AD-A050 768 INSTITUTE FOR DEFENSE ANALYSES ARLINGTON VA PROGRAM --ETC F/G 15/7
IDAHEX: A MANEUVER-ORIENTED MODEL OF CONVENTIONAL LAND WARFARE.--ETC(U)
NOV 76 P OLSEN
UNCLASSIFIED P-1221-VOL-2 SBIE-AD-E500 016 NL

1 OF 2
AD
A050 768

Copy 12 of 106 copies

AD-E500 016

IDA PAPER P-1221

12
B.S.

IDAHEX

A MANEUVER-ORIENTED MODEL OF CONVENTIONAL LAND WARFARE

VERSION 1.0

Volume 2: Game Designer's Manual

Paul Olsen

November 1976

INSTITUTE FOR DEFENSE ANALYSES
PROGRAM ANALYSIS DIVISION

IDA Log No. HQ 77-19000

18 SBIE | 19 AD-E500 016

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

14

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER P-1221-VOL-2	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) 6 IDAHEX: A Maneuver-Oriented Model of Conventional Land Warfare. Version 1.0. Volume [redacted] 2. Game Designer's Manual.		5. TYPE OF REPORT & PERIOD COVERED 9 Final rept., 6. PERFORMING ORG. REPORT NUMBER P-1221
7. AUTHOR(s) 10 Paul/Olsen		8. CONTRACT OR GRANT NUMBER(s) IDA Independent Research Program
9. PERFORMING ORGANIZATION NAME AND ADDRESS Institute for Defense Analyses Program Analysis Division 400 Army-Navy Drive, Arlington, VA 22202		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS N/A
11. CONTROLLING OFFICE NAME AND ADDRESS	11	12. REPORT DATE Nov[redacted]976 13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS (of this report) 12 761p. Unclassified 15a. DECLASSIFICATION DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

This document is unclassified and suitable for public release.

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

18. SUPPLEMENTARY NOTES

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

Land Warfare, Ground Combat, Simulation, Interactive Model, War Game, Computer-Assisted War Game, Ground Forces, Amphibious Landings, Airborne Operations, Maneuver

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

IDAHEX is an interactive computer model of two-sided conventional land warfare. It keeps the players informed of the situation and accepts their instructions to their forces. Units can move by land, sea, or air. A unit's movement rate is variable, depending upon its posture, the conditions of its movement, and the adequacy of transport. Attrition in engagements is assessed by a heterogeneous Lanchester square process. Air support can be played. Supplies consumption

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

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

403 219

20. (continued)

can be assessed, and logistics can be played. The model recognizes severed lines of retreat and lines of supply and imposes appropriate consequences. The documentation consists of three volumes: (1) A Guide for Potential Users; (2) Game Designer's Manual; (3) Player's Manual.

IDA PAPER P-1221

IDAHEX

A MANEUVER-ORIENTED MODEL OF CONVENTIONAL LAND WARFARE

VERSION 1.0

Volume 2: Game Designer's Manual

Paul Olsen

November 1976

INSTITUTE FOR DEFENSE ANALYSES
PROGRAM ANALYSIS DIVISION
400 Army-Navy Drive, Arlington, Virginia 22202

IDA Independent Research Program

PREFACE

IDAHEX is a computerized model of conventional land warfare at the theater level. Its documentation consists of:

Volume 1: *A Guide for Potential Users*
Volume 2: *Game Designer's Manual*
Volume 3: *Player's Manual*

Volume 1 outlines the model's fundamental characteristics. Volume 2 (this volume) comprehensively describes the model and its data base. Volume 3 contains enough information for someone with a modest knowledge of land warfare to play an IDAHEX game. It outlines the entire model, identifies information the game designer should give the players, and describes IDAHEX as a war game from the players' perspective.

Comments and inquiries are welcomed. They should be directed to the author (telephone: 703-558-1874).

RE: Classified references-
IDA Paper P-1221
Document should remain for unlimited distribution per Mrs. Doherty, IDA

CONTENTS

Preceding Page Blank

FIGURES

TABLES

1. UNDERSTANDING THE MANUAL

IDAHEX is a model of warfare. The model has been implemented as a computer program, written in FORTRAN. Usually, no distinction is drawn between the model and the program; this manual refers to both as "IDAHEX".

As a model, IDAHEX imposes some structure: there are exactly two sides in the conflict, "Red" and "Blue"; each side's force consists of individual "battle units"; the postures that a battle unit can assume are organized into rigidly sequenced classes; the area on which the Red and Blue forces move and fight, termed the "area of war", is approximately rectangular and is partitioned into regular hexagons; each battle unit's location is identified with a hexagon. Within this structure the "game designer" creates a game. He specifies the compositions of the Red and Blue forces, the resources held by each battle unit, the postures battle units can assume, the battle units' mobility, their resources' effectiveness in combat, the terrain of the area of war, and the size of the hexagons. Loosely speaking, the model is a war game whose rules are parameterized; the game designer sets the parameters, turning a general structure into a specific game. As a tool for designing war games, IDAHEX is valuable because:

(1) it is systematic, and therefore protects against design errors and omissions;

(2) it incorporates reasonably sophisticated procedures for assessing movement, combat, air support, and supplies consumption, obviating the time-consuming alternatives of writing *ad hoc* computer programs and making the assessments manually;

(3) it contains extensive logic for handling the consequences of maneuver.

The last point deserves emphasis. If a model plays maneuver at all--i.e., if battle units can move in more than one dimension--engagements may start and stop, battle units may be attacked from multiple directions, attackers may be attacked from the flank or rear, and enemy units may meet on the march. IDAHEX can handle these events. Without that capability, the game designer

For complete abstract see Vol. 1, AD-A050675.

would have to rely on a control team to make *ad hoc* judgments or would have to write a web of rules.

This Volume describes the IDAHEX model and shows how to use IDAHEX to design a war game. The Glossary briefly defines all variables input by the game designer as well as variables and functions used internally by the IDAHEX computer program. Detailed information on almost any variable or function in the Glossary can be found through the Index. Some variables and functions--easily identified because their names appear in brackets--do not actually exist in the IDAHEX program but should be treated just as any other variables and functions by the game designer. They have precise analogs in the program, analogs that are pedagogically inconvenient because of special coding to conserve storage. A variable whose name begins with a capital letter may not correspond to any program variable; it is only a pedagogical device. If a variable's value is set by inputs from the game designer--the "game design data"--the variable's name is italicized or underlined. In some cases, a variable's value is set by the game design data but may be altered later by IDAHEX; the altered variable is identified by the same name without italics or underlining. Examples:

(1) The array *katk* is set by the game design data, but IDAHEX almost immediately redefines it by multiplying each element by *tframe*. The resulting variable is named "katk" to distinguish it.

(2) The vector of battle units' locations, *buloc*, is initially set by the design data. But units' locations may change during the game. The vector variable containing updated unit locations is named "buloc".

A variable's name is never italicized or underlined simply because its value is derived from game design data: ultimately, every variable's value is determined by the game design data and the players' inputs.

The reader may be unaccustomed to variables' names containing lower-case letters. The IDAHEX documentation uses program variables' names as they appear in the MULTICS version of the IDAHEX computer program. In MULTICS FORTRAN and PL/I, the lower-case letters constitute the primary alphabet, of which the upper-case letters are an extension. Changing every lower-case letter in the IDAHEX source program to upper-case produces a logically equivalent program.

Unless the contrary is affirmed, a variable determining the number of elements in a set may be 0. For example, the game design data fix the value of *nss*(1), the number of types of Red supplies. Such size parameters let the game designer choose from a spectrum of complexity. At one end he can play several types of weapons, several types of transport, several types of supplies, and several types of personnel on each side. At the other end, he can play just one resource--an abstract index of strength--on each side.

IDAHEX distinguishes three roles for its user or users: the game designer, who provides the inputs that specify the game (the game design data); the Red player, who commands the Red force; and the Blue player, who commands the Blue force. If used in an interactive mode, IDAHEX gets the game design data from one file--ordinarily associated with the card reader, a tape data set, or a disc data set--and gets the players' inputs from one or two terminals. For maximum clarity, the documentation is written as though IDAHEX is used interactively.

The term "unit" means "battle unit" unless the context in which it is used indicates otherwise. The phrase "a Red type i resource", or "a Red resource of type i", means "a unit-quantity of Red resources of Red resource type i". (Here, "unit" means "unit of measure", not "battle unit".) Likewise for Blue. An element of a vector or a (two-dimensional) matrix may be indicated by use of parenthesized arguments instead of subscripts:

$$x(i) \text{ means } x_i,$$

$$a(i,j) \text{ means } a_{ij}.$$

The variable a(i,*) is row i of the matrix a. The variable a(*,j) is column j of the matrix a. These may also be written as

$$a_{i*} \text{ and } a_{*j} .$$

The symbol "ε", when used syntactically, means "in", "belongs to", or "is a member of". Example: if C is a set, "u ε C" means "u is a member of C". The same symbol with a line through it ("∉") means "not in", "does not belong to", or "is not a member of". If y and z are scalar variables, y*z denotes their product, and y**z denotes y raised to the power z.

2. THE ELEMENTS OF PLAY

This section explains how IDAHEX structures the area of war, the resources, and changes in unit postures and locations.

2.1 THE AREA OF WAR

The game board is termed the "area of war"--the area in which the forces exist. It is partitioned into congruent, regular hexagons, as Figure 2.1 illustrates. The hexagons are termed "cells". A cell's *depth* is defined as the distance from one side to the side directly opposite it; this distance equals the distance from a cell's center to any adjacent cell's center. (See Figure 2.2.) The variable *depth* is fixed by the game design data. The cells are always arranged in vertical ranks and numbered as Figure 2.1 illustrates. The number of cells in the first (the leftmost) rank, *nrank1*, is fixed by the design data. (It is 8 in Figure 2.1.) The next rank always has one less cell. The number of ranks is jointly determined by *nrank1* and *ncells*, the number of cells in the area of war. A cell may be "inactive"--in effect, excluded from the area of war. The bottom cell in every rank (the highest-numbered cell in every rank) is always inactive, regardless of the game design data. A cell's successors" are the cells that are adjacent to it and have larger numbers than its own. They are ordered as follows: a cell's first successor is the cell (if any) below it, its second successor is the cell to the upper right (if any), and its third successor is the cell to the lower right (if any). By definition, for $1 \leq n \leq ncells$ and $1 \leq k \leq 3$, [successor](n,k) is the cell number of the k-th successor of cell n unless cell n is inactive or its k-th successor is inactive or nonexistent, in which case [successor](n,k) = -1. For example, if all the cells in Figure 2.1 except the bottom cells (8, 15, 23, 30, 38, 45, 53, 60) are active,

```
[successor](12,1) = 13
[successor](12,2) = 19
[successor](12,3) = 20
[successor](7,3)  = -1
[successor](46,1) = 47
[successor](46,2) = -1
[successor](46,3) = 54
[successor](30,2) = -1
```

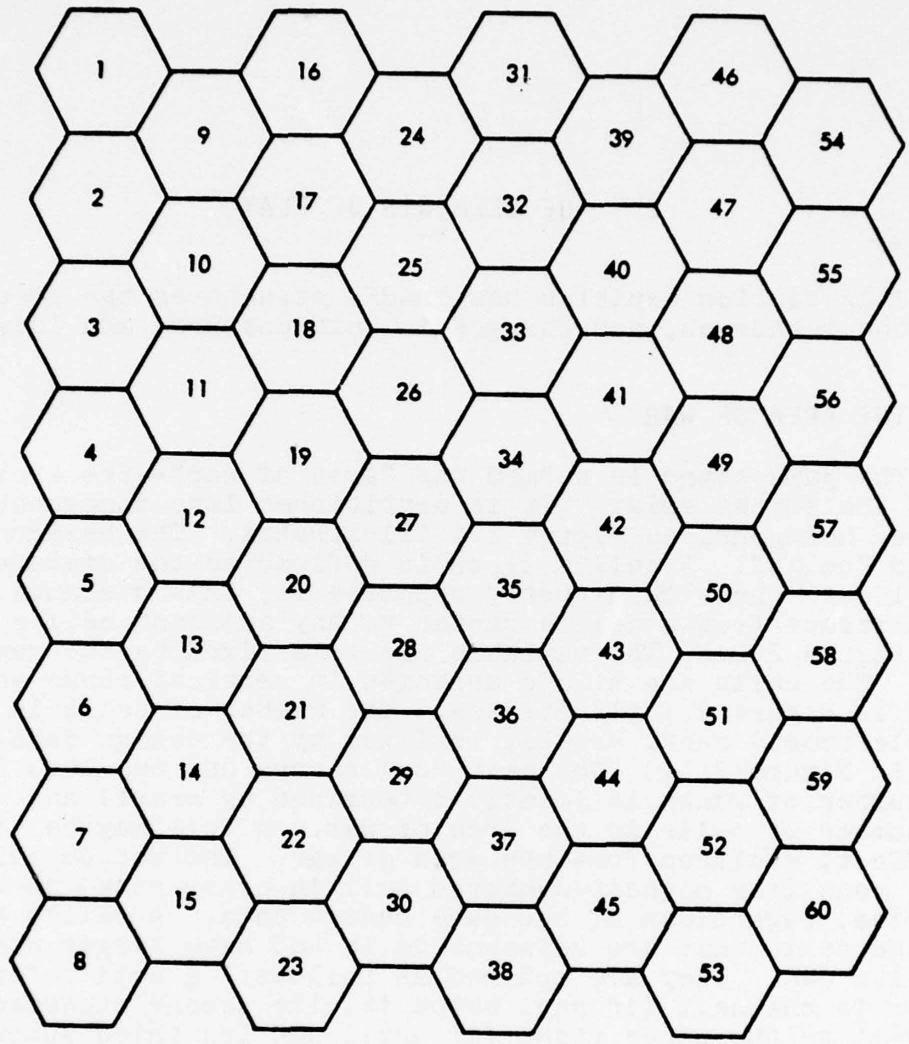


Figure 2.1. EXAMPLE OF AREA OF WAR

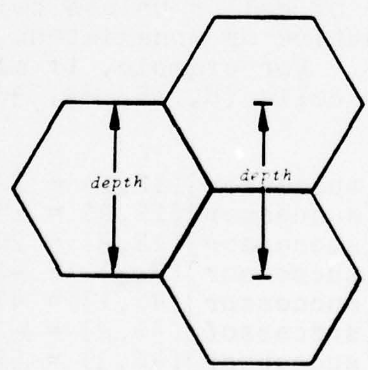


Figure 2.2. ILLUSTRATION OF CELL DEPTH

A cell's "environment" is the complex of physical conditions in the cell that affect combat or vulnerability to air strikes. Examples: clear, hilly, muddy, urban. The environment is assumed to be uniform throughout the cell. The environment of cell i is coded in [environment](i) as a positive integer.

The partitioning of the area of war into hexagons induces a network, formed by putting a node in each cell and creating an arc between each pair of nodes in adjacent cells. Figure 2.3 illustrates it. Coded characterizations of trafficability are associated with each arc (and therefore with each pair of adjacent cells). If cell i and cell j are adjacent, [rtetype](i,j), a positive integer, is the type of route between them. One could be more precise and say that [rtetype](i,j) characterizes the route between the node in cell i and the node in cell j, but such precision is spurious. There is ordinarily no single, geographically identifiable route between two cells. The datum [rtetype](i,j) is a general characterization of trafficability between the cells, not a characterization of a particular route of march. Indeed, for a task force in approach march, several parallel gravel roads would probably allow faster movement than a single paved road. By definition, [rtetype](i,j) ignores barriers between cell i and cell j. Another integer, [bartype](i,j), characterizes any barriers to movement. Barriers include rivers, ridges, and, in general, any natural or man-made obstacle that affects movement or attack. If [bartype](i,j) $\leq$ 0, it implies there are no barriers between cell i and cell j. If positive, it defines the type of barrier. Instead of listing multiple barriers between the cells, [bartype](i,j) gives one number that describes the barrier complex as a whole. IDAHEX assumes that

$$\text{[rtetype]}(i,j) = \text{[rtetype]}(j,i)$$
and
$$\text{[bartype]}(i,j) = \text{[bartype]}(j,i)$$

for each pair of adjacent cells, i and j. The format of the input data leaves the game designer no choice.

The game design data fix the values of the variables [*basic_env*], [*basic_rtetype*], and [*basic_bartype*], which are mapped into the actual environment type, actual route type, and actual barrier type by the game design variables *envmap*, *rtemap*, and *barmap*. For every $1 \leq i \leq$ *ncells*,

$$\text{[environment]}(i) = \mathit{envmap}(\text{[}\mathit{basic\_env}\text{]}(i)).$$

For every pair of adjacent cells, i and j,

$$\text{[rtetype]}(i,j) = \mathit{rtemap}(\text{[}\mathit{basic\_rtetype}\text{]}(i,j)),$$

and

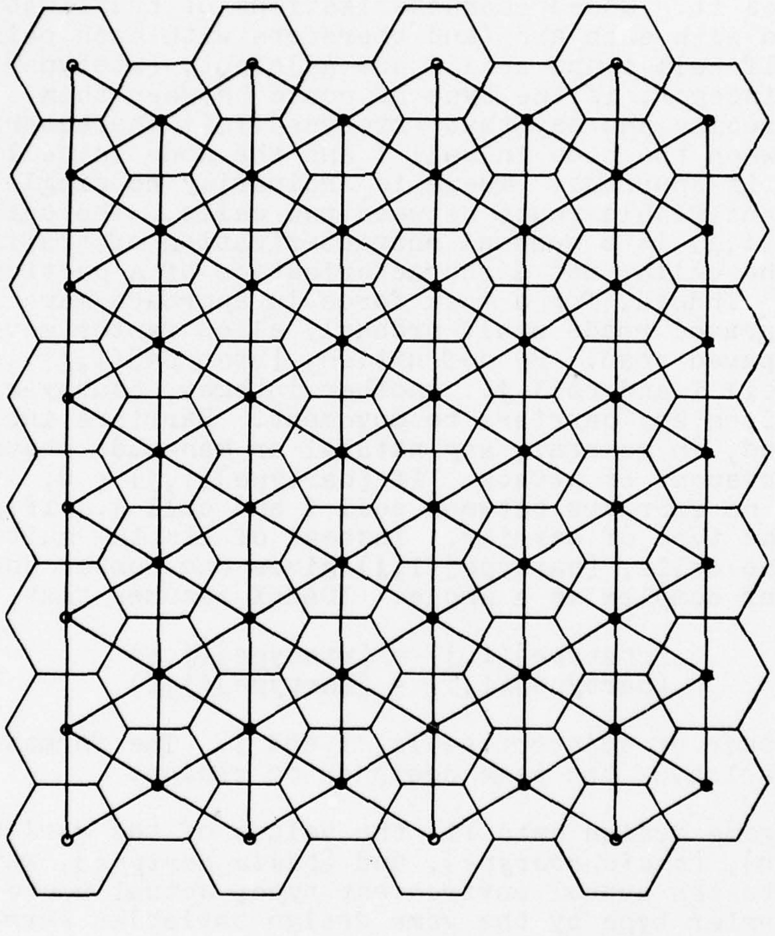


Figure 2.3. AREA OF WAR AND OVERLYING NETWORK

$$[\text{bartype}](i,j) = \begin{cases} barmap([basic\_bartype](i,j)) \\ \qquad \text{if } [basic\_bartype](i,j) > 0, \\ 0 \quad \text{otherwise.} \end{cases}$$

The game designer may want to use very detailed basic environment types, basic route types, and basic barrier types, not knowing how much detail will be needed. He may later find that this detail cannot be supported by the data on movement and attrition, which depend upon the environment, route, and barrier types. The maps *envmap*, *rtemap*, and *barmap* can be set to compress a large number of basic environment types, basic route types, and basic barrier types into a manageable number of actual types for the movement and attrition data. The maps can also be used to alter the actual environment types, route types, and barrier types during the course of the game--to reflect changes in weather, for example. Initially,

$$envmap(i) = i,\ rtemap(i) = i,\ barmap(i) = i$$

for every i. At the start of every cycle, including the first, the game design data can modify the maps, which then remain fixed unless and until the design data modify them again.[1] (To modify the maps is to change one or more elements of the vectors *envmap*, *rtemap*, and *barmap*.) IDAHEX advises the players of any modifications.

Although formally part of the area of war, a cell, i, can be effectively excluded by making its basic environment, [*basic_env*](i), a nonpositive integer. The cell is then "inactive". No unit is able to enter. Only one word of computer storage is used to record information about the cell. An attempt by the design data to set [*basic_rtetype*](i,j) or [*basic_bartype*](i,j) for any j is ignored.

2.2 THE FORCES

There are two forces, Red and Blue. Each force consists of indivisible "battle units", often called simply "units". The game designer assigns each unit a unique number, by which IDAHEX identifies it. A unit's number must be a positive integer. The number assigned to any Red unit must be less than any number assigned to a Blue unit, but units need not be numbered consecutively. Each unit has a "name"--a character string--and a type.

---

[1]A "cycle" is a subdivision of game time. It is defined in Section 2.3.

The complete set of unit types for a particular game might be:

1. Red motorized rifle division
2. Blue tank division
3. Red tank battalion
4. Red tank division
5. Red transport unit
6. Blue transport unit
7. Blue infantry division

Each unit type is identified by a positive integer. A unit's type is stored in the vector *butype*; in terms of the example above, if unit 45 is a Red tank division, then *butype*(45) = 4. Units of the same type must belong to the same side.

2.2.1 Battle Unit Status

A battle unit's "status" is described by its location, posture, and objective. Each battle unit is located in exactly one cell. The unit's location can not be fixed more precisely: the unit is never said to be, for example, 3 km east of the cell's center. Its location *is* the cell. Several units may have the same location, even if they belong to different sides.

At any moment of the game, each battle unit is in one of 6 "posture classes":

- -1. destroyed
- 0. inactive
- 1. hold
- 2. disengagement
- 3. movement
- 4. attack

A unit in posture class 2 is trying to break contact with any enemy units it may be fighting, as the first step in changing location. Its "objective" is the cell toward which it is disengaging. A unit in posture class 3 is moving from its location to another cell, its objective. Ordinarily, a unit in posture class 4 is trying to enter a new location, which may or may not contain enemy units, but in some cases it is trying to revert from posture class 2, 3, or 4 to posture class 1 without changing location, In the former instance, its objective is the cell it seeks to enter; in the latter; its objective is just its present location.

Posture class 1 embraces all remaining activities as well as simple idleness. In particular, a unit in posture class 1 is not in the process of changing location. It may or may not be engaged. Its objective is, by convention, its location.

A unit in posture class -1 or 0 is said to be "inactive". (Inversely, a unit in a positive posture class is said to be "active".) An inactive unit does not exist from the perspectives of other units. It can not move; it can not attack, nor can it be attacked. A unit in posture class -1 is a special kind of inactive unit: it was de-activated to represent its destruction, usually as a result of suffering intolerably high losses. A unit in posture class 0 is ordinarily a reinforcement or a package of replacements. It may become active (enter a positive posture class) later in the war. Its location is the cell where it is expected to enter the area of war if it becomes active, but while it remains inactive, it has no effect on enemy units passing through its location.

When a unit, say unit j, enters a new posture class, its "virtual time of entry", tentry(j), is updated. Normally, tentry(j) is set to the exact time at which unit j enters the new posture class, but in special situations it may be set to a later time. At the start of the game, tentry = *tentry*. The game design datum *tentry*(j) is most simply defined as the time at which unit j entered the cell where it is located at the start of the game. For example, if the game starts at time 0, if time is measured in days, and if unit 45 assumed its starting location 30 days prior to the starting time, then *tentry*(j) should be -30.0

Each positive posture class consists of at least one posture and no more than 10 postures. Posture class -1 consists of just one posture, numbered -10. The postures in posture class 0 are numbered 0 through 9, but IDAHEX does not distinguish among the postures in posture class 0. The postures in posture classes 1 through 4 are numbered as follows:

10-19	hold
20-29	disengagement
30-39	movement
40-49	attack

Notice that [floor](p/10) is the posture class to which posture p belongs.[1] There might, for example, be two different movement postures, representing surface movement and airborne movement. There might be several different attack postures, representing different degrees of willingness to trade casualties for space. Table 2.1 presents alternative ways of describing a unit's posture class. The game design datum *npost*(i) fixes the number of postures in posture class i ($1 \leq i \leq 4$). There must be one posture numbered 10, one numbered 20, one numbered 30, and one numbered 40. These are the standard hold, disengagement, movement, and attack postures; if IDAHEX knows a unit's posture class but

[1]See the Glossary for the definition of the function [floor].

Table 2.1. EQUIVALENT DESCRIPTIONS OF POSTURE CLASS

posture class 1;	in a hold posture;	holding
posture class 2;	in a disengagement posture;	disengaging
posture class 3;	in a movement posture;	moving
posture class 4;	in an attack posture;	attacking

has insufficient information to determine the posture, it assumes the standard posture in the posture class.

The postures within a posture class need not be numbered consecutively, but doing so may reduce storage requirements. Some postures within posture class 3 may represent land or sea movement whereas others may represent air movement. Numbering the surface movement postures before the air movement postures (if any) may reduce storage requirements.

2.2.2 Battle Unit Resources

The types of resources each side has are arranged in a list. For example, the list of Red resource types might be:

1. tanks
2. small arms and APCs
3. artillery
4. SAMs and AAA
5. trucks
6. ammunition
7. fuel and other consumables
8. tank crewmen
9. other personnel

The Blue list might be:

1. small arms and APCs
2. artillery
3. tanks
4. trucks
5. supplies
6. personnel

There is no correspondence between Red resource types and Blue resource types: in the example, Red type 3 resources are artillery while Blue type 3 resources are tanks, and Red has

SAMs and AAA while Blue has none. The resource types must be listed in the following order:

ground-to-ground weapons
ground-to-air weapons
transport
supplies
personnel

These five resource categories are combined to form larger categories:

materiel	ground-to-ground weapons	weapons	equipment
materiel	ground-to-air weapons	weapons	equipment
materiel	transport		equipment
materiel	supplies		support
	personnel		support

The preceding categories induce sublists in each side's list of resource types. In the example above, the list of Red ground-to-ground weapons is:

1. tanks
2. small arms and APCs
3. artillery

The list of Red weapons is:

1. tanks
2. small arms and APCs
3. artillery
4. SAMs and AAA

Thus, a Red type 2 ground-to-ground weapon is also a Red type 2 weapon, and is also a Red type 2 resource. The list of Blue weapons is:

1. small arms and APCs
2. artillery
3. tanks

The list of Red personnel is:

1. tank crewmen
2. other personnel

The list of Red support resources is:

1. ammunition
2. fuel and other consumables
3. tank crewmen
4. other personnel

Thus, Red type 2 personnel are also Red type 4 support resources and Red type 9 resources. Notice that the category of Blue ground-to-air weapons is empty. (Hence, the list of Blue weapons is identical to the list of Blue ground-to-ground weapons.) Any category except ground-to-ground weapons may be empty. It is permissible for a side to have only one type of ground-to-ground weapon, which would probably be not a physical entity but an abstract measure of strength.

A unit's type determines what types of resources it can possess. The game design data fix *nrst*(i), the number of different types of resources a unit of type i can have, and *iars*(*,i), a list of the types of resources it can have. Continuing the example above, suppose:

$$nrst(5) = 6$$

$$\begin{aligned} iars(1,5) &= 7 \\ iars(2,5) &= 6 \\ iars(3,5) &= 8 \\ iars(4,5) &= 9 \\ iars(5,5) &= 5 \\ iars(6,5) &= 2 \end{aligned}$$

This says that a unit of type 5 can have 6 types of resources: Red type 7 resources (fuel and other consumables), Red type 6 resources (ammunition), Red type 8 resources, Red type 9 resources, Red type 5 resources (trucks), and Red type 2 resources (small arms and APCs). The order in which the resource types appear in *iars*(*,5) affects only the order in which IDAHEX lists the resources of type 5 units internally. By keeping the elements of *nrst* small in value, the game designer can achieve substantial economies in computer storage utilization.

The value of *iars*(*,5) in the preceding example is reasonable if a type 5 unit is a Red transport unit (as in the example at the start of Section 2.2). Notice that *iars*(*,5) lists some resources for which a transport unit would have no use, such as tank crewmen. IDAHEX allows transfers of resources between units, and therefore resources might be attached to a unit simply to move them from one place to another. If *iars*(*,5) excluded tank crewmen, a type 5 unit could not accept them and therefore could never be used to take tank crewmen to a unit that needed them.

If i is the identification number of some battle unit, the design datum [*resources*](i,j) is defined as the quantity of type j resources in the unit at the start of the game. Of course, this quantity must be 0 if the unit is prohibited from having type j resources--i.e., if there is no $1 \leq k \leq nrst(butype(i))$

such that *iars*(k,*butype*(i)) = j. At any time during the game [resources](i,j) is the quantity of type j resources in unit i; it is set equal to [*resources*](i,j) at the start of the game.

The design datum *toe*(k,j) is defined as the planned effective quantity of type j resources in a type k battle unit. It might be based on the Table of Organization and Equipment for a type k unit. IDAHEX compares a unit's actual quantities of resources (given by [resources]) with its planned effective quantities in allocating supplies and replacements to it, estimating its strength, and estimating the size of its area of influence. Of course, *toe*(k,j) should be 0 if a type k unit is prohibited from having type j resources.

2.3 TIME

At the start of a game, the current time, t, equals *tinit*, which should be a nonnegative number. The game ends when t = *tend* or when a player stops it.

Time is divided into equal-length intervals called "cycles", which are subdivided into equal-length "periods", which are subdivided into equal-length "frames". Cycles, periods, and frames may all be the same length, but generally frames are shorter than periods. A "break" occurs at the start of each cycle, the start of each period, and the end of each frame. Each break causes execution of a procedure selected according to the cause of the break: at the start of each cycle, IDAHEX accepts players' air strike specifications; at the start of each period, it accepts players' commands; at the end of each frame, it assesses engagements and supplies consumption.

An "event" is a break or a change in a unit's status. At t = *tinit* (the start of the game), IDAHEX ascertains when the first event will occur. It advances t to that time (possibly the same as the current time) and lets the event occur. It then ascertains when the next event will occur, advances t to that time, and lets the event occur. It continues to advance t in jumps until t $\leq$ *tend* or a player stops the game after a break.

3. MANEUVER

A "task force" is a collection of one or more battle units --"task force elements"--that have the same status and will continue to have the same status as long as they remain in the task force. The elements of a task force must all belong to the same side. Each task force is identified by a positive integer; it is impossible to tell from this number alone the side to which the task force belongs.

3.1 EVENT SEQUENCING

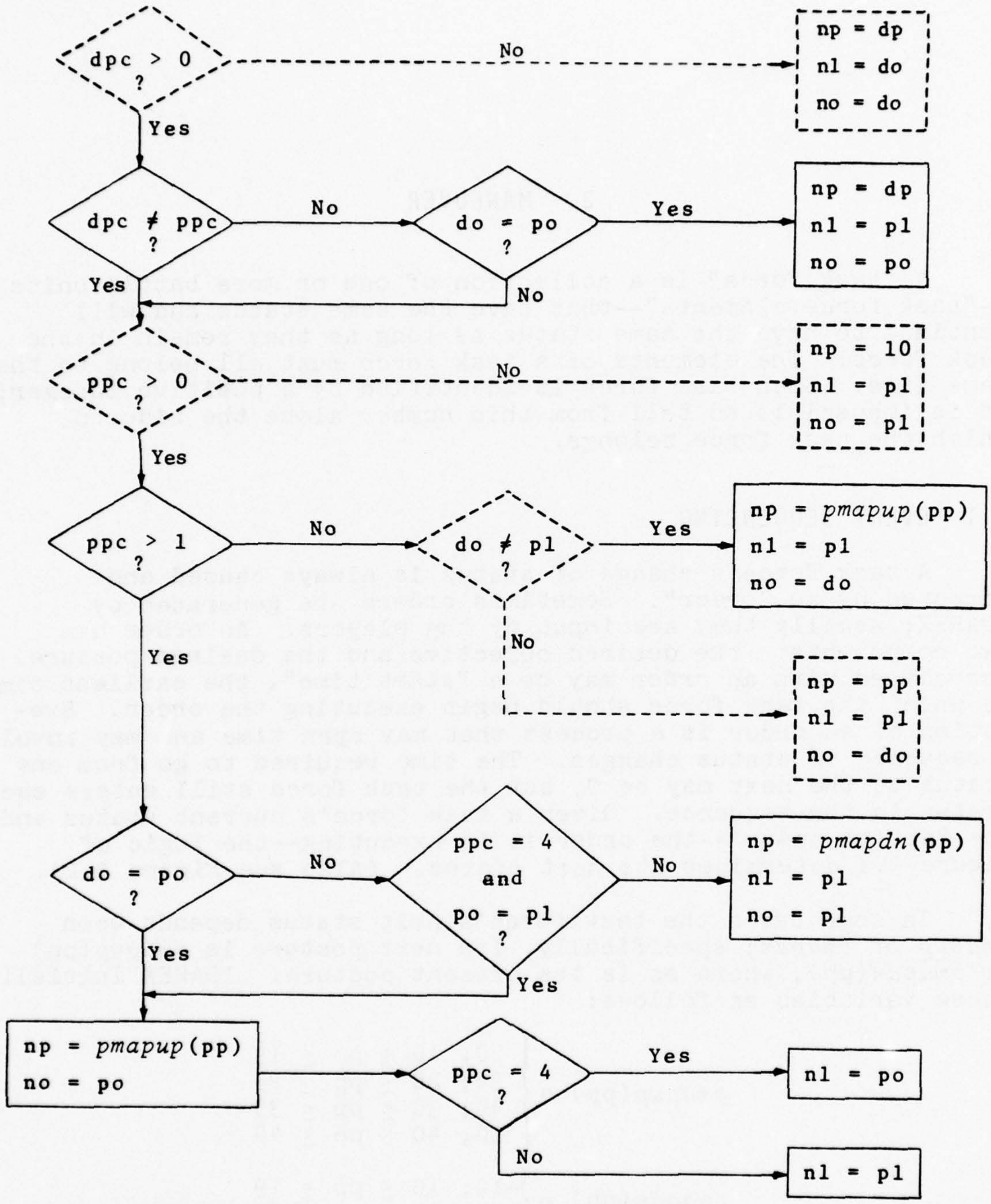
A task force's change of status is always caused and directed by an "order". Sometimes orders are generated by IDAHEX; usually they are input by the players. An order has two components: the desired objective and the desired posture. Associated with an order may be a "start time", the earliest time at which the task force should *begin* executing the order. Execution of an order is a process that may span time and may involve a sequence of status changes. The time required to go from one status to the next may be 0, but the task force still enters each status in the sequence. Given a task force's current status and its "active order"--the order it is executing--the logic of Figure 3.1 determines its next status. (Also see Figure 3.2).

In some cases the task force's next status depends upon *pmapup* or *pmapdn*; specifically, its next posture is *pmapup*(pp) or *pmapdn*(pp), where pp is its present posture. IDAHEX initializes these variables as follows:

$$\mathit{pmapup}(\mathrm{pp}) = \begin{cases} 20; & 10 \le \mathrm{pp} \le 19 \\ 30; & 20 \le \mathrm{pp} \le 29 \\ 40; & 30 \le \mathrm{pp} \le 39 \\ 10; & 40 \le \mathrm{pp} \le 49 \end{cases}$$

$$\mathit{pmapdn}(\mathrm{pp}) = \begin{cases} -10; & 10 \le \mathrm{pp} \le 19 \\ 40; & 20 \le \mathrm{pp} \le 49 \end{cases}$$

The game designer can modify these values, but the modified values must be such that:

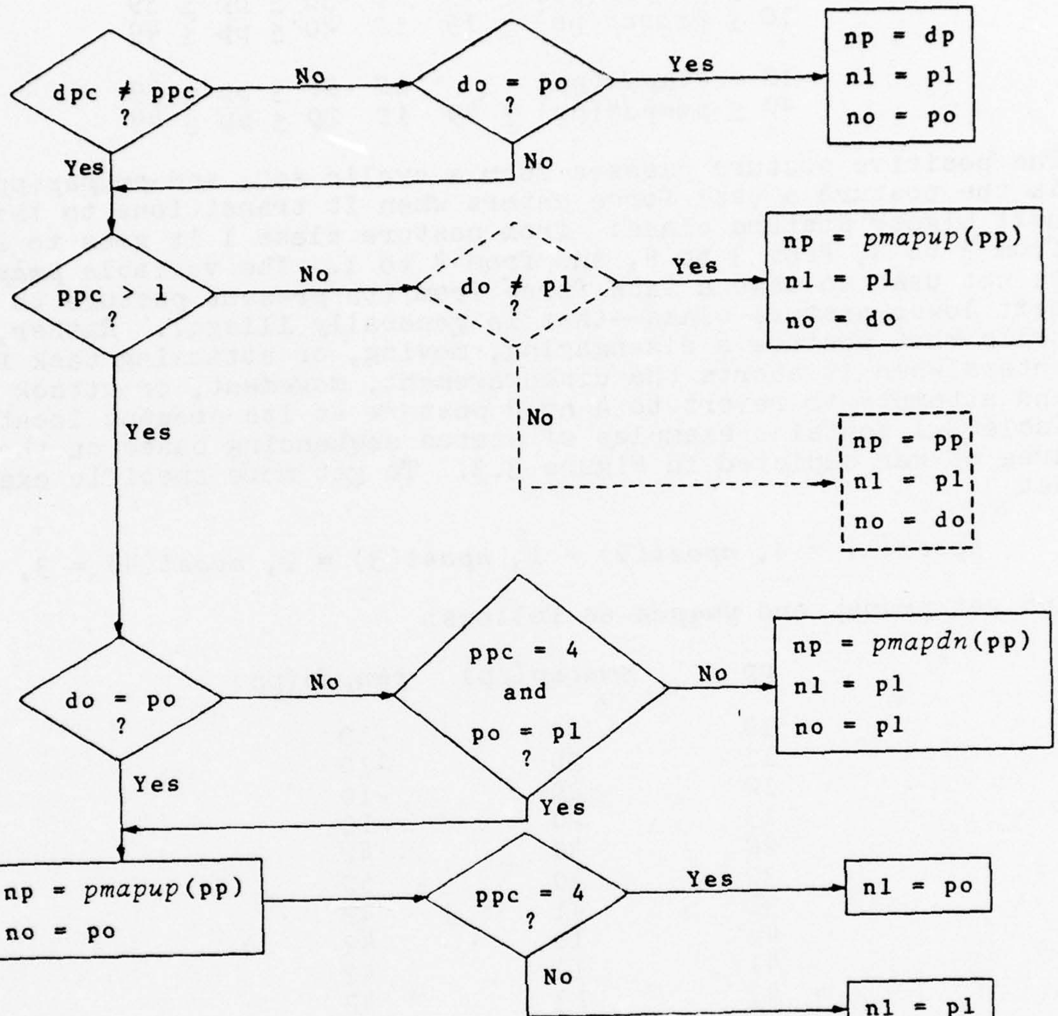


pp = present posture
ppc = present posture class
pl = present location
po = present objective

dp = desired posture
dpc = desired posture class
do = desired objective

np = next posture
nl = next location
no = next objective

Figure 3.1. STATUS SEQUENCING



pp = present posture
ppc = present posture class
pl = present location
po = present objective

dp = desired posture
dpc = desired posture class
do = desired objective

np = next posture
nl = next location
no = next objective

Figure 3.2. STATUS SEQUENCING FOR TASK FORCE WHOSE PRESENT POSTURE CLASS > 0 AND DESIRED POSTURE CLASS > 0

$$20 \le pmapup(\mathrm{pp}) \le 29 \quad \text{if} \quad 10 \le \mathrm{pp} \le 19$$
$$30 \le pmapup(\mathrm{pp}) \le 39 \quad \text{if} \quad 20 \le \mathrm{pp} \le 29$$
$$40 \le pmapup(\mathrm{pp}) \le 49 \quad \text{if} \quad 30 \le \mathrm{pp} \le 39$$
$$10 \le pmapup(\mathrm{pp}) \le 19 \quad \text{if} \quad 40 \le \mathrm{pp} \le 49$$

$$-10 = pmapdn(\mathrm{pp}) \quad \text{if} \quad 10 \le \mathrm{pp} \le 19$$
$$40 \le pmapdn(\mathrm{pp}) \le 49 \quad \text{if} \quad 20 \le \mathrm{pp} \le 49$$

The positive posture classes form a cyclic set, and *pmapup*(pp) is the posture a task force enters when it transitions to the next higher posture class: from posture class 1 it goes to 2, from 2 to 3, from 3 to 4, and from 4 to 1. The variable *pmapdn* is not used to take a task force from its present posture to the next lower posture class--that is generally illegal. Rather, it tells what posture a disengaging, moving, or attacking task force enters when it aborts the disengagement, movement, or attack and attempts to revert to a hold posture at its present location. Table 3.1 contains examples of status sequencing based on the area of war depicted in Figure 3.3. To get more specific examples, let

$$npost(1) = 4, \; npost(2) = 1, \; npost(3) = 2, \; npost(4) = 3,$$

and set *pmapup* and *pmapdn* as follows:

pp	*pmapup*(pp)	*pmapdn*(pp)
10	20	-10
11	20	-10
12	20	-10
13	20	-10
20	30	42
30	40	42
31	41	42
40	10	42
41	11	42
42	13	42

The preceding assignments are motivated by the following interpretations of the postures:

10 standard defense
11 halted
12 prepared for transferring resources to other units (*itrfp* = 12)
13 hasty, disorganized defense
20 disengaging
30 tactical march
31 administrative march
40 standard attack
41 attack from administrative march
42 hasty, disorganized attack

Table 3.1. EXAMPLES OF STATUS CHANGES
(Refer to Figure 3.3)

task force elements	present location	present posture	present objective	desired posture	desired objective	next location	next posture	next posture class	next objective
4,9	17	10	17	10	13	17	*pmapup*(10)	2	13
4,9	17	20	13	10	13	17	*pmapup*(20)	3	13
4,9	17	30	13	10	13	17	*pmapup*(30)	4	13
4,9	17	40	13	10	13	13	*pmapup*(40)	1	13
4,9	17	12	17	10	13	17	*pmapup*(12)	2	13
4,9	17	11	17	15	17	17	15	1	17
4,9	17	10	17	22	13	17	*pmapup*(10)	2	13
4,9	17	15	17	22	13	17	*pmapup*(15)	2	13
4,9	17	32	16	41	16	17	*pmapup*(32)	4	16
4,9	17	32	16	10	16	17	*pmapup*(32)	4	16
21	6	12	6	10	6	6	10	1	6
21	6	12	6	31	9	6	*pmapup*(12)	2	9
21	6	31	9	40	9	6	*pmapup*(31)	4	9
21	6	43	9	40	9	6	40	4	9
21	6	40	9	10	9	9	*pmapup*(40)	1	9
21	6	40	9	10	6	6	*pmapdn*(40)	4	6
21	6	42	9	11	6	6	*pmapdn*(42)	4	6
21	6	31	9	10	6	6	*pmapdn*(31)	4	6
21	6	23	9	14	6	6	*pmapdn*(23)	4	6
21	6	44	6	10	6	6	*pmapup*(44)	1	6
21	6	44	6	11	6	6	*pmapup*(44)	1	6

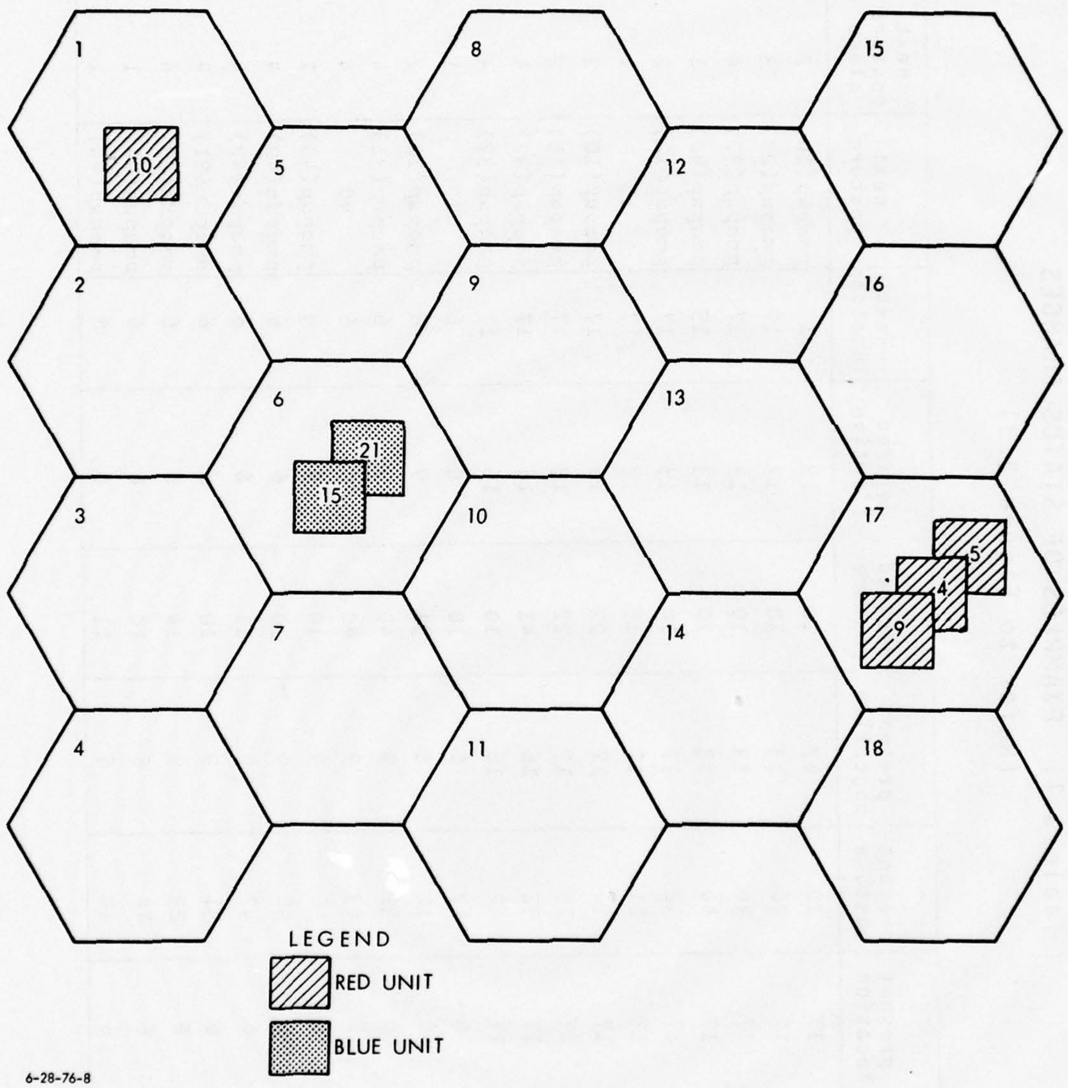


Figure 3.3. AREA OF WAR WITH BATTLE UNITS

Presumably, the ground combat attrition data make a unit less effective on defense in posture 13 than posture 11, and less effective in posture 11 than posture 10. Likewise, an attacker should be less effective in posture 41 than posture 40. Based on the above values of *pmapup* and *pmapdn* and the area of war in Figure 3.3, Table 3.2 shows the sequence of statuses induced by various orders. The last example in the table depicts a task force aborting an attack and reverting to a hold posture at its present location.

The preceding configuration can be simplified (at the risk of oversimplifying): let *npost*(1) = 2, *npost*(2) = *npost*(3) = *npost*(4) = 1, and accept the default values of *pmapup* and *pmapdn*. Let *itrfp* = 11. Then a task force in a hold posture in cell 6 whose desired posture is 11 and desired objective is 9 would go through the following sequence of statuses:

location	posture	objective
6	20	9
6	30	9
6	40	9
9	10	9
9	11	9

When the task force achieves posture 11, it will be ready and able to transfer resources to friendly units located in cell 9. If the movement of supplies and replacements is to be played explicitly, one hold posture should be set aside as a transfer posture, identified by the number *itrfp*. A task force whose

location = 6,
posture = 40,
objective = 9,

and whose

desired posture = 10,
desired objective = 6,

would go through the following sequence:

location	posture	objective
6	40	6
6	10	6

Thus, the task force aborts an attack and goes directly into the standard hold posture at its location; in contrast to the last example in Table 3.2, there is no "disorganized defense" posture

Table 3.2. EXAMPLES OF STATUS SEQUENCES

present location	present posture	present objective	desired posture	desired objective	Sequence of Statuses		
					location	posture	objective
6	10	6	10	9	6	20	9
					6	30	9
					6	40	9
					9	10	9
6	11	6	31	9	6	20	9
					6	30	9
					6	31	9
6	31	9	10	9	6	41	9
					9	11	9
					9	10	9
6	31	9	40	9	6	41	9
					6	40	9
6	0	6	10	6	6	10	6
6	40	9	10	6	6	42	6
					6	13	6
					6	10	6

in which to put it. Because this difficulty can arise whenever the game designer selects a skeleton configuration of postures, IDAHEX provides another way of reducing a task force's defensive capability in this situation: the task force can be credited with negative defense preparation time.[1]

In every example of task force movement thus far, the objective has been a cell adjacent to the task force's location, but Figures 3.1 and 3.2 do not require that. A task force may receive an order stating a desired objective not adjacent to its location. The task force will be able to execute the order only if: (1) it is airmobile and (2) the order causes it to enter an air movement posture.[2] A unit of type i is airmobile if and only if *mrair*(i) > 0. A movement posture pp is an air movement posture if and only if *airmove*(pp-29) = .true. Air movement is discussed in the next subsection.

3.2 EVENT SCHEDULING

Associated with any change of status is a delay time. The task force undergoing the change stays in its old status a length of time equal to the delay, and then enters its new status. The delay is computed by the IDAHEX function wait, *which is designed for easy modification or replacement.*

Throughout this subsection, $u_1,\ldots,u_n$ are the unit numbers of the task force elements. The side to which they belong is s; s = 1 if they are Red, s = 2 if they are Blue. The task force's location is cell pl. Its posture is pp. Its posture class is ppc. (ppc = [floor](pp/10).) Its objective is cell po. Its next location is nl, its next posture is np, its next posture class is npc (npc = [floor](np/10)), and its next objective is no. The preceding subsection reveals how the task force's present status (location pl, posture pp, objective no) and its active order determine its next status (location nl, posture np, objective no). This subsection shows how the delay for the transition from the present status to the next status is determined. Let d denote that delay.

---

[1]Preparation time's effect on attrition is discussed in Section 5.1.1. The way an aborted movement or attack can lead to negative preparation time is discussed in Section 3.3.6.

[2]The constraint is enforced by the event scheduling logic, described in the next subsection.

3.2.1 Transition within Positive Posture Class

npc = ppc, ppc > 0, no = po

Let

$$j = pp - 10*ppc + 1$$

and $$k = np - 10*ppc + 1.$$

Thus, posture pp is the j-th posture of posture class ppc, and posture k is the k-th posture. The delay is given by

$$d = \mathit{ptran}(ppc,j,k).$$

Regardless of the game design data, d = 0 if j = k.

3.2.2 From Hold Posture to Disengagement Posture

ppc = 1, npc = 2

In this case, d = 0.

3.2.3 From March Posture to Attack Posture

ppc = 3, npc = 4, no = po, airmove(pp-29) = .false.

The delay, d, in going from a movement posture to an attack posture with the same objective is the "movement delay". It corresponds to the time the task force would need to move physically from its present location to its objective if unimpeded by the enemy. A "march posture" is any movement posture other than an "air movement posture". (A task force in a march posture might be at sea.) If posture i is a movement posture ($30 \le i \le 39$), it is an air movement posture if and only if *airmove*(i-29) = .true. (*airmove* is a logical variable). Storage is utilized more efficiently if the movement postures are ordered (numbered) so that all the march postures occur before air movement postures--i.e., if *airmove*(k) = .true. for some $1 \le k \le$ *npost*(3), then *airmove*(m) = .true. for every $k < m \le$ *npost*(3). The movement delay for a task force in a march posture may be termed the march delay.

If cell po is nonexistent (po < 1 or po > *ncells*) or inactive, then d = +∞.[1] If cell po is not adjacent to cell pl, then d = +∞: a task force in a march posture cannot jump over cells. Otherwise, proceed.

[1] Usually, an infinite delay results from a mistake by the player. If IDAHEX suspects that is the case, it warns the player of the side to which the task force belongs, explaining why the delay is infinite.

Initially, suppose that the task force consists of a single unit, u_1.

The first step in finding d is ascertaining whether the task force has the supplies it needs in order to move; if *nss*(s) = 0, this step is skipped. For every $1 \le k \le nss(s)$, let ssstock(k) be the quantity of type k supplies in the unit:

$$\text{ssstock}(k) = [\text{resources}](u_1, \text{nequip}(s)+k).$$

Let

$$\text{ssuse}(k) = \sum_{irs=1}^{nrs(s)} \mathit{ssreqm}(k,irs,s) * [\text{resources}](u_1, irs)$$

for every $1 \le k \le nss(s)$. If ssuse(k) > ssstock(k) for some $1 \le k \le nss(s)$, set d = +∞--the unit cannot move. Otherwise, proceed to the next step to find d. The preceding test is crude since ssuse(k)--the amount of type k supplies required for movement--is independent of [rtetype](pl,po) and [bartype](pl,po). Perhaps the best strategy is to make *ssreqm* underestimate the supplies needed for a move. One risks letting the task force change location when it has insufficient supplies to complete the movement, but if *tframe* is suitably small, the risk is minor: each frame, supplies consumption is assessed, and if a task force in a movement posture exhausts any type of supplies, its movement delay is re-evaluated. If the delay is found to be +∞, the movement is aborted and the task force tries to revert to a hold posture in its present location.

Given that the unit has the supplies it needs in order to move, the next step is to determine how fast it can move. The unit's basic movement rate from cell pl to cell po is

$$\text{BMR} = \mathit{mr}(\ \mathit{butype}(u_1),\ \text{pp-29},\ [\text{rtetype}](\text{pl},\text{po})).$$

Thus, its movement rate depends upon its type, its particular movement posture, and the type of route between the cells.[1] The basic movement rate must be adjusted to reflect a deficit or

[1]To see why IDAHEX allows the triple dependence, consider an armored division making an administrative movement along roads through dense woods. If it were making a tactical movement instead, part of it would have to move off-road. If it were a straight-leg infantry division instead, it would have less trouble moving off-road.

surplus of transportation. Let TR be the total transport capacity available to the unit divided by its total demand for transport. (The latter two quantities are defined below.) The unit's adjusted movement rate is

$$\text{AMR} = \text{fmr}(\mathit{butype}(u_1),\text{TR}) * \text{BMR}.$$

The degradation (or possibly improvement) factor fmr(unit_type,TR) is computed as follows. If *fmr.f0*(unit_type) < 0, then

$$\text{fmr (unit\_type, TR)} = \begin{cases} \text{TR} / (2 - \text{TR}); & \text{TR} < 1 \\ 1; & \text{TR} \geq 1 . \end{cases}$$

The preceding formula assumes that the transport capacity cannot be stretched (by overloading vehicles and operating them at reduced speeds, for example); the transporting resources carry as much as they can, offload it, and return to the point of origin for another load, making as many trips as necessary. If, on the other hand, *fmr.f0*(unit_type) ≥ 0, then

$$\text{fmr (unit\_type, TR)} = \text{paf (TR, } \mathit{fmr.f0}\text{(unit\_type), } \mathit{fmr.f}\text{(unit\_type, *), } \mathit{fmr.x}\text{)}.$$

That is, fmr(unit_type,*) is a piecewise-affine (loosely speaking, piecewise-linear) function whose value at 0 is *fmr.f0*(unit_type), whose value at *fmr.x*(k) is *fmr.f*(unit_type,k) for any k, and whose value at TR is found by interpolation.

Thus far, the unit's movement rate, AMR, has been determined. By assumption, the distance it has to travel equals *depth*, which equals the distance between the centers of any two adjacent cells. There is yet another factor that may affect the movement delay: a barrier between cell pl and cell po. The delay imposed by a barrier depends upon the unit's type, the unit's posture, and the type of barrier. Let bt = ⌊bartype⌋(pl,po). A barrier between cells pl and po exists if and only if bt > 0. The unit's march delay is

$$\mathit{depth}/\text{AMR} + \mathit{bdelay}(\ \mathit{butype}(u_1),\ \text{pp-29},\ \text{bt}) \text{ if bt} > 0,$$

$$\mathit{depth}/\text{AMR if bt} = 0.$$

In summary, the march delay for a one-unit task force is found as follows:

(1) See if the unit has enough supplies to be able to move.

(2) Reference *mr* to find the basic rate at which the unit moves from its location to its objective, an adjacent cell.

(3) Adjust the basic movement rate according to the ratio of available transport capacity to the unit's aggregate demand for transport.

(4) Divide the estimated distance to be traveled, *depth*, by the adjusted movement rate; call the result d1.

(5) If there is a barrier between the unit's location and its objective, reference *bdelay* to find d2, the time needed to cross the barrier; d2 = 0 if no barrier exists.

(6) The movement delay is d1 + d2.

Now drop the assumption that the task force contains only one unit. Recall that the task force consists of the units $\{u_i: 1 \leq i \leq n\}$; n = 1 is allowed. The elements' location is cell pl; their posture is pp, and their objective is cell po. Also, recall that cell po must be adjacent to cell pl else the march delay, d, is automatically $+\infty$.

Every task force has the attribute "transport mode", a non-negative integer. It is 0 for a single-element task force. Let the variable named "mode" equal the transport mode of the task force in question. The task force is "stacked" if and only if mode > 0.

3.2.3.1 March Delay for Unstacked Task Force

In this subsection, the task force is assumed to be unstacked--i.e., mode = 0. As before, the movement delay is the sum of two terms, one proportional to the distance and inversely proportional to the movement rate, and the other dependent on the type of barrier encountered. By definition, the task force moves as an integral whole. Therefore, all along the route, it moves only as fast as its slowest element. The elements' supplies and transport are pooled.

The first step is to determine whether the task force has the supplies it needs in order to move. This step is skipped if *nss*(s) = 0. (Recall that the task force belongs to side s.) Let ssstock(k) be the amount of type k supplies in the task force for every $1 \leq k \leq$ *nss*(s). Let

$$\text{ssuse}(k) = \sum_{i=1}^{n} \sum_{irs=1}^{nrs(s)} \mathit{ssreqm}(k,irs,s) * [\text{resources}](u_i,irs)$$

for every $1 \leq k \leq$ *nss*(s). If ssuse(k) > sstock(k) for some $1 \leq k \leq$ *nss*(s), set d = $+\infty$--the task force cannot move. Otherwise, proceed.

The next step is to determine how fast the task force can move. To do that, it is necessary to determine how much transport capacity is made available to each element. This is done even if *ntrpt*(s) = 0 since resources other than transport may have transport capacity. The aggregate demand for transport is

$$\text{ttdemand} = \sum_{i=1}^{n} \sum_{irs=1}^{nrs(s)} \mathit{trnreq}(irs,s) * [\text{resources}](u_i,irs).$$

The aggregate transport capacity is

$$\text{ttcapacity} = \sum_{i=1}^{n} \sum_{irs=1}^{nrs(s)} \mathit{trncap}(irs,s) * [\text{resources}](u_i,irs).$$

Let

$$TR = \begin{cases} \text{ttcapacity/ttdemand; ttdemand} > 0 \\ +\infty\text{; ttdemand} = 0,\ \text{ttcapacity} > 0 \\ 1\text{; ttdemand} = \text{ttcapacity} = 0 \end{cases}$$

For the purpose of determining adjusted movement rates, each unit receives an allocation of transport capacity equal to its demand for transport *multiplied* by TR. Therefore, transport capacity and transport demand must be expressed in the same unit of measure, such as tons. The amount of transport a side s type i resource requires, *trnreq*(i,s), should be 0 if it can move itself. For example, if a type i resource is a tank, it can not only transport itself, so *trnreq*(i,s) = 0; it can transport other resources (a few people, for example), so *trncap*(i,s) > 0. Of course, personnel can move themselves, but if side s is preponderantly mechanized, its units' basic movement rates probably assume the personnel are mounted, and therefore their transport requirements should be positive.

Because each task force element enjoys a ratio of transport capacity to transport demand equal to TR, the adjusted rate of movement of element i ($1 \leq i \leq n$), denoted AMR(i), is given by

$$\text{AMR}(i) = \text{fmr}\ (\mathit{butype}(u_i),\ \text{TR}) * \mathit{mr}(\ \mathit{butype}(u_i),\ \text{pp-29},\ [\text{rtetype}](\text{pl},\text{po})).$$

The task force's adjusted movement rate is

$$\text{TFAMR} = \min\ \{\text{AMR}(i);\ 1 \leq i \leq n\}.$$

If TFAMR = 0, let dl = $+\infty$. Otherwise let dl = *depth* / TFAMR.

Let bt = [bartype](pl,po). If bt = 0, let d2 = 0; otherwise, let

$$d2 = \max \{bdelay(\ butype(u_i),\ pp-29,\ bt);\ 1 \le i \le n\}.$$

Then d = d1 + d2.

3.2.3.2 March Delay for Stacked Task Force

By assumption, mode > 0. Define the set of "carriers" in the task force as

$$C = \{u_i:\ trptcl(butype(u_i)) = \text{mode},\ 1 \le i \le n\},$$

i.e., the set of every task force element whose "transport class" agrees with the task force's transport mode. Define P, the set of "passengers", as the remaining elements of the task force. In effect, the passengers are loaded onto the carriers. The task force moves as fast as the carriers can, and the carriers are able to draw only on their own transport.

The first step is to determine whether the task force has the supplies it needs in order to move; this step is skipped if *nss*(s) = 0. For every $1 \le k \le nss(s)$, let ssstock(k) be the amount of type k supplies held by the task force, not just the carriers. For every $1 \le k \le nss(s)$, let

$$ssuse(k) = \sum_{u_i \varepsilon C} \sum_{irs=1}^{nrs(s)} ssreqm(k,irs,s) * [resources](u_i,irs),$$

the amount of type k supplies that the carriers' resources need in order to move. If ssuse(k) > ssstock(k) for some $1 \le k \le nss(s)$, then set $d = +\infty$. Otherwise, proceed.

The next step is to verify that the carriers have enough carrying capacity to accommodate the passengers. Only those carrier resources whose "load class" agrees with the task force's transport mode are eligible to help carry passengers. Let

$$R = \{i:\ loadcl(i,s) = \text{mode},\ 1 \le i \le nrs(s)\}.$$

Define the total carrying capacity as

$$CC = \sum_{u_i \varepsilon C} \sum_{j \varepsilon R} ldcap(j,s) * [resources](u_i,j).$$

Define the size of the load as

$$LOAD = \sum_{u_i \varepsilon P} \sum_{j=1}^{nrs(s)} \mathit{ldsize}(j,s) * [resources](u_i,j).$$

If LOAD > CC, set d = +∞ -- the task force cannot move. Otherwise, proceed.

The next step is to determine whether, after allocating resources to carry the passengers, the carriers have enough residual transport capacity to meet their own demand. The fraction of the carriers' type i resources allocated to carrying passengers is assumed to be the same for each $i \varepsilon R$; that fraction is LOAD/CC. The carriers' total capacity available for transporting their own resources is

$$ttcapacity = \sum_{u_i \varepsilon C} \sum_{j \varepsilon S} \mathit{trncap}(j,s)*[resources](u_i,j)$$

$$+ \sum_{u_i \varepsilon C} \sum_{j \varepsilon R} (LOAD/CC)*\mathit{trncap}(j,s)*[resources](u_i,j),$$

where $S = \{i: i \notin R, 1 \leq i \leq nrs(s)\}$.

Their resources' demand for transport is

$$ttdemand = \sum_{u_i \varepsilon C} \sum_{j=1}^{nrs(s)} \mathit{trnreq}(j,s) * [resources](u_i,j).$$

Let

$$TR = \begin{cases} ttcapacity/ttdemand; \ ttdemand > 0 \\ +\infty; \ ttdemand = 0, \ ttcapacity > 0 \\ 1; \ ttdemand = ttcapacity = 0 \end{cases}$$

The allocation of transport capacity to each carrier is assumed to equal its demand times TR. Therefore, the adjusted movement rate of the carrier u_i ($u_i \varepsilon C$) is given by

$$AMR(i) = fmr(\mathit{butype}(u_i), TR) * \\ mr(\mathit{butype}(u_i), pp\text{-}29, [rtetype](pl,po)).$$

The task force's movement rate is

$$TFAMR = \min \{AMR(i); u_i \varepsilon C\}.$$

Let

$$dl = \begin{cases} \text{depth / TFAMR if TFAMR} > 0, \\ +\infty \qquad \text{otherwise.} \end{cases}$$

Let bt = [bartype](pl,po). If bt = 0, let d2 = 0; otherwise, let

$$d2 = \max \{\mathit{bdelay}(\ \mathit{butype}(u_i),\ pp\text{-}29,\ bt);\ u_i \in C\}.$$

Then d = dl + d2. The task force moves as fast as the slowest carrier, in accordance with the concept that the passenger units as a group are borne by the carrier units as a group.

3.2.4 From Air Movement Posture to Attack Posture

ppc = 3, npc = 4, no = po, *airmove*(pp-29) = .true.

The air movement delay, d, is determined in basically the same way as the march delay (Section 3.2.3). Since the task force is moving above the surface, its movement rate is unaffected by route types and barrier types. In fact, it may go from one cell directly to a nonadjacent cell--i.e., cell po need not be adjacent to cell pl. Nevertheless, cell po must be in the area of war ($1 \leq po \leq \mathit{ncells}$) and must be active; otherwise, d is set immediately to $+\infty$.

3.2.4.1 Air Movement Delay for Unstacked Task Force

First, determine whether the task force has enough supplies to be able to move. (Skip this step if *nss*(s) = 0.) For every $1 \leq k \leq \mathit{nss}(s)$, let ssstock(k) be the amount of type k supplies in the task force, and let

$$\text{ssuse}(k) = \sum_{i=1}^{n} \sum_{irs=1}^{nrs(s)} \mathit{ssreqm}(k,irs,s) * [\text{resources}](u_i,irs).$$

If ssuse(k) > ssstock(k) for some $1 \leq k \leq \mathit{nss}(s)$, then set $d = +\infty$. Otherwise, proceed.[1]

---

[1] Suppose both aircraft and fuel are played explicitly as side s resources. *If side s type k* supplies include aviation fuel and side s type irs resources are aircraft, *ssreqm*(k,irs,s) might be the quantity of fuel typically carried by a unit-quantity of type irs resources. Making it unrealistically small is risky: the air movement delay might be too short to span a frame boundary; then the movement would escape supplies consumption assessment, which would prevent IDAHEX from observing the task force exhaust essential supplies before the movement were complete.

Next, let

$$\text{ttdemand} = \sum_{i=1}^{n} \sum_{irs=1}^{nrs(s)} \textit{trnreq}(irs,s) * [resources](u_i,irs),$$

$$\text{ttcapacity} = \sum_{i=1}^{n} \sum_{irs=1}^{nrs(s)} \textit{trncap}(irs,s) * [resources](u_i,irs).$$

Let

$$TR = \begin{cases} \text{ttcapacity/ttdemand; ttdemand} > 0 \\ +\infty\text{; ttdemand} = 0\text{, ttcapacity} > 0 \\ 1\text{; ttdemand} = \text{ttcapacity} = 0 \end{cases}$$

The task force's adjusted rate of movement (through the air) is

$$AMR = \min \{fmr(\textit{butype}(u_i),TR) * \textit{mrair}(\textit{butype}(u_i));\ 1 \le i \le n\}.$$

If AMR = 0, then $d = +\infty$. Thus, if a task force element cannot move itself by air, i.e., if

$$\textit{mrair}(\textit{butype}(u_i)) = 0$$

or

$$fmr(\textit{butype}(u_i),TR) = 0$$

for some $1 \le i \le n$, then the task force is unable to move. If AMR > 0, then the movement delay, d, is given by

$$d = dist(pl,po) / AMR;$$

dist(pl,po) is the straight-line distance from the center of cell pl to the center of cell po.

3.2.4.2 Air Movement Delay for Stacked Task Force

Define C, the set of carriers, and P, the set of passengers, as in Section 3.2.3.2.

First, determine whether the task force has enough supplies to be able to move. (Skip this step if *nss*(s) = 0.) For every $1 \le k \le nss(s)$, let ssstock(k) be the amount of type k supplies held by the task force, not just the carriers, and let

$$\text{ssuse}(k) = \sum_{u_i \in C} \sum_{irs=1}^{nrs(s)} \textit{ssreqm}(k,irs,s) * [resources](u_i,irs).$$

If ssuse(k) > ssstock(k) for some $1 \le k \le nss(s)$, then set $d = +\infty$. Otherwise, proceed.

Next, determine whether the carriers have enough carrying capacity to accommodate the passengers. Compute CC and LOAD as in Section 3.2.3.2. If LOAD > CC, set $d = +\infty$. Otherwise, proceed.

Find the ratio of the carriers' residual transport capacity to their own resources' total demand for transport: compute ttcapacity and ttdemand as in Section 3.2.3.2, and define TR as there. The task force's adjusted rate of movement (through the air) is

$$\text{AMR} = \min\{\text{fmr}(\textit{butype}(u_i),\text{TR}) * \textit{mrair}(\textit{butype}(u_i));\ u_i \in C\}.$$

Then

$$d = \begin{cases} \text{dist(pl,po) / AMR} & \text{if AMR} > 0, \\ +\infty & \text{if AMR} = 0. \end{cases}$$

3.2.5 From Disengagement Posture to Movement Posture

ppc = 2, npc = 3, no = po

The delay, d, in going from a disengagement posture to a movement posture with the same objective is the "disengagement delay". It is most simply interpreted as the time required to break contact with the enemy, but in reality a force being pursued by the enemy might never break contact completely. A better interpretation of the disengagement delay is the amount by which contact with the enemy increases the time needed for the task force to relocate from cell pl to cell po.

If the task force is not engaged, set d = 0. Otherwise, proceed.

If cell po is not part of the area of war or is inactive, or if cell po is not adjacent to cell pl, then set $d = +\infty$. Otherwise, proceed.

If *airmove*(np-29) = .true., set $d = +\infty$: an engaged task force cannot disengage and transition directly to an air movement posture. Otherwise, proceed.

Define two situations: (1) there are friendly units in hold postures in cell pl; (2) there are no friendly units in hold postures in cell pl. (No such unit could belong to the task force since its posture would differ from the task force's posture.)

3.2.5.1 Disengagement Delay When Enemy Can Not Pursue

In Situation (1) the friendly units are assumed to prevent the enemy from pursuing the task force during its movement to cell po. Therefore, the disengagement delay is independent of the movement:

$$d = \max\ \{\mathit{diseng}(\mathit{butype}(u_i),1);\ 1 \le i \le n\}.$$

The maximization is appropriate because the task force can not have disengaged until every element has.

3.2.5.2 Disengagement Delay When Enemy Can Pursue

In Situation (2) the enemy units with which the task force is engaged may be able to pursue it during its movement to the adjacent cell po. The disengagement delay therefore is the sum of two terms--one equal to the basic delay, d1, given by

$$d1 = \max\ \{\mathit{diseng}(\mathit{butype}(u_i),1);\ 1 \le i \le n\},$$

and the other term, d2, related to the anticipated movement delay.

Finding d2 parallels Section 3.2.3. Initially, assume that the task force is not stacked.

Defining the symbols as in Section 3.2.3.1, find sstock(k) and ssuse(k) for every $1 \le k \le \mathit{nss}(s)$. If ssuse(k) > sstock(k) for some $1 \le k \le \mathit{nss}(s)$, set $d2 = +\infty$. Otherwise, proceed.

Find TR as in Section 3.2.3.1. Posture np is a movement posture; for the purpose of determining d2, it is assumed to be the posture in which the task force will move to cell po. Define the adjusted movement rate of task force element i as

$$\mathrm{AMR}(i) = \mathrm{fmr}\ (\mathit{butype}(u_i),\ \mathrm{TR}) * \mathit{mr}(\ \mathit{butype}(u_i),\ np\text{-}29,\ [\mathrm{rtetype}](pl,po)).$$

(Cell po is adjacent to cell pl, or this point could not be reached.) If AMR(i) = 0 for any i, set $d2 = +\infty$. Otherwise, proceed. Let

$$w1 = \max\ \{\mathit{diseng}(\ \mathit{butype}(u_i),2) * (\mathit{depth}/\mathrm{AMR}(i));\ 1 \le i \le n\}.$$

Let bt = [bartype](pl,po). If bt = 0, let w2 = 0; otherwise, let

$$w2 = \max \{diseng(butype(u_i),2) * bdelay(butype(u_i),np\text{-}29,bt); \quad 1 \le i \le n\}.$$

Let $d2 = w1 + w2$.

The disengagement delay is computed as $d = d1 + d2$. The computation of w1 and w2 is consistent with the assumption that the task force moves as an integral whole throughout its journey; if w2 were smaller, one or more units must lag behind the rest at a barrier; if w1 were smaller, one or more units must lag behind along the route. The factor *diseng*(i,2) allows for differences in the abilities of different types of units to disengage.

Now assume that the task force is stacked. Define C, the set of carriers, as in Section 3.2.3.2. The basic delay is the same as before:

$$d1 = \max \{diseng(butype(u_i),1); \ 1 \le i \le n\}.$$

Added to it is a delay, d2, related to the anticipated time needed for movement to cell po; d2 depends only on the carriers' agility, not the passengers'.

Determine LOAD and CC as in Section 3.2.3.2. If LOAD > CC set $d2 = +\infty$. Otherwise, proceed.

Find TR as in Section 3.2.3.2. Define the adjusted movement rate of task force element i as

$$AMR(i) = fmr\ (butype(u_i),\ TR) * mr(\ butype(u_i),\ np\text{-}29,\ [rtetype](pl,po)).$$

If $AMR(i) = 0$ for any i such that $u_i \in C$, set $d2 = +\infty$. Otherwise, proceed. Let

$$w1 = \max \{diseng(butype(u_i),2) * (depth/AMR(i));\ u_i \in C\}.$$

Let bt = [bartype](pl,po). If bt = 0, let w2 = 0. Otherwise let

$$w2 = \max \{diseng(butype(u_i),2) * bdelay(butype(u_i),\ np\text{-}29,\ bt);\ u_i \in C\}.$$

Let $d2 = w1 + w2$.

The disengagement delay is computed as $d = d1 + d2$.

3.2.6 From Attack Posture to Hold Posture at New Location

$ppc = 4,\ npc = 1,\ no \neq pl$

The "attack delay" is 0 if cell po contains no enemy units in hold postures. Otherwise, the delay is indefinite: it depends upon the course of combat.

3.2.7 To Hold Posture at Present Location

$npc = 1 \neq ppc,\ no = pl$

If pp < 0, set $d = +\infty$. (A destroyed unit cannot come back to life.) Otherwise, proceed.

At this point, exactly three cases are possible: (1) $0 \le pp \le 9$; (2) $pp \ge 20$, $po \neq pl$, and $no = pl$; (3) $40 \le pp \le 49$ and $po = pl$. (See Section 3.1, especially Figure 3.1.) Case (1) occurs when units in posture class 0 are activated. Case (2) occurs when the task force is in a disengagement, movement, or attack posture and seeks to revert to a hold posture in cell pl; its next posture is an attack posture and its next objective is cell pl. And then Case (3) applies. In every case, d = 0.

3.2.8 Transition to or within Nonpositive Posture Class

$np < 10$

The delay is 0. Since there is normally no reason for a unit to enter posture class 0 from another posture class, a warning is issued to the game designer if that happens. The warning message is placed in the game designer's output file, file 51.

3.3 TACTICAL SITUATIONS

Because the forces can maneuver and the processes of maneuver span time, situations requiring special logic may arise. In many cases they require tactical decisions, in contrast to the player's operational decisions, and therefore should be handled by IDAHEX. In almost every case they must be handled by IDAHEX or absurd results might ensue. Handling these tactical situations with precision is not critical--indeed that would be inconsistent with the model's level of resolution.

Section 3.1 shows how events are determined, and Section 3.2 shows how they are scheduled. The events are arranged implicitly in a queue in order of scheduled occurrence; the event scheduled to occur next is at the front of the queue. A change of status by a task force in the attack posture class comes after a change of status by a task force in another posture class if both events are scheduled for the same time. When an event comes to the front of the queue, t is advanced to the time a at which it is scheduled to occur, and the event is passed to the subprogram xeq for execution. Instead of executing the event, xeq may alter the queue--adding events to it, or changing the times at which events are scheduled to occur and changing the order of events in the queue.

Some terms are needed. To "occupy" a cell is to change location of the cell. A side's "security force" in a cell consists of every friendly unit that is located in the cell and is in a hold posture. A unit or task force whose posture is disengagement, movement, or attack, and whose objective is cell j, is equivalently said to be in a disengagement, movement, or attack posture oriented toward cell j, or to be disengaging, moving, or attacking toward cell j.

The variable eps = *tframe* / 100.

3.3.1 Pursuit

Suppose a Blue task force in cell i enters a movement posture oriented toward cell j, an adjacent cell. Suppose that later a Red task force occupies cell i and subsequently enters a movement posture oriented toward cell j. If the Red task force is more mobile, its movement delay may be less than the Blue task force's delay--so much less that its movement delay ends before the Blue task force's. But because xeq implements the following rule, the Red task force cannot occupy cell j before the Blue task force.

Let task force m and task force n belong to opposite sides. Suppose the location, posture class, and objective of task force m coincide with the location, posture class, and objective of task force n. Also suppose that the next location, next posture class, and next objective of task force m coincide with the next location, next posture class, and next objective of task force n and the task forces' next posture class differs from their present posture class. Let $u_1, \ldots, u_j$ be the identification numbers of the units in task force m, and let $v_1, \ldots, v_k$ be the identification numbers of the units in task force n. If

$$\min \{tentry(u_i);\ 1 \le i \le j\} > \min \{tentry(v_i);\ 1 \le i \le k\},$$

then task force m may enter its next status no sooner than eps/4 after task force n.

3.3.2 Attack

An important variable in many tactical situations is [owner]; [owner](i) = 1 if cell i is owned by Red and 2 if the cell is owned by Blue. The game design data set [*owner*], and then IDAHEX sets [owner] = [*owner*]. Thus, the design data declare the ownership of each active cell at the start of the game. This subsection shows, among other things, how [owner] gets changed.

Suppose task force m, belonging to side sa (sa = 1 or sa = 2), is in an attack posture. Let sd = 3 - sa; side sd is its enemy. Suppose the task force's location is cell pl, its objective is cell po, its next posture is np, and its next objective is no; no = pl is permitted. Assume posture np is a hold posture. Assume the task force's attack delay (possibly 0) is complete, the task force has reached the front of the queue, and the subprogram xeq has been called to execute the task force's transition to its next status, a hold posture in cell po. The rest of this subsection charts the actions taken by xeq in this case. The verb "return" means "return from xeq to the calling program".

Step 1. If task force m is already engaged, go to Step 6. Search for side sd task forces whose location is cell po, whose posture class is 2, 3, or 4, and whose objective is cell pl. If none exist, go to Step 2. Do the following for each such task force: make its desired objective po; make its desired posture

pmapup(post)	if it is attacking,
pmapup(*pmapup*(post))	if it is moving,
pmapup(*pmapup*(*pmapup*(post)))	if it is disengaging,

where post is its posture; schedule its next change of status for time t, and place it ahead of task force m in the queue. Return. This procedure leads eventually to an engagement in which task force m is attacking side sd units holding in cell po; it obviates an entirely separate combat procedure for meeting engagements.

Step 2. If an engagement already exists at cell po, go to Step 4. If [owner](po) = sa, go to Step 3. Search for enemy task forces in movement or attack postures oriented toward cell po whose next change of status is scheduled to occur no later than t + eps. If none exist, go to Step 3. Reschedule the next change of status of each of these task forces to time t and place it ahead of task force m in the queue. Return. This step resolves virtual ties in times at which hostile units arrive at a cell in favor of the cell's current owner.

<u>Step 3</u>. Search for active side sd units located in cell po. If none exist, let task force m change status (let it occupy cell po), let [owner](po) = sa, and return. If cell po contains a side sd unit in a hold posture other than posture *itrfp*, go to Step 4. Let S be the set of every side sd unit whose location is po and whose posture is *itrfp*. Two cases are possible. Case 1: S is nonempty. In this case, constitute every member of S that does not belong to a task force as a task force, give it the order "desired objective = po, desired posture = 10", and position it in the queue according to the time of its next change of status. Let T be the set of every task force whose elements are members of S. For each task force in T, if there is a start time associated with the task force's active order, and it exceeds t, reset it to t and therefore reschedule the task force's next change of status. Now for each task force in T, if the task force's active order specifies a hold posture in cell po and the next change of status is scheduled to occur no later than t + eps, reschedule it to occur at t and move the task force ahead of task force m in the queue. Return. Case 2: S is empty. In this case, let T be the set of every side sd task force located in cell po whose objective is owned by side sa and whose objective contains one or more active side sa units. For each task force in T, determine whether the task force could execute the first change of status implied by the order "desired objective = po, desired posture = 10" no later than t + eps; if so, make that its active order, schedule its next change of status for time t, and place it ahead of task force m in the queue. If one or more task forces have received new orders in this way, return.

<u>Step 4</u>. If cell po contains no side sd security force, go to Step 5. If [owner](pl) = sd, change the active order of task force m to "desired objective = pl, desired posture = -10", schedule its next change of status for time t (keep task force m at the front of the queue), and return. (The units in task force m are destroyed because they are attacking at the same time the enemy owns their base. It is inappropriate to let their location be cell pl, and they have not been able to occupy cell po; therefore they must be removed from the area of war. Step 3 alters orders and re-sequences the queue to avert such catastrophes whenever possible.) If there is no engagement in progress at cell po, set one up between task force m and the side sd units in hold and disengagement postures in cell po. Otherwise, join task force m to the existing engagement. Reschedule the next change of status of task force m to occur at time t $+\infty$. Return.

Step 5. Reaching this point implies there is no side sd security force in cell po, but one or more active units from side sd are located there. Let S be the set of every side sd unit whose location is cell po and whose posture is a movement posture. For each unit $u \in S$, if tentry(u) > t - *delta*, change the unit's posture to *pmapup*(*pmapup*(*pmapup*(post))), where post is its present posture. (Side sd units that started moving from cell po within the interval *delta* of the arrival of task force m must revert to disengagement postures.) Take each side sd unit located in cell po and disengaging, and join it in an engagement with task force m. Take each side sd unit located in cell po that is attacking in some engagement, constitute it as a task force if not already an element of one, give the task force the active order "desired objective = po, desired posture = -10", and place the task force at the front of the queue. Let task force m enter its next status. (Let it occupy cell po.) Return.

Step 6. This point is reached if and only if task force m is already engaged. For that to happen, xeq must have been called once before to execute the task force's transition from an attack posture oriented toward cell po to a hold posture in cell po, and Step 4 must have joined the task force in an engagement. When that happened, xeq did not let the task force enter its next status; in fact, it rescheduled the change of status to occur after the end of the game. Subsequently, the change of status was rescheduled as Section 3.3.3 explains, and task force m again reached the front of the queue, inducing the current invocation of xeq. Proceed as follows. Take each side sd unit located in cell po that is in an attack posture and engaged, constitute it as a task force if it does not already belong to one, give the task force to which it belongs the active order "desired objective = po, desired posture = -10", and place the task force at the front of the queue. Let task force m enter its next status. (Let it occupy cell po.) Return.

3.3.3 Disappearance of a Security Force

Suppose cell pl is owned by side s (s = 1 or s = 2) and contains one or more units from side s in hold postures, and suppose one or more units from side 3-s are attacking toward cell pl. Suppose a task force consisting of the entire side s security force in cell pl now enters posture class -1, 0, or 2. Then the delay of every side s task force located in cell pl and disengaging is re-evaluated, possibly causing rescheduling of the task force's next change of status. This is necessary because a delay computed when a friendly security force existed might no longer be appropriate; in particular, a disengagement delay might have to be extended now that the enemy can pursue. Next, the active order of every side 3-s task force attacking toward cell pl is inspected. If the order implies that the task force's next change of status is something other than just a transition to another attack posture oriented toward cell pl, the change of status is rescheduled to t and moved to the front of the queue (giving the task force the opportunity to occupy cell pl). Otherwise, the order is discarded, so that the next order, if any, in the task force's mission becomes the active order, and the test is repeated.[1] The process continues until either the test is passed or no orders remain in the task force's mission.

3.3.4 Counterattack

IDAHEX structures every engagement in such a way that units from one side are attacking and units from the other side are defending. A defender is in a hold posture or a disengagement posture. It is possible that all defenders in the engagement are holding, or all disengaging, or some holding and some disengaging. The defenders are all located in the same cell, the cell under attack, while the attackers may be located in different cells. An attacker is in an attack posture unless its location is the same as the defenders'. In a counterattack, a task force consisting of one or more defenders disengages, moves, and attacks toward the location of one or more of the attackers. Suppose the defenders' location is cell i. Suppose task force m consists of one or more of the defenders in hold postures, and xeq has been called to execute its transition to a disengagement posture oriented toward cell j, the location of one or more of the attackers. Let A be the set of the attackers located in cell j. If task force m is not stronger than A, the task force's active order is changed to "desired objective = i, desired posture = 10", its next change of status is scheduled for time t, and it is placed at the front of the queue. If task force m is stronger than A, A's attack is aborted: each unit in A that does not belong to a task force is constituted as one; each task force contained in A is given the active order "desired objective = j, desired posture = 10", its next change of status is schdeuled for time t, and it is placed ahead of task force m in the queue; xeq returns without executing the transition of task force m to its next status. The criterion for deciding whether task force m is stronger than A is as follows. Let $u_1,\ldots,u_n$ be the task force's elements, identified by unit numbers. Let s = 1 if the task force is Red and s= 2 if it is Blue. The attack strength of task force m is given by

[1]Missions are explained in Section 4. Basically, a mission is a sequence of orders for a task force.

$$f0 = \sum_{k=1}^{n} \sum_{irs=1}^{nrs(s)} rsvala(irs,s) * [resources](u_k,irs).$$

The number rsvala(irs,s) is the standard value of a side s type irs resource on attack; its computation is explained in Section 5.3. Basically, rsvala(irs,s) measures the contribution of a single type irs resource belonging to a standard side s force attacking a standard enemy force in a standard engagement. The defense strength of task force m is given by

$$g0 = \sum_{k=1}^{n} \sum_{irs=1}^{nrs(s)} rsvald(irs,s) * [resources](u_k,irs).$$

The number rsvald(irs,s) is the standard value of a side s type irs resource on defense. Let $v_1,\ldots,v_r$ be the units in the set A, identified by their numbers. Let $s' = 3 - s$. The attack strength of A is given by

$$f1 = \sum_{k=1}^{r} rsvala(irs,s') * [resources](v_k,irs).$$

The defense strength of A is given by

$$g1 = \sum_{k=1}^{r} rsvald(irs,s') * [resources](v_k,irs).$$

Task force m is considered stronger than A if and only if

$$\frac{f0}{g1} > \frac{f1}{g0}.$$

3.3.5 Activation of Inactive Task Force

Suppose xeq has been called to execute the transition of a task force whose elements are in a nonpositive posture class to a positive posture class. If the task force's next location (the cell where it will become active) is owned by the enemy, or if one or more active enemy units are located there, xeq does not execute the change of status and, instead, reschedules it for t = +∞ and warns the player of the side to which the task force belongs.

3.3.6 Virtual Time of Posture Class Entry

When a task force transitions from its present status to its next status, tentry may be reset for each of its elements. Let ppc be the task force's present posture class, po its present objective, and pl is present location. Let npc be its next posture class and no is next objective. Let units $\{u_i; 1 \leq i \leq n\}$ be its elements.

If npc = ppc, tentry is not changed. Henceforth, assume npc $\neq$ ppc.

If npc = 2 or npc = 3, IDAHEX sets

$$\text{tentry}(u_i) = t$$

for every $1 \leq i \leq n$. That is, the virtual time at which the units enter the next posture class equals the actual time.

If npc = 1, or if npc = 4 and no $\neq$ pl, IDAHEX sets

$$\text{tentry}(u_i) = \max\{t, \text{tentry}(u_i)\}$$

for every $1 \leq i \leq n$.

In the remaining case, npc = 4 and no = pl; the task force is aborting a disengagement, movement, or attack and trying to revert to a hold posture in cell pl. If there is no enemy task force whose objective is pl, whose location is not po, and whose posture class is 2, 3, or 4, then

$$\text{tentry}(u_i) \leftarrow \max\{t, \text{tentry}(u_i)\}$$

for every $1 \leq i \leq n$. Otherwise, tentry is determined as follows. If ppc = 2, then

$$\text{tentry}(u_i) \leftarrow \max\{t, \text{tentry}(u_i)\}$$

for every $1 \leq i \leq n$. If ppc = 3, then

$$\text{tentry}(u_i) \leftarrow \max\{t + (t - \text{tentry}(u_i)), t\}$$

for every $1 \leq i \leq n$; that is, the virtual time of entry is set ahead of the current time by the length of time the unit has been moving. Finally, if ppc = 4, $\text{tentry}(u_i)$ is, for every $1 \leq i \leq n$, set equal to t plus the movement delay that would be computed for the (entire) task force were it moving from cell po to cell pl in posture 30.

Thus, if a task force aborts a movement or attack and an enemy unit directly threatens to seize its location from the flank or rear, tentry for its elements is set ahead in time to indicate just how far out of position it is. Because of the way tentry is set for transitions into posture class 1, this penalty is retained when the task force subsequently reverts to holding its present location. The combat procedure uses t - tentry(j) as a measure of the length of time unit j has had to prepare a defense; if unit j has aborted a movement or attack and reverted to holding its location, its preparation time may be negative.

The combat procedure may also reset tentry.[1] If, during one frame of a given engagement, the FEBA (measured by the variable feba) advances, then for each defending unit, say unit i, tentry(i) is reset to t0 + *tframe*, where t0 is the value of tentry(i) at the start of the frame. Thus, the defenders' level of preparation cannot increase while the attackers are making progress.

3.3.7 Engagement Termination

Suppose engaged task force m goes from a disengagement posture to a movement posture, or enters a nonpositive posture class (its units are destroyed or de-activated), or breaks off an attack and tries to revert to a hold posture in its own location. Then xeq deletes the task force's elements from their engagement. If no units from their side remain in the engagement, then the engagement terminates. In this event, xeq may re-schedule the times at which enemy units that were engaged enter new statuses. Suppose task force n, an enemy of task force m, was participating in the terminated engagement. The time at which it is scheduled to enter its next status is reset to min {t0, t1}, where t0 is the time at which it is presently scheduled to enter its next status, and t1 is the time at which the task force (which is now not engaged) would enter its next status if it were just beginning its transition to its next status. The result is that disengaging task forces can immediately enter movement postures.

[1]Understanding this paragraph precisely requires some knowledge of Section 5 ("Combat").

4. THE PRIMARY COMMANDS

At the start of each period, the Red player and the Blue player input commands to IDAHEX. A command is an instruction to battle units or a request for information. IDAHEX prevents a player from issuing instructions to enemy units or obtaining the enemy player's instructions to his units. The commands are fully described in the *Player's Manual*. This section discusses only the three most important commands, which are all instructions to battle units.

4.1 MISSION COMMAND

Recall from Section 3.1 that a task force's change of status is always caused and directed by an order. A mission is a sequence of orders. Every task force has a mission, and every mission is assigned to exactly one task force (but two task forces may have identical missions). The same positive integer that identifies the task force identifies its mission. A mission's orders are stored in a pop-up stack, and are executed in sequence, from the top to the bottom. The order at the top of the stack is termed the "active order". If there is a start time associated with it, execution does not begin until the current time equals or exceeds the start time. When execution of the order is completed, it is removed from the stack, and the next order, if any, pops to the top.

A mission is created or modified by the mission command. If the player is modifying an existing mission, he identifies it by number and then lists the new orders in the sequence in which they are to be executed. The new orders completely replace the old orders. If the player is creating a new mission, he lists the orders, the elements of the task force (identified by their unit numbers), and finally, if there is more than one element, he selects the task force's transport mode. Creation of the mission also creates the task force. When the mission ends, because it is accomplished or canceled, the task force ceases to exist as an organizational entity, and the number assigned to it and its mission becomes available for identifying a new task force and mission.

The following two examples are based on the area of war in Figure 3.3 and the posture configuration assumed by Table 3.2, namely:

npost(1) = 4, *npost*(2) = 1, *npost*(3) = 2, *npost*(4) = 3

pp	*pmapup*(pp)	*pmapdn*(pp)
10	20	-10
11	20	-10
12	20	-10
13	20	-10
20	30	42
30	40	42
31	41	42
40	10	42
41	11	42
42	13	42

Example 1. Assume units 4 and 9 are both in the same hold posture in cell 17. In the following communications with IDAHEX, the Red player constitutes units 4 and 9 as a task force and assigns it a mission. Every line that IDAHEX writes on a player's terminal is preceded by a question mark to distinguish it. (IDAHEX does not actually write the question mark.) The player's replies are enclosed in quotation marks.

```
? Enter command.
"mission"
? Enter orders.
"16, 31, 0"
"16, 11, 0"
"12, 10, 2.65"
""
? List task force.
"4,9"
? Enter transport mode.
"0"
```

Each of the three lines after the prompting phrase "Enter orders." states an order: the first number is the desired objective, the second the desired posture, and the third is the order's start time. The mission implies the following sequence of statuses for the task force consisting of units 4 and 9:

location	posture	objective
17	20	16
17	30	16
17	31	16
17	41	16
16	11	16
16	20	12
16	30	12
16	40	12
12	10	12

Example 2. Assume the posture class of unit 21 is 0. In the following communications with IDAHEX, the Blue player creates a mission for the task force consisting of unit 21:

```
? Enter command.
  "mission"
? Enter orders.
  "6, 12, 0"
  "9, 12, 0"
  ""
? List task force.
  "21"
```

The mission implies the following sequence of statuses for unit 21:

location	posture	objective
6	10	6
6	12	6
6	20	9
6	30	9
6	40	9
9	10	9
9	12	9

The example illustrates one way of accomplishing re-supply and replacement: if new resources should enter the area of war in cell i at time tr, the game design data should incorporate them into a battle unit whose initial location is i and initial posture class is 0, and then when t $\geq$ tr the player whose side should receive the resources can issue a mission command to activate the unit. An inactive unit first assumes posture 10 when it is activated. (See Figure 3.1.)

Example 3. Assume the posture class of unit 21 is 0. In the following communications with IDAHEX, the Blue player activates unit 21 in cell 8 instead of its present location, cell 6:

```
? Enter command.
  "mission"
? Enter orders.
  "8, 0, 0"
  "8, 10, 0"
  ""
? List task force.
  "21"
```

The mission implies the following sequence of statuses for unit 21:

location	posture	objective
8	0	8
8	10	8

Thus, a player can activate one of his units in a cell different from its initial location; to do so, he must first change its location while it remains in posture class 0. This capability is necessary since the location where a package of supplies and replacements should become available might depend on the course of the game: in the first place, IDAHEX prohibits activation of a unit in a cell owned by the enemy or containing enemy units; and it may be convenient to design the game so that supplies and replacements originate in corps, army, or front depots, which relocate to keep up with the combat forces, rather than fixed, theater depots. A player could use the capability to change inactive units' locations in order to cheat, activating units wherever (and whenever) he pleased. Therefore, IDAHEX places an advisory message in the game designer's output file, file 51, whenever an inactive unit changes location.

In every example the mission's last order declares a hold posture as the desired posture. That is not essential because the player can always extend (modify) a mission some time after creating it. But he should avoid letting a task force complete its mission in a posture class other than -1, 0, or 1: to save time IDAHEX occasionally assumes that every disengaging, moving, or attacking unit belongs to a task force.

4.2 REDISTRIBUTING RESOURCES

One set of active units, called the "givers", can transfer resources to another set of active units, called the "takers", subject to these restrictions: the givers and the takers must all belong to the same side, the givers and the takers must all have the same location, and the givers must be in the transfer posture, posture *itrfp*. A taker may be in any of the postures 10 through 49, including posture *itrfp*, and a unit may be both a giver and a taker. If unit j is a taker, it can accept any quantity of type irs resources, even if $toe(butype(j),irs) = 0$, provided $irs = iars(i, butype(j))$ for some $1 \leq i \leq nrst(butype(j))$.

4.2.1 The Transfer Command

The transfer command causes an immediate, instantaneous transfer of resources from the givers to the takers. The command includes a list of the givers, a list of the takers, and the amount of each type of resource to be transferred from the set of givers to the set of takers. As an essential part of the command, the player declares the transfer location--the givers' and takers' location. If the player declines to furnish a list of givers, the list consists by default of every friendly unit whose location is the transfer location and whose posture is the transfer posture. If he fails to furnish a list of takers, the list consists by default of every active, friendly unit whose location is the transfer location and whose posture is not *itrfp*. If, despite the defaults, there are no givers or no takers, no transfer is made, and the player is warned. The player may also decline to declare the transfer amounts of one or more types of resources.

Let G be the set of givers, identified by their unit numbers, and T the set of takers, identified by their unit numbers. Let s = 1 if the units are Red and s = 2 if they are Blue. If G = T, then regardless of what transfer amounts the player specifies, all resources are pooled, and apportioned among the units. Assume G = T. Let $1 \leq irs \leq nrs(s)$. Define

$$T' = \{k \in T: iars(j,butype(k)) = irs \text{ for some } 1 \leq j \leq nrst(butype(k))\}.$$

T′ is the set of takers that can have type irs resources.

Let

$$T'' = \{k \in T': toe(butype(k), irs) > 0\}.$$

If T'' is nonempty, the resources of type irs are redistributed so that after redistribution

$$[resources](k, irs) \; / \; toe(butype(k), irs)$$

is the same for every $k \in T''$ and $[resources](k, irs) = 0$ for every $k \in T - T''$.[1] Alternatively, if T'' is empty, the resources are redistributed so that $[resources](k, irs)$ is the same for every $k \in T'$ (and 0 for every $k \in T - T'$).

Henceforth, assume $G \neq T$. If, for any i, the player does not declare the amount of type i resources to be transferred, it is determined as

$$\min\{demand, supply\},$$

where

$$demand = \sum_{k \in T} \max\{toe(butype(k), i) - [resources](k, i), 0\}$$

and

$$supply = \sum_{k \in G} \max\{[resources](k, i) - toe(butype(k), i), 0\}.$$

That is, each taker demands the amount by which its stock falls short of its planned effective stock, and each giver demands the amount by which its stock exceeds its planned effective stock. If the player does declare the amount of type i resources to be transferred, it is reset if necessary so that it does not exceed

$$\sum_{k \in G} [resources](k, i),$$

the amount available. Let amt(irs) be the amount of type irs resources to be transferred.

The first step in accomplishing the transfer is allocating the resources among the takers. Let $1 \leq irs \leq nrs(s)$. Define T' and T'' as before. For any $k \in T$, let q(k) be the quantity of type irs resources to be transferred to unit k, which must now be determined. If T' is empty, q(k) is set to 0 for every

---

[1] $T - T''$ is the set of every k such that $k \in T$ but $k \notin T''$.

k, and amt(irs) is reset to 0. Hence, assume T' is nonempty. Case 1: T'' is nonempty. Then the quantity amt(irs) is distributed among the battle units of T'' so as to equalize as much as possible their ratios of actual stock to planned effective stock. To be precise, q(k) is set to 0 for every $k \in T - T''$, and q(k) is chosen for every $k \in T''$ to

$$\text{minimize} \sum_{k\in T''} \left(1 - \frac{[\text{resources}](k,\text{irs}) + q(k)}{\mathit{toe}(\mathit{butype}(k),\text{irs})}\right)^2$$

$$\text{subject to} \sum_{k\in T''} q(k) = \text{amt(irs)}$$

$$q(k) \geq 0 \text{ for every } k.$$

Alternatively, assume Case 2: T'' is empty. Then the quantity amt(irs) is distributed among the battle units of T' so as to equalize their stocks as much as possible. To be precise, q(k) is set to 0 for every $k \in T - T'$, and q(k) is chosen for every $k \in T'$ to

$$\text{minimize} \sum_{k\in T'} \left([\text{resources}](k,\text{irs}) + q(k)\right)^2$$

$$\text{subject to} \sum_{k\in T'} q(k) = \text{amt(irs)}$$

$$q(k) \geq 0 \text{ for every } k.$$

In both cases, after q is determined the transfer occurs: [resources](k,irs) is increased by the quantity q(k) for each $k \in T$.

To complete the transfer, the givers must be assessed for the resources that have already been distributed to the takers. Let $1 \leq \text{irs} \leq \text{nrs}(s)$. For any $k \in G$, let q(k) be the quantity of type irs resources to be taken from unit k, which must now be determined. Define

$$G' = \{k \in G: \mathit{toe}(\mathit{butype}(k),\text{irs}) = 0\}.$$

Let

$$Q = \sum_{k\in G'} [\text{resources}](k,\text{irs}).$$

For every $k \in G'$, q(k) is set equal to

$$\min\{\text{amt(irs)}/Q, 1\} * [\text{resources}](k,\text{irs}).$$

If $Q \geq \text{amt(irs)}$, q(k) is set to 0 for every $k \in G - G'$. Otherwise,

$q(k)$ is chosen for every $k \in G - G'$ to equalize as much as possible the units' ratios of actual stocks to planned effective stocks: $q(k)$ is chosen for every $k \in G - G'$ to

$$\text{minimize} \sum_{k \in G-G'} \left(\frac{[\text{resources}](k,\text{irs}) - q(k)}{toe(butype(k),\text{irs})} - 1 \right)^2$$

$$\text{subject to} \sum_{k \in G-G'} q(k) = \text{amt}(\text{irs})$$

$$q(k) \geq 0 \text{ for every } k.$$

After q is determined the transfer occurs:

$$[\text{resources}](k,\text{irs}) \leftarrow [\text{resources}](k,\text{irs}) - q(k)$$

for every $k \in G$.

4.2.2 The Delivery Command

The delivery command allows the player to arrange a transfer of resources that will occur automatically, at the earliest possible moment. The command does this by creating a "delivery order" (not to be confused with the "orders" in a mission). A delivery order has four components: (1) the delivery task force; (2) the delivery destination; (3) the delivery size; (4) the intended recipients of the delivery. The delivery task force, identified by number, is the set of battle units intended to transfer resources to another set of units. The delivery destination is the cell where the delivery will occur. The delivery size is a number between 0 and 1, inclusive, that indicates how much should be transferred. The intended recipients must all belong to the player's side. The list of intended recipients may be empty. Once created, a delivery order continues to exist until the player cancels it or it is executed. Two or more delivery orders may name the same delivery task force, but if the delivery destinations are the same as well, confusion may result.

Suppose task force m has just entered posture *itrfp* in cell dd. IDAHEX must decide whether the transfer of resources will be governed by a transfer command that the player will issue later or by a delivery order. *IDAHEX infers that the player intends to issue a transfer command, and therefore makes no delivery of resources at this time, if either of the following conditions holds:*

(1) with this change of status, the task force has accomplished its mission;

(2) with this change of status, the task force has completed execution of its active order, and its new active order has a start time that exceeds the current time.

If neither condition holds, IDAHEX *searches for a delivery order--one whose delivery task force is* m *and delivery destination is* dd. *If none is found, a delivery order is generated, with the delivery task force* = m, *delivery cell* = dd, *delivery size* = 1.0, *and intended recipients = the empty set; a generated delivery order is treated as any other delivery order.*

Execution of the delivery order is a procedure very similar to the one initiated by a transfer command. Let G be the set of elements of task force m, identified by their unit numbers. G is the set of "givers". Let lambda be the delivery size, and let R be the set of intended recipients, identified by their unit numbers. If R is nonempty, let T be the set of every $k \in R$ such that unit k is active and located in cell dd; if R is empty, let T be the set of every active, friendly unit located in cell dd whose posture is not *itrfp*. T is the set of "takers". If T is empty, of course, no transfer occurs.

If G = T, then the resources are redistributed exactly as described for that case in Section 4.2.1.

Assume $G \neq T$. The amount of type i resources to be transferred is determined as

$$\min \{\text{demand}, \text{supply}\},$$

where

$$\text{demand} = \sum_{k \in T} \max \{\textit{toe}(\textit{butype}(k),i) - [\text{resources}](k,i),\ 0\}$$

and

$$\text{supply} = \text{lambda} * \sum_{k \in G} \max \{[\text{resources}](k,i) - \textit{toe}(\textit{butype}(k),i),\ 0\}.$$

Since the delivery size, lambda, may not exceed 1, a giver can only give away resources to the extent they exceed its planned effective stock. The first step in accomplishing the delivery is allocating the resources among the takers. The procedure is exactly the same as described in the previous subsection. The second step is assessing the givers for the resources that have been distributed to the takers. The procedure is exactly the same as described in the previous subsection.

5. COMBAT

As Section 3.3.2 explains, an engagement arises when units from one side attempt to occupy a cell containing enemy units in hold or disengagement postures. An engagement is not precipitated merely by a task force's entering an attack posture oriented toward a cell containing enemy units in hold or disengagement postures. The engagement arises when the task force attempts to change status from the attack posture oriented toward the enemy-owned cell to a hold posture in the enemy-owned cell.[1] The cell is termed the "engagement location". The force that precipitates the engagement, by attempting to occupy an enemy-owned cell, constitutes the engagement "attackers". Other friendly units may join the engagement later, possibly attacking from different locations; they, too, become "attackers". The enemy units whose location is the attacker's objective and whose postures are hold or disengagement constitute the engagement "defenders". Thus, at the outset of the engagement, one side is the attacker and the other side is the defender. These roles remain fixed throughout the engagement: even if the attackers succeed in occupying the engagement location, so that they are in hold postures and no longer attack postures, they are still the "attackers". An engagement ends when all its attackers have left or all its defenders have left. If an attacker's location is not the engagement location, it leaves its engagement when its objective becomes a cell other than the engagement location. If an attacker's location is the engagement location, it leaves its engagement when it enters a posture class other than 1 or 2. A defender leaves its engagement when it enters a posture class other than 1 or 2. Therefore, an attacker or defender leaves its engagement if it is destroyed (posture class -1).

Usually, a defender leaves its engagement by entering a movement posture.[2] While it is moving, the enemy cannot engage it. That is one reason for the disengagement delay, and especially for making one term of the delay proportional to the anticipated movement delay. (See Section 3.2.5.2.) Loosely speaking, if the

[1]Sometimes, as Section 3.3.2 explains, the attempt by a task force in a attack posture to occupy its objective causes enemy units located there to divert to hold postures. After they have done so, it re-attempts occupation, precipitating an engagement.

[2]It is impossible for a unit to enter a movement posture oriented toward its own location.

tactical situation implies that the unit is vulnerable to pursuit by engagement attackers, its disengagement delay (hence, the interval during which it is engaged) is extended to account for the combat that its rearguard would have in reality with pursuing enemy units.

Each engagement has a stylized FEBA that measures the attackers' progress. In any given engagement, the variable feba expresses the FEBA position as a fraction of *depth*. At the start of the engagement, feba = 0. At that point, all the attackers are in attack postures oriented toward the engagement location. If the attackers are sufficiently strong relative to the defenders, the FEBA advances--feba increases, to a maximum of 1. One might imagine that when feba is increasing the attackers are beating back the defenders; a more general, and more contemporary, interpretation is that the attackers are penetrating the defenders' formation. The game design datum *febad* is the criterion for deciding when the attackers have penetrated sufficiently to be allowed to occupy the engagement location. As soon as feba $\geq$ *febad*, ownership of the engagement location passes to the attackers' side, the attackers are allowed to enter the cell, and the defenders are forced to disengage and move out or be destroyed.

An engagement's FEBA is independent of other engagements' FEBAs and the general disposition of forces in the area of war. It may be interpreted as a measure of the attackers' penetration of the engagement location. But essentially it is just an abstraction used to determine how long the engagement lasts before the attackers defeat the defenders.

At the end of each frame, the results of every engagement during the frame are evaluated. If an engagement starts during a frame, the attackers cannot possibly occupy the engagement location until the end of the frame, when the engagement's feba is updated. Therefore, *tframe* should be short enough to avoid delaying attackers excessively.

5.1 THE ATTRITION PROCESS

Attrition is essentially a Lanchester square process. The game design datum *katk*(i,j,k) is the quantity of enemy type j materiel destroyed in one unit of time by a single side k type i ground-to-ground weapon belonging to an attacker, under the assumption that the side k weapon allocates all its fire to enemy type j materiel. The quantity destroyed in one frame is

$$\text{katk}(i,j,k) = \textit{tframe} * \textit{katk}(i,j,k).$$

The datum *kdef*(i,j,k) is the quantity of enemy type j materiel destroyed in one unit of time by a single side k type i ground-to-ground weapon belonging to a defender, under the assumption that the side k weapon allocates all its fire to enemy type j materiel. The quantity destroyed in one frame is

$$\text{kdef}(i,j,k) = \textit{tframe} * \textit{kdef}(i,j,k).$$

Let cell loc be the engagement location of a given engagement. Let units $\{\text{atk}(i);\ 1 \le i \le \text{natk}\}$ be the attackers and units $\{\text{def}(i);\ 1 \le i \le \text{ndef}\}$ the defenders. Let sideA = 1 if the attackers are Red and sideA = 2 if they are Blue; let sideD = 3 - sideA. For each $1 \le i \le \text{nrs(sideA)}$, let

$$\text{rsatk}(i) = \sum_{k=1}^{\text{natk}} \text{frinv}(i,\text{atk}(k)) * [\text{resources}](\text{atk}(k),i),$$

the attackers' total quantity of type i resources that can become actively involved in combat or combat support. The function frinv is explicated in Section 5.4. Briefly, frinv(i,j) is the fraction of type i resources held by unit j that are available for combat, if the unit's type i resources are equipment, or that are available and needed for combat support if its type i resources are support resources. For each $1 \le i \le \text{nrs(sideD)}$, let

$$\text{rsdef}(i) = \sum_{k=1}^{\text{ndef}} \text{frinv}(i,\text{def}(k)) * [\text{resources}](\text{def}(k),i),$$

the defenders' total quantity of type i resources that can become actively involved in combat or combat support. The current time, t, must coincide with the end of a frame. This subsection's goal is to derive the attrition suffered by each attacker and each defender during the frame just ended.

5.1.1 Determining the Kill Matrices

Select an attacker-defender pair: for some $1 \le i \le \text{natk}$ and some $1 \le j \le \text{ndef}$, let

$$\text{unitA} = \text{atk}(i),\ \text{unitD} = \text{def}(j).$$

Of course, unitA is a positive integer identifying a battle unit. The phrase "battle unit unitA" is abbreviated below as simply "unitA". The phrase "battle unit unitD" is abbreviated as simply "unitD".

If one of unitA's type i ground-to-ground weapons ($1 \le i \le$ *nggwep*(sideA)) allocates all its fire to unitD's type j materiel ($1 \le j \le \text{nmat(sideD)}$), the basic quantity of enemy type j materiel it destroys in the frame is katk(i,j,sideA). But a weapon normally does not allocate all its fire to a single type of enemy materiel. This is not just a matter of doctrine. In reality, there might be several different types of materiel at which a weapon would fire; what it actually fired at would depend upon what targets it detected, and that would depend upon the composition and deployment of the enemy force. Two variables are used to adjust katk(i,j,sideA) for the allocation of fire-- *stdtgt*(*,sideD) and *aggatk*(i,*,sideA). The game design datum *stdtgt*(j,sideD) is, for $1 \leq j \leq$ nmat(sideD), the quantity of type j materiel in a standard side sideD combat force. The design datum *aggatk*(i,j,sideA) is the fraction of fire of a type i weapon from side sideA that is allocated to enemy type j materiel if the enemy materiel belongs to a standard enemy force. Let n = nmat(sideD). The fraction of fire of unitA's type i ground-to-ground weapons that is allocated to unitD's type j materiel is

$$\text{alpha}(i,j) = \frac{\mathit{aggatk}(i,j,\text{sideA}) * (\text{frinv}(j,\text{unitD}) * [\text{resources}](\text{unitD},j))}{\mathit{stdtgt}(j,\text{sideD}) * \text{DEN}},$$

where

$$\text{DEN} = \sum_{k=1}^{n} \mathit{aggatk}(i,k,\text{sideA}) * (\text{rsdef}(k) \; / \; \mathit{stdtgt}(k,\text{sideD})).$$

The divisor DEN is just a normalizer, to ensure that the fractions of fire allocated to the various types of enemy materiel sum to 1.[1] Thus, if type j materiel is overrepresented compared with the standard force, more fire is allocated to it; if no type j materiel is present, no fire is allocated to it.

The basic quantity of unitD's type j materiel that a single unitA type i weapon destroys in the frame is

$$\text{katk}(i,j,\text{sideA}) * \text{alpha}(i,j).$$

This quantity must be adjusted for the specific conditions of the engagement. Each adjustment affects either the lethality potential of unitA's type i weapons or the vulnerability of unitD's type j materiel to all enemy fire. In the former case, the adjustment takes the form of a factor applied to the row katk(i,*,sideA); in the latter, it takes the form of a factor applied to the column katk(*,j,sideA). The adjusted quantity

---

[1]Because of this normalization, it is not necessary that

$$\sum_{j} \mathit{aggatk}(i,j,\text{sideA}) = 1.$$

of unitD's type j materiel that a unitA type i weapon destroys in the frame is

```
K(i,j,unitA,unitD) = katk(i,j,sideA) * alpha(i,j)
                                     * PA * PD
                                     * EA * ED
                                     * B
                                     * PREP
```

The sequel defines the factors.

Let postA be unitA's posture if unitA is in an attack posture; let postA = 40 if not. Let

PA = [fckar](i,sideA,postA),

which equals *fckar*(i,sideA,*poff*(postA)) by definition. The factor PA adjusts the lethality of unitA's type i weapons according to unitA's attack posture. Let postD be unitD's posture. Let

PD = [fckac](j,sideD,postD),

which equals *fckac*(j,sideD,*poff*(postD)) by definition. The factor PD adjusts the vulnerability of unitD's type j materiel according to unitD's defense posture.

Let e be the environment in the engagement location: e = [environment](loc). Let

EA = *fckare*(i,sideA,e).

The factor EA adjusts the lethality of unitA's type i weapons according to the environment in which the combat occurs. Let

ED = *fckace*(j,sideD,e).

The factor ED adjusts the vulnerability of unitD's type j materiel. The combat is tacitly assumed to occur in the engagement location; hence, the environment in unitA's location is irrelevant. This does not imply that the attacker benefits or suffers from terrain equally as the defender. The variables *fckare* and *fckace* provide factors that are applied only to *katk*(*,*,sideA), the attacker's kill matrix; other variables, namely *fckdre* and *fckdce*, provide factors that are applied to *kdef*(*,*,sideD), the defender's kill matrix.

Let bt be the type of barrier between unitA's location and unitD's location: if cell locA is unitA's location, bt = [bartype](locA,loc). If bt = 0, let B = 1. (If there is no barrier, no adjustment is needed.) If bt > 0 and

$$feba \le febab(bt) \; / \; depth,$$

let B = *barier*(i,sideA,bt); otherwise, let B = 1. Thus, even if a barrier exists, its effects cease when the attackers have progressed sufficiently.

The area of the "area of influence" of unit def(k) ($1 \le k \le ndef$) is zrarea(def(k)).[1] The total area of the defenders' combined area of influence is computed as

$$defarea = \sum_{k=1}^{ndef} zrarea(def(k)).$$

The engagement variable front indicates the length of the defenders' line of contact with the attackers. Like FEBA, it is an abstraction, a way of measuring how far the defenders are stretched. If one or more attackers are located in cell loc, then front = $+\infty$. If not, the value of front depends on the number of directions from which the attack is coming. If the attackers are all located in the same cell, front equals the length of any side of a square equal in area to cell loc (a hexagon). If the attackers are located in k different cells, where $k > 1$, then front equals k times the length of any side of cell loc. The depth of the defense, defdepth, is given by

$$defdepth = \min \{defarea/front, \; depth\}.$$

The defenders' prepared positions, if any, are assumed to extend only to the depth defdepth. The factor PREP has two purposes: to reduce the vulnerability of a defender holding prepared positions, and to increase the vulnerability of a defender whose defense is hasty or disorganized. If unitD is not in a hold posture, let PREP = 1. Alternatively, assume that it is. The virtual length of time it has had to prepare its defense is t - tentry(unitD), which may be negative. (See Section 3.3.6.) Let

$$pf = prep \; (j, \text{side D}, t - tentry(unitD)).$$

(The function prep is explicated in Section 5.4.) If pf < 1, unitD's preparation time is sufficient to reduce the vulnerability of its type j materiel provided it still holds prepared positions. Hence, let

---

[1]The function zrarea is explicated in Section 5.4.

$$\text{PREP} = \begin{cases} \text{pf if pf} \le 1 \text{ and feba} < \text{defdepth}/\textit{depth}, \\ 1 \text{ if pf} \le 1 \text{ and feba} \ge \text{defdepth}/\textit{depth}. \end{cases}$$

On the other hand, pf > 1 indicates a hasty, disorganized defense, a condition unlikely to improve just because the attackers progress. Hence, let

$$\text{PREP} = \text{pf if pf} > 1.$$

That completes derivation of K(i,j,unitA,unitD), the "potential" quantity of unitD's type j materiel ($1 \le j \le$ nmat(sideD)) destroyed in the frame by a unitA type i ground-to-ground weapon ($1 \le i \le$ *nggwep*(sideA)). Similarly, the potential quantity of unitA's type j materiel ($1 \le j \le$ nmat(sideA)) destroyed in the frame by a unitD type i weapon ($1 \le i \le$ *nggwep*(sideD)) is

$$\begin{aligned} K(i,j,\text{unitD},\text{unitA}) = \text{kdef}(i,j,\text{sideD}) &* \text{alpha}'(i,j) \\ &* \text{PD}' * \text{PA}' \\ &* \text{ED}' * \text{EA}'. \end{aligned}$$

The factors' definitions are analogous to those given above.

Let m = nmat(sideA). The allocation factor

$$\text{alpha}(i,j) = \frac{\mathit{aggdef}(i,j,\text{sideD}) * (\text{frinv}(j,\text{unitA}) * [\text{resources}](\text{unitA},j))}{\mathit{stdtgt}(j,\text{sideA}) * \text{DEN}},$$

where

$$\text{DEN} = \sum_{k=1}^{m} \mathit{aggdef}(i,k,\text{sideD}) * (\text{rsatk}(k) \,/\, \mathit{stdtgt}(k,\text{sideA})).$$

Let postD be unitD's posture. Let

$$\text{PD}' = [\text{fckdr}](i,\text{sideD},\text{postD}),$$

which equals *fckdr*(i,sideD,*poff*(postD)) by definition. Let postA be unitA's posture if unitA is in an attack posture; let postA = 40 if not. Let

$$\text{PA}' = [\text{fckdc}](j,\text{sideA},\text{postA}),$$

which equals *fckdc*(j,sideA,*poff*(postA)) by definition. Let e = [environment](loc). Let

$$\text{ED}' = \mathit{fckdre}(i,\text{sideD},e),$$

$$\text{EA}' = \mathit{fckdce}(j,\text{sideA},e).$$

That completes deri ation of the two potential kill matrices for the attacker-defender pair unitA-unitD, K(*,*,unitA,unitD) and K(*,*,unitD,unitA). Potential kill matrices are derived for each attacker-defender pair.[1]

For any battle unit ibu and resource type irs, define

ERS(ibu,irs) = freff(ibu) * frinv(irs,ibu) * [resources](ibu,irs).

It is the effective quantity of the unit's type irs resources that can become actively involved in combat or combat support. The function freff is explicated in Section 5.4. Briefly, it adjusts a battle unit's effectiveness according to the density of friendly forces in its location. Let IGA = *nggwep*(sideA). Battle unit unitD's potential loss of type j materiel ($1 \leq j \leq$ nmat(sideD)) in the frame due to all enemy ground fire is, by definition,

$$\text{ploss(unitD,j)} = \sum_{k=1}^{\text{natk}} \sum_{i=1}^{\text{IGA}} \text{K(i,j,atk(k),unitD)} * \text{ERS(atk(k),i)}.$$

Let IGD = *nggwep*(sideD). Battle unit unitA's potential loss of type j materiel ($1 \leq j \leq$ nmat(sideA)) in the frame due to all enemy ground fire is, by definition,

$$\text{ploss(unitA,j)} = \sum_{k=1}^{\text{ndef}} \sum_{i=1}^{\text{IGD}} \text{K(i,j,def(k),unitA)} * \text{ERS(def(k),i)}.$$

Associated with potential losses of materiel are potential losses of personnel. (Notice that no fire is allocated directly to personnel.) Let nm = nmat(sideD). Unit unitD's potential loss of type p personnel ($1 \leq p \leq$ *npers*(sideD)) due to all enemy ground fire is, by definition,

$$\text{ploss(unitD,nm+p)} = \sum_{j=1}^{\text{nm}} \sum_{k=1}^{\text{natk}} \sum_{i=1}^{\text{IGA}} \Big(\text{K(i,j,atk(k),unitD)} * \text{ERS(atk(k),i)}\Big) * \textit{dpersr}\text{(p,i,j)}$$

---

[1]To conserve storage, the IDAHEX computer program uses none of these matrices. Of course, it gets the same results.

if sideD = 1 and

ploss(unitD,nm+p) =

$$\sum_{j=1}^{nm} \sum_{k=1}^{natk} \sum_{i=1}^{IGA} \Big(K(i,j,atk(k),unitD) * ERS(atk(k),i) \Big) * dpersb(p,i,j)$$

if sideD = 2. Let nm = nmat(sideA). Unit unitA's potential loss of type p personnel ($1 \le p \le npers(sideA)$) due to all enemy ground fire is

ploss(unitA,nm+p) =

$$\sum_{j=1}^{nm} \sum_{k=1}^{ndef} \sum_{i=1}^{IGD} \Big(K(i,j,def(k),unitA) * ERS(def(k),i) \Big) * dpersr(p,i,j)$$

if sideA = 1 and

ploss(unitA,nm+p) =

$$\sum_{j=1}^{nm} \sum_{k=1}^{ndef} \sum_{i=1}^{IGD} \Big(K(i,j,def(k),unitA) * ERS(def(k),i) \Big) * dpersb(p,i,j)$$

if sideA = 2.

A unit's potential losses might exceed what it has. To determine actual losses, a sequence of adjustments are made. The first step is determining the values for the resources in the engagement. The values are returned by the subprogram app, whose arguments include a single kill matrix for all the attackers and a single kill matrix for all the defenders. This subsection concludes by explaining the derivation of these average kill matrices. The next subsection explains app.

Let $1 \le i \le IGA$ and $1 \le j \le IGD$. Let $1 \le k \le natk$. The total potential destruction of enemy type j weapons attributed to all the type i weapons of attacker k equals

$$\sum_{\ell=1}^{ndef} K(i,j,atk(k),def(\ell)) * ERS(atk(k),i).$$

The formula commits no double-counting because the array K takes into account the allocation of type i weapons' fire to the various types of materiel in the various enemy units. The total potential destruction of enemy type j weapons attributed to all the attackers' type i weapons equals

$$\sum_{k=1}^{natk} \sum_{\ell=1}^{ndef} K(i,j,atk(k),def(\ell)) * ERS(atk(k),i).$$

Therefore, the average potential destruction of enemy type j weapons attributed to a type i weapon that is effectively, actively involved in combat is

$$A(i,j) = \frac{\displaystyle\sum_{k=1}^{natk} \sum_{\ell=1}^{ndef} K(i,j,atk(k),def(\ell)) * ERS(atk(k),i)}{\displaystyle\sum_{k=1}^{natk} ERS(atk(k),i)} .$$

The matrix A is an average kill matrix for the attackers as a whole. The defenders' average kill matrix, D, is defined analogously: for $1 \le i \le IGD$ and $1 \le j \le IGA$,

$$D(i,j) = \frac{\displaystyle\sum_{k=1}^{ndef} \sum_{\ell=1}^{natk} K(i,j,def(k),atk(\ell)) * ERS(def(k),i)}{\displaystyle\sum_{k=1}^{ndef} ERS(def(k),i)} .$$

The matrices A and D are passed to the subprogram app for use in the antipotential potential method. In that context, a theoretically rigorous approach would create A and D not by averaging (as above) but by using artificial weapon types. Unless

$$K(i,j,atk(k'),def(\ell)) = K(i,j,atk(k''),def(\ell))$$

and

$$K(j,i,def(\ell),atk(k')) = K(j,i,def(\ell),atk(k''))$$

for every $1 \le j \le IGD$ and $1 \le \ell \le ndef$, it would re-classify type i weapons belonging to attacker k′ and type i weapons belonging to attacker k″ as two different types of weapons. And unless

$$K(i,j,def(k'),atk(\ell)) = K(i,j,def(k''),atk(\ell))$$

and

$$K(j,i,atk(\ell),def(k')) = K(j,i,atk(\ell),def(k''))$$

for every $1 \leq j \leq$ IGA and $1 \leq \ell \leq$ natk, it would re-classify type i weapons belonging to defender k′ and type i weapons belonging to defender k′′ as two different types of weapons. Corresponding to an increase in the number of different types of weapons the attackers and defenders had would be an increase in the number of rows and columns of A and D. The matrices might grow so large that they required too much main storage and led to excessive execution times for app.

5.1.2 Determining Weapons' Values

The antipotential potential method finds consistent values (antipotential potentials) for weapons based on the rates at which they destroy enemy weapons. It was discovered independently by Spudich [6] (also see [7]), by Dare and James [3], and by Thrall and Howes [5]. Their work was synthesized by Anderson [2]. The IDAHEX subprogram app determines the value of each type of weapon in a given engagement. The present version of app computes these values from the kill matrices A and D, derived in Section 5.1.1, by Holter's version of the anti-potential potential method [4].

Recall that A(i,j) is the (average) rate at which a type i ground-to-ground weapon belonging to the attackers kills the defenders' type j ground-to-ground weapons, and D(i,j) is the (average) rate at which a type i ground-to-ground weapon belonging to the defenders kills the attackers' type j ground-to-ground weapons. Let

$$m = nggwep(\text{sideA}),\ n = nggwep(\text{sideD}).$$

The matrix A is $m \times n$, and D is $n \times m$. Let wa be the m-vector whose i-th component is the amount of type i weapons held by the attackers, and let wd be the n-vector whose j-th component is the amount of type j weapons held by the defenders. Let va be an m-vector and vd an n-vector. The component va(i) $(1 \leq i \leq m)$ is the value of a type i weapon belonging to the attackers, and vd(j) $(1 \leq j \leq n)$ is the value of a type j weapon belonging to the defenders; the values are derived below.

Some notation is needed. Suppose v and w are real s-vectors, and M is a real $r \times s$ matrix. Then

$$\langle v, w \rangle = \sum_{i=1}^{s} v(i) * w(i),$$

and M * v is the r-vector whose i-th component equals

$$\sum_{j=1}^{s} M(i,j) * v(j).$$

(Unless noted otherwise, all vectors are column vectors.) The transpose of M is denoted "M^t": $M^t(i,j) = M(j,i)$ for every $1 \le i \le r$ and $1 \le j \le s$.

The antipotential potential method defines va and vd so that, for some scalar alpha,

$$\text{(1)} \quad \text{alpha} * \text{va}(i) = \sum_{j=1}^{n} A(i,j) * \text{vd}(j) \text{ for every } 1 \le i \le m$$

and, for some scalar delta,

$$\text{(2)} \quad \text{delta} * \text{vd}(i) = \sum_{j=1}^{m} D(i,j) * \text{va}(j) \text{ for every } 1 \le i \le n.$$

Thus, each weapon's value is proportional to the rate at which it destroys enemy value. By equation (2),

$$\text{vd}(j) = (1/\text{delta}) * \sum_{k=1}^{m} D(j,k) * \text{va}(k).$$

Substitute for vd in equation (1), to conclude

$$\text{(3)} \quad (\text{alpha} * \text{delta}) * \text{va}(i)$$

$$= \sum_{k=1}^{m} \sum_{j=1}^{n} A(i,j) * D(j,k) * \text{va}(k)$$

$$= \sum_{k=1}^{m} AD(i,k) * \text{va}(k),$$

where AD is the matrix product of A and D. Let

$$\text{lambda} = \text{alpha} * \text{delta}.$$

Equation (3) says that va is an eigenvector of the matrix AD and lambda is an eigenvalue. According to the Frobenius Theorem, if AD is nonnegative and irreducible, then equation (3) has a solution in which lambda > 0 and va $\ge$ 0, and such a solution is unique up to multiplication of va by a positive scalar. Of course, AD is nonnegative. The matrix AD is "irreducible" if and only if it is not "reducible". By definition, AD is reducible if and only if re-ordering its rows and columns can put it in the form

$$\left[\begin{array}{c|c} M1 & 0 \\ \hline M3 & M2 \end{array}\right],$$

where M1 and M2 are square matrices and all the elements in the upper right-hand block are zero. Permuting the rows and columns of AD is equivalent to permuting the rows of A and the columns of D before calculating the product matrix. It follows that the non-negative matrix AD is reducible if and only if there are subsets A1 and A2 of the set $\{i: 1 \le i \le m\}$ such that: the number of elements in A2 exceeds 0 and equals m minus the number of elements in A1; and if $A(i,j) > 0$ for some $i \varepsilon A1$ and $1 \le j \le n$, then $D(j,k) = 0$ for every $k \varepsilon A2$. The condition holds if, for example, the attackers' weapons of a certain type are invulnerable to the defenders' fire.[1] Thrall argues that the weapon values obtained by the antipotential potential method are meaningful even if AD is reducible [5].[2]

Several ways of scaling va and resolving lambda into the factors alpha and delta have been proposed. Each of the following sets of assumptions uniquely determines va (determines how it should be scaled) and alpha and delta:

$$\text{(i)} \sum_{i=1}^{m} va(i) = 1, \sum_{i=1}^{n} vd(i) = 1 \quad \text{(Dare and James)}$$

$$\text{(ii) delta} = \sum_{i=1}^{m} va(i), \text{ alpha} = \sum_{i=1}^{n} vd(i)$$

(Thrall and Howes)

$$\text{(iii) delta} = \langle va,wa \rangle, \text{ alpha} = \langle vd,wd \rangle$$

(Spudich in TATAWS III)

---

[1]The matrix AD is reducible if $D(*,j) = 0$ for some j, which is necessarily true (because of the allocation of fire) if the attackers have no type j weapons. IDAHEX circumvents this problem by working, in effect, with an irreducible submatrix of AD.

[2]His argument posits that the antipotential potential method finds the weapon values by an iterative procedure starting with all-positive values. Such is the app procedure.

(iv) alpha = delta, va(kw) = 1 (Holter).

In (iv) kw is an integer in the interval [1,m]. The requirement va(kw) = 1 merely fixes the scaling of va; choice of kw does not affect the relative proportions of the elements of va and vd.[1] The present version of app implements (iv).

For arguments in favor of scaling assumption (iv) and against the three alternatives, see [4]. The primary consideration in selecting a scaling assumption is the reasonableness of the resulting force ratio:

$$FR = \frac{\langle va, wa \rangle}{\langle vd, wd \rangle} .$$

It should indicate which side is dominant. The attackers are said to dominate if the force ratio rises as combat continues. That happens if and only if the defenders' rate of value loss is bigger in proportion to their total value than the attackers'--i.e., the quantity

$$FR2 = \left(\frac{\langle vd, A^t * wa \rangle}{\langle vd, wd \rangle}\right) \Big/ \left(\frac{\langle va, D^t * wd \rangle}{\langle va, wa \rangle}\right)$$

exceeds 1. But

$$FR2 = \frac{\langle A*vd, wa \rangle}{\langle D*va, wd \rangle} \cdot \frac{\langle va, wa \rangle}{\langle vd, wd \rangle}$$

$$= \frac{alpha * (\langle va, wa \rangle)^2}{delta * (\langle vd, wd \rangle)^2} .$$

The first of the two preceding equalities reveals that the value of FR2 is independent of how va and vd are scaled. Under scaling assumption (iv), the force ratio, FR, equals the square root of FR2 (and therefore exceeds 1 if and only if FR2 exceeds 1). Under assumptions (i) and (ii), it is possible that FR > 1 while FR2 < 1,

---

[1] Some antipotential potentials may be 0. Of course, if va(kw) = 0, no rescaling can make va(kw) = 1. IDAHEX's subprogram app chooses kw to avoid this contradiction if possible. The contradiction is avoidable unless the only nonnegative solution of equations (1) and (2) is alpha = delta = 0, va = 0, vd = 0.

and *vice versa*. Under assumption (iii), FR > 1 if and only if FR2 > 1, but regardless of the force ratio the attackers lose value at the same rate as the defenders:

$$\begin{aligned}\langle va, D^t * wd\rangle &= \langle D*va, wd\rangle \\ &= delta * \langle vd, wd\rangle \\ &= \langle va, wa\rangle * alpha \\ &= \langle A*vd, wa\rangle \\ &= \langle vd, A^t * wa\rangle.\end{aligned}$$

Hence, assumption (iv) appears to be the most suitable.

The subprogram app actually determines the value of every resource, not just ground-to-ground weapons. Let mm = nrs(sideA) and nn = nrs(sideD). Let

$$vala(i) = \begin{cases} va(i); & 1 \le i \le m \\ 0 \quad ; & m < i \le mm \end{cases}$$

$$vald(i) = \begin{cases} vd(i); & 1 \le i \le n \\ 0 \quad ; & n < i \le nn \end{cases}$$

Since the resources other than ground-to-ground weapons cannot destroy enemy resources, giving them zero value is completely consistent with the antipotential potential method. Indeed, one might expand A and D to include all resource types, so A would have mm rows and nn columns and D would have nn rows and mm columns. Of course, A(i,j) would be 0 unless $i \le m$, and D(i,j) would be 0 unless $i \le n$. The vectors vala and vald defined above would satisfy equations (1) and (2) using the expanded A and D:

$$alpha * vala(i) = \sum_{j=1}^{n} A(i,j) * vald(j) + \sum_{j=n+1}^{nn} A(i,j) * vald(j)$$

for every $1 \le i \le mm$, and

$$\text{delta} * \text{vald}(i) = \sum_{j=1}^{m} D(i,j) * \text{vald}(j)$$

$$+ \sum_{j=m+1}^{mm} D(i,j) * \text{vald}(j)$$

for every $1 \le i \le nn$.

5.1.3 Finding Actual Losses

Section 5.1.1 derives the potential losses of materiel suffered by each battle unit in the given engagement. Section 5.1.2 derives the values of the resources in the engagement. Those subsections' notation remains in force. Recall that ERS(ibu,i) is the effective quantity of type i resources belonging to battle unit ibu that can become actively involved in combat or combat support. Let

$$\text{ersatk}(i) = \sum_{k=1}^{natk} \text{ERS}(\text{atk}(k),i)$$

for every $1 \le i \le mm$ (mm = nrs(sideA)), and

$$\text{ersdef}(i) = \sum_{k=1}^{ndef} \text{ERS}(\text{def}(k),i)$$

for every $1 \le i \le nn$ (nn = nrs(sideD)). The attackers' total value, fgrd, is defined by

$$\text{fgrd} = \sum_{i=1}^{mm} \text{ersatk}(i) * \text{vala}(i).$$

The defenders' total value, ggrd, is defined by

$$\text{ggrd} = \sum_{i=1}^{nn} \text{ersdef}(i) * \text{vald}(i).$$

The engagement's ground force ratio is

$$\text{FRGRD} = \text{fgrd} / \text{ggrd}.$$

The calculation of the family of kill matrices K(*,*,atk(k),def(ℓ)); the average kill matrices, A and D; and the potential losses, ploss; considered several influences, listed in Table 5.1. But the values assigned to the relevant variables by the game design data may not adequately represent all these influences, necessitating adjustments to ploss. In addition, ploss must be scaled according to the intensity of combat. Finally, no unit should be assessed losses in excess of what it has.

The first step in the adjustment process is determining a representative posture for the engagement attackers and one for the defenders. The value of attacker k is, by definition,

$$\sum_{i=1}^{mm} \text{ERS(atk(k),i)} * \text{vala(i)}.$$

Let postA be that posture such that the total value of the attackers in it is greatest; as before, an attacker's posture is taken to be 40 if not an attack posture. Let postD be that posture such that the total value of the defenders in it is greatest.

The next step compares the attackers' value loss implied by ploss with the value loss prescribed by the engagement's force ratio.[1] The attackers' potential loss of value is

$$\text{delval} = \sum_{k=1}^{natk} \sum_{i=1}^{mm} \text{ploss(atk(k),i)} * \text{vala(i)}.$$

Let temp = frdval(FRGRD,postA). (The function frdval is explicated in Section 5.4.) *If* temp < 0, *this step is skipped.* Thus, by appropriately defining the game design data used by frdval, the game designer can selectively avert this step. If temp $\geq$ 0, let

$$\text{scalar} = \text{temp} / (\text{delval}/\text{fgrd}),$$

and redefine ploss: for every $1 \leq k \leq$ natk and $1 \leq$ irs $\leq$ nrs(sideA)

$$\text{ploss(atk(k),irs)} \longleftarrow \text{scalar} * \text{ploss(atk(k),irs)}.$$

[1]This step is basically the same as one in IDAGAM's attrition procedure [1]. Indeed, the basic structure of IDAHEX's attrition procedure--a scaled Lanchester square process--originated with IDAGAM.

Table 5.1. INFLUENCES ON ATTRITION

Influence	How Represented
attack *vs.* defense	katk *vs.* kdef
posture	*fckar, fckac, fckdr, fckdc*
environment	*fckare, fckace, fckdre, fckdce*
barriers	*barier*
defensive preparation	prep

That is, the attackers' potential losses are scaled so that the attackers' total potential loss of value agrees with what is predicted from the force ratio.

Next, the same operation occurs for the defenders. Let

$$\text{delval} = \sum_{k=1}^{ndef} \sum_{i=1}^{nn} \text{ploss(def(k),i)} * \text{vald(i)}.$$

Let temp = frdval(FRGRD,postD). *If* temp < 0, *this step is skipped.* Otherwise, let

$$\text{scalar} = \text{temp} \;/\; (\text{delval/ggrd}),$$

and redefine ploss: for every $1 \le k \le$ ndef and $1 \le$ irs $\le$ nrs(sideD),

$$\text{ploss(def(k),irs)} \leftarrow \text{scalar} * \text{ploss(def(k),irs)}.$$

The final step is scaling ploss according to the intensity of combat, which depends upon the tactical overlap of the attacking force and the defending force. The tactical overlap is defined as the depth of the attackers' penetration of the defenders' cell (feba * *depth*) plus the effective range of the attackers' fire, which depends upon the combat environment. To be precise, the tactical overlap is defined as

$$\text{TO} = \min \{\text{feba} * \textit{depth} + \textit{td}([\text{environment}](\text{loc})),\ \text{defdepth}\}.$$

(Recall that cell loc is the engagement location, and defdepth, defined in Section 5.1.1, is the depth of the defense.) The intensity of combat is indicated by

$$\text{TI} = \text{TO} \;/\; \text{defdepth},$$

a number between 0 and 1. If the attackers and defenders are colocated, then feba = 1, and TI = 1. The potential losses are scaled by TI:

$$\text{ploss}(\text{atk}(k),\text{irs}) \longleftarrow \text{TI} * \text{ploss}(\text{atk}(k),\text{irs}),$$

for every $1 \leq k \leq \text{natk}$ and $1 \leq \text{irs} \leq \text{nrs(sideA)}$, and

$$\text{ploss}(\text{def}(k),\text{irs}) \longleftarrow \text{TI} * \text{ploss}(\text{def}(k),\text{irs})$$

for every $1 \leq k \leq \text{ndef}$ and $1 \leq \text{irs} \leq \text{nrs(sideD)}$.

The losses can now be assessed. Usually, a unit can only lose resources that are actively involved in combat. If FRGRD $\geq$ .0001, then for every $1 \leq k \leq \text{natk}$, [resources](atk(k),irs) is reduced by the quantity

$$\min \{\text{ploss}(\text{atk}(k),\text{irs}),\ \text{frinv}(\text{irs},\text{atk}(k)) * [\text{resources}](\text{atk}(k),\text{irs})\}$$

for every $1 \leq \text{irs} \leq \text{nrs(sideA)}$. But if FRGRD < .0001, the attackers lose everything: for every $1 \leq k \leq \text{natk}$

$$[\text{resources}](\text{atk}(k),\text{irs}) \longleftarrow 0$$

for every $1 \leq \text{irs} \leq \text{nrs(sideA)}$. That eliminates the possibility of dummy attacks, in which the attackers have no ground-to-ground weapons available for combat and suffer no losses. If FRGRD $\leq$ 10,000, then for every $1 \leq k \leq \text{ndef}$, [resources](def(k),irs) is reduced by the quantity

$$\min \{\text{ploss}(\text{def}(k),\text{irs}),\ \text{frinv}(\text{irs},\text{def}(k)) * [\text{resources}](\text{def}(k),\text{irs})\}$$

for every $1 \leq \text{irs} \leq \text{nrs(sideD)}$. If FRGRD > 10,000, then for every $1 \leq k \leq \text{ndef}$

$$[\text{resources}](\text{def}(k),\text{irs}) \longleftarrow 0$$

for every $1 \leq \text{irs} \leq \text{nrs(sideD)}$.

The preceding assessment procedure may err in the case of personnel, assuming that *dpersr* and *dpersb* give actual personnel losses associated with actual materiel losses, rather than potential materiel losses. Inconsistency occurs when and only when the potential loss of some type of materiel (given by ploss) exceeds the actual (assessed) loss (which cannot happen if the initial quantity is 0, for then the potential loss is 0). The inconsistency is one facet of a larger phenomenon. If the potential loss of some type of materiel exceeds the actual loss, the force to which it belongs loses a smaller fraction of its value than it should, assuming frdval determines that fraction. These inconsistencies are probably small in magnitude and are always fleeting: if the potential loss of some type of materiel exceeds the actual loss, then, barring other changes, in the next frame none of it will be available for the engagement and the potential loss of it will be 0.

5.2 FEBA MOVEMENT

Recall that each engagement has its own FEBA, measured by the variable feba, whose primary purpose is to determine when the attackers are allowed to occupy the engagement location. This subsection explains how any given engagement's feba is updated at the end of a frame to reflect the combat during the frame. The notation of Section 5.1 remains in force.

The change in feba from the start of the frame to the end of the frame depends on the attackers' posture, the defenders' posture, and a force ratio that includes the contribution of close air support (CAS). Air support is assessed at the start of every cycle, as Section 6 explains. (A cycle consists of one or more frames.) The losses of ground-to-ground weapons inflicted by CAS are recorded for use by the combat procedure. For every $1 \leq j \leq$ *nggwep*(sideD), let CASATK(j) be the amount of the defenders' type j weapons destroyed by air strikes made (by side sideA) in close support of the attackers; of course, if the attackers received no CAS in the cycle, CASATK(j) = 0. For every $1 \leq j \leq$ *nggwep*(sideA), let CASDEF(j) be the amount of the attackers' type i weapons destroyed by air strikes made (by side sideD) in close support of the defenders. These losses were determined at the start of the current cycle, and are assumed to be spread uniformly over the cycle. Therefore, to find CAS's effect on the engagement in the frame now ending, CASATK and CASDEF must be divided by nframe, defined as the number of frames in a cycle.

To find a force ratio that reflects both the ground forces and the air forces in the engagement, it is necessary to assign a value to CAS's contribution in a way consistent with the way the ground values are determined. The antipotential potential method facilitates this. Recall that the attackers' ground value is

$$\text{fgrd} = \sum_{i=1}^{m} \text{ersatk(i)} * \text{vala(i)},$$

where m = *nggwep*(sideA) (vala(i) = 0 if i > m). For every $1 \leq i \leq m$

$$\text{vala}(i) = (1/\text{alpha}) * \sum_{j=1}^{n} A(i,j) * \text{vald}(j),$$

where n = *nggwep*(sideD). Therefore,

$$\text{fgrd} = (1/\text{alpha}) * \sum_{j=1}^{n} \left(\sum_{i=1}^{m} A(i,j) * \text{ersatk}(i) \right) * \text{vald}(j).$$

The sum in parentheses is side sideD's total potential loss of type j materiel in the frame. The air value of side sideA in the engagement, fair, is computed the same way

$$\text{fair} = (1/\text{alpha}) * \sum_{j=1}^{n} (\text{CASATK}(j)/\ \text{nframe}) * \text{vald}(j).$$

Analogously, the air value of side sideD in the engagement is defined as

$$\text{gair} = (1/\text{delta}) * \sum_{j=1}^{m} (\text{CASDEF}(j)\ /\ \text{nframe}) * \text{vala}(j).$$

The combined ground-air force ratio is

$$\text{FRGA} = \frac{\text{fgrd} + \text{fair}}{\text{ggrd} + \text{gair}}\ .$$

Let postA be the attackers' representative posture and postD the defenders' representative posture; postA and postD are defined in Section 5.1.3. The function value

vfeba (FRGA, postA, postD, sideA)

is the velocity of an engagement's FEBA when the combined ground-air force ratio is FRGA, the attackers' representative posture is postA, the defenders' representative posture is postD, and the attackers belong to side sideA. (The function vfeba is explicated in Section 5.4.) Let

temp = vfeba(FRGA,postA,postD,sideA) * *tframe*.

This number may be negative. If feba0 is the value of feba at the start of the frame, then at the end of the frame

feba = min {max {feba0 + temp/*depth*, 0}, 1}.

5.3 ELIMINATION AND RETREAT

After the losses in one frame of an engagement are assessed, each of its attackers and defenders is examined to see if it is so weak it should be eliminated. The evaluation is based on the resources' "standard values": for s = 1 or s = 2 and $1 \le i \le nrs(s)$, rsvald(i,s) is the "standard value of a side s type i resource on defense". It is found by putting the resource on defense in a nominal engagement. Let s1 = 1 and s2 = 2. For every $1 \le i \le$ *nggwep*(s1) and $1 \le j \le$ *nggwep*(s2), let

$$DSTD(i,j) = kdef(i,j,s1) * \mathit{aggdef}(i,j,s1) \Big/ \sum_{k=1}^{nm} \mathit{aggdef}(i,k,s1) .$$

where nm = nmat(s2). For every $1 \le i \le$ *nggwep*(s2) and $1 \le j \le$ *nggwep*(s1), let

$$ASTD(i,j) = katk(i,j,s2) * \mathit{aggatk}(i,j,s2) \Big/ \sum_{k=1}^{nm} \mathit{aggatk}(i,k,s2) ,$$

where nm = nmat(s1). The subprogram app is called with the kill matrices ASTD and DSTD as arguments; it returns the values of the side s1 resources (on defense), which define rsvald(*,s1). The values of the side s2 resources (on attack) define rsvala(*,s2)--"the standard values of side s2 resources on attack"--which are used to resolve mutual attacks (Section 3.3.4). To compute rsvald(*,s2)--the Blue resources' standard values on defense--the process is repeated with s1 = 2 and s2 = 1.

Let unit ibu be an attacker or defender in the engagement. Let s = 1 if it belongs to Red and s = 2 if it belongs to Blue. Let n = nrs(s). Let

$$sv = \sum_{i=1}^{n} \mathit{toe}(\mathit{butype}(ibu),i) * rsvald(i,s),$$

$$cv = \sum_{i=1}^{n} [resources](ibu,i) * rsvald(i,s).$$

If

$$cv < \mathit{vanish}(\mathit{butype}(ibu)) * sv - 10^{-5},$$

then unit ibu is eliminated: it is assigned the mission whose only order declares -10 as the desired posture.

If, at the end of a frame, feba $\geq$ *febad* in a given engagement, the defenders are declared defeated, and the combat procedure calls the tactical subprogram haven to ascertain whether the defenders have a line of retreat. To be admissible as a direction of retreat, a cell must be active and adjacent to the engagement location, and it must satisfy the following conditions: (i) it contains none of the attackers in the engagement; (ii) if it contains an active unit belonging to the attackers' side, then it must also contain an active unit belonging to the defenders' side and be owned by the defenders' side. If the game design variable *haven.zoc* has the value .true., a direction of retreat can also be blocked by the presence of attackers in cells flanking it. To be precise, suppose cell i satisfies all the preceding conditions for admissibility as a direction of retreat. If *haven.zoc* = .true., in order to be an admissible direction of retreat, cell i must satisfy the additional condition: (iii) if cell j is adjacent to both cell i and the engagement location and it contains one of the attackers, then cell i must contain an active unit from the defenders' side and must be owned by the defenders' side.

If the defenders have no admissible direction of retreat, they are eliminated. If they have an admissible direction of retreat, haven selects the most desirable one. Each admissible direction of retreat is scored as follows:

(1) Initially, let its score be 0.

(2) If it is exactly two cells away from a cell containing one of the attackers, let its score be -1.

(3) If it is adjacent to a cell (other than the engagement location) containing one of the attackers, let its score be -2.

(4) If it is owned by the attackers' side, decrease its score by .5.

(5) If it is owned by the defenders' side and contains an active unit from their side, increase its score by 1.8.

(6) Let s = 1 if the defenders are Red and s = 2 if they are Blue. Let k = *pthome*(s). If the cell is the k-th rim cell of the engagement location increase its score by .01.

The "rim cells" of a given cell are the cells adjacent to it. They are ordered by number, from lowest to highest. For example, in Figure 3.3 (page 3-6), the first rim cell of cell 6 is cell 2, the second is cell 3, the third is cell 5, the fourth is cell 7, the fifth is cell 9, and the sixth is cell 10; the fourth rim cell of cell 1 is cell 2, the sixth is cell 5, and the other rim cells of cell 1 do not exist; the fifth rim cell of cell 14 is cell 17.

Let cell r be the admissible direction of retreat with the highest score; ties are broken by minimizing r. Each defender that is not already disengaging is forced to disengage immediately toward cell r: it is assigned the active order

desired objective = r,
desired posture = *pmapup*(*pmapup*(*pmapup*(*pmapup*(p)))),

where p is its current posture (a hold posture). Once all the defenders are disengaging, the attackers are allowed to occupy the engagement location, as Section 3.3.3 explains.

5.4 THE COMBAT FUNCTIONS

This subsection explains the functions frinv, zrarea, freff, prep, frdval, and vfeba, which the subprogram combat invokes. They are piecewise-affine (loosely speaking, piecewise-linear) functions mapping the real line into the real line. Such a function is specified by listing points in its domain--x(1), x(2),...,x(n)--and its value at each of these points--y(1), y(2),...,y(n). By requirement, $x(1) \le x(2) \le \ldots \le x(n)$. The function pafgen evaluates a piecewise-affine function. Let w be a real number. If $w \le x(1)$, then

$$\text{pafgen}(w,y,x) = y(1).$$

If $w \ge x(n)$, then

$$\text{pafgen}(w,y,x) = y(n).$$

Suppose $x(1) < w < x(n)$. Let

$$i1 = \max \{i: \ x(i) \le w, \ 1 \le i \le n\},$$
$$i2 = \min \{i: \ x(i) > w, \ 1 \le i \le n\}.$$

Then

$$\text{pafgen}(w,y,x) = y(i1) + \frac{w - x(i1)}{x(i2) - x(i1)} * (y(i2) - y(i1)).$$

(The IDAHEX function pafgen actually has an additional argument--n, the number of components of the vector x or y.)

A similar function, paf, is used to evaluate a piecewise-affine function whose domain is the nonnegative reals. Such a function is specified by listing its value at 0, which is denoted y0; listing points in its domain--x(1),...,x(n); and listing its values at these points--y(1),...,y(n). By requirement, $0 \leq x(1) \leq \ldots \leq x(n)$. Let w be a real number. Define the vector ylong, with n+1 components, as follows:

$$\text{ylong}(1) = \text{y0},$$

$$\text{ylong}(i+1) = y(i) \text{ for } 1 \leq i \leq n.$$

Define the vector xlong, with n+1 components, as follows:

$$\text{xlong}(1) = \text{x0},$$

$$\text{xlong}(i+1) = x(i) \text{ for } 1 \leq i \leq n.$$

Then

$$\text{paf}(w,\text{y0},y,x) = \text{pafgen}(w,\text{ylong},\text{xlong}).$$

(The IDAHEX function paf actually has an additional argument--n, the number of components of the vector x.)

5.4.1 Resource Availability for Combat - frinv

The function frinv, as called by the combat procedure, has two essential arguments: a unit number, ibu; and a resource type, irsarg. If the unit's resources of type irsarg are equipment (weapons or transport), frinv returns the fraction of them that are available for combat; equipment is available for combat if and only if its requirements for support and protection are met. If the type irsarg resources are support resources (supplies or personnel), frinv returns the fraction of them that are available and needed. The neutral term "fractional involvement" designates the number returned in either case. In the process of determining the fractional involvement of type irsarg resources, frinv determines the fractional involvement of every type of resources in unit ibu. Let s = 1 if unit ibu is Red and s = 2 if it is Blue. Let fi(irs) be the fractional involvement of type irs resources in unit ibu for every $1 \leq \text{irs} \leq \text{nrs}(s)$. The sequel explains how it is determined.

A unit loaded on other units cannot participate in combat: if unit ibu is a passenger in a stacked task force--i.e., if the task force's transport mode is positive and *trptcl*(*butype*(ibu)) equals it--then $\text{fi}(i) \equiv 0$.

Henceforth, assume unit ibu is not a passenger. Given that frinv has been called by the combat procedure, unit ibu must be engaged. Other units from its side may be participating in the same engagement; frinv assumes that all such units with the same location as unit ibu perform as an integral whole, sharing their support and using their weapons in concert. Let L be the set of every unit that is from the same side, participating in the same engagement, and located in the same cell as unit ibu. Delete from L every unit that is a passenger in a stacked task force. For $1 \leq \text{irs} \leq \text{nrs(s)}$, define amount(irs) as the total quantity of type irs resources held by the force L:

$$\text{amount(irs)} = \sum_{i \in L} \text{[resources](i,irs)}.$$

Let

$$\text{nsp} = \textit{nss}(\text{s}) + \textit{npers}(\text{s}),$$

the number of types of side s support resources. Suppose nsp = 0. Then fi is determined solely by considerations of equipment protection. The game design variable *pg* organizes equipment into protection groups, numbered 1, 2,...; *pg*(i,s) is the protection group to which type i equipment of side s belongs. At least one type of the side's ground-to-ground weapons should belong to protection group 1. Any equipment in protection group 1 can protect itself and equipment in higher protection groups. Equipment in a protection group higher than 1 cannot protect itself, but can protect equipment in protection groups higher than its own. The quantity of type i equipment that a unit-quantity of type j equipment can protect, provided *pg*(j) < *pg*(i), is *prot*(i,j,s) by definition. Equipment other than ground-to-ground weapons, although it may conceivably belong to protection group 1 and be able to protect itself, is assumed to be unable to protect other equipment--i.e., *prot*(i,j,s) is assumed to be 0 if j > *nggwep*(s). In the present case, where support is ignored, fi(i) = 1 for every i such that *pg*(i,s) = 1. The fractional involvement of equipment in higher protection groups, if any, is determined inductively. Suppose that for some

$$k < \max \{\textit{pg}(\text{i,s});\ 1 \leq \text{i} \leq \text{nequip(s)}\}.$$

fi(i) has been determined for every i in the set

$$\text{I} = \{\text{i}:\ \textit{pg}(\text{i,s}) \leq \text{k},\ \ 1 \leq \text{i} \leq \text{nequip(s)}\}.$$

For each j such that *pg*(j,s) = k + 1, let

$$\text{QP(j)} = \sum_{i \in I} \textit{prot}(\text{j,i,s}) * \text{fi(i)} * \text{amount(i)},$$

the quantity of type j equipment that can be protected. Set

$$fi(j) = \frac{\min \{QP(j),\ amount(j)\}}{amount(j)} .$$

That completes the induction step. If possible k is incremented by 1 and the step is repeated.

Typically, small arms belong to protection group 1, tanks to group 2, artillery to group 3, and ground-to-air weapons and transport to group 4. Notice that protecting one type of equipment does not reduce a weapon's ability to protect other types of equipment. One might think of the equipment in protection group 1 as being deployed near the front of the formation, the equipment in protection group 2 deployed behind it, and so on, with the equipment in each protection group acting as a screen for the equipment deployed behind it.

Henceforth, assume that nsp > 0. To determine fi(i) for every $1 \le i \le nrs(s)$, frinv implicitly allocates support to the various types of resources. The allocation is reasonable, but not optimal: it does not maximize the force L's value in combat. It should not; the allocation is partly prescriptive. It is designed to field a balanced combat force--one in line with *stdtgt*(*,s), with no unprotected equipment.

Let

$$neq = nequip(s).$$

For every $1 \le k \le nsp$ let

$$suppt(k) = amount(neq+k),$$

the total quantity of type k support held by the units in L. If personnel are played--i.e., if *npers*(s) > 0--the quantities of personnel available to support materiel must be reduced by overhead requirements:

$$suppt(k) \longleftarrow suppt(k) - \sum_{i \varepsilon L} ppoh(k, butype(i))$$

for every $1 \le k \le npers(s)$. (*ppoh*(k,j) is defined as the overhead of type k personnel in a type j battle unit--a quantity that is independent of the unit's actual size.)

Let

$$i0 = \begin{cases} 0 & \text{if } s = 1, \\ nrs(1) & \text{if } s = 2. \end{cases}$$

The game design datum *spdd*(k,i0+irs) is the demand of a unit-quantity of side s type irs resources ($1 \leq irs \leq nrs(s)$) for type k support ($1 \leq k \leq nsp$).[1] The IDAHEX computer program assumes that supplies' demand for supplies is 0 and personnel's demand for personnel is 0. The total demand of the force's resources of type irs for support of type k is computed as

$$dd(k) = amount(irs) * spdd(k,i0+irs).$$

Let ss(k) be the quantity of type k support allocated to the force's type irs resources. For every $1 \leq k \leq nsp$, let

$$sigma(k) = paf\ (ss(k)/dd(k),\ frinv.f0(k,i0+irs),\ frinv.f(k,i0+irs,*),\ frinv.x(s,*))$$

if $dd(k) > 0$, and let sigma(k) = 1 if not. The fractional involvement of the force's type irs resources, fi(irs), is given by

$$fi(irs) = \min\ \{sigma(k);\ 1 \leq k \leq nsp\}.$$

Thus, allocation of support to each type of resources determines their fractional involvement. To be sure that no more of any type of support is allocated than is available, the vector alloc is used to keep track of the allocation; alloc(k), for $1 \leq k \leq nsp$, is the total quantity of type k support allocated. Initially, alloc $\equiv$ 0.

First, personnel are allocated to supplies. For each $1 \leq kpp \leq npers(s)$, the total demand for type kpp personnel by the force's supplies is

$$Q = \sum_{kss=1}^{nss(s)} suppt(kss) * spdd(\ nss(s)+kpp,\ i0+neq+kss);$$

the demand of type kss supplies alone is

$$dd = suppt(kss) * spdd(\ nss(s)+kpp,\ i0+neq+kss)$$

($1 \leq kss \leq nss(s)$); the allocation of type kpp personnel to type kss supplies is chosen as

$$\min\ \{dd,\ dd * (suppt(kpp)\ /\ Q)\},$$

[1]Equipment's requirement for support normally should include the personnel needed to operate it in combat and, in addition, personnel needed to keep it operational (by maintenance and repair, for example). With respect to the latter, the game designer must avoid counting personnel requirements twice--once in resources' requirements (*spdd*) and once in overhead (*ppoh*).

and alloc(nss(s)+kpp) is increased by this quantity. As explained above, the allocation determines fi(kss). Only that fraction of type kss supplies are available for allocation; redefine suppt(kss) for every $1 \le \text{kss} \le nss(\text{s})$:

$$\text{suppt(kss)} \leftarrow \text{fi(kss)} * \text{suppt(kss)}.$$

Next, supplies are allocated to personnel, but fi(irs) is set to 1 for each nmat(s) < irs $\le$ nrs(s) whether or not the allocation satisfies personnel's demand. Record the allocation of supplies:

$$\text{alloc(kss)} \leftarrow \text{alloc(kss)} + \sum_{\text{kpp}=1}^{npers(\text{s})} \text{amount(nmat(s)+kpp)} * spdd\text{(kss,10+nmat(s)+kpp)}.$$

(If nss(s) = 0 or $npers$(s) = 0, both preceding steps are vacuous.)

Next, support is allocated to equipment. Let

$$\text{I} = \{\text{ieq}: stdtgt(\text{ieq,s}) > 0,\ 1 \le \text{ieq} \le \text{neq}\}.$$

If i $\le$ neq but i $\notin$ I, fi(i) is set to 0 and never changed. If i ε I and amount(i) = 0, fi(i) is set to 1 and never changed. Initially, fi(i) = 0 for every other i ε I. It is increased in small increments by increasing the support allocated to each type of equipment in the set I. Let rgain be a small positive number--.01, for example. At the start of any given iteration of the algorithm, let

$$\text{q(j)} = \text{fi(j)} * \text{amount(j)}$$

for every j. (fi may have been redefined in prior iterations.) The iteration consists of performing the following sequence of operations for each i ε I for which amount(i) > 0.

Step 1: If pg(i,s) = 1, let qp = $+\infty$ and go to Step 2. Let P be the set of every j ε I such that pg(j,s) < pg(i,s). Let

$$\text{qp} = \sum_{\text{j} \varepsilon \text{P}} prot(\text{i,j,s}) * \text{q(j)},$$

the quantity of type i equipment that can be protected by the equipment presently available for combat.

Step 2: Let

$$\text{qr} = \text{rgain} * stdtgt(\text{i,s}),$$

the amount by which q(i) would have to increase in order to increase

$$q(i) \,/\, \mathit{stdtgt}(i,s)$$

by the amount rgain. Let

$$qadd = \min \{qp, qr\}.$$

For every $1 \le ksp \le nsp$, let REQ(ksp) be the amount of additional type ksp support that must be allocated to type i equipment to increase q(i) by the amount qadd--i.e., to increase fi(i) by the amount

$$qadd \,/\, amount(i).$$

If

$$alloc(ksp) + REQ(ksp) \le suppt(ksp)$$

for every $1 \le ksp \le nsp$, then allocate the support and update fi and q(i):

$$alloc(ksp) \leftarrow alloc(ksp) + REQ(ksp) \text{ for every } 1 \le ksp \le nsp,$$

$$fi(i) \leftarrow fi(i) + qadd \,/\, amount(i),$$

$$q(i) \leftarrow fi(i) * amount(i).$$

If not, fi(i) cannot be increased.

Thus, the algorithm tends to field a balanced combat force--i.e., it strives to equalize

$$\frac{fi(i) * amount(i)}{\mathit{stdtgt}(i,s)}$$

over every $i \in I$ and never commits unprotected equipment to combat.

Support not actually needed by resources for combat--i.e., unallocated support--should not be actively involved in combat (and subject to enemy fire). The final step reduces the fractional involvement of support that is in surplus: for every $neq < irs \le nrs(s)$ such that $amount(irs) > 0$, fi(irs) is redefined as

$$\min \{fi(irs),\ alloc(irs-neq) \,/\, amount(irs)\}.$$

The preceding explains the derivation of frinv(irsarg,ibu) in the case where unit ibu is engaged; that case always applies when frinv is called by the combat procedure. The derivation actually involves finding the fractional involvement of resources in a set of units, L, that share their support and use their weapons in concert. Sometimes, for a player's information, it is useful to find the fractional involvement for a specified force L, rather than a force inferred by frinv from the argument ibu. The IDAHEX entry point frinv actually has two additional arguments: a vector, list; and an integer, nlist. If ibu $\leq$ 0, frinv constructs the set L from the vector list, whose first nlist elements must be the identification numbers of friendly battle units; frinv then proceeds as above to find the fractional involvement of resources in the force L.

5.4.2 Area of Area of Influence - zrarea

The function value zrarea(ibu) is the area of the area of influence of battle unit ibu. It is 0 if the unit is inactive. Assume unit ibu is active. Let s = 1 if it is Red and s = 2 if it is Blue. Its current value, measured in terms of the standard resource values, is

$$\text{cv} = \sum_{\text{irs}=1}^{\text{nrs(s)}} \text{rsvald(irs,s)} * [\text{resources}](\text{ibu,irs}).$$

Its value at *toe* strength would be

$$\text{sv} = \sum_{\text{irs}=1}^{\text{nrs(s)}} \text{rsvald(irs,s)} * \mathit{toe}(\mathit{butype}(\text{ibu}),\text{irs}).$$

The size of its area of influence is assumed to be proportional to the size at *toe* strength. The latter depends upon the unit's type and posture class. Let pc be the unit's posture class:

$$\text{zrarea(ibu)} = (\text{cv/sv}) * \mathit{aisize}(\mathit{butype}(\text{ibu}),\text{pc}).$$

5.4.3 Battle Unit Effectiveness - freff

A battle unit's effectiveness may depend upon the density of friendly forces in its location. If the density is too low, the friendly force is vulnerable to infiltration and turning maneuvers. If the density is too high, the friendly force is more vulnerable to area fire, and congestion of the trafficable areas reduces the maneuver battalions' tactical mobility. In many models the degradation of effectiveness due to high density is implemented indirectly by a rule limiting the number of units located in the same cell. Since units may vary greatly in size, especially late in the game, IDAHEX uses a more flexible method.

Suppose the location of unit ibu, an active battle unit, is cell i. Let F be the set of every active, friendly unit located in cell i, identified by number. The total area of their areas of invluence is

$$A = \sum_{j \varepsilon F} \text{zrarea}(j).$$

The friendly force density is A divided by the cell area; let d equal this quotient. Let s = 1 if the units in F are Red and s = 2 if they are Blue. The fractional effectiveness of unit ibu, or any unit in F, is

$$\text{freff(ibu)} = \text{paf}\ (d,\ \textit{freff.f0}(s),\ \textit{freff.f}(s,*),\ \textit{freff.x}(s,*)).$$

Normally, this is a number in the interval [0,1]. It can exceed 1 only if *freff.f0*(s) > 1 or *freff.f*(s,j) > 1 for some j.

5.4.4 Defensive Preparation - prep

The vulnerability of materiel varies with the time its battle unit has had to prepare a defense. Suppose s = 1 or s = 2, and suppose $1 \leq i \leq \text{nmat}(s)$. The function value prep(i,s,h) is the factor the combat procedure applies to type i materiel belonging to a side s unit whose defense preparation time equals h.

$$\text{prep}(i,s,h) = \text{pafgen}\ (h,\ \textit{prep.f}(i,s,*),\ \textit{prep.x}(s,*)).$$

Because of peculiarities in the way preparation time is calculated, h may be negative. The game designer should allow for this possibility by choosing

$$\textit{prep.x}(s,1) < 0.$$

5.4.5 Fraction of Value Lost - frdval

This function finds the fraction of value that a side in combat loses given the side's posture and the engagement's ground force ratio. Let post be the side's posture and FR the force ratio. Let k = *poff*(post). Let

$$\text{temp} = \text{paf}\ (\text{FR},\ \textit{frdval.f0atk}(k),\ \textit{frdval.fatk}(k,*),\ \textit{frdval.x})$$

if post $\geq$ 40 (the side is the attacker in the engagement), and

$$\text{temp} = \text{paf}\ (\text{FR},\ \mathit{frdval.f0def}(\text{k}),\ \mathit{frdval.fdef}(\text{k},*),\ \mathit{frdval.x})$$

if post < 40 (the side is the defender). The number temp gives the fraction of value lost in one unit of time, but the combat procedure needs to know the fraction lost in one frame. Therefore,

$$\text{frdval (FR, post)} = \begin{cases} 1 - (1 - \text{temp})\text{**tframe}; & \text{temp} \geq 0 \\ \text{temp}; & \text{temp} < 0. \end{cases}$$

The combat procedure, which calls frdval, interprets frdval(FR,post) < 0 as a signal that no prediction of the side's losses should be made from the force ratio and therefore that the side's losses should not be scaled according to it.

5.4.6 FEBA Velocity - vfeba

The function value vfeba (FR, pa, pd, sa) is the velocity of the FEBA (measured by *depth* * feba) in an engagement in which the force ratio is FR, the attackers are from side sa, the attackers are in posture pa, and the defenders are in posture pd. Let

$$\text{ka} = \begin{cases} \mathit{poff}(\text{pa}) & ; \quad \text{sa} = 1 \\ \mathit{poff}(\text{pa}) + \text{vfeba.npa}; & \quad \text{sa} = 2. \end{cases}$$

The offset vfeba.npa is defined by the entry point vfeba0. If the number of attack postures is large (i.e., if *npost*(4) is close to 10), it may be necessary to increase vfeba.npa and the dimensions of certain variables declared by vfeba0. In that event, IDAHEX will advise the game designer with a message in file 51 (which is described in Section 8). Let kd = *poff*(pd). Then

$$\text{vfeba (FR, pa, pd, sa)} = \text{paf (FR},\ \mathit{vfeba.f0}(\text{ka,kd}),\ \mathit{vfeba.f}(\text{ka,kd},*),\ \mathit{vfeba.fr}).$$

This number may be negative.

In defining *vfeba.f0* and *vfeba.f* the game designer should keep in mind that the attackers have already been charged with the time needed to go from their locations to the engagement location, and if they occupy the engagement location and then leave, they will be charged with the time needed to go from the engagement location to their new locations; the movement delay takes care of unopposed movement. The feba velocity is used to determine an *additional* delay caused by opposition. Consequently, if the force ratio is very high, the feba velocity should be very high; it should not be limited by the unopposed movement rate.

6. AIR SUPPORT

At the start of every cycle (including t = *tinit*), each player may enter air strikes. IDAHEX contains no air warfare model and therefore has no way of ascertaining what air assets a side can allocate against enemy ground forces. It assumes that any air strike a player enters is within his side's capability. In practice, the game designer adopts either of two solutions: he gives each player a list of the air assets available in each cycle for use against enemy ground forces, or he runs an air warfare model concurrently with IDAHEX. The first solution is suitable when the course of the air war is easy to predict--usually because one side clearly dominates.

Suppose the side s player (s = 1 or s = 2) is inputting an air strike. His first line of input tells IDAHEX the "target cell" and the "strike role". The target cell is the cell toward which the strike is directed. The strike role is either close air support (CAS) or air interdiction of battle units. If the strike role is interdiction, the player's next input line defines the four-component vector asprty, which is a list of the four positive posture classes in order of priority. The next input line sets ascomp; ascomp(i) is the number of type i aircraft participating in the strike ($1 \leq i \leq$ *nactyp*(s)).

Suppose the air strike role is CAS. If there is no engagement whose location is the target cell, the player is warned and no strike occurs. If such an engagement exists, let V be the set of every enemy unit in the engagement, identified by unit number. If the enemy units are the defenders in the engagement, and if at least one of them is in a hold posture, then delete from V every unit in a disengagement posture.

On the other hand, suppose the strike role is interdiction. Let k be the smallest integer such that asprty(k) equals the posture class of some active enemy unit located in the target cell. Thus, asprty(k) is the highest priority posture class that appears among enemy units in the target cell. Let pc = asprty(k). Define V as the set of every active enemy unit, identified by unit number, whose location is the target cell and posture class is pc. These units are the targets of

AD-A050 768 INSTITUTE FOR DEFENSE ANALYSES ARLINGTON VA PROGRAM --ETC F/G 15/7
IDAHEX: A MANEUVER-ORIENTED MODEL OF CONVENTIONAL LAND WARFARE.--ETC(U)
NOV 76 P OLSEN

UNCLASSIFIED P-1221-VOL-2 SBIE-AD-E500 016 NL

2 OF 2

AD
A050 768

END
DATE
FILMED
4-78
DDC

the strike. Behind this definition of V is an implicit assumption that the strike aircraft can only distinguish enemy units from each other by location (cell) and posture class.

Let

$$nw = nagwep(s).$$

For every $1 \le iw \le nw$, the amount of type iw air-to-ground weapons in the strike is

$$agwep(iw) = \sum_{i=1}^{n} agload(i,iw,s) * ascomp(i)$$

where $n = nactyp(s)$. Let $v = 3 - s$. (Side v is the enemy of side s.) For $1 \le j \le nmat(v)$, the amount of type j materiel in the target battle units is

$$grdrs(j) = \sum_{i \varepsilon V} [resources](i,j).$$

Let env be the environment type of the target cell: if the target cell is cell i,

$$env = [environment](i).$$

Choose an air-to-ground weapon type, iw; $i \le iw \le nw$. For every $1 \le j \le nmat(v)$, define

$$aag(j) = \begin{cases} aagatk(iw,j,s) & \text{if the strike role is CAS and side s is the engagement attacker,} \\ aagdef(iw,j,s) & \text{if the strike role is CAS and side s is the engagement defender,} \\ aagred(iw,j,pc) & \text{if the strike role is interdiction and } s = 1 \\ aagblu(iw,j,pc) & \text{if the strike role is interdiction and } s = 2 \end{cases}$$

(Recall that pc is the posture class of the target battle units, assuming the strike role is interdiction.) For $1 \le j \le nmat(j)$, the fraction of fire of type iw weapons allocated to the target units' type j materiel is computed as

$$\text{alpha}(j) = \frac{\text{aag}(j) * (\text{grdrs}(j) \,/\, \textit{stdtgt}(j,v))}{\sum_i \text{aag}(i) * (\text{grdrs}(i) \,/\, \textit{stdtgt}(i,v))} .$$

This method of allocating fire is analogous to the method used in ground combat.

Choose ibu ε V and $1 \leq j \leq$ nmat(v). The goal is to determine the potential destruction of type j *materiel in* unit ibu by the type iw weapons in the strike, denoted K(iw,j,ibu). In parallel with the ground combat attrition procedure, this quantity is found by taking a basic kill rate and applying factors that each adjust either the shooting weapon's lethality or the target materiel's vulnerability. The basic kill rate depends upon the game design datum *kag*(iw,j,s) and the allocation of fire. The adjustment factors depend upon the posture class of unit ibu--denoted by pc--and the target cell environment. By definition,

$$\begin{aligned} K(iw,j,ibu) = \textit{kag}(iw,j,s) &* \textit{fcagrp}(iw,s,pc) * \textit{fcagcp}(j,v,pc) \\ &* \textit{fcagre}(iw,s,env) * \textit{fcagce}(j,v,env) \\ &* Q, \end{aligned}$$

where Q is the amount of fire from type iw weapons allocated to type j materiel in unit ibu:

$$Q = \Big(\text{alpha}(j) * ([\text{resources}](ibu,j) \,/\, \text{grdrs}(j))\Big) * \text{agwep}(iw).$$

The total potential loss of type j materiel by all the target units is

$$\sum_{i \varepsilon V} \sum_{iw=1}^{nw} K(iw,j,i)$$

If the strike role is CAS and $j \leq$ *nggwep*(v), *this quantity is* recorded in the array casfx for later use by the combat procedure.

Choose ibu ε *V and* $1 \leq j \leq$ nmat(v). The actual loss of type j materiel by unit ibu is computed as follows. Initially, set iw = 1. Let

$$L = \min \{K(iw,j,ibu),\ [\text{resources}](ibu,j)\},$$

and reduce the unit's stocks of type j materiel by that quantity:

$$[\text{resources}](ibu,j) \longleftarrow [\text{resources}](ibu,j) - L.$$

This loss of type j materiel implies a loss of personnel. If unit ibu is Red, then for each $1 \le k \le \mathit{npers}(1)$, the number of type k personnel in the unit is reduced by the quantity

$$\min \{\mathit{dgpred}(k,iw,j) * L, [\text{resources}](ibu,\text{nmat}(1)+k)\}.$$

If unit ibu is Blue, then for each $1 \le k \le \mathit{npers}(2)$, the number of type k personnel in the unit is reduced by the quantity

$$\min \{\mathit{dgpblu}(k,iw,j) * L, [\text{resources}](ibu,\text{nmat}(2)+k)\}.$$

If iw < nw, iw is incremented by 1 and the process (starting with the definition of L) is repeated. The preceding is an efficient way of computing the attrition, but leads to an unfortunate anomaly: the way the air-to-ground weapons are ordered can affect personnel losses. The anomaly arises only when the battle unit has some type j materiel but so little that

$$\sum_{iw=1}^{nw} K(iw,j,ibu) > [\text{resources}](ibu,j)$$

(before [resources](ibu,j) is reduced). Losses of materiel are never affected by the ordering of air-to-ground weapons.

7. SUPPLIES CONSUMPTION

Every unit's consumption of supplies is assessed at the end of each frame, immediately after all engagements are evaluated and the resulting attrition is assessed. An inactive unit (one in posture class -1 or 0) consumes no supplies; therefore, the rest of this section applies only to active units.

Let s = 1 or s = 2. If *nss*(s) = 0--side s supplies are not played--then nothing is done. Otherwise, consumption of supplies by side s battle units in a given frame is determined as follows:

Let unit ibu be a side s battle unit. Let $1 \le k \le nss(s)$. Denote the unit's demand for type k supplies by D(ibu,k). Suppose the unit is not engaged or it is a passenger in a stacked task force. In the latter case, let pc = 1; otherwise let pc be its posture class. If s = 1, D(ibu,k) is defined by

$$D(ibu,k) = \sum_{irs=1}^{nrs(1)} \mathit{tframe} * \mathit{ssvncr}(k,irs,pc) * [resources](ibu,irs).$$

If s = 2

$$D(ibu,k) = \sum_{irs=1}^{nrs(2)} \mathit{tframe} * \mathit{ssvncb}(k,irs,pc) * [resources](ibu,irs).$$

Thus, every resource demands supplies according to its unit's posture class, and the unit's demand is the sum of its resources' demands. Alternatively, suppose unit ibu is engaged and is not a passenger in a stacked task force. Let pp be its posture, and let

$$p = \begin{cases} pp - 19; & pp \ge 40 \\ pp - 9; & pp < 40. \end{cases}$$

Let

$$index = mapps(s,p).$$

For every $1 \le irs \le nrs(s)$, let

$$lambda(irs) = frinv(irs,ibu),$$

the fraction of the unit's resources of type irs that are actively involved in combat. Then

$$D(ibu,k) = \sum_{irs=1}^{nrs(s)} \mathit{tframe} * \mathit{ssvact}(k,irs,index) * (lambda(irs) * [resources](ibu,irs)) + \sum_{irs=1}^{nrs(s)} \mathit{tframe} * \mathit{ssvres}(k,irs,index) * (1 - lambda(irs)) * [resources](ibu,irs).$$

Because the rate of supplies consumption might depend strongly on the attack or defense posture, the game design variables *ssvact* and *ssvres* can distinguish different attack or defense postures. The variable *mapps*, which induces the third subscript of *ssvact* and *ssvres*, can be used to consolidate attack postures (40-49) or defense postures (10-29), thereby reducing the storage requirements of *ssvact* and *ssvres*.

The preceding defines any battle unit's demands for supplies. Again choose a side s battle unit, unit ibu. Suppose it does not belong to a task force. Let $1 \le k \le \mathit{nss}(s)$. The unit's present stock of type k supplies, stk, is given by

$$stk = [resources](ibu,nequip(s)+k).$$

The quantity of type k supplies it consumes is computed as

$$C = \min \{D(ibu,k),\ stk\}$$

(it cannot consume more than it has), and therefore its stock of type k supplies at the end of the frame is redefined as follows:

$$[resources](ibu,nequip(s)+k) \longleftarrow stk - C.$$

Alternatively, suppose unit ibu is an element of a task force (possibly the only element). Let TF be the set of every unit in the task force, identified by unit number. Choose $1 \le k \le \mathit{nss}(s)$. The goal is to determine how much of the type k supplies held by unit ibu are consumed in the frame. The task force's total demand for type k supplies is

$$dd = \sum_{i \varepsilon TF} D(i,k).$$

Its total stock of type k supplies is

$$stk = \sum_{i \varepsilon TF} [resources](i,nequip(s)+k).$$

The amount of type k supplies consumed by the task force is computed as

$$C = \min \{dd, stk\}.$$

Each element of the task force is assessed the same fraction of its stock of type k supplies:

$$[resources](i,nequip(s)+k)$$

$$\leftarrow \frac{stk - C}{stk} * [resources](i,nequip(s)+k)$$

for every $i \varepsilon TF$ and, in particular, for i = ibu. Thus, the elements of a task force share their supplies.

After assessing supplies consumption by a task force in a movement posture, IDAHEX ascertains whether the task force has exhausted its supplies of any type (assuming *nss*(s) > 0). If so, the task force might lack supplies it needs in order to move and should not be allowed to change location. IDAHEX finds what its movement delay would be if it were just starting its movement, in its present posture. If that delay equals or exceeds 10**9, the task force's mission is changed to a single order specifying 10 as the desired posture and its present location as the desired objective--which causes the task force to abort its movement and attempt to revert to a hold posture in its present location.

8. COMMUNICATING WITH THE IDAHEX COMPUTER PROGRAM

IDAHEX uses the following files:

file10 - Red player input
file11 - Red player output
file20 - Blue player input
file21 - Blue player output
file50 - game design (input) data
file51 - game designer's output file
file60 - game design (input) data

The program references a file by using its number--10, 11, 20, 21, 50, 51, or 60--as the data set reference number in a FORTRAN formatted read or write statement or by using its name (file10, file11, etc.) as the file name in a PL/I get or put statement.

File 50 contains all the game design data except the values of *envmap, rtemap,* and *barmap. File 60 contains the data that* define *envmap, rtemap,* and *barmap* in each cycle. The format and sequence of the data in file 50 and file 60 are explained in Section 9. File 51 contains IDAHEX's interpretation of the data in file 50, and warning or error messages if IDAHEX questions the correctness of the data. An error message indicates that IDAHEX was unable to interpret the input data. It may continue processing the game design data, but it will terminate execution before the players can enter air strike specifications or commands. A warning draws the game designer's attention to a possible error in the design data; execution continues. If execution is allowed to proceed and a game is played, file 51 also contains a history of the game.

The game design datum *nprint* indicates the number of distinct data sets that are being used. If *nprint* = 1, IDAHEX expects files 50, 10, and 20 to be associated with the same data set (usually card reader input) and all the output files to be associated with the same data set (usually high speed printer output). If *nprint* = 2, IDAHEX expects file 50 to be associated with a different data set than files 10 and 20, which it expects to be associated with the same data set, and it expects file 51 to be associated with a different data set than files 11 and 21, which it expects to be associated with the same data set. If *nprint* = 3, IDAHEX expects every file to be associated with a different data set. No matter what the value of *nprint*, file 60 must be associated with a distinct data set for which the rewind operation is permitted. Normally, *nprint* = 1 means that IDAHEX is being used in a batch processing mode; *nprint* = 2 means it is being used interactively with one terminal, which the players share; and *nprint* = 3 means it is being used with two terminals, one for the Red player and one for the Blue player.

By using the save command (see Volume 3, Section 4), a player can save the game situation in an unformatted, rewindable file that he designates by number. At least one file should be set aside for this purpose. It is wise to set aside more than one because, if not, every save will necessarily overwrite the previous one.

The file associations must be in effect when IDAHEX is invoked. The following MULTICS commands illustrate how the file associations are established when IDAHEX is to be played from exactly one terminal (*nprint* = 1).

```
io attach file10 syn_ user_input
io attach file11 syn_ user_output
io attach file20 syn_ user_input
io attach file21 syn_ user_output
io attach file50 vfile_ Sinai_dd
io attach file60 vfile_ Sinai_terrain_maps
io attach file90 vfile_ Sinai_dd_unformatted
io attach file91 vfile_ Sinai_game.1
io attach file92 vfile_ Sinai_game.2
io attach file93 vfile_ Sinai_game.3
set_cc file51 -on
set_cc file11 -on
set_cc file21 -on
line_length 115
```

The files 90, 91, 92, and 93 identified above are intended as places to save the game situation. The first character of every line output to files 11, 21, and 51 is a carriage control character; hence, the files' carriage control attribute is set to "on".

The IDAHEX main program is named cgcm. Invoking it invokes IDAHEX.

The game design variable *iprint* governs the output's level of detail. If *iprint* $\geq$ 1, file 51 will contain a complete description of every significant change in a battle unit's status. If *iprint* $\geq$ 5, the players will be informed of every significant change in a unit's status. If *iprint* $\geq$ 7, file 51 will contain a complete description of every change in a unit's status. File 51 will always contain a detailed description of every engagement. If *iprint* ≥ 15, the players will receive the same description. If *iprint* < 15, they will not be informed of an engagement's average kill matrices (denoted A and D in Sections 5.1.1 and 5.1.2). If *iprint* < 9, they will not be informed of the values of the attackers' and defenders' weapons (Section 5.1.2). If *iprint* < 5, they will not be informed of the losses in the engagement. A value of 9 is generally best.

9. GAME DESIGN DATA INPUT

The game design data are read from files 50 and 60 in a sequence of groups. Section 9.1 describes the groups of data in the order in which they are read. The description of a group consists of: (1) a line listing the variables whose values the group fixes and, on the right-hand-side, the name of the IDAHEX entry point that reads the group; and (2) FORTRAN statements indicating how the group is read and therefore the correct order of the data within the group. The FORTRAN statements do not correspond exactly to IDAHEX source code, and although generally written according to MULTICS FORTRAN language conventions, are not necessarily valid source code for any compiler; their sole purpose is to explain how the contents of files 50 and 60 fix the values of the game design variables. Contrary to FORTRAN convention, the FORTRAN code in this section assumes that the statements in a do loop are not executed even once if the lower bound specified in the do statement exceeds the upper bound.

Section 9.2 contains a complete example of files 50 and 60 as a sequence of lines representing card images. The lines are grouped to correspond to the data groups of Section 9.1. The first line of each group ends with the code number used for the group in Section 9.1 (the number at the start of the line naming the game design variables and the entry point).

9.1 SEQUENCE AND FORMAT

Game design variables' names are not italicized in this section. The only variables mentioned that are not game design variables are do loop indices and the following: nnsyl (fixed by cgcm); vtemp, wtemp, i, j, k, side, itemp, name, jtemp, temp, old, kap (defined in cmbt0), kdp (defined in cmbt0), nequip, nmat, nrs.

In accordance with the rest of the manual, some variables' names contain two components--for example, frinv.f, frinv.x, freff.f. Such a variable is referenced in only one subprogram; it takes the first component of its name from the subprogram's name. In the actual IDAHEX source program, the variable's name is simply the second component of the two-component name used to identify it in this manual.

The following format statements are cited by many read statements in this subsection:

```
2 format (8i10)
3 format (8f10.0)
```

9.1.1 File 50

1. iprint, nprint — cgcm

```
read(50,2) iprint, nprint
```

2. tinit, tend, tframe, tcycle, tpd, delta — time0

```
read(50,3) tinit, tend, tframe, tcycle, tpd, delta
```

3. ncells, nrankl — net0

```
read(50,2) ncells, nrankl
```

4. ename0 — net0

```
1 read(50,4) i, (name(k), k = 1, nnsyl)
4 format (i5,5x,6a8)
  if (i.le.0) go to 6
  do 5 k = 1, nnsyl
     ename0(i,k) = name(k)
5    continue
  go to 1
6 continue
```

5. nenv — net0

```
read(50,2) nenv
```

6. ename — net0

```
  do 5 i = 1, nenv
  read(50,4) (ename(i,j), j = 1, nnsyl)
4 format (6a8)
5 continue
```

7. [basic_env] net0

```
1 read(50,2) i, itemp
  if (i.le.0) go to 5
  [basic_env](i) = itemp
  go to 1
5 continue
```

8. rname0 net0

```
1 read(50,4) i, (name(k), k = 1, nnsyl)
4 format (i5,5x,6a8)
  if (i.le.0) go to 6
  do 5 k = 1, nnsyl
     rname0(i,k) = name(k)
5    continue
  go to 1
6 continue
```

9. bname0 net0

```
1 read(50,4) i, (name(k), k = 1, nnsyl)
4 format(i5,5x,6a8)
  if (i.le.0) go to 6
  do 5 k = 1, nnsyl
     bname0(i,k) = name(k)
5    continue
  go to 1
6 continue
```

10. nrtety net0

```
read(50,2) nrtety
```

11. rname net0

```
  do 5 i = 1, nrtety
  read(50,4) (rname(i,j), j = 1, nnsyl)
4 format (6a8)
5 continue
```

12. nbarty net0

```
read(50,2) nbarty
```

13. bname net0

```
    do 5 i = 1, nbarty
    read(50,4) (bname(i,j), j = 1, nnsyl)
  4 format (6a8)
  5 continue
```

14. [basic_rtetype], [basic_bartype] net0

```
  1 read(50,2) i, (vtemp(k), k = 1,3),
                  (wtemp(k), k = 1,3)
    if (i.le.0) go to 5
    do 4 k = 1,3
       j = [successor](i,k)
       if (j.le.0) go to 4
       [basic_rtetype](i,j) = vtemp(k)
       [basic_bartype](i,j) = wtemp(k)
  4    continue
    go to 1
  5 continue
```

15. depth net0

```
read(50,3) depth
```

16. iblul, nsyl, nutype bu0

```
read(50,2) iblul, nsyl, nutype
```

17. npost bu0

```
read(50,2) (npost(i), i = 1,4)
```

18. itrfp bu0

```
read(50,2) itrfp
```

19. nggwep, ngawep, ntrpt, nss, npers bu0

```
    do 5 i = 1,2
  5 read(50,2) nggwep(i), ngawep(i),
               ntrpt(i), nss(i), npers(i)
```

20. rsname buO

```
      do 5 k = 1,2
         do 5 i = 1, nrs(k)
            read(50,4) (rsname(i,j,k) j = 1,2)
4           format (a5,1x,a5)
5           continue
```

21. flag buO

```
      read(50,2) (flag(i), i = 1, nutype)
```

22. nrst, iars buO

```
      do 5 i = 1, nutype
    5 read(50,2) nrst(i), (iars(j,i), j = 1, nrst(i))
```

23. toe buO

```
      do 5 i = 1, nutype
    5 read(50,3) (toe(i,j), j = 1, nrs(flag(i)))
```

24. aisize buO

```
      do 5 i = 1, nutype
    5 read(50,3) (aisize(i,j), j = 1,4)
```

25. buname, butype, buloc, bupost, tentry, [resources] buO

```
    1 read(50,4) i, (vtemp(j), j = 1, nsyl)
    4 format (i5,5x,7a8)
      if (i.le.0) go to 10
      do 5 j = 1, nsyl
         buname(i,j) = vtemp(j)
    5    continue
      read(50,6) butype(i), buloc(i), bupost(i), tentry(i)
    6 format (3i10,f10.0)
      read(50,3) ([resources](i,j), j = 1, nrs(flag(butype(i))))
      go to 1
   10 continue
```

26. [owner] buO

```
      read(50,2) border, side
      do 5 i = 1, border
         [owner](i) = side
```

```
5    continue
  side = 3 - side
  do 6 i = border + 1, ncells
     [owner](i) = side
6    continue
7 read(50,2) i, itemp
  if (i.le.0) go to 8
  [owner](i) = itemp
  go to 7
8 continue
```

27. pmapup, pmapdn wait0

```
   do 5 i = 10, 19
      pmapup(i) = 20
5     pmapdn(i) = -10
   do 6 i = 20, 29
      pmapup(i) = 30
6     pmapdn(i) = 40
   do 7 i = 30, 39
      pmapup(i) = 40
7     pmapdn(i) = 40
   do 8 i = 40, 49
      pmapup(i) = 10
8     pmapdn(i) = 40
10 read(50,2) i, itemp, jtemp
   if (i.le.0) go to 11
   pmapup(i) = itemp
   pmapdn(i) = jtemp
   go to 10
11 continue
```

28. ptran wait0

```
   do 5 i = 1, 4
      do 5 j = 1, npost(i)
5        read(50,3) (ptran(i,j,k), k = 9, npost(i))
```

29. diseng wait0

```
   do 5 i = 1, nutype
5     read(50,3) (diseng(i,j), j = 1,2)
```

30. airmove wait0

```
   read(50,4) (airmove(i), i = 1, npost(3))
4  format (8ℓ10)
```

31. mrair wait0

```
   read(50,3) (mrair(i), i = 1, nutype)
```

32. bdelay, mr wait0

```
        do 5 i = 1, nutype
           do 5 j = 1, npost(3)
              read(50,3) (bdelay(i,j,k), k = 1, nbarty)
              read(50,3) (mr(i,j,k), k = 1, nrtety)
     5        continue
```

33. trnreq, trncap, ssreqm wait0

```
        do 10 k = 1, 2
           read(50,3) (trnreq(i), i = 1, nrs(k))
           read(50,3) (trncap(i), i = 1, nrs(k))
           do 8 j = 1, nrs(k)
     8        read(50,3) (ssreqm(i,j,k), i = 1, nss(k))
    10     continue
```

34. trptcl wait0

```
        read(50,2) (trptcl(i), i = 1, nutype)
```

35. loadcl wait0

```
        do 5 j = 1, 2
     5     read(50,2) (loadcl(i,j), i = 1, nrs(j))
```

36. nlc, ldsize, ldcap wait0

```
        do 10 k = 1,2
           read(50,2) nlc(k)
           do 8 i = 1,nrs(k)
     8        read(50,3) (ldsize(i,j,k), j = 1, nlc(k))
           read(50,3) (ldcap(i,k), i = 1, nrs(k))
    10     continue
```

37. fmr.f0, fmr.f, fmr.x fmr0

```
        do 5 i = 1, nutype
     5     read(50,3) fmr.f0(i), (fmr.f(i,j), j = 1,6)
        read(50,3) temp, (fmr.x(j), j = 1,6)
```

38. ssvncr ssuse0

```
        do 5 i = 1, nss(1)
           do 5 j = 1, nrs(1)
     5        read(50,3) (ssvncr(i,j,k), k = 1,3)
```

39. ssvncb ssuse0

```
      do 5 i = 1, nss(2)
        do 5 j = 1, nrs(2)
   5      read(50,3) (ssvncb(i,j,k), k = 1,3)
```

40. mapps, ssvact, ssvres ssuse0

```
      do 10 side = 1,2
   1    read(50,2) i, k
        if (i.le.0) go to 10
        mapps(side,[kpost](i)) = k
        read(50,4) old
   4    format (ℓ10)
        if (old) go to 1
        do 6 i = 1, nss(side)
        read(50,3) (ssvact(i,j,k), j = 1, nrs(side))
        read(50,3) (ssvres(i,j,k), j = 1, nrs(side))
   6    continue
        go to 1
  10    continue
```

41. poff cmbt0

```
      k = 0
      do 5 i = 10, 9 + npost(1)
        k = k + 1
        poff(i) = k
   5    continue
      do 6 i = 20, 19 + npost(2)
        k = k + 1
        poff(i) = k
   6    continue
      do 7 i = 40, 39 + npost(4)
        k = k + 1
        poff(i) = k
   7    continue
      do 8 i = 10 + npost(1), 19
   8    poff(i) = poff(10)
      do 9 i = 20 + npost(2), 29
   9    poff(i) = poff(20)
      do 10 i = 40 + npost(4), 49
  10    poff(i) = poff(40)
  11  read(50,2) i, k
      if (i.le.0) go to 12
      if (i.le.29) poff(i) = max(k,poff(10))
      if (i.ge.40) poff(i) = max(k,poff(40))
      go to 11
  12  continue
```

42. stdtgt cmbt0

```
      do 5 j = 1, 2
5        read(50,3) (stdtgt(i,j), i = 1, nrs(j))
```

43. aggatk cmbt0

```
      do 5 k = 1, 2
         do 5 i = 1, nggwep(k)
5           read(50,3) (aggatk(i,j,k), j = 1, nmat(3-k))
```

44. aggdef cmbt0

```
      do 5 k = 1, 2
         do 5 i = 1, nggwep(k)
5           read(50,3) (aggdef(i,j,k), j = 1, nmat(3-k))
```

45. katk cmbt0

```
      do 5 k = 1, 2
         do 5 = 1, nggwep(k)
5           read(50,3) (katk(i,j,k), j = 1, nmat(3-k))
```

46. kdef cmbt0

```
      do 5 k = 1, 2,
         do 5 i = 1, nggwep(k)
5           read(50,3) (kdef(i,j,k), j = 1, nmat(3-k))
```

47. fckar cmbt0

```
      kap = 0
      do 5 i = 40, 49
5        kap = max(poff(i), kap)
      do 6 j = 1, 2
         do 6 i = 1, nggwep(j)
6           read(50,3) (fckar(i,j,k), k = 1, kap)
```

48. fckdr cmbt0

```
      kdp = 0
      do 5 i = 10, 29
5        kdp = max(poff(i), kdp)
      do 6 j = 1, 2
         do 6 i = 1, nggwep(j)
6           read(50,3) (fckdr(i,j,k), k = 1, kdp)
```

49. fckac cmbt0

```
      do 5 j = 1, 2
         do 5 i = 1, nmat(j)
5           read(50,3) (fckac(i,j,k), k = 1, kdp)
```

50. fckdc cmbt0

```
      do 5 j = 1, 2
         do 5 i = 1, nmat(j)
5           read(50,3) (fckdc(i,j,k), k = 1, kap)
```

51. fckare cmbt0

```
      do 5 j = 1, 2
         do 5 i = 1, nggwep(j)
5           read(50,3) (fckare(i,j,k), k = 1, nenv)
```

52. fckdre cmbt0

```
      do 5 j = 1, 2
         do 5 i = 1, nggwep(j)
5           read(50,3) (fckdre(i,j,k), k = 1, nenv)
```

53. fckace cmbt0

```
      do 5 j = 1, 2
         do 5 i = 1, nmat(j)
5           read(50,3) (fckace(i,j,k), k = 1, nenv)
```

54. fckdce cmbt0

```
      do 5 j = 1, 2
         do 5 i = 1, nmat(j)
5           read(50,3) (fckdce(i,j,k), k = 1, nenv)
```

55. barier cmbt0

```
      do 5 j = 1, 2
         do 5 i = 1, nggwep(j)
5           read(50,3) (barier(i,j,k), k = 1, nbarty)
```

56. dpersr cmbt0

```
      do 5 i = 1, npers(1)
         do 5 j = 1, nggwep(2)
    5       read(50,3) (dpersr(i,j,k), k = 1, nmat(1))
```

57. dpersb cmbt0

```
      do 5 i = 1, npers(2)
         do 5 j = 1, nggwep(1)
    5       read(50,3) (dpersb(i,j,k), k = 1, nmat(2))
```

58. td cmbt0

```
      read(50,3) (td(i), i = 1, nenv)
```

59. febab cmbt0

```
      read(50,3) (febab(i), i = 1, nbarty)
```

60. febad cmbt0

```
      read(50,3) febad
```

61. vanish cmbt0

```
      read(50,3) (vanish(i), i = 1, nutype)
```

62. frdval.f0atk, frdval.fatk frdv0

```
      do 5 i = 40, 39+npost(4)
    5    read(50,3) frdval.f0atk(poff(i)),
                    (frdval.fatk(poff(i),j), j = 1,7)
```

63. frdval.f0def, frdval.fdef frdv0

```
      do 5 i = 10, 9+npost(1)
    5    read(50,3) frdval.f0def(poff(i)),
                    (frdval.fdef(poff(i),j), j = 1,7)
      do 6 i = 20, 19+npost(2)
    6    read(50,3) frdval.f0def(poff(i)),
                    (frdval.fdef(poff(i),j), j = 1,7)
```

64. frdval.x frdv0

```
      read(50,3) (frdval.x(i), i = 1,7)
    4 format (10x, 7f10.0)
```

65. vfeba.f0, vfeba.f vfeba0

```
      vfeba.npa = 6
    1 read(50,2) i, j
      if (i.le.0) go to 5
      read(50,3) vfeba.f0(poff(i), poff(j)),
                 (vfeba.f(poff(i),poff(j),k), k = 1,7)
      go to 1
    5 read(50,2) i, j
      if (i.le.0) go to 6
      read(50,3) vfeba.f0(vfeba.npa+poff(i), poff(j)),
                 (vfeba.f(vfeba.npa+poff(i), poff(j),k), k = 1,7)
      go to 5
    6 continue
```

66. vfeba.fr vfeba0

```
      read(50,4) (vfeba.fr(i), i = 1,7)
    4 format (10x, 7f10.0)
```

67. ppoh frinv0

```
      do 5 j = 1, nutype
    5   read(50,3) (ppoh(i,j), i = 1, npers(flag(j)))
```

68. spdd frinv0

```
      do 5 j = 1, nrs(1)
    5   read(50,3) (spdd(i,j), i = 1, nsp(i))
```

69. frinv.f0, frinv.f frinv0

```
      do 5 j = 1, nmat(1)
        do 5 i = 1, nsp(1)
    5     read(50,3) frinv.f0(i,j)
                     (frinv.f(i,j,k), k = 1,6)
```

70. frinv.x(1,*) frinv0

```
      read(50,4) (frinv.x(1,i), i = 1,6)
    4 format (10x, 7f10.0)
```

71. spdd frinv0

```
      do 5 j = 1, nrs(2)
    5    read(50,3) (spdd(i,nrs(1)+j), i = 1, nsp(2))
```

72. frinv.f0, frinv.f frinv0

```
      do 5 j = 1, nmat(2)
         do 5 i = 1, nsp(2)
    5       read(50,3) (frinv.f0(i,nrs(1)+j),
                       (frinv.f(i,nrs(1)+j,k), k = 1,6))
```

73. frinv.x(2,*) frinv0

```
      read(50,4) (frinv.x(2,i), i = 1,6)
    4 format (10x, 7f10.0)
```

74. pg(*,1), prot(*,*,1) frinv0

```
      read(50,3) (pg(i,1), i = 1, nequip(1))
      do 5 i = 1, nequip(1)
    5    read(50,3) (prot(i,j,1), j = 1, nggwep(1))
```

75. pg(*,2), prot(*,*,2) frinv0

```
      read(r0,3) (pg(i,2), i = 1, nequip(2))
      do 5 i = 1, nequip(2)
    5    read(50,3) (prot(i,j,2), j = 1, nggwep(2))
```

76. freff.f0, freff.f, freff.x freff0

```
      do 5 i = 1, 2
         read(50,3) freff.f0(i), (freff.f(i,j), j = 1,7)
         read(50,4) (freff.x(i,j), j = 1,7)
    4    format (10x, 7f10.0)
    5    continue
```

77. prep.f, prep.x prep0

```
      do 6 j = 1, 2
         do 5 i = 1, nmat(j)
 5          read(50,3) (prep.f(i,j,k), k = 1,7)
            continue
         read(50,3) (prep.x(j,k), k = 1,7)
 6       continue
```

78. nactyp, nagwep air0

```
      do 5 i = 1, 2
 5       read(50,2) nactyp(i), nagwep(i)
```

79. kag air0

```
      do 5 k = 1, 2
         do 5 i = 1, nagwep(k)
 5          read(50,3) (kag(i,j,k), j = 1, nmat(3-k))
```

80. fcagrp air0

```
      do 5 j = 1, 2
         do 5 i = 1, nagwep(j)
 5          read(50,3) (fcagrp(i,j,k), k = 1,4)
```

81. fcagcp air0

```
      do 5 j = 1, 2
         do 5 i = 1, nmat(j)
 5          read(50,3) (fcagcp(i,j,k), k = 1,4)
```

82. fcagre air0

```
      do 5 j = 1, 2
         do 5 i = 1, nagwep(j)
 5          read(50,3) (fcagre(i,j,k), k = 1, nenv)
```

83. fcagce air0

```
      do 5 j = 1, 2
         do 5 i = 1, nmat(j)
 5          read(50,3) (fcagce(i,j,k), k = 1, nenv)
```

84. dgpred air0

```
      do 5 k = 1, nmat(1)
         do 5 i = 1, npers(1)
5           read(50,3) (dgpred(i,j,k), j = 1, nagwep(2))
```

85. dgpblu air0

```
      do 5 k = 1, nmat(2)
         do 5 i = 1, npers(2)
5           read(50,3) (dgpblu(i,j,k), j = 1, nagwep(1))
```

86. agload air0

```
      do 5 k = 1, 2
         do 5 i = 1, nactyp(k)
5           read(50,3) (agload(i,j,k), j = 1, nagwep(k))
```

87. aagatk(*,*,1) air0

```
      do 5 i = 1, nagwep(1)
5        read(50,3) (aagatk(i,j,1), j = 1, nmat(2))
```

88. aagdef(*,*,1) air0

```
      do 5 i = 1, nagwep(1)
5        read(50,3) (aagdef(i,j,1), j = 1, nmat(2))
```

89. aagatk(*,*,2) air0

```
      do 5 i = 1, nagwep(2)
5        read(50,3) (aagatk(i,j,2), j = 1, nmat(1))
```

90. aagdef(*,*,2) air0

```
      do 5 i = 1, nagwep(2)
5        read(50,3) (aagdef(i,j,2), j = 1, nmat(1))
```

91. aagred air0

```
      do 5 i = 1, nagwep(1)
         do 5 j = 1, nmat(2)
5           read(50,3) (aagred(i,j,k), k = 1,3)
```

92. aagblu air0

```
      do 5 i = 1, nagwep(2)
         do 5 j = 1, nmat(1)
 5          read(50,3) (aagblu(i,j,k), k = 1,3)
```

93. haven.zoc, haven.pthome haven0

```
      read(50,4) haven.zoc,
                 (haven.pthome(i), i = 1,2)
 4    format (ℓ10, 2i10)
```

9.1.2 File 60

The following data are read at the start of a cycle.

94. envmap cgcm

```
 1    read(60,2,end=5) i, j
      if (i.le.0) go to 5
      envmap(i) = j
      go to 1
 5    continue
```

95. rtemap wait1

```
 1    read(60,2,end=5) i, j
      if (i.le.0) go to 5
      rtemap(i) = j
      go to 1
 5    continue
```

96. barmap wait1

```
 1    read(60,2,end=5) i, j
      if (i.le.0) go to 5
      barmap(i) = j
      go to 1
 5    continue
```

Before the start of a game, IDAHEX sets

```
envmap(i) = i for every i
rtemap(i) = i for every i
barmap(i) = i for every i,
```

before any data redefining them are read. At the start of each cycle, including the first, it reads file 60 for redefinitions of envmap, rtemap, and barmap as described above.

9.2 SAMPLE DATA

This subsection illustrates a complete set of game design data. Each line, except for identification codes at the end, represents an 80-column card image.

9.2.1 File 50

BEST AVAILABLE COPY

```
    100        2                                                        01
      0    1000.       .25       .5       1.0        .1                02
    115        12                                                       03
  2     open desert                                                     04
  3     hilly or mountainous
  1     clear, flat or gently rolling
  4     urban
     -9
      4                                                                 05
clear                                                                   06
open desert
rough
urban
      1        0                                                        07
      2        1
      3        1
      4        1
      5        2
      6        2
      7        1
      8        1
      9        3
     10        3
     11        0
     12        3
     14        1
     15        1
     16        2
     17        2
     18        2
     19        1
     20        3
     21        3
     22        0
     24        4
     25        0
     26        1
     27        1
     28        2
     29        2
     30        1
     31        1
     32        3
     33        3
     34        3
     37        0
     38        2
     39        2
     40        2
     41        2
     43        1
     44        3
     45        3
     48        0
     49        1
     50        2
     51        2
     52        2
     53        2
     54        1
```

55	1
56	4
60	2
61	2
62	2
63	2
64	2
65	2
66	2
67	2
68	2
72	2
73	2
74	2
75	2
76	2
77	2
78	2
79	2
80	1
83	2
84	2
85	2
86	2
87	2
88	1
89	3
90	3
91	3
94	2
95	2
96	2
97	2
98	2
99	1
100	3
101	3
102	3
103	3
104	
105	2
106	2
107	2
108	2
109	3
110	2
111	3
112	1
113	1
114	1

-9

```
4      loose sand, no roads                                  08
1      excellent trafficability
2      poor roads
3      firm sand, no roads
5      rough
6      mountainous
   -9                                                        09
1      Suez Canal
2      impassable
```

BEST AVAILABLE COPY

-9
.6 10
excellent trafficability 11
poor roads
firm sand, no roads
loose sand, no roads
rough
mountainous
2 12
Suez Canal 13
impassable

2	2	0			2	0	14
3	2	0	2	0	2	0	
4	3	0	2	0	3	0	
5	4	0	2	0	4	0	
6	3	0	2	0	2	0	
7	2	0	2	0	1	0	
8	5	0	2	0	5	0	
9	5	0	2	0	5	0	
10	5	0	5	0	6	0	
14	2	0			2	0	
15	3	0	2	0	2	0	
16	2	0	2	0	4	0	
17	4	0	2	0	2	0	
18	3	0	2	0	2	0	
19	5	0	1	0	2	0	
20	6	0	5	0	2	0	
21	6	0	6	0	6	0	
26	2	0			3	1	
27	1	0	3	1	3	1	
28	1	0	4	1	4	1	
29	1	0	4	1	4	1	
30	1	0	3	1			
31	5	0			1	0	
32	6	0	5	0	2	0	
33	6	0	6	0	6	0	
34			6	0			
38							
39	1	0	3	0	4	0	
40	1	0	4	0	4	0	
41			2	0	1	0	
43	5	0	2	2	1	0	
44	6	0	5	0	2	0	
45			5	0			
49	3	0	2	0	2	0	
50	4	0	4	0	2	0	
51	4	0	4	0	2	0	
52	4	0	2	0	4	0	
53	2	0	4	0	2	0	
54	2	2	3	0	2	0	
55	1	0	3	1	3	1	
56			3	1	3	1	
60	4	0			2	0	
61	2	0	2	0	4	0	
62	2	0	4	0	4	0	
63	2	0	0	0	2	0	
64	4	0	4	0	2	0	
65	2	0	4	0	4	0	
66	1	0	2	0	4	0	
67	1	0	4	0	2	0	

68	1	0	2	0	2	0	
72	4	0	2	0	4	0	
73	4	0	4	0	4	0	
74	4	0	4	0	4	0	
75	4	0	4	0	2	0	
76	2	0	4	0	3	0	
77	2	0	2	0	2	0	
78	2	0	5	0	5	0	
79	2	0	2	0	5	0	
80			5	0			
83	4	0	4	0	2	0	
84	4	0	4	0	4	0	
85	4	0	4	0	4	0	
86	4	0	4	0	4	0	
87	3	0	4	0	2	0	
88	5	0	1	0	5	0	
89	6	0	2	0	6	0	
90	6	0	2	0	6	0	
91			6	0	6	0	
94	4	0	4	0	4	0	
95	4	0	2	0	4	0	
96	4	0	4	0	4	0	
97	4	0	4	0	5	0	
98	3	0	5	0	4	0	
99	5	0	2	0	2	0	
100	6	0	6	0	2	0	
101	6	0	5	0	2	0	
102	6	0	5	0	5	0	
103			5	0			
105	4	0					
106	4	0					
107	4	0					
108	5	0					
109	5	0					
110	5	0					
111	5	0					
112	2	0					
113	2	0					
38	1	0	1	0	4	0	
-9							
15,0							15
150	2	8					16
2	1	1	1				17
11							18
4	1	1	2	1			19
4	0	1	2	1			
small arms							20
anti- tank							
tanks							
arty							
SAMS & AAA							
trpt							
ammo							
fuel& other							
pers							
small arms							
anti tank							
tanks							
arty							
trans -port							

BEST AVAILABLE COPY

	ammo								
	fuel& other								
	pers								
	1	1	1	1	1	2	2	2	21
	9	1	2	3	4	5	6	7	22
	8	9							
	9	1	2	3	4	5	6	7	
	8	9							
	6	1	4	6	7	8	9	IARS	
	7	1	2	5	6	7	8	9	
	8	1	2	3	4	6	7	8	
	9								
	8	1	2	3	4	5	6	7	
	8								
	8	1	2	3	4	5	6	7	
	8								
	8	1	2	3	4	5	6	7	
	8								
	155	35	32	12	6	255.	70	165.	23
	2550					TOE			
	25	3	97	0	2	155.	50	75.	
	1210					TOE			
	12	0	0	55	0	275.	80	75.	
	1250					TOE			
	10	1	0	0	25	170	30	30	
	520					TOF			
	5	0	0	0	0	500	0	20	
	650					TOE			
	185	50	95	25	1100.	330	390.	5900	
	130	35	127	25	1170.	400	320	5600	
	10	0	0	0	500	10	30	750	
	30	15	10	15					24
	30	15	10	15					
	15	7.5	12	7.8					
	7	7	5	7					
	2	2	2	2					
	40	20	10	20					
	45	25	12	20					
	2	2	1.9	2					
2	7TH MOT RIFLE								25
	1	26	10	-10					
	160	35	32	12	6	255.	75	165.	
	2550								
4	112TH MOT RIFLE								
	1	26	10	-5					
	170	38	32	12	7	245.	80	165.	
	2545								
3	10TH TANK REG								
	2	27	10	0					
	30	3	98	0	3	140.	70	70.	
	1210								
5	5TH MOT RIFLE								
	1	26	10	0					
	175	38	32	12	6.5	255.	85	160	
	2550								
7	102ND MOT RIFLE								
	1	27	10	0					
	170	35	32	12	6	255.	80	160.	
	2540								
8	8TH MOT RIFLE								

BEST AVAILABLE COPY

```
         1         27        10         0
       165         37        32        12         6     250.       70     155.
      2550
  14       140TH MOT RIFLE
         1         29        10         0
       160         35        32        12         6     255.       80     165.
      2540
  15       15TH MOT RIFLE
         1         29        10         0
       160         33        30        12       6.3    250.       70     160.
      2535
  20       66TH ARTY REG
         3         55        10        -1
        13          0         0        53         0     260.       75      75.
      1245
  21       RED ARTY REG
         3         56        10        -3
        12          0         0        52         0     265.       70      75.
      1250
  22       22ND MOT RIFLE
         1         55        10      -2.5
       155         38        30        12         4     195.       75     105
      2550
  62       62ND TANK REG
         2         56        10      -2.5
        25          4        90         0         0     150.       55      75
      1205
  25       2ND TANK REG
         2         28        10         0
        30        3.5        95         0       2.5     155.       50      70.
      1210
  37       RED AAW REG
         4         29        10        -5
        11          0         0         0        25     170.       30      25
       520
  39       RED AAW REG
         4         43        10      -9.5
        10          0         0         0        24     165.       30      25
       520
  50       RED TRANSPORT UNIT
         5         27        10         0
         7          0        5.         0         0     490.     50.     100.
       645
  51       RED TRANSPORT UNIT
         5         29        10         0
       5.5          0         0         0         0     485.   50000   40000.
       650
  52       RED TRANSPORT UNIT
         5         56        10         0
         5          0         0         0         0     500.        5      20
       650
  53       RED TRANSPORT UNIT
         5         30        10         0
         5          0         0         0         0     500.        5      20
       650
 150       3RD MECH INF BDE.
         6         99        10       -20
       215         50        95        20   1100.        330    380.     5900
 153       BLUE MECH INF BDE
         6         39        10       -20
```

BEST AVAILABLE COPY

	215	49	95	18	1080.	330	380.	5550	
154	54TH BLUE MECH INF								
	6	39	10	0					
	215	50	95	12	1075.	330	380.	5800	
156	2ND ARMORED BDE								
	7	106	0	0					
	150	35	125	10	1170.	400	360	5550	
158	EZEKIEL ARMORED								
	7	110	0	0					
	145	31	124	12	1150	380	350.	5540	
159	BLUE ARMORED BDE.								
	7	99	10	-30					
	150	35	127	20	1170.	400	360.	5560	
160	DEATH'S HEAD								
	7	112	0	0					
	150	35	127	18	1168.	400	360.	5600	
170	BLUE TRANSPORT UNIT								
	8	99	10	0					
	10.	0.	25.	0.	500.	41000.	54000.	750.	
171	BLUE TRANSPORT UNIT								
	8	99	10	0					
	10	0	0	0	490	150.	350.	750	
172	BLUE TRANSPORT UNIT								
	8	106	10	0					
	10	0	0	0	495	5	25	745	
176	BLUE MECH INF BDE								
	6	67	10	-15					
	180	50	90	20	1000.	300	380.	5700	
174	BLUE TRANSPORT UNIT								
	8	65	10	0					
	11	0	0	0	500	5	30	720	
	-9								
	35								26
	43	1							
	44	1							
	45	1							
	46	1							
	55	1							
	56	1							
	-9								
	-9								27
	0	.16667							28
	.041667	0							
	0								
	0								
	0								
	.10	.5							29
	.05	.4							
	.20	.6							
	.10	.6							
	.05	.0							
	.09	.40							
	.06	.35							
	.04	.40							
	1								30
	0	0	0	0	0	0	0	0	31
	.25	100							32
	250	220	148	75	40	4			
	.30	100					2		
	280	250	165	80	42	3			

BEST AVAILABLE COPY

.30	100					3		
330	300	188	75	39	3			
.10	100					4		
310	280	185	78	44	10			
.15	100					5		
430	400	238	75	43	10			
1.0	100							
330	300	198	95	55	15			
1.0	100					7		
350	320	210	100	57	13			
.2	100							
480	450	268	85	55	25			
0	.001	0	0	0	0	1	1	33
.0875								
.96	0	.55	0	0	1	0	0	
.001								
0	.0001							
0	.0001							
0	.0001							
0	.0001							
0	.0001							
0	.0001							
0	.0001							
0	.0001							
0	0							
0	.001	0	0	00	1.0	1	.0880	
.83	0	.5	0	1	0	0	.001	
0	.00001							
0	.00001							
0	.00001							
0	.00001							
0	.00001							
0	.00001							
0	.00001							
0	0							
0	0	0	0	1	0	0	1	34
0	0	0	0	0	1	0	0	35
1								
0	0	0	0	1	0	0	0	
1								36
999999999								
.05								
999999999								
999999999								
999999999								
999999999								
1.								
1.								
.03								
0	0	0	0	0	1.2	0	0	
0								
1								
999999999								
.06								
999999999								
999999999								
999999999								
1.								
1.								
.035								

BEST AVAILABLE COPY

0	0	0	0	1.5	0	0	0	
0	.5	1.0	1.0	1.0	1.0	1.0		37
0	.6	1.0	1.0	1.0	1.0	1.0		
0	.5	1.0	1.0	1.0	1.0	1.0		
0	.5	1.0	1.0	1.0	1.0	1.0		
0	.45	1.0	1.0	1.0	1.0	1.0		
-1.								
0	.5	1.0	1.0	1.0	1.0	1.0		
0	.5	1.0	1.0	1.0	1.0	1.0		
0	.75	1.0	1.0	1.0	1.0	1.0		
0	0	0						38
0	0	0						
0	0							
0	0	0						
0	0	0						
0	0	0						
0	0	0						
0	0	0						
0	0	.001						
0	0	0						
0	0	0						
0	0	0						
0	0	0						
0	0	0						
0	0	0						
0	0	0						
0	0	0						
.01	.020	.034						
0	0	0						39
0	0	0						
0	0	0						
0	0	0						
0	0	0						
0	0	0						
0	0	0						
0	0	.0005						
0	0	0						
0	0	0						
0	0	0						
0	0	0						
0	0	0						
0	0	0						
0	0	0						
.011	.022	.037						
10	1							40
*								
0	0	0	2.75	0	0	0	0	
.02								
0	0	0	0000	0	0	0	0	
.001								
0	0	0	000		0	0	0	
.020								
0	0	0	0	0	0	0	0	
.010								
11	2							
*								
0	0	0	2.00	0	0	0	0	
.020								
0	0	0	0	0	0	0	0	
0								

BEST AVAILABLE COPY

0	0	0	0	0	0	0	0	
.020								
0	0	0	0	0	0	0	0	
.020								
20	3							
4								
0	0	0	2.00	0	0	0	0	
.01								
0	0	0	0	0	0	0	0	
.001								
0	0	0	0	0		0	0	
.030								
0	0	0	0	0	0	0	0	
.020								
40	4							
4								
0	0	0	3.0	0	0	0	0	
.02								
0	0	0	0	0	0	0	0	
.001								
0	0	0	0	0	0	0	0	
.024								
0	0	0	0	0	0	0	0	
.01								
-9								
10	5							
4								
0	0	0	2.8	0	0	0	.02	
0	0	0	0	0	0	0	.02	
0	0	0	0	0	0	0	.0230	
0	0	0	0	0	0	0	.015	
11	6							
4								
0	0	0	0	0	0	0	.001	
0	0	0	0	0	0	0	.001	
0	0	0	0	0	0	0	.017	
0	0	0	0	0	0	0	.015	
20	7							
4								
0	0	0	01.5	0	0	0	.015	
0	0	0	0	0	0	0	.010	
0	0	0	0	0	0	0	.022	
0	0	0	0	0	0	0	.020	
40	8							
4								
0	0	0	2.8	0	0	0	.025	
0	0	0	0	0	0	0	.001	
0	0	0	0	0	0	0	.030	
0	0	0	0	0	0	0	.020	
-9								
0								41
512	109	193	91	45	1200.	360	600.	42
10650								
185	50	95	20	800.	330	380.	5900	
.885	.100	.001	.001	.01	.001	.001		43
.685	.505	.25	.020	.020	.010	.01		
.250	.150	.55	.04	.030	.010	.01		
.600	.100	.150	.400	.055	.025	.025		
.785	.200	.001	.001	.001	.010	.001	.001	
.270	.040	.700	.010	.001	.005	.001	.001	

.070	.050	.850	.020	.001	.050	.011	.001	
.000	.050	.300	.500	.001	.100	.005	.005	
.880	.080	.010	.001	.010	.010	.010		44
.700	.005	.020	.001	.001	.001	.001		
.270	.050	.500	.070	.050	.025	.025		
.550	.050	.150	.330	.050	.025	.025		
.900	.060	.050	.020	.001	.040	.025	.025	
.270	.030	.750	.001	.001	.020	.001	.001	
.050	.02	.85	.02	.001	.08	.01	.01	
.400	.050	.250	.300	.001	.05	.050	.050	
.150	.200	.020	.010	.250	.135	.135		45
.100	.050	.200	.080	.100	.050	.050		
.550	.400	.340	.550	.700	.400	.400		
.250	.300	.100	.550	.800	.500	.500		
.150	.200	.005	.020	.150	.250	.160	.135	
.250	.100	.400	.300	.250	.150	.075	.075	
.570	.300	.500	.590	.400	.800	.400	.400	
.220	.350	.110	.590	.210	.800	.450	.450	
.250	.200	.150	.250	.230	.130	.100		46
.300	.250	.400	.200	.200	.100	.100		
1.050	.800	.350	.550	.500	.250	.250		
.350	.300	.100	.600	.600	.300	.300		
.250	.250	.005	.020	.050	.100	0.05	.05	
.650	.200	.500	.700	.700	.900	.500	.400	
.700	.600	.750	.800	.800	.800	.450	.350	
.300	.400	.100	.610	.210	.900	.480	.420	
1.0								47
1.0								
1.0								
1.0								
1.0								
1.0								
1.0								
1.0								
1.0	1.0	.50						48
1.0	1.0	.70						
1.0	1.0	.50						
1.0	1.0	.20						
1.0	1.0	.55						
1.0	1.0	.75						
1.0	1.0	.45						
1.0	1.0	.20						
1.0	1.0	1.20						49
1.0	1.0	1.15						
1.0	1.0	1.05						
1.0	1.0	1.05						
1.0	1.0	1.00						
1.0	1.0	1.05						
1.0	1.0	1.02						
1.0	1.0	1.02						
1.0	1.0	1.19						
1.0	1.0	1.14						
1.0	1.0	1.00						
1.0	1.0	1.05						
1.0	1.0	1.00						
1.0	1.0	1.00						
1.0	1.0	1.00						
1.0								50
1.0								
1.0								

BEST AVAILABLE COPY

1.0				
1.0				
1.0				
1.0				
1.0				
1.0				
1.0				
1.0				
1.0				
1.0				
1.0				
1.0				
1.00	1.00	.85	.80	51
1.00	1.10	1.05	1.20	
1.00	.85	.50	.65	
1.00	.90	.70	.50	
1.00	.96	.90	.80	
1.00	1.25	1.05	1.18	
1.00	.87	.55	.60	
1.00	.95	.65	.45	52
1.00	1.00	.95	.85	
1.00	1.30	1.50	1.80	
1.00	.90	.70	.75	
1.00	.95	.80	.85	
1.00	1.00	1.5	.90	
1.00	1.35	1.50	1.75	
1.00	.95	.65	.70	
1.00	.97	.85	.85	
1.0	1.00	.50	.20	53
1.0	.95	.70	.40	
1.0	.9	.85	1.00	
1.0	1.00	.90	.95	
1.0	1.00	.95	.90	
1.0	1.00	.85	.90	
1.0	1.00	.85	.95	
1.0	1.00	.90	.90	
1.0	1.00	.49	.19	
1.0	1.00	.69	.39	
1.0	1.00	.84	.99	
1.0	1.00	.89	.94	
1.0	1.0	.84	.89	
1.0	1.00	.84	.94	
1.0	1.00	.89	.89	
1.0	1.05	1.15	1.10	54
1.0	1.05	1.10	1.00	
1.0	1.25	1.45	1.70	
1.0	1.02	1.05	1.05	
1.0	1.00	1.00	1.00	
1.0	1.09	1.07	1.08	
1.0	1.00	.96	1.04	
1.0	1.00	.97	1.04	
1.0	1.05	1.14	1.09	
1.0	1.04	1.09	.99	
1.0	1.24	1.45	1.69	
1.0	1.01	1.04	1.04	
1.0	1.08	1.06	1.07	
1.0	1.05	.95	1.03	
1.0	1.06	.96	1.03	
.33				55
.30				

BEST AVAILABLE COPY

.25								
.60								
.40								
.35								
.30								
.50								
7.0	3.5	2.0	4.5	4.5	2.0	.01	.01	56
7.5	3.5	4.0	3.0	3.0	2.0	0	0	
7.5	3.5	4.0	5.0	5.0	2.0	0	0	
15.0	4.0	3.0	6.0	5.0	3.0	.01	.01	
5.0	4.0	1.0	8.5	3.0	.01	.01		57
5.5	4.0	4.0	7.0	3.0	0	0		
5.5	4.0	4.0	9.0	3.0	0	0		
11.0	4.5	3.0	11.0	4.0	.01	.01		
5	5	3	2					58
10	15							59
.99								60
.02	.05	.16	0	0	.04	.04	0	61
1.00	.24	.15	.10	.08	.05	.04	.001	62
0	.08	.11	.16	.23	.28	.35	.70	63
0	.16	.22	.32	.46	.56	.65	.98	
0	.05	.06	.09	.12	.14	.17	.25	
0	.5	1.0	2.0	3.0	4.0	6.	50.0	64
40	11							65
-15.	1.0	2.0	3.30	4.60	7.00	9.00	41.00	
40	10							
-20.	0	.4	2.30	4.60	6.00	8.00	40.00	
-9								
40	10							
-21.	0	.6	2.70	5.00	7.00	9.00	50.00	
40	11							
-13.	2.0	2.5	3.70	6.00	8.00	10.00	51.00	
-9								
0	1.0	1.5	2.0	3.0	4.0	5.0	50.0	66
375								67
295								
268								
120								
50								
2350								
2250								
100								
.12	.21	11						68
.30	0	1						
1.15	.56	4						
3.50	0	5						
0	0	5						
0	0	1.0						
0	0	.1						
0	0	.1						
0	0	0						
0	.35	.65	.80	1.0	1.0	1.0		69
.2	.50	.85	1.0	1.0	1.0	1.0		
0	.25	.50	.80	1.0	1.0	1.0		
0	.35	.70	.60	1.0	1.0	1.0		
0	.80	.95	1.0	1.0	1.0	1.0		
0	.25	.50	.75	1.0	1.0	1.0		
0	.35	.65	.85	1.0	1.0	1.0		
0	.30	.60	.90	1.0	1.0	1.0		
0	.25	.50	.75	1.0	1.0	1.0		

BEST AVAILABLE COPY

0	.25	.55	.80	1.0	1.0	1.0			
0	.50	.70	.90	1.0	1.0	1.0	24		
0	.25	.55	.80	1.0	1.0	1.0	34		
0	.25	.50	.75	1.0	1.0	1.0			
0	.25	.50	.75	1.0	1.0	1.0	25		
0	.25	.50	.75	1.0	1.0	1.0	35		
1.00	1.00	1.00	1.0	1.0	1.0	1.0			
0	.25	.50	.75	1.0	1.0	1.0	26		
0	.25	.50	.75	1.0	1.0	1.0	36		
1.00	1.00	1.00	1.00	1.0	1.0	1.0			
1.00	1.00	1.00	1.00	1.0	1.0	1.0	27		
0	.45	.85	.95	1.0	1.0	1.0	37		
1.00	1.00	1.00	1.0	1.0	1.0	1.0			
1.00	1.00	1.00	1.00	1.0	1.0	1.0	28		
0	.50	.90	.99	1.0	1.0	1.0	38		
0	.25	.50	.75	1.0	1.0	1.0			70
.13	.25	8.6							71
.34	0	1							
1.25	.60	4							
3.30	0	9							
0	0	1.00							
0	0	.1							
0	0	.1							
0	0	0							
0	.30	.60	.70	1.00	1.0	1.0			72
.15	.40	.65	.80	1.00	1.0	1.0			
0	.25	.50	.75	1.00	1.0	1.0			
0	.40	.75	.85	1.00	1.0	1.0			
0	.80	.90	.95	1.00	1.0	1.0			
0	.25	.50	1.00	1.00	1.0	1.0			
0	.30	.60	.80	1.00	1.0	1.0			
0	.30	.55	.85	1.00	1.0	1.0			
0	.25	.50	.75	1.00	1.0	1.0			
0	.30	.55	.80	1.00	1.0	1.0			
0	.50	.70	.90	1.00	1.0	1.0			
0	.35	.60	.65	1.00	1.0	1.0			
1.00	1.00	1.00	1.00	1.00	1.0	1.0			
0	.	.50	.75	1.00	1.0	1.0			
0	.25	.50	.75	1.00	1.0	1.0			
1.	1.	1.	1.	1.00	1.0	1.0			
1.	1.	1.	1.	1.00	1.0	1.0			
0	.50	.75	.95	1.00	1.0	1.0			
1.	1.	1.	1.	1.00	1.0	1.0			
1.	1.	1.	1.	1.	1.	1.			
0	.50	.75	1.00	1.00	1.0	1.0			
0	.25	.50	.75	1.0	1.0	1.0			73
1	2	2	3	4	4				74
1.00									
1.00									
1.40									
1.40	1.00	1.00							
1.50	1.00	2.00	1.00						
7.00	2.00	10.00	15.00						
1	2	5	3	4					75
1.0									
1.0									
1.0	1.0								
1.0	1.0	1.0							
5.0	2.0	10.0	15.0						
1.00	1.0	.9	.6	.35	.20	.10		0	76

BEST AVAILABLE COPY

0	.8	1.0	1.5	2.0	2.5	3.0	3.5	
1.00	1.0	.85	.60	.35	.20	.10	0	
0	1.0	1.25	1.5	2.0	2.5	3.0	3.5	
3.00	1.7	1.20	1.00	.95	.85	.50		77
3.00	1.75	1.20	1.00	.90	.85	.50		
3.00	1.75	1.20	1.00	.95	.85	.50		
3.00	1.75	1.20	1.00	.95	.85	.50		
3.00	1.75	1.20	1.00	.95	.85	.50		
3.20	1.80	1.25	1.00	.95	.85	.50		
3.20	1.80	1.25	1.00	.90	.85	.50		
3.20	1.80	1.25	1.00	.90	.85	.50		
-1.	-.5	0	1.0	2.0	10.	60.		
2.80	1.70	1.15	1.00	.90	.80	.50		
2.80	1.70	1.15	1.00	.90	.80	.50		
2.80	1.70	1.15	1.00	.95	.80	.50		
2.80	1.70	1.15	1.00	.90	.80	.50		
2.90	1.75	1.20	1.00	.90	.80	.50		
2.90	1.75	1.20	1.00	.90	.80	.50		
2.90	1.75	1.20	1.00	.90	.80	.50		
-1.	-.5	0	1	2	10	60		
1	1							78
3	3							
.004	.002	.001	.030	.050	.100	.110		77
.85	.65	.60	.84	.84	.90	.015	.010	
.30	.20	.20	.35	.35	.45	.030	.025	
.02	.20	.002	.03	.03	.05	.120	.100	
1.00	1.10	1.25	1.15					80
1.00	1.00	1.00	1.0					
1.00	1.00	1.10	1.00					
1.00	1.10	1.25	1.15					
1.00	1.10	1.30	1.20					81
1.00	1.10	1.15	1.10					
1.00	1.00	1.10	1.00					
1.00	1.00	1.00	1.00					
1.00	1.00	1.00	1.00					
1.00	1.00	1.00	1.00					
1.00	1.00	1.00	1.00					
1.00	1.00	1.00	1.00					
1.00	1.05	1.25	1.15					
1.00	1.05	1.10	1.00					
1.00	1.00	1.15	1.00					
1.00	1.00	1.00	1.00					
1.00	1.00	1.00	1.00					
1.00	1.00	1.00	1.00					
1.00	1.00	1.00	1.00					
1.00	1.00	.80	.50					82
1.00	.90	.75	.50					
1.00	1.00	.85	.50					
1.00	1.00	.85	.50					
1.00	1.00	.75	.60					83
1.00	1.00	.50	.20					
1.00	1.00	1.10	.85					
1.00	1.00	1.00	.85					
1.00	1.00	.90	.80					
1.00	1.00	.90	.75					
1.00	1.00	.60	.30					
1.00	1.00	.65	.40					
1.00	1.00	.70	.55					
1.00	1.00	.45	.15					
1.00	1.00	1.00	.90					

BEST AVAILABLE COPY

1.00	1.00	1.00	.90					
1.00	1.00	.85	.80					
1.00	1.00	.85	.35					
1.00	1.00	.60	.35					
13.0	13.0	15.0						84
2.5	2.6	3.0						
4.0	4.0	1.0						
4.5	4.6	6.0						
4.5	4.6	6.0						
1.0	1.0	1.0						
.01	.01	.01						
.01	.01	.01						
11.0								85
4.0								
4.0								
9.0								
1.2								
.01								
.01								
6.2								86
.90	1.80	.75						
.19	.45	.20						
.12	.25	.15						
.54	.05	.28	.10	.10	.01	.01		87
.60	0	.29	.10	.10	.01	.01		88
.16	.04	.70	.04	.03	.05	.01	.01	89
.16	.04	.70	.04	.03	.10	.01	.01	
.16	.10	.20	.20	.02	.10	.05	.05	
.16	.04	.65	.04	.03	.10	.01	.01	90
.16	.04	.70	.04	.03	.10	.01	.01	
.16	.10	.20	.20	.02	.10	.05	.05	
.35	.60	.20						91
.05	0	.05						
.25	.29	.20						
.15	.10	.20						
.15	.10	.25						
.10	.01	.05						
.05	.01	.05						
.10	.20	.25						92
.01	.01	.01						
.75	.70	.25						
.10	.01	.20						
.05	.01	.20						
.05	.05	.20						
.01	.01	.01						
.01	.01	.01						
.15	.25	.20						
.01	.01	.01						
.70	.65	.20						
.10	.01	.20						
.05	.01	.20						
.05	.05	.20						
.01	.01	.01						
.01	.01	.01						
.20	.15	.10						
.05	.15	.05						
.20	.15	.10						
.25	.20	.10						
.20	.02	.10						
.10	.10	.30						
.10	.05	.15						
.10	.05	.15						
4	1	5						93

9.2.2 File 60

```
          1          2                                      ENVMAP
         -9                                                 ENVMAP
          3          4                                      RTEMPA
         -9                                                 RTEMAP
         -9                                                 BARMAP
         -9                                                 ENVMAP
          3          3
         -9                                                 RTEMAP
         -9                                                 BARMAP
```

10. GLOSSARY

This section contains an alphabetical glossary of variables and functions mentioned in this volume. For each variable such that array dimensions or consistency of the game design data implies a finite upper or lower bound on the variable's value, that bound is given; "UB" and "LB" are abbreviations of "upper bound" and "lower bound".

Name	Description	Type
aagatk(i,j,k)	fraction of fire of side k air-to-ground weapon of type i allocated to enemy materiel of type j when enemy materiel belongs to engaged battle unit in attack posture LB = 0	real
aagblu(i,j,k)	fraction of fire of Blue air-to-ground weapons of type i allocated to enemy materiel of type j when enemy materiel belongs to unengaged battle unit in posture class k LB = 0	real
aagdef(i,j,k)	fraction of fire of side k air-to-ground weapons of type i allocated to enemy materiel of type j when enemy materiel belongs to engaged battle unit in hold or disengagement posture LB = 0	real
aagred(i,j,k)	fraction of fire of Red air-to-ground weapons of type i allocated to enemy materiel of type j when enemy materiel belongs to unengaged battle unit in posture class k LB = 0	real
aggatk(i,j,k)	fraction of fire of side k ground-to-ground weapons of type i allocated to enemy materiel of type j if side k is engagement attacker	real
aggdef(i,j,k)	fraction of fire of side k ground-to-ground weapons of type i allocated to enemy materiel of type j if side k is engagement defender	real
agload(i,j,k)	notional load of side k air-to-ground weapons of type j on side k aircraft of type i LB = 0	real
airmove(i)	true if i-th movement posture implies air movement; false if not	logical

Name	Description	Type
aisize(ij)	area of area of responsibility of a unit of type i in posture class j if its resources coincide with *toe*(i,*) LB = 0	real
barier(i,j,k)	factor applied to katk(i,*,j) if weapon i belongs to battle unit attacking across barrier of type k and engagement feba ≤ *febab*(k)/*depth* LB = 0	real
barmap(i)	barrier type if basic barrier type is i LB = 0, UB = *nbarty*	integer
[bartype](i,j)	type of barrier between cell i and cell j (0 signifies no barrier); undefined unless cells are adjacent LB = 0, UB = *nbarty*	integer
[*basic_bartype*](i,j)	basic type of barrier between cell i and cell j (0 signifies no barrier); undefined unless cells are adjacent LB = 1, UB = nbarraw	integer
[*basic_env*](i)	basic environment in cell i UB = nenvraw	integer
[*basic_rtetype*](i,j)	basic type of route between cell i and cell j; undefined unless cells are adjacent LB = 1, UB = nrteraw	integer
bdelay(i,j,k)	barrier delay for a unit of type i in j-th movement posture crossing a barrier of type k LB = 0	real
bname(i,*)	description of barrier type i	character
bname0(i,*)	description of basic barrier type i	character
buloc(i)	location of unit i	integer
buloc(i)	location of unit i (a cell number) at t = *tinit* LB = 1, UB = *ncells*	integer
buname(i,*)	name of unit i	character

Name	Description	Type
bupost(i)	posture of unit i	integer
bupost(i)	posture of unit i at t = *tinit* LB = 0, UB = 19	integer
butype(i)	type of unit i LB = 1, UB = *nutype*	integer
delta	length of time a unit must be in movement posture before arrival of enemy unit at its location (point of origin) to avoid reversion to disengagement posture LB = 0	real
depth	distance from center of any cell to center of adjacent cell LB > 0	real
dgpblu(i,j,k)	loss of Blue personnel of type i associated with destruction of unit-quantity of Blue materiel of type k by Red air-to-ground weapons of type j LB = 0	real
dgpred(i,j,k)	loss of Red personnel of type i associated with destruction of a unit-quantity of Red materiel of type k by Blue air-to-ground weapons of type j LB = 0	real
diseng(i,1)	minimum time required for a type i unit to disengage LB = 0	real
diseng(i,2)	factor applied to movement delay to determine additional disengagement delay imposed on type i unit disengaging without a rearguard LB = 0	real
dpersb(i,j,k)	loss of Blue personnel of type i associated with destruction of a unit-quantity of Blue materiel	real

Name	Description	Type
	of type k by Red ground-to-ground weapons on type j LB = 0	
dpersr(i,j,k)	loss of Red personnel of type i associated with destruction of a unit-quantity of Red materiel of type k by Blue ground-to-ground weapons of type j LB = 0	real
ename(i,*)	description of environment type i	character
ename0(i,*)	description of basic environment type i	character
[environment](i)	type of environment in cell i LB = 1, UB = *nenv*	integer
envmap(i)	environment type if basic environment type is i LB = 1, UB = *nenv*	integer
fcagce(i,j,k)	factor applied to *kag*(*,i,3-j) if target battle unit is in environment k LB = 0	real
fcagcp(i,j,k)	factor applied to *kag*(*,i,3-j) if target battle unit is in posture class k LB = 0	real
fcagre(i,j,k)	factor applied to *kag*(i,*,j) if target battle unit is in environment k LB = 0	real
fcagrp(i,j,k)	factor applied to *kag*(i,*,j) if target battle unit is in posture class k LB = 0	real
fckac(i,j,k)	factor applied to katk(*,i,3-j) if materiel belongs to battle unit in posture p ($10 \leq p \leq 29$); k = *poff*(p) LB = 0	real
[fckac](i,j,k)	factor applied to katk(*,i,3-j) if materiel belongs to battle unit in posture k; it equals *fckac*(i,j,poff(k)) LB = 0	real

Name	Description	Type
fckace(i,j,k)	factor applied to katk(*,i,3-j) if environment in engagement cell is type k LB = 0	real
fckar(i,j,k)	factor applied to katk(i,*,j) if weapon belongs to battle unit in posture p ($40 \leq p \leq 49$); k = *poff*(p) LB = 0	real
[fckar](i,j,k)	factor applied to katk(i,*,j) if weapon belongs to battle unit in posture k; it equals *fckar*(i,j,poff(k)) LB = 0	real
fckare(i,j,k)	factor applied to katk(i,*,j) if environment in engagement cell is type k LB = 0	real
fckdc(i,j,k)	factor applied to kdef(*,i,3-j) if materiel belongs to battle unit in posture p ($10 \leq p \leq 29$); k = *poff*(p) LB = 0	real
[fckdc](i,j,k)	factor applied to kdef(*,i,3-j) if materiel belongs to battle unit in posture k; it equals *fckdc*(i,j,poff(k)) LB = 0	real
fckdce(i,j,k)	factor applied to kdef(*,i,3-j) if environment in engagement cell is type k LB = 0	real
fckdr(i,j,k)	factor applied to kdef(i,*,j) if weapon belongs to battle unit in posture p ($10 \leq p \leq 29$); k = poff(p) LB = 0	real
[fckdr](i,j,k)	factor applied to kdef(i,*,j) if weapon belongs to battle unit in posture k; it equals *fckdr*(i,j,poff(k)) LB = 0	real

Name	Description	Type
fckdre(i,j,k)	factor applied to kdef(i,*,j) if environment in engagement cell is type k LB = 0	real
febab(i)	depth of attacker penetration of defender's cell at which effect of type i barrier ceases LB = 0, UB = *depth*	real
febad	degree of attacker penetration of defender's cell at which defenders must disengage and retreat to another cell LB = 0, UB < 1	real
flag(i)	side to which unit of type i belongs	integer
[floor](a)	largest integer ≤ a	integer
fmr.f(i,j)	factor applied to movement rate of type i unit when its ratio of transport capacity to transport demand is *fmr.x*(j) LB = 0	real
fmr.f0(i)	factor applied to movement rate of type i unit when its ratio of transport capacity to transport demand is 0 LB = 0	real
fmr.x(j)	ordinate corresponding to *fmr.f*(i,j) for any i LB = 0	real
frdval.fatk(i,j)	fraction of value lost by attacker in 1 unit of time when attacker-to-defender force ratio is *frdval.x*(j) and attacker is in posture p; i = *poff*(p) LB = 0, UB = 1	real
frdval.fdef(i,j)	fraction of value lost by defender in 1 unit of time when attacker-to-defender force ratio is *frdval.x*(j) and defender is in posture p; i = *poff*(p) LB = 0, UB = 1	real

Name	Description	Type
frdval.f0atk(i)	fraction of value lost by attacker in 1 unit of time when attacker-to-defender force ratio is 0 and attacker is in posture p; i = *poff*(p) LB = 0, UB = 1	real
frdval.f0def(i)	fraction of value lost by defender in 1 unit of time when attacker-to-defender force ratio is 0 and defender is in posture p; i = *poff*(p) LB = 0	real
frdval.x(i)	ordinate corresponding to *frdval.fatk*(j,i) and *frdval.fdef*(j,i) for any j LB = 0	real
freff.f(i,j)	fractional effectiveness in combat of one or more side i battle units located in same cell if total area of their areas of responsibility divided by area of cell equals *freff.x*(i,j) LB = 0	real
freff.f0(i)	fractional effectiveness in combat of one or more side i battle units located in same cell if total area of their zone of responsibility is 0 LB = 0	real
freff.x(i,j)	ordinate corresponding to *freff.f*(i,j) LB = 0	real
frinv.f(i,j,k)	fraction of type r resources available for combat in side s battle unit if available quantity of type i support resources divided by demand for type i support resources equals *frinv.x*(s,k); j = r if s = 1, j = nrs(1)+r if s = 2 LB = 0	real

Name	Description	Type
frinv.f0(i,j)	fraction of type r resources available for combat in side s battle unit if available quantity of type i support resources divided by demand for type i support resources equals 0; j = r if s = 1, j = nrs(1)+r if s = 2 LB = 0	real
frinv.x(i,j)	ordinate corresponding to frinv.f(k,ℓ,j) for any (k,ℓ) associated with side i LB = 0	real
haven.pthome(i)	preferred direction of retreat for side i LB = 1, UB = 6	integer
haven.zoc	truth value of "attacker's zone of control extends into adjacent cells"	logical
iars(i,j)	absolute index of i-th resource on list of resources in a unit of type j LB = 0, UB = nrs(*flag*(j))	integer
iblul	index number of lowest-numbered Blue unit LB = 2, UB = nbumax	integer
iprint	level of detail in terminal output LB = 0	integer
itrfp	index of transfer posture ($10 \leq i \leq 19$) LB = 10, UB = 10 + *npost*(1)	integer
kag(i,j,k)	amount of enemy materiel of type j destroyed by a single side k air-to-ground weapon of type i if all of the air-to-ground weapon's fire is allocated to enemy materiel of type j LB = 0	real
kap	max [*poff*(i); $40 \leq i \leq 49$]	integer

Name	Description	Type
katk(i,j,k)	amount of enemy materiel of type j destroyed in 1 unit of time by a single side k ground-to-ground weapon of type i if the weapon allocates all its fire to enemy materiel of type j; side k is the attacker in the engagement LB = 0	real
katk(i,j,k)	*tframe* * *katk*(i,j,k)	real
kdef(i,j,k)	amount of enemy materiel of type i destroyed in 1 unit of time by a single side k ground-to-ground weapon of type i if the weapon allocates all its fire to enemy materiel of type j; side k is the defender in the engagement LB = 0	real
kdef(i,j,k)	*tframe* * *kdef*(i,j,k)	real
kdp	max [*poff*(i); 10 ≤ i ≤ 29]	integer
[kpost](p)	p - 9 if p < 40, p - 19 if p ≥ 40	integer
ldcap(i,j)	load capacity of resource of type i belonging to side j LB = 0	real
ldsize(i,j,k)	load size of a single side k resource of type i relative to load class j LB = 0	real
loadcl(i,j)	load class of a resource of type i belonging to side j LB = 0, UB = nlcmax	integer
mapps(i,j)	pointer used to reference data on supplies consumption in engaged side i battle unit in posture p, where j = [kpost](p)	integer

Name	Description	Type
	(see *ssvact* and *ssvres*) LB = 1, UB = ssuse.npsmax	
mr(i,j,k)	movement rate of a unit of type i in j-th movement posture along a route type k LB = 0	real
mrair(i)	air movement rate of unit of type i LB = 0	real
nactyp(i)	number of side i aircraft types LB = 0, UB = nacmax	integer
nagwep(i)	number of sides i air-to-ground weapon types LB = 0, UB = nagwmx	integer
nbarty	number of types of barriers (between cells) LB = 0, UB = nbarmx	integer
ncells	number of cells (largest identification number of any cell) in area of war LB = 1, UB = ncelmx	integer
nenv	number of types of cell environments LB = 1, UB = nenvmx	integer
nequip	number of types of side i equipment (nwep(i) + *ntrpt*(i)) LB = 1	integer
ngawep(i)	number of types of side i ground-to-air weapons LB = 0	integer
nggwep(i)	number of types of side i ground-to-ground weapons LB = 1, UB = nggwmx	integer
nlc(i)	highest load class of side i resources LB = 0, UB = nlcmax	integer

Name	Description	Type
nmarchp	max {k: *airmove*(k) = false.} LB = 1, UB = *npost*(3)	integer
nmat(i)	number of types of side i material (nwep(i) + *ntrpt*(i) + *nss*(i)) LB = 1, UB = nmatmx	integer
nnsyl	number of computer double-words occupied by name or any environment, route, or barrier type	integer
npers(i)	number of types of side i personnel LB = 0	integer
npost(i)	number of postures in posture class i LB = 1, UB = 10	integer
nprint	number of output devices (printer & terminals) to be used LB = 1, UB = 3	integer
*nrank*1	number of cells in first row of area of war LB = 2, UB = *ncells*	integer
nrs(i)	number of types of side i resources LB = 1, UB = nrsmax	integer
nrst(i)	number of types of resources that a unit of type i may have LB = 1, UB = nrs(*flag*(i))	integer
nrtety	number of types of routes (between cells) LB = 1, UB = natmax	integer
nsp(i)	number of types of side i support resources (*nss*(i) + *npers*(i))	integer
nss(i)	number of types of side i supplies LB = 0, UB = nssmax	integer

Name	Description	Type
nsyl	number of computer double-words occupied by name of any unit LB = 1, UB = 2	integer
ntrpt(i)	number of types of side i transport vehicles LB = 0, UB = ntrnmx	integer
nutype	number of types of units LB = 1, UB = nutymx	integer
nwep(i)	number of types of side i weapons (*nggwep*(i) + *ngawep*(i)) LB = 1, UB = nwepmx	integer
[owner](i)	1 if Red owns cell i, 2 if Blue	integer
[*owner*](i)	side that owns cell i at t = *tinit*	integer
pg(i,j)	protection group to which side j resources of type i belong LB = 1	integer
pmapdn(i)	first posture a unit enters when it transitions from posture i to a lower posture class	integer
pmapup(i)	first posture a unit enters when it transitions from posture i to a higher posture class	integer
poff(i)	offset pointer used to reference ground combat data for a unit in posture i (see *frdval.fatk*, *frdval.fdef*, *vfeba.f*)	integer
ppoh(i,j)	number of overhead type i personnel in type j battle unit LB = 0	real
prep.f(i,j,k)	factor applied to katk(*,i,3-j) if defending unit, a member of side j, is credited with defense preparation time equal to *prep.x*(j,k) LB = 0	real
prep.x(i,j)	ordinate corresponding to *prep.f*(k,i,j) for any k	real

Name	Description	Type
prot(i,j,k)	amount of side k resources of type i that a side k ground-to-ground weapon of type j can protect LB = 0	real
ptran(i,j,k)	time required to transition from j-th posture in posture class i to k-th posture in posture class i	real
[resources](i,j)	quantity of resources of type j in unit i (classification of resources by type depends on unit's side)	real
[*resources*](i,j)	quantity of resources of type j in unit i at t = *tinit* LB = 0	real
rname(i,*)	description of route type i	character
rname0(i,*)	description of basic route type i	character
rsname(i,*,j)	description of side j resource type i	character
rsvala(i,j)	standard value of side j type i resource on attack	real
rsvald(i,j)	standard value of side j type i resource on defense	
rtemap(i)	route type if basic route type is i LB = 1, UB = *nrtety*	integer
[rtetype](i,j)	type of route between cell i and cell j; undefined unless cells are adjacent LB = 1, UB = *nrtety*	integer
spdd(i,j)	demand for type i support resources by a unit quantity of type r resources; j = r if resources belong to Red battle unit; j = nrs(1)+r if resources belong to Blue battle unit LB = 0	real
ssreqm(i,j,k)	quantity of side k supplies of type i required by a side k resource of type j in order to move LB = 0	real

Name	Description	Type
ssvact(i,j,k)	quantity of type i supplies consumed in one unit of time by a type j resource actively involved in ground combat; supplies and resource belong to a battle unit from side s in posture p; k = *mapps*(s,[kpost](p)) LB = 0	real
ssvncb(i,j,k)	amount of type i supplies consumed in one unit of time by a type j resource in a Blue battle unit in posture class k; $1 \leq k \leq 3$ LB = 0	real
ssvncr(i,j,k)	amount of type i supplies consumed in one unit of time by a type j resource in a Red battle unit in posture class k; $1 \leq k \leq 3$ LB = 0	real
ssvres(i,j,k)	consumption of type i supplies in one unit of time by a type j resource not actively involved in combat but in an engaged battle unit; battle unit belongs to side s and is in posture p; k = *mapps*(s,[kpost](p)) LB = 0	real
stdtgt(i,j)	quantity of resource i in a standard side j ground force LB = 0	real
[successor](i,j)	j-th successor of cell i ($1 \leq j \leq 3$)	integer
t	current game time (t = *tinit* at start of game)	real
tcycle	length of cycle LB = *tpd*	real
td(i)	depth of defender's tactical zone when environment in engagement cell is type i LB = 0	real

Name	Description	Type
tend	time at which game ends LB = *tinit*	real
tentry(i)	time at which unit i entered location and posture class it is in at start of game	real
tentry(i)	virtual time at which unit i entered its present posture class	real
tframe	length of frame LB = 0, UB = *tpd*	real
tinit	time at start of game LB = 0	real
toe(i,j)	planned effective quantity of type j resources in a unit of type i LB = 0	real
tpd	length of period LB = *tframe*, UB = *tcycle*	real
trncap(i,j)	transport capacity of each side j resource of type i LB = 0	real
trnreq(i,j)	transport requirement of each side j resource of type i LB = 0	real
trptcl(i)	transport class of a unit of type i LB = 0	integer
vanish(i)	fraction of standard value at which battle unit of type i vanishes LB = 0, UB = 1	real

Name	Description	Type
vfeba.f0(i,j)	signed distance of FEBA movement in 1 unit of time if attacker-to-defender force ratio is 0, attacker is in posture p´, and defender is in posture p´´; j = *poff*(p´´); i = *poff*(p´) if attacker is Red; i = vfeba.npa + *poff*(p´) if attacker is Blue	real
vfeba.f(i,j,k)	signed distance of FEBA movement in 1 unit of time if attacker-to-defender force ratio is *vfeba.fr*(i), attacker is in posture p´, defender is in posture p´´; j = *poff*(p´´); i = *poff*(p´) if attacker is Red; i = vfeba.npa + *poff*(p´) if attacker is Blue LB = 0	real
vfeba.npa	maximum number of attack postures subprogram vfeba can accommodate given current array dimensions	integer
vfeba.fr(i)	ordinate corresponding to *vfeba.f*(j,k,i) for any (j,k) LB = 0	real

11. INDEX OF VARIABLES

12. REFERENCES

[1] Anderson, Lowell Bruce, *et al. IDA Ground-Air Model I (IDAGAM I).* Vol. I, R-199, Arlington, VA: Institute for Defense Analyses, October 1974.

[2] Anderson Lowell Bruce. *A Method for Determining Linear Weighting Values for Individual Weapons Systems,* WP-4, Improved Methodologies for General Purpose Forces Planning (New Methods Study), Arlington, VA: Institute for Defense Anlyses, Revised April 1973.

[3] Dare, D.P., and B.A.P. James. *The Derivation of Some Parameters for a Corps/ Division Model from a Battle Group Model,* Defence Operational Analysis Establishment Memorandum 7120, U.K.: Ministry of Defence, July 1971, CONFIDENTIAL.

[4] Holter, William H., *et al. Appendix D: NATO Combat Capabilities Analysis II (COMCAP II)* (U), GRC Report OAD-CR-8, McLean, VA: General Research Corporation, August 1973, SECRET (Appendix D is UNCLASSIFIED).

[5] Howes, David R., and Robert M. Thrall. "A Theory of Ideal Linear Weights for Heterogeneous Combat Forces," *Naval Research Logistics Quarterly,* Vol. 20, No. 4, December 1973. (Or see Robert M. Thrall and Associates, Chapter 2C, *Final Report to US Army Strategy and Tactics Analysis Group* (RMT-200-R4-33), May 1972.)

[6] Spudich, John. *The Relative Kill Productivity Exchange Ratio Technique,* Combined Arms Research Office, Booz-Allen Applied Research, Inc., n.d.

[7] US Army, Combat Developments Command, Headquarters, TAB E, Appendix II to Annex L, *Tank, Antitank and Assault Weapons Requirements Study* (U), Phase III (TATAWS III), December 1968, SECRET-NOFORN (Tab E, Appendix II to Annex L is UNCLASSIFIED).