.');
Writeln ('The necessary technology has already been developed for');
Writeln ('the launcher units in the game; therefore, they skip the');
Writeln ('Feasible and Research Phases of the Acquisition Cycle.');
Writeln ('The screen displays the name and number of all the');
Writeln ('expendable launchers currently in the pipeline. It also');
Writeln ('shows the number available for launch.');
Writeln ('Each update takes two years to complete. The Transition');
Writeln ('Columns were added to help the player track the launchers');
Writeln ('through the acquisition cycle. The Development column is the');
Writeln ('only acquisition phase the player may modify. The computer');
Writeln ('automatically updates the other columns at the beginning of');
Writeln ('each year. To update the Development column, enter the');
Writeln ('launcher name. The computer will then ask how many of');
Writeln ('these missile types you want to update. ');
GotoXY (2,23) ; TextColor (white);
Write ('Hit any key to continue to Expendable Launcher ACB Screen');
Read (kbd,Ch) ; TextColor (yellow);
END

3 : BEGIN
GotoXY (10,1) ; TextColor (White);
Write ('Adding Survivability Features Help Screen');
TextColor (Yellow) ; GotoXY (1,2);
WRiteln ('Enter a '3' from the Acquisition Phase Main Menu to access');
Writeln ('the Add Survivability Features Screen. These features are');
Writeln ('modifications to the satellite to enable it operate better');
Writeln ('in a hostile environment. There are seven different features');
WRiteln ('Only satellites in the Feasible, Research, or Development');
WRiteln ('stages can be modified. Modifying Satellites in the');
Writeln ('Development stage are penalized due to the late requirements');
Writeln ('They have a choice of a 50% overrun on the mods or delaying');
Writeln ('the acquisition cycle of that unit by two years.');
WRiteln ('The cost of each feature is a function of the satellite type');
Writeln ('To incorporate a particular feature into the satellite, ');
Writeln ('enter the corresponding number. Features 5, 6, and 7 increase');
Writeln ('the weight of the satellite. If one of these features is ');
Writeln ('selected, the program will display the modified mass and ask');
Writeln ('for confirmation. If you change your mind about adding a ');
Writeln ('feature, enter the number again and it will be removed.');
Writeln ('The maneuver feature represent the number of maneuvers');
Writeln ('the satellite will be able to perform once it is been launched.');
Writeln ('The program will ask how many maneuvers you want. You may');
Writeln ('off-load fuel by entering a maneuver number less than');
Writeln ('currently on board. Watch the satellite mass closely!!!');
GotoXY (3,23) ; TextColor (white);
Write ('Hit any key to continue to the Add Survivability Screen');
Read (kbd,Ch) ; TextColor (Yellow);
END
A Composite Satellite is a system which has a piggyback system added on which performs its own mission. These add-on systems add 50% of their mass to the overall system. There is a maximum of 5 add-on systems on a satellite.

Once a system has been added on it cannot be taken off. Do not design a Composite Satellite too heavy to launch.

Choose a satellite to launch
Choose the launch location
Choose the launcher type
Choose the launch pad
IF NASA oks the launch,
then off it goes!

After choosing the launch pad, the computer gives you a opportunity to create a Multiple Launch. A Multiple Launch is performed by placing more than one payload onto the same launch vehicle. The extra payloads add 75% of their mass to the overall lift mass the launcher must place into orbit.

There can be no more than four payloads and they must be targeted for the same orbit altitude (GEO, VLEO, etc.).

For a satellite to successfully achieve orbit, the satellite must first be successfully launched. This is a function of the launcher reliability. If the launcher fails, the launcher and payload is destroyed, and the Pad is damaged ($10 Mill.).

If launched ok, the satellites must be successfully deployed
There is a set 97% chance of a successful deployment for each payload on board. If the payload fails to deploy, it is destroyed. If this happens, the launcher and other possible payloads are unaffected.
6 : BEGIN
Gotoxy (1,2) ;
writeln ('ADAM consists of 10 one-year turns. Each turn contains');
writeln ('the following phases:');
writeln ('');
writeln ('1. Read Scenario Phase');
writeln ('');
writeln ('2. Database Update Phase');
writeln ('');
writeln ('3. Acquisition Phase');
writeln ('');
writeln ('4. Assign Ground Support Phase');
writeln ('');
writeln ('5. Deployment Phase');
writeln ('');
writeln ('6. Scoring Phase');
GotoXY (19,23) ; TextColor (white) ;
Write ('Hit any key to continue') ; TextColor (Yellow) ;
Gotoxy (1,1) ;
writeln ('In the Read Scenario Phase, the Game Controller describes');
writeln ('the conditions the players must abide by for that turn.');
writeln ('');
writeln ('In the Database Update Phase, the computer advances the');
writeln ('status of units upgraded last turn. The computer also');
writeln ('performs a health status check of all active satellites.');
writeln ('');
writeln ('In the Acquisition Phase, the players give orders for units');
writeln ('to begin the next Milestone in the Acquisition Cycle.');
writeln ('');
writeln ('In the Ground Support Phase, the players assign the ground');
writeln ('support nodes required for all satellites before launch.');
writeln ('');
writeln ('In the Deployment Phase, the players match payloads with');
writeln ('launch vehicles and launch locations before launching.');
writeln ('');
writeln ('In the Scoring Phase, the Game Controller determines the');
writeln ('mission requirements the players have met in the turn');
writeln ('A point is assigned to each filled requirement. This phase');
writeln ('is optional and only used as a device for measuring the');
writeln ('effectiveness of the space force developed.');
Gotoxy (10,23) ; TextColor (White) ;
Write ('Hit any key to continue the game') ; TextColor (Yellow) ;
Read (kbd, Ch) ;
END ;

7 : BEGIN
GotoXY (14,1) ; TextColor (White) ;
Write ('ASSIGN GROUND SUPPORT HELP SCREEN') ;
All satellites must have their ground support elements filled before they may be launched. The ground support consists of three elements:

1. Ground Control
2. Data Processing
3. User Terminal

Some ground units can support more than one satellite. The more technologically advanced the ground unit is, the more satellites it can support.

OVERLAY PROCEDURE Check_If_Second_Copy (Root : UnitPtr ; WorkUnit : Unit) ;

VAR
   Current : UnitPtr ;
   ThisName : string10 ;
   ThisNumber : integer ;
   ThisMission : string10 ;
   Copy : boolean ;

BEGIN
   Copy := False ;
   Current := Root ;
   ThisName := WorkUnit.Name ;
   ThisNumber := WorkUnit.TailNumber ;
   ThisMission := WorkUnit.Mission ;

   WHILE (Current <> Nil) AND (Current^.Mission <> ThisMission) DO
      Current := Current^.Next ;

   WHILE (Current <> Nil) AND (Current^.Mission = ThisMission) AND (NOT Copy) DO
      BEGIN
         IF Current^.Name = ThisName THEN
            begin

184
IF (Current^.Status = 'PRODUCTION') OR (Current^.Status = 'READY') OR (Current^.Status = 'ACTIVE') OR (Current^.Status = 'SPARE') OR (Current^.Status = 'DEAD')
THEN Copy := True;
end;
Current := Current^.Next;
END;

IF Copy THEN WorkUnit.Status := 'DEVELOPMENT'
ELSE WorkUnit.Status := 'RESEARCH';
END;

(used by controller when units are added to database. Also used when)
(developing a new database. spots duplicate entries and asks if)
(controller wants to delete old entry)

OVERLAY PROCEDURE AddUnit (VAR Root : UnitPtr);

VAR
I : integer;
Abandon : boolean;
WorkUnit : Unit;
Last : UnitPtr;
Current : UnitPtr;
Ch : char;
Temp : string[10];
Holder : UnitPtr;
BEGIN
REPEAT (until user answers 'N' to 'MORE?' question ...)
ClrScr;
FillChar (WorkUnit, SizeOf (WorkUnit), CHR(0));
GotoXY (1,1);
WITH WorkUnit DO { fill record with good data }
BEGIN
Write ('>> Mission: '); Readln (Mission);
Write ('>> Name: '); Readln (Name);
IF Mission = 'LAUNCHER' THEN
BEGIN
Write ('>> Tail Number '); Readln (TailNumber);
Write ('>> Reusable Type: '); Readln (ReuseField);
IF (ReuseField = 'REUSABLE') OR (ReuseField = 'RECYCLABLE') OR (ReuseField = 'R') OR (ReuseField = 'C') THEN
BEGIN
Write ('>> Annual Sorties Left: '); Readln (Sorties);
Write ('>> System Launches Rem: '); Readln (RemUses);
Write ('>> Launch/Refurbish Cost: '); Readln (OMCost);
Write ('>> Status: '); Readln (Status);
END;
ELSE
BEGIN
  Write ('> > # Active: ') ; Readln (NumActive);
  Write ('> > # Prod-Active: ') ; Readln (ProdActive);
  Write ('> > # Production: ') ; Readln (NumProd);
  Write ('> > # Dev-Prod: ') ; Readln (DevProd);
  Write ('> > # Development: ') ; Readln (NumDev);
END;

Write ('> > Realability: ') ; Readln (Reliability);
Write ('> > Deployed: ') ; Readln (Deployed);
Write ('> > R&D Cost: ') ; Readln (RDCost);
Write ('> > ACQ Cost: ') ; Readln (ACQCost);
Write ('> > Mass to VLEO: ') ; Readln (VLEO);
Write ('> > Mass to LEO: ') ; Readln (LEO);
Write ('> > Mass to MOL: ') ; Readln (MOL);
Write ('> > Mass to MEO: ') ; Readln (MEO);
Write ('> > Mass to GEO: ') ; Readln (GEO);
Write ('> > Update: ') ; Readln (Update);
Write ('> > Tech Level: ') ; Readln (TechLevel);
END

ELSE IF Mission = 'GROUND' THEN
BEGIN
  Write ('> > TailNumber: ') ; Readln (TailNumber);
  Write ('> > LifeTime: ') ; Readln (LifeTime);
  Write ('> > Type: ') ; Readln (Kind);
  Write ('> > Deployed (Year): ') ; Readln (Deployed);
  Write ('> > R&D Cost (Mill $): ') ; Readln (RDCost);
  Write ('> > ACQ Cost (Mill $): ') ; Readln (ACQCost);
  Write ('> > O&M Cost (Mill $): ') ; Readln (OMCost);
  Write ('> > Status: ') ; Readln (Status);
  Write ('> > Supporting Tail Number ') ; Readln (SupportNumber);
  Write ('> > Update: ') ; Readln (Update);
  Write ('> > Tech Level: ') ; Readln (TechLevel);
END

ELSE IF Mission = 'PAD' THEN
BEGIN
  Write ('> > TailNumber: ') ; Readln (TailNumber);
  Write ('> > Location: ') ; Readln (Location);
  Write ('> > Status: ') ; Readln (Status);
  Write ('> > Update: ') ; Readln (Update);
  Write ('> > Deployed: ') ; Readln (Deployed);
  Write ('> > R&D Cost: ') ; Readln (RDCost);
  Write ('> > ACQ Cost: ') ; Readln (ACQCost);
  Write ('> > O&M Cost: ') ; Readln (OMCost);
  Write ('> > Launches/Year: ') ; Readln (LaunchRate);
  Write ('> > Launches/Rem: ') ; Readln (NumRemaining);
END

ELSE
BEGIN

END
Write ('>') TailNumber: ' '); Readln (TailNumber);  
Write ('>') Mass: ' '); Readln (Mass);  
Write ('>') LifeTime: ' '); Readln (LifeTime);  
Write ('>') Deployed (Year): ' '); Readln (Deployed);  
Write ('>') R&D Cost (Mill $): ' '); Readln (RDCost);  
Write ('>') ACQ Cost (Mill $): ' '); Readln (ACQCost);  
Write ('>') Status: ' '); Readln (Status);  
Write ('>') Orbit: ' '); Readln (Orbit);  
Write ('>') Inclination: ' '); Readln (Inclination);  
Write ('>') Ascending Node: ' '); Readln (AscendingNode);  
Write ('>') Update: ' '); Readln (Update);  
Write ('>') Tech Level: ' '); Readln (TechLevel);  
Write ('>') Controller: ' '); Readln (ControlName);  
Write ('>') Processor: ' '); Readln (ProcessorName);  
Write ('>') User: ' '); Readln (UserName);  
Write ('>') * # ' '); Readln (ControlNumber);  
Write ('>') Add-On : 1 ' '); Readln (AddOnName1);  
Write ('>') # ' '); Readln (AddOnNum1);  
Write ('>') Add-On: 2 ' '); Readln (AddOnName2);  
Write ('>') # ' '); read (AddOnNum2);  
GotoXY (40,1);  
Write ('>') Add-On : 3 ' '); Read (AddOnName3); GotoXY (40,2);  
Write ('>') # ' '); Read (AddOnNum3); GotoXY (40,3);  
Write ('>') Add-On : 4 ' '); Read (AddOnName4); GotoXY (40,4);  
Write ('>') # ' '); Read (AddOnNum4); GotoXY (40,5);  
Write ('>') Add-On: 5 ' '); Read (AddOnName5); GotoXY (40,6);  
Write ('>') # ' '); read (AddOnNum5); GotoXY (40,7);  
Write ('>') Anti-Jam : Data ')');  
Read (AntiJamDataLink); GotoXY (40,8);  
Write ('>') Low Anti-Jam : Corn ')');  
Read (LowAntiJamComLink); GotoXY (40,9);  
Write ('>') Med Anti-Jam : Corn ')');  
Read (MedAntiJamComLink); GotoXY (40,10);  
Write ('>') Hi Anti-Jam : Corn ')');  
read (HighAntiJamComLink); GotoXY (40,11);  
Write ('>') 15 Day Autonomy ')');  
Read (Autonomyl5); GotoXY (40,12);  
write ('>') 180 Day Autonomy ')');  
Read (Autonomy180); GotoXY (40,13);  
Write ('>') * of Manuvers ')');  
Read (Manuvers); GotoXY (40,14);  
END;  

IF Mission = 'COMM' THEN  
BEGIN  
Write ('>') # of Channels: ' ');  
Read (Channels);  
END;
IF Mission = 'WEATHER' THEN

BEGIN
    Write ('>> DownLink ');
    Read (DownLink); GotoXY (40,15);
    Write ('>> DayVideo ');
    Read (DayVideo); GotoXY (40,16);
    Write ('>> NightVideo ');
    Read (NightVideo); GotoXY (40,17);
    Write ('>> DayImage ');
    Read (DayImage); GotoXY (40,18);
    Write ('>> NightImage ');
    Read (NightImage); GotoXY (40,19);
    Write ('>> Temp ');
    Read (Temp); GotoXY (40,20);
    Write ('>> Solar ');
    Read (Solar);
END;

IF Mission = 'NAV' THEN

BEGIN
    Write ('>> Accuracy (#) ');
    Read (Accuracy); GotoXY (40,15);
    Write ('>> Position (real) ');
    Read (Position); GotoXY (40,16);
    Write ('>> Redundancy (#) ');
    Read (Redundancy);
END;

IF Mission = 'IMINT' THEN

BEGIN
    Write ('>> DownLink ');
    Read (DLink); GotoXY (40,15);
    Write ('>> DayVideo ');
    Read (DVideo); GotoXY (40,16);
    Write ('>> NightVideo ');
    Read (NVideo); GotoXY (40,17);
    Write ('>> DayImage ');
    Read (DImage); GotoXY (40,18);
    Write ('>> NightImage ');
    Read (NImage);
END;

IF Mission = 'SIGINT' THEN

BEGIN
    Write ('Data Link ');
    Read (DataLink); GotoXY (40,15);
    Write ('Signal Identidy ');
    Read (Signal); GotoXY (40,16);
END;

IF Mission = 'WARNING' THEN
BEGIN
    Write ('>> Aircraft (S/M/L) ') ;
    Read (Aircraft) ; GotoXY (40,15) ;
    Write ('>> ICBM (Y/N) ') ;
    Read (ICBM) ; GotoXY (40,16) ;
    Write ('>> SLBM (Y/N) ') ;
    Read (SLBM) ; GotoXY (40,17) ;
    Write ('>> Target (char) ') ;
    Read (Target) ; GotoXY (40,18) ;
    Write ('>> Count (char) ') ;
    Read (Count) ;
END ;

IF Mission = 'OFFENSE' THEN
BEGIN
    Write ('>> Low Earth Orbit ') ;
    Read (LowOrbit) ; GotoXY (40,15) ;
    Write ('>> Medium Earth Orbit ') ;
    Read (MedOrbit) ; GotoXY (40,16) ;
    Write ('>> High Earth Orbit ') ;
    Read (HighOrbit) ; GotoXY (40,17) ;
    Write ('>> Ground Targets ') ;
    Read (GroundTarget) ; GotoXY (40,18) ;
    Write ('>> Air Craft ') ;
    Read (Planes) ; GotoXY (40,19) ;
    Write ('>> Ships ') ;
    Read (Ships) ; GotoXY (40,20) ;
    Write ('>> Time Urgent ') ;
    Read (TimeUrgent) ;
END ;

IF Mission = 'DEFENSE' THEN
BEGIN
    Write ('>> Boost Phase ') ;
    Read (Boost) ; GotoXY (40,15) ;
    Write ('>> MidCourse Phase ') ;
    Read (MidCourse) ; GotoXY (40,16) ;
    Write ('>> Reentry Phase ') ;
    Read (Reentry) ; GotoXY (40,17) ;
    Write ('>> Air Craft ') ;
    Read (AC) ; GotoXY (40,18) ;
    Write ('>> Cruise Missiles ') ;
    Read (CruiseMissile) ;
END ;

END ;
Abandon := False ;

{ look for duplicates here }

IF Root = Nil THEN { if list is empty point Root to unit }
BEGIN

189
New (Root) ;
WorkUnit.Next := Nil ; ( ensure list is terminated by Nil )
Root^ := WorkUnit ; Head := Root ;
END
ELSE ( ... if theres somethin in the list already )
BEGIN
Current := Root ; ( start traverse at Root of list )
REPEAT
IF (Current^.Name = WorkUnit.Name)
AND (Current^.TailNumber = WorkUnit.TailNumber) THEN
{ duplicate found }
BEGIN
ShowUnit (Current^) ;
ClearLines (23,25) ;
Center-Line (24,
'Duplicate entry exists, delete old record (Y/N)') ;
IF Yes THEN Abandon := True ELSE Abandon := False ;
END ;
IF Abandon THEN
BEGIN
Holder := Current^.Next ;
Dispose (Current) ;
New (Last^.Next) ;
WorkUnit.Next := Holder ;
Last^.Next^ := WorkUnit ;
END
ELSE
BEGIN
Last := Current ;
Current := Current^.Next
END ;
UNTIL (Current = Nil) OR Abandon OR
(Current^.Mission >= WorkUnit.Mission) OR
((Current^.Mission = WorkUnit.Mission) AND
 (Current^.Name >= WorkUnit.Name)) ;
{ sorts list by mission and name}
IF NOT Abandon THEN { add UnitRec to linked list }
IF (Root^.Mission > WorkUnit.Mission) AND
 (Root^.Name > WorkUnit.Name) THEN { new root item }
BEGIN
New (Root) ; { create new dynamic Unit }
WorkUnit.Next := Last ; { point new unit at old root }
Root^ := WorkUnit { point new root at workUnit }
END
ELSE
BEGIN
NEW (Last^.Next) ; { create new dynamic Unit }
WorkUnit.Next := Current ; { points its next to current }
Last^.Next^ := WorkUnit ; { and assign WorkUnit to it }
END ;
GotoXY (1,24) ; Write ('')) Add another unit to list? (Y/N): ') ;
Readln (Ch) ;
UNTIL (Ch = 'N') OR (Ch = 'n') ;

END ;

OVERLAY PROCEDURE Add_Copy (Root : UnitPtr ; Var WorkUnit : Unit) ;

VAR
  Current,Present, Last : UnitPtr ;
  Number : integer ;

BEGIN
  ClrScr ;
  Current := Root ;
  Number := 0 ;
  WHILE (Current <> Nil) AND (Current^.Mission <> WorkUnit.Mission) DO
    Current := Current^.Next ;
  REPEAT
    IF (Current^.Name = WorkUnit.Name) AND (Current^.TailNumber > Number)
    THEN Number := Current^.TailNumber ;
    Last := Current ;
    Current := Current^.Next ;
  UNTIL (Current = Nil) OR (Current^.Mission <> WorkUnit.Mission) ;

  New (Last^.Next) ;
  Last^.Next^.Next := WorkUnit ;
  Last^.Next^.TailNumber := Number + 1 ;
  Search (Root, WorkUnit.Name, Number + 1, 'YES') ;
  Where^.Status := 'RESEARCH' ;
  IF Where^.Mission = 'PAD' THEN
    BEGIN
      ClearLines (20,25) ;
      GotoXY (24,20) ;
      TextColor (White) ;
      Write ('WHERE DO YOU WANT TO BUILD ',
             Where^.Name:10, Where^.TailNumber:3) ;
      GotoXY (21,22) ;
      Write ('Enter V for Vandenberg or C for Canaveral ') ;
      Read (kbd,Ch) ;
      Case Ch OF
        'V', 'v' : Where^.Location := 'VANDENBERG' ;
        'C', 'c' : Where^.Location := 'CANAVERAL' ;
      END ;
    END ;
  END ;
Check_If_Second_Copy (Root, Where) ;
END ;

OVERLAY PROCEDURE Delete_Unit (VAR Root : UnitPtr ; ThisName : string10 ;
                                  ThisNumber : integer) ;

VAR
    Holder : UnitPtr ;
    Current : UnitPtr ;
    Last : UnitPtr ;

BEGIN
    Current := Root ;
    Last := Root ;

    WHILE (Current <> Nil) AND ((Current^.Name <> ThisName)
                                OR (Current^.TailNumber <> ThisNumber)) DO
    BEGIN
        Last := Current ;
        Current := Current^.Next ;
    END ;

    IF (Current^.Name = ThisName) AND (Current^.TailNumber = ThisNumber) THEN
    BEGIN
        Holder := Current^.Next ;
        Dispose (Current) ;
        Last^.Next := Holder ;
    END ;

END ;

(shows the entire database, entry by entry)

OVERLAY PROCEDURE ViewList (Root : UnitPtr) ;

VAR
    WorkFile : UnitFile ;
    Current : UnitPtr ;
    Last : UnitPtr ;

BEGIN
    Current := Root ;

    IF Root = Nil THEN  { nothing is now in list }
    BEGIN
        Clrscr ;
        Borderl ;

    END ;
Center_Line (12, '<< Your list is empty! >> ') ;
Center_Line (14, ' Press ANY KEY to continue ') ;
Readln (kbd, Ch) ;
END
ELSE
BEGIN

Current := Root ;

REPEAT
    ShowUnit (Current") ;
    ClearLines (24,25) ;
    Center_Line (25,
        '>> Press <CR> to view next unit in list or "Q" to quit') ;
    Read (kbd, Ch) ;
    Current := Current^.Next ;
    UNTIL (Ch IN ['Q','q']) OR (Current = Nil) ;
END ;
END ;

{used to save current database to disk}

OVERLAY PROCEDURE SaveList (Root : UnitPtr) ;

VAR
    WorkName : string [30] ;
    WorkFile : UnitFile ;
    Current : UnitPtr ;
    I : integer ;

BEGIN
    TextBackground (1) ;
    ClrScr ;
    Border1 ;
    TextBackground (1) ;
    Center_Line (12, 'Enter filename for saving the database ') ;
    Readln (WorkName) ;
    Assign (WorkFile,WorkName) ; ( open file for write access )
    Rewrite (WorkFile) ;
    Current := Root ;
    WHILE Current <> Nil DO  ( traverse and write )
        BEGIN
            Write (WorkFile,Current") ;
            Current := Current^.Next
        END ;
    Close (WorkFile)
END ;

{Called beginning of each turn to determine status of all active}
OVERLAY PROCEDURE Check_Failure (VAR Root : UnitPtr) ;

VAR
  Age : integer ;
  Current : UnitPtr ;
  FailRate : real ;
  Check : integer ;
  Line : integer ;
  AtLeastOneFailed : boolean ;

BEGIN
  TextBackground (1) ;
  Center_Line (12, ' Center_Line (14, ');
  AtLeastOneFailed := False ;
  Line := 6 ;
  Current := Root ;
  TextColor (White) ;
  While Current <> Nil DO
    BEGIN
      IF (Current^.Status = 'ACTIVE') AND (Current^.Mission <> 'LAUNCHER') AND
          (Current^.Mission <> 'PAD') AND (Current^.Mission <> 'GROUND') THEN
        BEGIN
          Age := Year - Current^.Deployed ;
          Check := Current^.LifeTime - Age ;

          IF Check < 0 THEN FailRate := 1.0
          ELSE IF Check = 0 THEN FailRate := 0.58
          ELSE IF Check < 2 THEN FailRate := 0.08
          ELSE FailRate := 0.003 ;

          IF Random < FailRate THEN
            BEGIN
              GotoXY (25,Line) ;
              Write ('System ', Current^.Name:10, Current^.TailNumber:3,
                ' Failed at Age ', Age) ;
              Current^.Status := 'DEAD' ;
              AtLeastOneFailed := True ;
              Line := Line + 1 ;
            END ;
        END ; (status ACTIVE and Mission <> Launcher/Pad/Ground)
      Current := Current^.Next ;
    END ; (while not nil)
    IF AtLeastOneFailed THEN
      BEGIN
        GotoXY (30,23) ;
        WRITE ('Hit any Key to continue') ; Read (kbd,Ch) ;
      END ;
    TextColor (Yellow) ;
    Textbackground (1) ;
OVERLAY PROCEDURE Check_Budget (VAR WorkUnit : Unit) ;

VAR
  Cost : integer ;
BEGIN

  WITH WorkUnit DO
  BEGIN
    IF (Status = 'FEASIBLE') OR (Status = 'RESEARCH')
    THEN Cost := Round (RDCost / 2)
    ELSE Cost := ACQCost ;

    IF (BUDGET - Cost < 0 ) THEN
      BEGIN
        ClearLines (23,25)
        GotoXY (15,23)
        Write ('cannot afford to buy/update ',
        Name:10, ' check budget')
       TextColor (White)
        Center_Line (25, 'Hit Any Key to continue') ; read (kbd,Ch)
        textColor (Yellow)
      END ;

    IF (BUDGET - Cost >= 0 ) THEN
      BEGIN
        IF Update = 99 THEN Update := 4 (late surviv. mods)
        ELSE Update := 2 ;
        BUDGET := BUDGET - Cost ;
      END ;
  END ;
END ;

FUNCTION Max (A,B,C,D,E : integer) : integer ;

VAR
  Temp : integer ;
BEGIN
  IF A > B THEN Temp := A ELSE Temp := B ;
  IF Temp < C THEN Temp := C ;

  (called when updating units on ACB to determine affordability)
  (called during Count_Systems to determine where to begin writing)
  (next entry on ACB Screen)
IF Temp < D THEN Temp := D ;
IF Temp < E THEN Temp := E ;
Max := Temp ;
END ;

;display all the systems on the ACB)

PROCEDURE Count_Systems (VAR Root : UnitPtr) ;

VAR
  Current : UnitPtr ;
  ThisMission : string10 ;
  ThisStatus : string11 ;
  BuyLine, RD1Line, RD2Line, ACQLine, LAULine : integer ;
  NextLine : integer ;
  ReusableLV : boolean ;
  Updated : string [2] ;
  ThisName : string [12] ;

BEGIN
  Updated := 'x ' ;
  ClearLines (7,25) ;
  Current := Root ;
  NextLine := 8 ;
  WHILE (Current <> Nil) AND (NextLine < 23) DO
    BEGIN
      IF (Current’.Mission = 'LAUNCHER') AND
      ((Current’.ReuseField = 'REUSABLE') OR
       (Current’.ReuseField = 'RECYCLABLE') OR
       (Current’.ReuseField = 'R') OR (Current’.ReuseField = 'C'))
      THEN ReusableLV := True
      ELSE ReusableLV := False ;

      IF (Current’.Mission <> 'LAUNCHER') OR (ReusableLV) THEN
        BEGIN
          ThisMission := Current’.Mission ;
          BuyLine := NextLine ; RD1Line := NextLine ; RD2Line := NextLine ;
          ACQLine := NextLine ; LAULine := NextLine ;
          GotoXY (3,NextLine) ; Write (ThisMission) ;
          GotoXY (23, NextLine) ; Write ('-') ;
          GotoXY (40,NextLine) ; Write ('-') ;
          GotoXY (56,NextLine) ; Write ('-') ;
          GotoXY (71,NextLine) ; Write ('-') ;
          WHILE ThisMission = Current’.Mission DO
            BEGIN
              ThisStatus := Current’.Status ;
              IF ThisStatus = 'FEASIBLE' THEN

              END ;
            END ;
          END ;
        END ;
      END ;
    END ;
  END ;
END ;
BEGIN
  GotoXY (15, BuyLine) ;
  IF (Current^.Update > 0) AND (Current^.Update <> 99)
    THEN ThisName := Updated + Current^.Name
  ELSE ThisName := Current^.Name ;
  Write (ThisName:12, Current^.TailNumber:3) ;
  BuyLine := BuyLine + 1 ;
END ;

IF ThisStatus = 'RESEARCH' THEN
  BEGIN
    GotoXY (31, RD1Line) ;
    IF (Current^.Update > 0) AND (Current^.Update <> 99)
      THEN ThisName := Updated + Current^.Name
    ELSE ThisName := Current^.Name ;
    Write (ThisName:12, Current^.TailNumber:3) ;
    RD1Line := RD1Line + 1 ;
  END ;

IF ThisStatus = 'DEVELOPMENT' THEN
  BEGIN
    GotoXY (47, RD2Line) ;
    IF (Current^.Update > 0) AND (Current^.Update <> 99)
      THEN ThisName := Updated + Current^.Name
    ELSE ThisName := Current^.Name ;
    Write (ThisName:12, Current^.TailNumber:3) ;
    RD2Line := RD2Line + 1 ;
  END ;

IF ThisStatus = 'PRODUCTION' THEN
  BEGIN
    GotoXY (63, ACQLine) ;
    IF (Current^.Update > 0) AND (Current^.Update <> 99)
      THEN ThisName := Updated + Current^.Name
    ELSE ThisName := Current^.Name ;
    Write (ThisName:12, Current^.TailNumber:3) ;
    ACQLine := ACQLine + 1 ;
  END ;

  Current := Current^.Next ;
END ; (Mission = ThisMission)

END (Mission is sat or resuable)
ELSE Current := Current^.Next ;

NextLine := Max (BuyLine, RD1Line, RD2Line, ACQLine, LAULine) ;
NextLine := NextLine + 2 ;

END ; (Nil and line check)

IF (NextLine >= 23) AND (Current <> Nil) THEN
  BEGIN
    ClearLines (23,25) ;
    Center_Line (25, 'Enter 'C' to continue listing') ;
IF Continue THEN Count_Systems (Current) ;
END ;

IF Current = Nil THEN
BEGIN
    ClearLines (23,25) ;
    Center_Line (25, 'Hit "C" to restart listing or <CR> to begin updating') ;
    IF Continue THEN Count_Systems (Head) ;
END ;

END ;

{displays ACB screen and asks for updates }
{determines if player wants to add another copy of a unit. player does}
{this by entering a tail number of 0'}

OVERLAY PROCEDURE Display_Satellite_ACB_Screen (Root : UnitPtr ) ;

VAR
Update : boolean ;
OnACB : boolean ;
AllReadyUpdated : boolean ;
Cost : integer ;
Copy : boolean ;

BEGIN
    REPEAT
        Copy := False ;
        AllReadyUpdated := False ;
        OnACB := True ;
       ClrScr ;
        Center_Line (2, 'SATELLITE ACQUISITION BOARD') ;
        GotoXY (63,2) ; Write ('YEAR: ', Year:4) ;
        Center_Line (3, '* Indicates Unit Has Been Updated') ;
        GotoXY (33,4) ; Write ('BUDGET: ', BUDGET) ;
       TextColor (White) ;
        GotoXY (4,6) ; Write ('MISSION') ;
        GotoXY (20,6) ; Write ('FEASIBLE') ;
        GotoXY (37,6) ; Write ('RESEARCH') ;
        GotoXY (52,6) ; Write ('DEVELOPMENT') ;
        GotoXY (68,6) ; Write ('PRODUCTION') ;
       TextColor (Yellow) ;
        Writeln ;
        Update := True ;

        Count_Systems (Root) ;
        ClearLines (23,25) ;

        Center_Line (25, 'Hit "Q" if you do not want to update units ') ;
        Read (kbd,Ch) ;
IF (Ch = 'Q') OR (Ch = 'q') THEN Update := False;
IF Update THEN BEGIN

ClearLines (23,25);
Center_Line (24, 'Enter name of unit to be updated: (DSCS II, Eagle etc)');
Center_Line (25, 'Enter tail number: (0 will update "Feasible" unit) ');
GotoXY (68,24); Read (SearchName);
SearchNumber := getInt (StringNumber, 68,25);

IF SearchNumber = 0 THEN {adding copy}
BEGIN
Search (Root, SearchName, SearchNumber, 'NO');
IF Found THEN BEGIN
Copy := True;
Add_Copy (Root, Where^);
END;
END {searchnumber = 0}
ELSE Search (Root, SearchName, SearchNumber, 'YES');{already on ACB}

IF Found THEN BEGIN
IF (Where^.Update > 0) THEN BEGIN
ClearLines (23,25);
Center_Line (23, 'Unit is currently being upgraded !!')
Center_Line (25, 'Hit any key to continue'); Read (kbd,Ch);
AllReadyUpdated := True;
END;

IF (Where^.Status <> 'FEASIBLE') AND (Where^.Status <> 'RESEARCH')
AND (Where^.Status <> 'DEVELOPMENT') THEN BEGIN
ClearLines (23,25);
Center_Line (23, 'Unit is not in the Acquisition Process')
Center_Line (25, 'Hit any key to continue'); Read (kbd,Ch);
OnACB := False;
END;

IF (NOT AllReadyUpdated) AND OnACB THEN BEGIN
ClearLines (24,25);
IF (Where^.Status = 'FEASIBLE') OR
(Where^.Status = 'RESEARCH') THEN
Cost := Where^.RDCost Div 2
ELSE Cost := Where^.ACQCost;
GotoXY (20,24);
Write ('Updating this unit will cost ', Cost, ' Million');
Center_Line (25, 'Do you wish to upgrade this unit (Y/N)? ');
IF Yes THEN Check_Budget (Where^)
ELSE IF Copy

199
BEGIN

TextBackground (1);
ClrScr;
Current := Root;
Update := True;
CenterLine (2, 'EXPENDABLE LAUNCHER ACQUISITION SCREEN');
GotoXY (60, 3); Write ('YEAR:', Year:5);
GotoXY (33, 4); Write ('BUDGET:', BUDGET);
TextColor (White);
GotoXY (30, 6); Write ('DEV-PROD');
GotoXY (57, 6); Write ('PROD-ACTIVE');
GotoXY (6, 7); Write ('NAME');
GotoXY (14, 7); Write ('DEVELOPMENT');
GotoXY (29, 7); Write ('TRANSITION');
GotoXY (43, 7); Write ('PRODUCTION');
GotoXY (58, 7); Write ('TRANSITION');
GotoXY (73, 7); Write ('ACTIVE');
TextColor (Yellow);
GotoXY (1, 9);

While (Current <> Nil) AND (Current^.Mission <> 'LAUNCHER') DO
  Current := Current^.Next;

While (Current <> Nil) AND (Current^.Mission = 'LAUNCHER') DO
  BEGIN
    IF (Current^.ReuseField = '') OR (Current^.ReuseField = 'NO') THEN
      Writeln (Current^.Name:10, Current^.NumDev:10, Current^.DevProd:15,
                Current^.NumProd:13, Current^.ProdActive:13,
                Current^.NumActive:15);
    END;
  Current := Current^.Next;
END;

ClearLines (23, 25);
Center_Line (23, 'Hit 'Q' if you do not want to update any launchers') ;
Read (kbd,Ch) ;
IF Ch IN ['Q', 'q'] THEN Update := False ;
IF Update THEN
BEGIN
GotoXY (5,23) ;
Write ( 'Enter name of launcher to be upgraded (ATLAS, TITAN 34D etc): ' ) ;
Read (LauncherName) ;
Search (Root, LauncherName, 1, 'NO') ;
IF (Found) AND ((Where^.ReuseField = '') OR (Where^.ReuseField = 'NO')) THEN
begin
GotoXY (20,24) ;
Write ('Enter # of ', LauncherName, ' to be put in production ') ;
Number := GetInt (StringNumber, 65,20) ;
IF (Where^.NumDev - Number) < 0 THEN
BEGIN
ClearLines (23,25) ; GotoXY (25,24) ;
Write ('Dont have that many ', LauncherName, 's') ;
Center_Line (25, 'Hit any key to continue') ; Read (kbd,Ch) ;
END
ELSE
BEGIN
Cost := Where^.ACQCost ;
IF BUDGET - Cost < 0 THEN
BEGIN
ClearLines (23,25) ; GotoXY (20,24) ;
Write ('cant afford to update ',Number, SearchName:10) ;
Center_Line (25, 'Please hit <CR> to continue') ; Read ;
END
ELSE
BEGIN
BUDGET := BUDGET - Cost ;
Where^.Update := 0 ;
END ; (within budget)
END ; (Not toomany)
END ; (Found and expendable)
END ; (Update)
END ;

(used by controller if he wants to change the status of a unit)

OVERLAY PROCEDURE Change_Status (Root : UnitPtr) ;

VAR
ThisStatus : string11 ;

201
BEGIN

TextBackground (1);
ClrScr;
Center.Line (2, 'CHANGE SATELLITE STATUS');
Center.Line (12, 'Enter name of unit ');
Center.Line (14, 'Enter tail number ');
GotoXy (50,12); Read (SearchName);
SearchNumber := GetInt (StringNumber, 50,14);
Search (Root, SearchName, SearchNumber, 'YES');
IF Found THEN
  begin
  ClearLines (23,25);
  Center.Line (24, 'Enter new status (FEASIBLE, RESEARCH ,etc ') ;
  Read (ThisStatus);
  Where^.Status := ThisStatus ;
  END ; (if found)
END ;

(called from Ground_Track, displays the ground swaths of satellites)
(each satellite has own FOV. also function of altitude)
(data files contain pre-calculated ground points)

OVERLAY Procedure Trace (Orbit : string5 ; Inc : integer ;
  Ascending : integer ; Mission : string10) ;

VAR
  First : boolean ;
  FOV : integer ;
  X,Y,X1,Y1,X2,Y2 : integer ;
  Bottom, Top, Right, Left : integer ;
  Lat, Lon, TempLon : real ;
  Ch : char ;
  Trace : text ;
  Stop : boolean ;
  Radius : integer ;

BEGIN
  Radius := 75 ;

  IF Orbit = 'GEO' THEN
    BEGIN
      Ascending := Round (Ascending * 0.9 + 160) ;
      Circle (Ascending, 100, Radius, 2) ;
      (check for off screen coverage)
      IF Ascending >= 245
        THEN Circle (Ascending - 320, 100, Radius, 2) ;
      IF Ascending <= 75
        THEN Circle (Ascending + 320, 100, Radius, 2) ;
    END
  ELSE
    BEGIN

IF (Orbit = 'VLEO') OR (Orbit = 'LEO') THEN
BEGIN
  Case Inc OF
    10..40 : Assign (Trace, 'A:VLEO30.DAT');
    41..70 : Assign (Trace, 'A:VLEO60.DAT');
    71..95 : Assign (Trace, 'A:VLEO95.DAT');
END ; (Case)
END ; (VLEO)

IF Orbit = 'MEO' THEN
BEGIN
  Case Inc OF
    10..40 : Assign (Trace, 'A:MEO30.DAT');
    41..70 : Assign (Trace, 'A:MEO60.DAT');
    71..95 : Assign (Trace, 'A:MEO95.DAT');
END ; (Case)
END ; (MEO)

IF (Orbit = 'MOL') THEN Assign (Trace, 'A:MOL.DAT');

Reset (Trace);
First := True;

IF Mission = 'NAV' THEN
BEGIN
  IF Orbit = 'VLEO' THEN FOV := 5;
  IF Orbit = 'LEO' THEN FOV := 10;
  IF Orbit = 'MEO' THEN FOV := 20;
END;

IF Mission = 'WEATHER' THEN
BEGIN
  IF Orbit = 'VLEO' THEN FOV := 6;
  IF Orbit = 'LEO' THEN FOV := 12;
  IF Orbit = 'MEO' THEN FOV := 3;
END;

IF Mission = 'IMINT' THEN
BEGIN
  IF Orbit = 'VLEO' THEN FOV := 5;
  IF Orbit = 'LEO' THEN FOV := 3;
  IF Orbit = 'MEO' THEN FOV := 2;
END;
IF Mission = 'SIGINT' THEN
BEGIN
  IF Orbit = 'VLEO' THEN FOV := 10;
  IF Orbit = 'LEO' THEN FOV := 5;
  IF Orbit = 'MEO' THEN FOV := 3;
END;

IF FOV = 0 THEN FOV := 5;  \{ensure program doesn't bomb\}

While NOT EOF (Trace) DO
BEGIN
  Readln (Trace, Lat, Lon);

  IF Lat >= -60 THEN
  BEGIN
    Lon := Lon + Ascending;
    Templon := Lon;
    IF (Lon >= 180) AND (Lon <= 360) THEN Templon := Lon - 360;
    IF (Lon <= -180) AND (Lon >= -360) THEN Templon := Lon + 360;

    Lon := Templon;

    Y := Round (120 - Lat * 1.2);
    X := Round (Lon * 0.9 + 160);

    IF abs (X1 - X) > 100 THEN First := True;

    IF First THEN
    BEGIN
      X1 := X;
      X2 := X;
      Y1 := Y;
      Y2 := Y;
      First := False;
    END;

    IF Orbit <> 'MOL' THEN
    BEGIN
      Circle (X,Y,FOV,2);
      X1 := X; Y1 := Y;
    END \{not MOL\}
    ELSE
    BEGIN
      Draw (X1,Y1,X,Y,2);
      X1 := X;
      Y1 := Y;
    END \{MOL\}
  END;

END; \{lat > -55\}
END ; (EOF)

Close (Trace) ;

END ; (Not comm)

END ;

(called from main at game start to draw global picture. stores it in)
(global 'world' array)

OVERLAY PROCEDURE Draw_World ;

VAR
Start : boolean ;
Outline : text ;
Lat, Lon : real ;
X,Y,X1,Y1 : integer ;

BEGIN
ClrScr ;
GraphColorMode ;
write ('loading world data');
Start := True ;
Assign (Outline, 'A:World.dat') ;
Reset (Outline) ;

While NOT EOF (Outline) DO
BEGIN
Readln (Outline, Lat, Lon) ;
IF Lat > -55.0 THEN
BEGIN
IF (Lat <> 99) OR (Lon <> 99) THEN
BEGIN
Lon := - Lon ;
Y := Round (120 - Lat * 1.2) ;
X := Round (Lon * 0.9 + 160) ;
IF Abs (X1 - X) > 100 Then Start := True ;
IF Start THEN
BEGIN
X1 := X ;
Y1 := Y ;
Start := False ;
END ;
Draw (X1,Y1,X,Y,1) ;
X1 := X ;
Y1 := Y ;
END
ELSE Start := True ;
END ; ( < -55)
(Orbit_Correction_Factor calculates the correction factor as a function) (of launcher type and launch inclination. The shuttle types have worse) (adjust factor. This factor is multiplied by lift weight of launcher to) (determine the mass-to-orbit capability)

FUNCTION Orbit_Correction_Factor
  (LauncherName : string10 ; Inclination : integer) : real ;

VAR
  VandWindow : Window ;
  CapeWindow : Window ;
  InRange : boolean ;
  Factor : real ;
  ShuttleType : boolean ;
BEGIN
  ShuttleType := False ;
  Factor := 1.0 ;
  VandWindow := [65..122] ;
  CapeWindow := [29..48] ;
  InRange := False ;
  IF (LaunchSite = 'VANDENBERG') AND (Inclination IN VandWindow)
     THEN InRange := True ;
  IF (LaunchSite = 'CANAVERAL') AND (Inclination IN CapeWindow)
     THEN InRange := True ;
  IF LauncherName = 'SPACE TAXI' THEN Factor := 1.0 ;
  IF (Pos ('STS', LauncherName) <> 0) OR
    (Pos ('Shuttle', LauncherName) <> 0) THEN
    BEGIN
      IF InRange THEN Factor := 0.7
      ELSE Factor := 0.5 ;
      ShuttleType := True ;
    END ;
  IF (NOT ShuttleType) AND (LauncherName <> 'SPACE TAXI') THEN
    BEGIN
      IF InRange THEN Factor := 0.9
      ELSE Factor := 0.7 ;
    END ;
  END ;
(called by the launch procedure when a launch or deployment fails)
(leaves unit in database but marks it as dead)

PROCEDURE Kill_Unit (Root : UnitPtr ; VAR Name : string10 ;
                      Number : integer ; Blank : string5) ;

BEGIN
    Search (Root, Name, Number, 'NO') ;
    IF Found THEN
        BEGIN
            Where^.Status := 'DEAD' ;
            IF Blank = 'YES' THEN Name := '' ;
        END ;
END ;

(called by the launch procedure when a deployment succeeds)
(set status to active)

PROCEDURE Activate_Unit (VAR Root : UnitPtr ; VAR Name : string10 ;
                         Number : integer ; Blank : string5) ;

BEGIN
    Search (Root, Name, Number, 'NO') ;
    IF Found THEN
        BEGIN
            Where^.Status := 'ACTIVE' ;
            Where^.Deployed := Year ;
            IF Blank = 'YES' THEN Name := '' ;
        END ;
END ;

(one of main routines, determines launch and deployment successes)
(takes care of main load and any additional payloads or add-ons)

OVERLAY PROCEDURE Launch (Root : UnitPtr ; VAR WorkUnit : Unit) ;

VAR
    Current : UnitPtr ;
    LaunchProb : real ;
    DeployProb : real ;

BEGIN
    Current := Root ;

    While (Current <> Nil) AND ((Current^.Name <> LiftVehicle) OR
                                (Current^.TailNumber <> LiftVehicleNum)) DO
        Current := Current^.Next ;

    WITH WorkUnit DO
    BEGIN

    END ;
IF Current <> Nil THEN
BEGIN
  LaunchProb := Random;

  IF LaunchProb > Current^.Reliability THEN
  BEGIN
    ClrScr;
    TextBackground (1); Delay (800); ClrScr;
    TextBackground (2);
    ClrScr; Delay (800); TextBackground (3); ClrScr;
    Delay (800);
    Border1; Border2;
    TextColor (White);
    GotoXY (20,10); Write (Where^.Name, Where^.TailNumber:3,
    'EXPLODES ON THE PAD !!!!!!!!
    '); Center_Line (14, 'PayLoad and Launcher are completely destroyed');
    TextColor (Yellow);
    Center_Line (20, 'Please hit any key to continue');
    Read (kbd,Ch);
    Search (Root, LaunchPad, LaunchPadNum, 'NO');
    IF Found THEN
    BEGIN
      IF BUDGET > 10 THEN BUDGET := BUDGET - 10
      ELSE BUDGET := 0;
      END;
      IF PayLoad1 <> '' THEN
        Kill_Unit (Root, PayLoad1, Num1, 'YES');
      IF PayLoad2 <> '' THEN
        Kill_Unit (Root, PayLoad2, Num2, 'YES');
      IF PayLoad3 <> '' THEN
        Kill_Unit (Root, PayLoad3, Num3, 'YES');
      IF PayLoad4 <> '' THEN
        Kill_Unit (Root, PayLoad4, Num4, 'YES');
      IF AddOnName1 <> '' THEN
        Kill_Unit (Root, AddOnName1, AddOnNum1, 'NO');
      IF AddOnName2 <> '' THEN
        Kill_Unit (Root, AddOnName2, AddOnNum2, 'NO');
      IF AddOnName3 <> '' THEN
        Kill_Unit (Root, AddOnName3, AddOnNum3, 'NO');
      IF AddOnName4 <> '' THEN
        Kill_Unit (Root, AddOnName4, AddOnNum4, 'NO');
      IF AddOnName5 <> '' THEN
        Kill_Unit (Root, AddOnName5, AddOnNum5, 'NO');
      WorkUnit.Status := 'DEAD';
  END;

208
END ; (launch prob > reliability)

IF LaunchProb <= Current^.Reliability THEN
BEGIN
  TextBackground (1);
  ClrScr;
  Border1; TextBackground (1);
  Center_Line (12, 'Successful launch');
  Delay (2000);
  DeployProb := Random;
  IF DeployProb > 0.97 THEN
  BEGIN
    TextBackground (1); ClrScr; Delay (800);
    TextBackground (2); ClrScr; Delay (800);
    TextBackground (3); ClrScr; Delay (800);
    Center_Line (12, 'Deployment Failure');
    Center_Line (14, 'Satellite is lost');
    Delay (2000);
    Status := 'DEAD';
    IF PayLoad1 <> ''
    THEN Kill_Unit (Root, PayLoad1, Num1, 'YES');
    IF PayLoad2 <> ''
    THEN Kill_Unit (Root, PayLoad2, Num2, 'YES');
    IF PayLoad3 <> ''
    THEN Kill_Unit (Root, PayLoad3, Num3, 'YES');
    IF PayLoad4 <> ''
    THEN Kill_Unit (Root, PayLoad4, Num4, 'YES');
    IF AddOnName1 <> ''
    THEN Kill_Unit (Root, AddOnName1, AddonNum1, 'NO');
    IF AddOnName2 <> ''
    THEN Kill_Unit (Root, AddOnName2, AddonNum2, 'NO');
    IF AddOnName3 <> ''
    THEN Kill_Unit (Root, AddOnName3, AddonNum3, 'NO');
    IF AddOnName4 <> ''
    THEN Kill_Unit (Root, AddOnName4, AddonNum4, 'NO');
    IF AddOnName5 <> ''
    THEN Kill_Unit (Root, AddOnName5, AddonNum5, 'NO');
    Center_Line (22, 'Better Luck Next Time');
    Center_Line (24, 'Hit any key to continue');
    Read (kbd,Ch);
  END
END ; (Deploy Prob > 0.97)
IF DeployProb <= 0.97 THEN
BEGIN
  TextBackground (1);
  Clrscr; Border1;
  TextBackground (1);
  CenterLine (14, 'successful deployment');
  Delay (2000);
  Status := 'ACTIVE';
  Deployed := Year;
  IF PayLoad1 <> '' THEN
    Activate_Unit (Root, PayLoad1, Num1, 'YES');
  IF PayLoad2 <> '' THEN
    Activate_Unit (Root, PayLoad2, Num2, 'YES');
  IF PayLoad3 <> '' THEN
    Activate_Unit (Root, PayLoad3, Num3, 'YES');
  IF PayLoad4 <> '' THEN
    Activate_Unit (Root, PayLoad4, Num4, 'YES');
  IF AddOnName1 <> '' THEN
    Activate_Unit (Root, AddOnName1, AddOnNum1, 'NO');
  IF AddOnName2 <> '' THEN
    Activate_Unit (Root, AddOnName2, AddOnNum2, 'NO');
  IF AddOnName3 <> '' THEN
    Activate_Unit (Root, AddOnName3, AddOnNum3, 'NO');
  IF AddOnName4 <> '' THEN
    Activate_Unit (Root, AddOnName4, AddOnNum4, 'NO');
  IF AddOnName5 <> '' THEN
    Activate_Unit (Root, AddOnName5, AddOnNum5, 'NO');
END; (successful deployment)

END; (successful launch)

IF (Current^.ReuseField = '') OR (Current^.ReuseField = 'NO') THEN
  Current^.NumActive := Current^.NumActive - 1
  (decrement * launchers)
ELSE
BEGIN
  Current^.RemUses := Current^.RemUses - 1;
  Current^.Sorties := Current^.Sorties - 1;
  BUDGET := BUDGET - Current^.OMCost;
END;
END; (not nil (liftvehicle found)
END; ('With)

NASAOK := False;
MultipleLaunch := False;
Search (Root, LaunchPad, LaunchPadNum, 'NO');
END;

(called before every launch, gives civilians the opportunity to cancel)
launch due to violating civilian constraint)
is NASA players kills a launch, the units go back to a Ready status)

OVERLAY PROCEDURE NASA_Check (WorkUnit : Unit);

BEGIN
  WITH WorkUnit DO
  BEGIN
    Textbackground (1);
    ClrScr;
    Border1;
    Border2;
    Textbackground (1);
    NASAOK := False;
    Center Line (7, 'NASA, Please Confirm Launch of ');
    GotoXY (33,10);
    Write (Name:10, TailNumber:3);
    TextColor (White);
    GotoXY (20,13); Write ('Add-on Packages');
    Gotoxy (50,13); Write ('Additional Payloads');
    TextColor (Yellow);

    IF AddOnName1 <> '' THEN
      BEGIN
        GotoXY (18,15);
        Write (AddOnName1:10, AddOnNum1:3);
      END
    ELSE
      BEGIN
        GotoXY (23,15); Write ('NONE');
      END;

    IF AddOnName2 <> '' THEN
      BEGIN
        GotoXY (18,16);
        Write (AddOnName2:10, AddOnNum2:3);
      END;

    IF AddOnName3 <> '' THEN
      BEGIN

211
GotoXY (18,17);
Write (AddOnName3:10, AddOnNum3:3);
END;

IF AddOnName4 <> '' THEN
BEGIN
  GotoXY (18,18);
  Write (AddOnName4:10, AddOnNum4:3);
END;

IF AddOnName5 <> '' THEN
BEGIN
  GotoXY (18,19);
  Write (AddOnName5:10, AddOnNum5:3);
END;

IF PayLoad2 <> '' THEN
BEGIN
  Gotoxy (55,15);
  Write (PayLoad2, Num2:3);
END
ELSE
BEGIN
  GotoXY (58,15);
  Write ('NONE');
END;

IF PayLoad3 <> '' THEN
BEGIN
  Gotoxy (55,16);
  Write (PayLoad3, Num3:3);
END;

IF PayLoad4 <> '' THEN
BEGIN
  Gotoxy (55,17);
  Write (PayLoad4, Num4:3);
END;

Center-Line (21, '(YIN) ');

IF Yes THEN NASAOQ := True
ELSE
BEGIN
  (clear launch variables)
  NASAOK := False;
  AddOnName1 := '';
  AddOnName2 := '';
  AddOnName3 := '';
  AddOnName4 := '';
  AddOnName5 := '';
  PayLoad1 := '';
  PayLoad2 := '';
  PayLoad3 := '';
  PayLoad4 := '';
  Status := 'ACTIVE';
END;

END ; (With workUnit)
END ;
(Displays the launchers capable of lifting the designated payload)  
(function of throw weight and orbit correction factor)

OVERLAY PROCEDURE Display_Launchers (Root : UnitPtr ; VAR WorkUnit : Unit);

VAR
Current : UnitPtr ;
ThisOrbit : string5 ;
AtLeastOne : boolean ;
Correction : real ;

BEGIN
AtLeastOne := False ;
Current := Root ;
ClearLines (8,25) ;
GotoXY (1,9) ;

WHILE (Current <> Nil) AND (Current^.Mission <> 'LAUNCHER' ) DO
  Current := Current^.Next ;

WITH WorkUnit DO
  BEGIN
    ThisOrbit := Orbit ;
    
    WHILE (Current <> Nil) AND (Current^.Mission = 'LAUNCHER') DO
      BEGIN
      Correction := Orbit_Correction_Factor
                   (Current^.Name, Inclination) ;

      IF ThisOrbit = 'VLEO'
      THEN LiftWeight := Current^.VLEO * Correction ;

      IF ThisOrbit = 'LEO'
      THEN LiftWeight := Current^.LEO * Correction ;

      IF ThisOrbit = 'MOL'
      THEN LiftWeight := Current^.MOL * Correction ;

      IF ThisOrbit = 'MEO'
      THEN LiftWeight := Current^.MEO * Correction ;

      IF ThisOrbit = 'GEO'
      THEN LiftWeight := Current^.GEO * Correction ;

      RemWeight := LiftWeight ;

      IF (LiftWeight >= Mass) AND
         ((Current^.ReuseField <> '') AND
         (Current^.ReuseField <> 'NO') AND

213
(Current^.Status = 'ACTIVE') OR
(Current^.ReuseField = '') OR
(Current^.ReuseField = 'NO') AND
(Current^.NumActive > 0)) THEN
BEGIN
  RemWeight := RemWeight - Mass;
  Writeln (Current^.Name:10, Current^.TailNumber:3,
       Current^.NumActive:9,
       TenBlanks,' ', LiftWeight:6:0,
       TenBlanks,' ', Mass:6:0,
       TenBlanks,' ', RemWeight:6:0);

  AtLeastOne := True;

END;
Current := Current^.Next;

END; (Mission = Launcher)
IF NOT AtLeastOne THEN
BEGIN
  Center_Line (20, 'Need a bigger rocket in your inventory!!!');
  Center_Line (25, 'Please hit <CR> to Continue'); Read;
END;

END; (With)
END;

{contains the system level commands availabla only to game controller}
{accessed at various points in the game by enter the Shift and Tab key}
{at the same time}

PROCEDURE System_Level_Commands (Root : UnitPtr);
BEGIN
  REPEAT
    TextBackground (1);
    ClrScr ; Quit := False ;
    Border1 ;
    Textbackground (1);
    Center_Line (3, 'SYSTEM LEVEL COMMANDS');
    Center_Line (5, '[L]oad List');
    Center_Line (7, '[A]dd Unit');
    Center_Line (9, '[V]iew List');
    Center_Line (11, '[S]ave List');
    Center_Line (13, '[B]udget Adjustment');
    Center_Line (15, '[D]elete Unit');
    Center_Line (17, '[C]hange status');
    Center_Line (19, '[X] Add Copy');
    Center_Line (21, '[Y] Add Expenable Launchers');
  END;
Center_Line (23, '"Q"]uit ') ;

GotoXY (40,24) ;
Read (kbd,Ch) ;

CASE Ch OF

' A', 'a' : AddUnit (Root) ;

'L', 'l' : LoadList (Root) ;

'S', 's' : SaveList (Root) ;

'V', 'v' : BEGIN
  ViewList (Root) ;
  END ;

'B', 'b' : BEGIN
  ClrScr ;
  GotoXY (30,12) ;
  Write ('Current Budget: ', BUDGET:5) ;
  GotoXY (30,14) ;
  Write ('Updated Budget:') ;
  BUDGET :=GetInt (StringNumber, 46,14) ;
  END ;

'D', 'd' : BEGIN
  ClrScr ;
  Center_Line (12, 'Enter name of unit to be copied') ;
  Center_Line (14, 'Enter tail number') ;
  GotoXY (60,12) ; Read (SearchName) ;
  SearchNumber := GetInt (StringNumber, 60,14) ;
  Search (Root, SearchName, SearchNumber, 'NO') ;
  IF Found THEN
    Add-Copy (Root, Where^.Name, Where^.TailNumber) ;
  END ;

'C', 'c' : Change_Status (Root) ;

'X', 'x' : Begin
  ClrScr ;
  Center_Line (12, 'Enter name of unit to be copied') ;
  Center_Line (14, 'Enter tail number') ;
  GotoXY (60,12) ; Read (SearchName) ;
  GotoXY (60,14) ; Read (SearchNumber) ;
  Search (Root, SearchName, SearchNumber, 'NO') ;
  IF Found THEN Add_Copy (Root, Where^) ;
  END ;
'Y', 'y' : BEGIN
ClrScr;
Center_Line (4, 'Adding Expendable Launchers to Data Base');
Center_Line (12, 'Enter name of expendable launcher');
Read (SearchName);
Search (Root, SearchName, 1, 'YES');
IF Found THEN
BEGIN
ClearLines (23, 25);
Center_Line (23, 'Enter # to add to development');
Read (SearchNumber);
Where^.NumDev := SearchNumber;
END; (Found)
END;

'Q', 'q' : Quit := True;
END; (CASE)
UNTIL Quit;

END;

{displays all pertinent data when creating a composite satellite}

PROCEDURE Composite_Satellite_Header (VAR Root : UnitPtr);

VAR
Current : UnitPtr;
ThisName : string10;
ThisNumber : integer;
CountLines : integer;

BEGIN
Current := Root;
ClrScr;
Center_Line (2, 'Composite Satellite Screen');
Center_Line (3, 'Add-on systems add 50 % of their weight to overall system');
Center_Line (6, 'Satellites Ready For Launch');
GotoXY (1, 9) ; Write (' Mission') ; GotoXY (20, 9) ; Write ('Name');
GotoXY (30, 9) ; Write ('Tail Number') ; GotoXY (45, 9) ; Write ('Orbit');
GotoXY (55, 9) ; Write ('Inclination') ; GotoXY (72, 9) ; Write ('Mass');
GotoXY (1, 11);
CountLines := 11;

WHILE (Current <> Nil) AND (CountLines < 23) DO
BEGIN
IF (Current^.Mission <> 'GROUND') AND
  (Current^.Mission <> 'LAUNCHER')
AND (Current^.Mission <> 'PAD') AND

(Current^.Status = 'READY') THEN

BEGIN
  Writeln (Current^.Mission:10, Current^.Name:15,
            Current^.TailNumber:11, Current^.Orbit:12,
            Current^.Inclination:12, TenBlanks,
            Current^.Mass:5:0);

  CountLines := CountLines + 1;
END ;

Current := Current^.Next ;

END ; (Not nil and Lines < 23)

IF CountLines >= 23 THEN
BEGIN
  ClearLines (23,25);
  Center_line (24, "Hit 'C' to continue listing ');
  IF Continue THEN Composite_Satellite_Header (Current);
END ; (lines >= 23)

END ;

PROCEDURE Count_Ground (Var Root : UnitPtr ; Catagory : string10);

VAR
  Current : UnitPtr ;
  CountLines : integer ;
  Line : integer ;
BEGIN
  Current := Root ;
  CountLines := 0 ;
  Line := 5 ;
  WHILE (Current <> Nil) AND (CountLines < 6) DO
    BEGIN
      IF (Current^.Mission = 'GROUND') AND (Current^.Kind = Catagory) AND
      ((Current^.SupportName = '') OR (Current^.SupportName = 'NONE'))
      AND (Current^.Status = 'ACTIVE') THEN
        BEGIN
          IF Catagory = 'CONTROL' THEN
            BEGIN
              GotoXY (5, Line) ;
              Write (Current^.Name:10, Current^.tailNumber:3) ;
              Line := Line + 1 ;
            END ;
          IF Catagory = 'PROCESSOR' THEN
            BEGIN

GotoXY (30,Line);
Write (Current^.Name:10, Current^.tailNumber:3);
Line := Line + 1;
END;

IF Category = 'USER' THEN
BEGIN
GotoXY (55, Line);
Write (Current^.Name:10, Current^.TailNumber:3);
Line := Line + 1;
END;
END; (if ground/catagory/active)

Current := Current^.Next;
END; {not nil and < 6}

IF CountLines > 6 THEN
BEGIN
ClearLines (12,12);
CenterLine (12, 'Hit 'C' to continue listing');
IF Continue THEN Count_Ground (Current, Category);
END;
END;

(displays all the satellites currently in the production state and the) (ground elements already assigned to them)

PROCEDURE Display_System_CheckList (Var Root : UnitPtr);

VAR
Current : UnitPtr;
CountLines : integer;

BEGIN
Window (1,1,80,13);
Textbackground (1);
ClrScr;
TextColor (white);
Center_line (1, 'SYSTEM CHECK LIST');
Center_LIne (2, 'For Satellites in Production Status');
GotoXY (20,4); Write ('GROUND CONTROL');
GotoXY (40,4); Write ('DATA PROCESSING');
GotoXY (60,4); Write ('USER TERMINAL');
GotoXY (2,5); Write ('Satellite *');
GotoXY (23,5); Write ('System *');
GotoXY (43,5); Write ('System *');
GotoXY (63,5); Write ('System *');
TextColor (Yellow);
CountLines := 7;
Current := Root;
GotoXY (1,7);
While (Current <> Nil) AND (CountLines < 13) DO
BEGIN
  IF (Current^.Mission <> 'GROUND') AND (Current^.Mission <> 'LAUNCHER')
     AND (Current^.Mission <> 'PAD') AND
     (Current^.Status = 'PRODUCTION') THEN
    BEGIN
      Writeln (Current^.Name:10, Current^.TailNumber:4,
                Current^.ControlName:15, Current^.ControlNumber:3,
                Current^.ProcessorName:17, Current^.ProcessorNumber:3,
                Current^.UserName:17, Current^.UserNumber:3);
      CountLines := CountLines + 1;
    END;
  Current := Current^.Next;
END; (While)
IF CountLines >= 13 THEN
BEGIN
  ClearLines (13,13);
  Center_Line (13, 'hit 'C' to continue listing' );
  IF Continue THEN Display_System_CheckList (Current)
  ELSE ClearLines (13,13);
END ;
IF Current = Nil THEN
BEGIN
  ClearLines (13,13);
  Center_Line (13, 'HIT 'C' to restart listing or <CR> to assign' );
  IF Continue THEN Display_System_CheckList (Head)
  ELSE ClearLines (13,13);
END ;
END ;
Bibliography


15. Roseland, Larry, Major, USAF. Chief, Space Applications and Concepts Branch, Air Command and Staff College, Maxwell AFB, AL. Telephone interview. 3 September 1987.


19. Thieman, Bruce, Major, USAF. Space Curriculum Phase Manager, Air Command and Staff College, Maxwell AFB, AL. Personal Interview, 12 August 1987.


VITA

Captain Jeffrey E. Heier was born on 24 March 1961 in Grosse Pointe, Michigan. He graduated from high school in Clayton, Ohio, in 1979. He attended the Illinois Institute of Technology, Chicago, Illinois, where he received the degree of Bachelor of Science in Computer Science in May 1983. He was a Distinguished Military Graduate and received a Regular commission through the ROTC program. He reported for active duty in October 1983, and served as a Missile Warning Systems Analyst at Headquarters NORAD/Space Command, Colorado Springs, Colorado, until entering the School of Engineering, Air Force Institute of Technology, in May 1986.

Permanent address: 6500 Cheri Lynne
Dayton, Ohio 45415
ADAM Acquisition Deployment And Maneuvering The Space Game

Jeffrey E. Heier, B.S., Captain, USAF

The purpose of this effort was to provide middle to senior level managers a tool which would enhance their knowledge of how space assets are acquired, deployed, and maneuvered in response to the ground situation. This effort had two basic objectives: (1) Provide a user friendly model which effectively simulates the acquisition and deployment process of space assets. (2) Develop a model which could be used as a baseline for follow on efforts. The model was developed on a micro-computer because it was perceived this media would provide a more user-friendly and expandable environment.