

A Concise User's Guide to ISAAC_FL: ISAAC's *Mission-Fitness Landscape Mapper Program (U)*

Andrew Ilachinski

DISTRIBUTION STATEMENT A
Approved for Public Release
Distribution Unlimited

Center for Naval Analyses

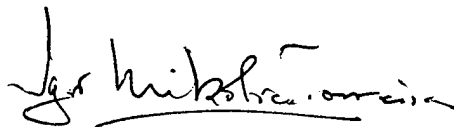
4401 Ford Avenue • Alexandria, Virginia 22302-1498

DTIC QUALITY INSPECTED 2

19990429 045

Approved for distribution:

Septer



Dr. Igor Mikolic-Torreira, Director
Systems and Tactics Team
Operating Forces Division

CNA's annotated briefings are either condensed presentations of the results of formal CNA studies that have been documented elsewhere or stand-alone presentations of research reviewed and endorsed by CNA. These briefings are the best opinion of CNA at the time of issue. They do not necessarily represent the opinion of the Department.

CLEARED FOR PUBLIC RELEASE

~~Distribution limited to DOD agencies. Specific authority: N00014-96-D-0001~~

For copies of this document call: CNA Document Control and Distribution Section (703)824-2943.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 074-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE September 1997	3. REPORT TYPE AND DATES COVERED Final	
4. TITLE AND SUBTITLE A Concise User's Guide to ISAAC-FL: ISAAC's Mission-Fitness Landscape Mapper Program			5. FUNDING NUMBERS C - N00014-96-D-0001	
6. AUTHOR(S) A Ilachinski				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Center for Naval Analyses 4401 Ford Avenue Alexandria, Virginia 22302-1498			8. PERFORMING ORGANIZATION REPORT NUMBER CAB 97-88	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)			10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Distribution unlimited			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 Words) ISAAC (Irreducible Semi-Autonomous Adaptive Combat) is a simple multiagent-based "toy model" of land combat design to illustrate how certain aspects of land combat can be viewed as emergent phenomena resulting from the collective, nonlinear, decentralized interactions among notional combatants. ISAAC takes a bottom-up, synthesist approach to the modeling of combat, vice the more traditional top-down, reductionist approach, and represents a first step toward developing a complex systems theoretic analyst's toolbox for identifying, exploring, and possibly exploiting emergent collective patterns of behavior on the battlefield. This model was developed as part of a recently completed project, sponsored by the Marine Corps Combat Development Command, that assessed the general applicability of "complex systems theory." The focus of this brief is a stand-alone Mission-Fitness Landscape Mapper that uses the core engine to "map-out" the behavior over a user-defined d-dimensional slice of ISAAC's total N-dimensional phase-space.				
14. SUBJECT TERMS artificial intelligence, computer programs, computerized simulation, ground combat, ISAAC (irreducible semi-autonomous adaptive combat), computer applications, land warfare, modeling and simulation (M&S), user manuals, war games			15. NUMBER OF PAGES 31	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT	

**A Concise User's Guide to
ISAAC_FL:
ISAAC's Mission-Fitness Landscape
"Mapper" Program (U)**

Andy Ilachinski
Systems and Tactics Team (Operating Forces Division)
Center for Naval Analyses
(703) 824-2045
ilachina@cna.org

CNA

What is ISAAC?

➔ *An adaptive-personality-driven local rule- and agent-based "toy model" of combat ("Combat from the ground up")*

- Basic element is an ISAACA = ISAAC Agent
 - Represents primitive combat unit (infantryman, tank, etc.)
- Each ISAACA is equipped with:
 - **Doctrine:**
 - A default *internal rule set* specifying how to act in a generic environment
 - A *hierarchical rule set*, consisting of orders passed down echelon via a dynamic C² topology
 - A *global rule set* that determines combat attrition and reinforcement
 - **Mission:** Goals directing behavior
 - **Situational Awareness:** Sensors generating an internal map of environment
 - **Adaptability:** An internal mechanism to alter behavior and/or rules

CNA

[2]

ISAAC (Irrducible Semi-Autonomous Addaptive Combat) is a simple multiagent-based "toy model" of land combat designed to illustrate how certain aspects of land combat can be viewed as emergent phenomena resulting from the collective, nonlinear, decentralized interactions among notional combatants. ISAAC takes a *bottom-up, synthesist* approach to the modeling of combat, vice the more traditional top-down, or reductionist approach, and represents a first step toward developing a complex systems theoretic analyst's toolbox for identifying, exploring, and possibly exploiting emergent collective patterns of behavior on the battlefield. This model was developed as part of a recently completed project, sponsored by the Marine Corps Combat Development Command (MCCDC), that assessed the general applicability of "complex systems theory" to land warfare (see [1] and [2]).

Models based on differential equations homogenize the properties of entire populations and ignore the spatial component altogether. Partial differential equations--by introducing a physical space to account for troop movement--fare somewhat better, but still treat the agent population as a continuum. In contrast, ISAAC consists of a discrete heterogeneous set of spatially distributed individual agents (i.e., combatants), each of which has its own characteristic properties and rules of behavior. These properties can also change (i.e., adapt) as an individual agent evolves in time.

The basic element of ISAAC is an ISAAC Agent (or ISAACA), which represents a primitive combat unit (infantryman, tank, transport vehicle, etc.) that is equipped with the following characteristics (see [3] for additional details):

- **Doctrine:** a default local-rule set specifying how to act in a generic environment
- **Mission:** goals directing behavior
- **Situational Awareness:** sensors generating an internal map of environment
- **Adaptability:** an internal mechanism to alter behavior and/or rules

In ISAAC, the "final outcome" of a battle -- as defined, say, by measuring the surviving force strengths -- takes second stage to exploring how two forces might "co-evolve" during combat.

Modules Making up ISAAC

PROGRAM	DESCRIPTION
ISAAC_CE	The fully interactive (multi-squad version of the) <i>Core Engine</i> that includes all of the features described in [3]
ISAAC_SQ	The <i>single-squad</i> version of the core engine that can be used to display files generated by ISAAC_GA
ISAAC_GA	A stand-alone <i>Genetic Algorithm Evolver</i> that uses the core engine to evolve personalities "best-fit" to perform a specified mission
ISAAC_PM	A stand-alone <i>Parameter-Space Mapper</i> that uses the core engine to "map-out" the behavior over a user-defined two-dimensional slice of ISAAC's total N-dimensional phase-space
ISAAC_FL	A stand-alone <i>Mission-Fitness Landscape Mapper</i> that uses the core engine to "map-out" the behavior over a user-defined d-dimensional slice of ISAAC's total N-dimensional phase-space
ISAAC_PB	A stand-alone program that can be used to <i>Play-Back</i> previously recorded runs; in particular, all of the *.out files on the distribution diskette [4]

CNA

[3]

The ISAAC "toy-world" system currently consists of five separate, but interwoven, programs: an interactive *Core Engine* that incorporates all of the behavioral and dynamical features that are described in [3]; a standalone *Genetic Algorithm Evolver* that uses a slightly older version of ISAAC to evolve force characteristics that are "best-fit" for performing user-defined missions (ISAAC_GA); a *Single-Squad* version of the core engine that can be used to run and display data files generated by ISAAC_GA (ISAAC_SQ); a *Parameter-Space Mapper* that also uses a slightly older version of ISAAC to map out the behavior of a system over a two-dimensional "slice" of ISAAC's total N-dimensional phase space (ISAAC_PM); and an enhanced *Mission-Fitness Landscape Mapper* program (ISAAC_FL) that allows the user to simultaneously sample d-dimensional hypervolumes with 15 separate fitness measures. A sixth program, ISAAC_PB, can be used to "play-back" the recorded runs described in [3] and stored in [4].

The program that is in the shaded box--ISAAC_FL [5]--is the focus of this brief.

Appendix B of this CAB describes a minor patch to ISAAC_CE (version 1.8.4b)

Four Classes of Run "Modes"

- **Interactive Mode**

- No learning
- Fixed set of local CA-like rules applied at each time step

- **Data Collection Mode**

- Time-series of various changing quantities
- Keep track of mission "success" measures

- Behavioral profiles on "mission fitness landscapes"

- **Genetic Algorithm "Evolver" Mode**

- Fixed set of adaptive personalities
- Personalities designed ("evolved") prior to run-time

- **Adaptive Learning Mode**

- Real-time adaptive learning strategies
- ISAACAs "discover" new rules as scenario develops

CNA

[4]

ISAAC is designed to allow the user to explore the evolving patterns of macroscopic behavior that result from the collective interactions of individual agents, as well as the feedback that these patterns might have on the rules governing the individual agents' behavior. ISAAC can currently be run in four different "modes":

- *Interactive Mode*, in which ISAAC's core engine is run interactively using a fixed set of rules. This mode, which allows the user to make 'on-the-fly' changes to the values of any (or all) parameters defining a given run (including the "decision-making personality" of individual ISAACAs), is particularly well suited for quickly and easily playing simple "What if?" scenarios. This purely graphical run mode is also useful for interactively "searching" for interesting emergent behavior.
- *Data-Collection Mode*, in which the user can (1) generate time series of various changing quantities describing the step-by-step evolution of a battle, and (2) keep track of certain measures of how well mission objectives are met at a battle's conclusion. ISAAC_FL allows the user to explore d-dimensional hypervolumes of ISAAC's N-dimensional parameter space.
- *Genetic Algorithm "Evolver" Mode*, in which a genetic algorithm is used to breed a personality for one side that is "best" suited for performing some well-defined mission against a fixed personality (and force disposition) for the other. This mode illustrates how programs such as this can eventually be used to evolve real-world "tactics" and "strategies."

A fourth mode, planned for future releases, is an adaptive mode in which individual ISAACAs learn from their experience during a given run.

Data Collection

Class of data	Appropriate "flag"	Associated Output files
Force sizes	stat_flag	STATS_1.dat
ISAACA interpoint distance distributions	interpoint_flag	STATS_2.dat -- STATS_6.dat
ISAACA:enemy-flag interpoint distance distributions	interpoint_flag	STATS_7.dat & STATS_8.dat
ISAACA neighbor-number distributions	neighbors_flag	STATS_14.dat -- STATS_19.dat & STATS_21.dat
ISAACA cluster-size distributions	cluster_1_flag & cluster_2_flag	STATS_10.dat -- STATS_13.dat
Center-of-mass positions	center_mass_flag	STATS_20.dat
Spatial entropy	entropy_flag	STATS_9.dat

¹ This refers to variables appearing in the statistics parameters section of ISAAC's data input file; see Contents of ISAAC's Data Input File in [3]

CNA

[5]

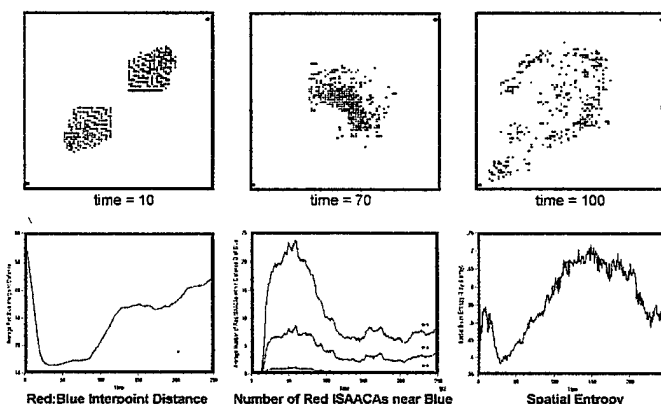
ISAAC is equipped with a rudimentary data collection capability. Specifically, ISAAC's *Core Engine* (i.e., the program ISAAC_CE) has facilities to calculate seven basic classes of information (each as a function of time; see discussion on pages 135-140 in [3]):

- Class 1: Force sizes
- Class 2: ISAACA interpoint distance distributions
- Class 3: ISAACA neighbor-number distributions
- Class 4: ISAACA:enemy-flag interpoint distance distributions
- Class 5: ISAACA cluster-size distributions
- Class 6: Center-of-mass positions
- Class 7: Spatial Entropy

Data collection is enabled by setting the stat_flag variable appearing in ISAAC's input data file equal to 1 (see Statistics Parameters section in Contents of ISAAC's Data Input File). The calculation of each of the above classes of statistics is toggled by individual parameter "flags" appearing in the statistics parameters section of ISAAC's input data file.

The total output of ISAAC's data collection routines (assuming all statistics flags are set equal to 1) is distributed among 21 consecutively numbered files STATS_1.dat through STATS_21.dat. The slide above provides a brief description of their contents. A complete listing of the data fields appearing in each of these output files is given in Appendix E: STATS_X.dat Data Fields in [3].

Data Collection: Sample



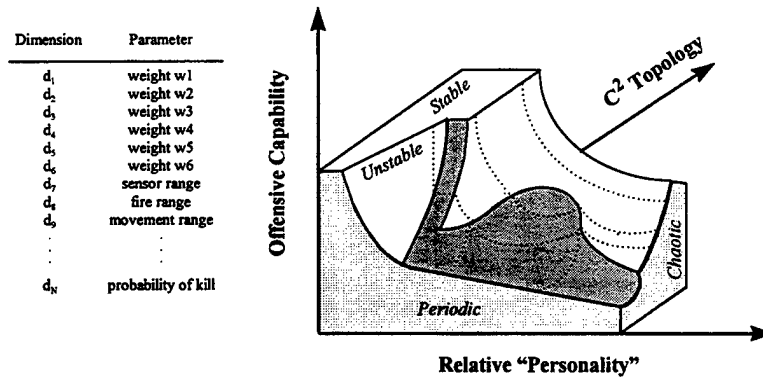
CNA

Consider the fairly complex ISAAC evolution depicted in this slide. There are 200 ISAACs per side and both sides are relatively aggressive: red's combat threshold is equal to -3 and blue's to -6.

One can see that the behavior of these two "personalities" proceeds in essentially four stages. The first stage ($t = 1$ through $t = 20$) consists of an initial internal jostling for position (on both sides) and a steady march toward the enemy corner. The second stage (between $t = 25$ and $t = 70$) consists of "close combat" within a tight central cluster of closely packed red and blue ISAACs. The third stage (between $t = 80$ and $t = 120$) is marked by a relatively rapid "expansion" of forces outward from the central region of the battlefield. The fourth, and final, stage (for $t > 140$) consists mostly of small local skirmishes that are distributed throughout most of the battlefield.

The three plots along the bottom provide a sampling of the kind of information contained in the statistics output files STATS_1.dat through STATS_21.dat (for details see pages 140-145 in [3]): (1) The left-most plot shows the average distance between red and blue ISAACs as a function of time. The curve starts out at a distance $d \sim 57$ that is equivalent to the initial separation distance between the starting "boxes" of the red and blue forces. As the two forces move toward their enemy's flag and thus approach one another, the average inter-force distance steadily diminishes, reaching a minimum as the two forces "collide" near the center of the battlefield. (2) The middle plot shows the average number of red ISAACs that are within a distance $D=1$ (lower curve), $D=3$ (middle curve) or $D=5$ (upper curve) of blue ISAACs. The largest relative numbers, of course, are found between the times $t \sim 25$ and $t \sim 80$ during which the scenario includes some very close combat near the central region of the battlefield. (3) The right-most plot shows the spatial entropy as a function of time. Spatial entropy is here computed by partitioning the 80-by-80 battlefield into an 8-by-8 array of 10-by-10 sub-blocks (see **Spatial Entropy** in [3]).

N-Dimensional Combat "Phase Space"



CNA

[7]

Assume, for sake of argument, that each of the parameters that appears in ISAAC's input data file under the heading "ISAACA Parameters" represents a single independent "degree-of-freedom" (or axis) in the total ISAACA parameter-space. Even if we were to do away with all parameters having to do with communications, all command and control functions, notional defense, fratricide and reconstitution, that still leaves us with a roughly 28-dimensional parameter space: 12 parameters for defining the individual components of the personality weight vector (6 alive + 6 injured) + 5 parameters for defining various ranges + 6 parameters for defining the 6 threshold conditions (3 alive + 3 injured) + 2 parameters for defining an ISAACA's single-shot probability and the maximum number of enemy targets that it can simultaneously target. Even granting that our assumption that each of these parameters can be treated as an independent parameter is obviously false (many parameters are interrelated and not all parameters are equally "important" in defining an ISAACA's overall behavior), an ISAACA's genome clearly "lives" in a very large dimensional space. ISAAC_FL is designed to allow the user to sample over d -dimensional hyper-volumes of ISAAC's ostensibly N -dimensional phase space.

Mission-Fitness Measures

Mission Objective "Primitive"	Description
m_1	minimize time to goal
m_2	minimize red (i.e. friendly) casualties
m_3	maximize blue (i.e. enemy) casualties
m_4	maximize red-to-blue (i.e. friendly-to-enemy) survival ratio
m_5	minimize red (i.e. friendly) center-of-mass distance to blue (i.e. enemy) flag
m_6	maximize blue (i.e. enemy) center-of-mass distance to red (i.e. friendly) flag
m_7	maximize number of red (i.e. friendly) within an SC-defined distance of the blue (i.e. enemy) flag
m_8	minimize number of blue (i.e. enemy) within an SC-defined distance of the red (i.e. friendly) flag
m_9	minimize number of red fratricide "hits" (i.e. on friendly side)
m_{10}	maximize number of blue fratricide "hits" (i.e. on enemy side)

$$F = \alpha_1 m_1 + \alpha_2 m_2 + \dots + \alpha_{10} m_{10}$$

CNA

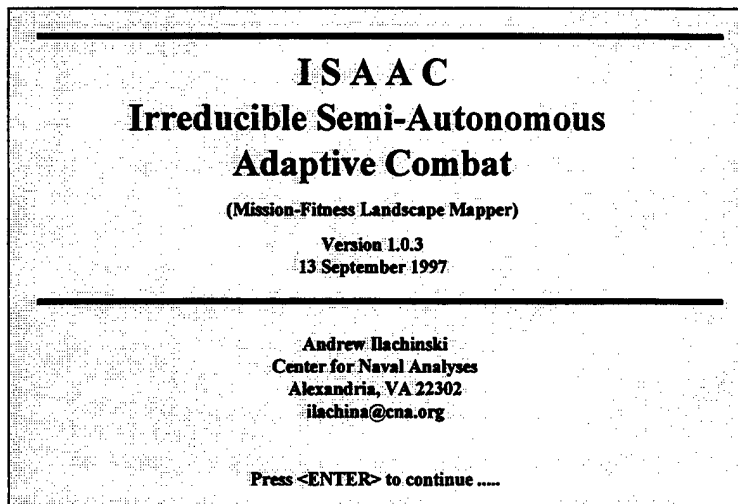
[8]

The objective--or *mission fitness*--is a measure of how well red ISAACAs have performed a user-defined mission. Typical missions might be to "get to blue flag as quickly as possible," "minimize red casualties," "maximize the ratio of blue to red casualties," and so on, or some combination of these.

More specifically, the user can assign up to ten weights ($a_1, a_2, \dots, a_{10}$) to represent the relative degree of importance afforded to a particular mission objective "primitive," m_i .

At this introductory level, of course, the list of mission primitives is still fairly short and simple, though it is flexible enough to enable the user to define many non-trivial objectives. The actual mission objective, or fitness function, F , is a weighted sum of mission primitives.

In later versions of ISAAC, the mission fitness function will be enhanced in at least two ways: (1) it will be linked with ISAAC's built-in data collection capability (so, for example, a SC can choose to look for behaviors leading to low/high spatial entropy as a mission primitive), and (2) the user will be able to define arbitrary mission fitness measures using a script language.



CNA

[9]

This is the opening screen of the program ISAAC_FL, containing the name of the program, build version and date of compilation. The program can be run either in DOS (by typing ISAAC_FL, followed by <ENTER>), or as a DOS-program in Windows 3.1/95.

The maximum allowable values of some pertinent parameters in v1.0.3 are as follows:

- *Battlefield size* < 80
- *Sensor / Fire range* < 20
- *Number of ISAACAs* < 150
- *Initial Conditions* < 100

The next screen--containing prompts for input and output data file names--is accessed by hitting the <ENTER> key.

Input/Output Files

- **Input**

- ISAAC.dat (user defined)
- FITNESS.dat (user defined)

- **Output**

- FIT_AVE.dat (user defined)
- FIT_DEV.dat (user defined)
- FIT_HIST.dat (user defined)
- F[X]A_TAG.dat; X = 1,2,...,15 (automatic)
- F[X]D_TAG.dat; X = 1,2,...,15 (automatic)

CNA

[10]

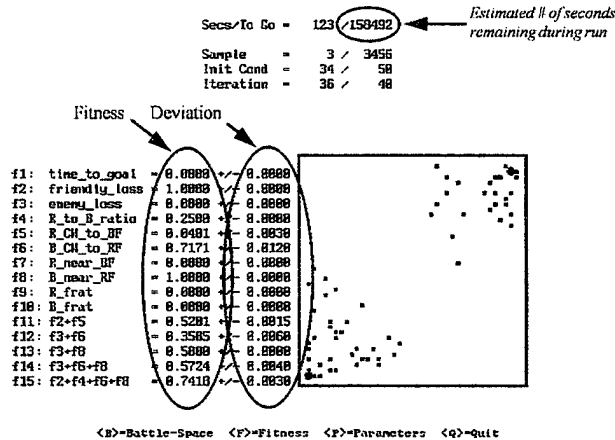
After displaying the opening screen, ISAAC_FL prompts the user for the names of *five* files, two of which are input data files, and the remaining three of which are output data files:

- ISAAC.dat is the default name of the file that contains a truncated version of ISAAC's input data file (see the section **Contents of ISAAC's Input File** in [3])
- FITNESS.dat is the default name of the file containing various parameters that control the calculations to be performed by ISAAC_FL during the given run
- FIT_AVE.dat is the default name of the file that contains a complete list of mission-fitness averages computed during the run
- FIT_DEV.dat is the default name of the file that contains a complete list of mission-fitness deviations¹ computed during the run
- FIT_HIST.dat is the default name of the file that contains the distribution of fitness averages and deviations, sorted according to the number of bins specified in the file FITNESS.dat

The 30 additional "tag" files (not all of which need be open for a given run)-- F1A_TAG.dat, F2A_TAG.dat,...--contain parameter listings filtered for specific fitness-average or fitness-deviation "windows." For example, F1A_TAG.dat may be defined (by software flags defined in FITNESS.dat) to record parameter listings only when f_1 is greater than 0.9.

¹ The "deviation" of mission-fitness f refers to *mean absolute deviation* $\delta f = N^{-1} \sum_j |f_j - \langle f \rangle|$.

ISAAC FITNESS-LANDSCAPE MAPPER Version 1.0.3



CNA

[11]

Once the user has selected the name of both input files and the parameter-mapper's output data file, ISAAC_FL runs through its initialization routine and displays the main graphics page. A sample graphics page is shown in this slide. Note that this figure assumes that the hot-keys 'B' (for Battle-Space), and 'F' (for Fitness) have both been pressed.

Notice that the display is broken up into four separate regions:

- A *banner-display* region, located at the top of the display and containing a large bold font, that identifies the program and release version.
- A text-based *progress-report* region, located directly beneath the banner-display region, that provides an update of the progress made thus far during the run: Sample refers to the current sample number S being processed (expressed as a fraction of the total number of samples), Iteration refers to the current initial condition C (expressed as a fraction of the total number of initial conditions that the program will average the fitness value over, C/C_{total}), Time refers to the current time step t of the sample that is being run for the C th initial condition for the S th sample (expressed as a fraction of the maximal allotted time for this run, t/t_{total}). The number of seconds thus far elapsed during the run, and an estimate of the number of seconds remaining, appears at the top of the report region.¹
- A *fitness-measure-report* region, appearing to the bottom left of the battlefield, that lists the fitness measures (+/- absolute deviations) of the previous sample run.
- A "hot-key" menu region, appearing at the bottom of the display, that contains a short menu of "hot keys" that the user can use to interrupt a run at any time to perform a variety of functions: "B" (for Battle-Space) toggles the graphics display of the notional battlefield, "F" (for Fitness) toggles the display of the mission-fitness values, "P" (for Parameters) toggles a display of red's parameter values, and "Q" (for Quit) closes all output data files (saving intermediate results) and quits the program.

¹ This estimate is, initially, rather poor but steadily improves as the run proceeds

FITNESS.dat

- **Fitness Landscape Parameters**
- **Penalty weights**
- **Termination Parameters**

CNA

[12]

FITNESS.dat is similar to the input files PHASE.dat and GA.dat, used by ISAAC_PM.exe and ISAAC_GA.exe, respectively (see pages 152 and 179 in [3]).

It consist of three categories of parameters:

- *Fitness Landscape Parameters*, that define the d-dimensional hypervolume that ISAAC_FL will sample ISAAC's behavior on. This section also includes parameters fixing the number of initial conditions to be averaged over for a given run, specifying the number of bins to be used for histogram counts and thresholds to be used to "tag" certain subsets of ISAAC's parameter space.
- *Penalty Weights*, that define the 15 mission-fitness measures that ISAAC_FL will be keeping track of during the run
- *Termination Parameters*, that specify the termination conditions. These are essentially the same as discussed on pages 182-184 in [3].

Note that to reduce ISAAC's effective dimensionality, the following symmetry conditions are imposed on the red ISAACAs' weight set:

- (1) $w_2 = w_1$
- (2) $w_4 = w_3$
- (3) $w_{\text{injured}} = w_{\text{alive}}$

A Concise User's Guide to ISAAC_FL: ISAAC's Mission-Fitness Landscape "Mapper" Program

Fitness Landscape parameters															
num_initial_conds	50														
max_time_to_goal	40														
penalty_power	2														
TAG_fitness_ave_flag	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0
TAG_fitness_dev_flag	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0
fitness_AVE_min_for_TAG	.9	.9	.9	.9	.9	.9	.9	.9	.9	.9	.9	.9	.9	.9	.9
fitness_AVE_max_for_TAG	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
fitness_DEV_min_for_TAG	.4	.4	.4	.4	.4	.4	.4	.4	.4	.4	.4	.4	.4	.4	.4
fitness_DEV_max_for_TAG	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
number_of_bins_for_histogram	20														
max_value_for_AVE_histogram	1														
max_value_for_DEV_histogram	.5														
number_of_red_forces	35	35	1												
weight_w1/w1_AIR/IR	-100	100	5												
weight_w2/w2_AIR/IR	-100	100	5												
weight_w3_BV	0	0	1												
weight_w4_BV	0	100	5												
sensor_range	1	7	3												
fire_range	3	3	1												
advance_threshold	1	2	1												
cluster_threshold	5	5	1												
combat_threshold	-15	0	3												
single_shot_prob	.05	.05	1												
max_engage_num	6	6	1												

... (continued)

CNA

[13]

num_initial_conds = total number of randomized initial spatial configurations of red and blue ISAACs that will be averaged over in calculating a mission fitness for a given personality (typically, 25 - 100).

max_time_to_goal = maximum number of iteration steps allowable per each run of the evolution (typical values for battlefield sizes of ~ 80-by-80 are between 100-150 steps).

penalty_power = the power *n* used in defining the fitness function, *f*, for each of the ten mission primitives (see figure 68 in [3]). The value of *penalty_power* effectively determines how rapidly *f* falls off from its maximal to minimal value (*n*=1 yields a linear fall-off, *n*=2 yields a quadratic fall-off, and so on).

TAG_fitness_ave_flag = 1 if fitness averages are to be "tagged" and stored in a separate file. There is one flag value for each of 15 mission fitness measures.

TAG_fitness_dev_flag = 1 if fitness deviations are to be "tagged" and stored in a separate file. There is one flag value for each of 15 mission fitness measures.

fitness_AVE_min_for_TAG = minimum value for the "window" of values for which the fitness averages will be tagged.

fitness_AVE_max_for_TAG = maximum value for the "window" of values for which the fitness averages will be tagged.

fitness_DEV_min_for_TAG = minimum value for the "window" of values for which the fitness deviations will be tagged.

fitness_DEV_max_for_TAG = maximum value for the "window" of values for which the fitness deviations will be tagged.

number_of_bins_for_histogram = the number of bins into which the (min, max) tag window will be partitioned in order to keep track of the number of times the fitness averages (or fitness deviation) falls within a given range.

max_value_for_AVE/DEV_histogram = the upper limit value for keeping a histogram record.

The three columns for each of the remaining 12 entries define the lower limit, upper limit, and the total number of samples for each entry. Thus, for example, the entry "*sensor_range* 1 7 3" means that ISAAC_FL will sweep through sensor ranges $r_s = 1, 3, 5, 7$.

```

.....
*
*      penalty weights (0-100)
*
*.....
w1_time_to_goal      10 0 0 0 0 0 0 0 0 0 0 0 0 0 0
w2_friendly_loss     0 10 0 0 0 0 0 0 0 0 10 0 0 0 10
w3_enemy_loss        0 0 10 0 0 0 0 0 0 0 0 10 10 10 0
w4_red_to_blue_survival_ratio 0 0 0 10 0 0 0 0 0 0 0 0 0 0 10
w5_friendly_CM_to_enemy_flag 0 0 0 0 0 10 0 0 0 0 0 10 0 0 0
w6_enemy_CM_to_friendly_flag 0 0 0 0 0 0 10 0 0 0 0 0 10 0 10
w7_friendly_near_enemy_flag 0 0 0 0 0 0 0 10 0 0 0 0 0 0 0
w8_enemy_near_friendly_flag 0 0 0 0 0 0 0 0 10 0 0 0 10 10 10
w9_red fratricide_hits 0 0 0 0 0 0 0 0 0 10 0 0 0 0 0
w10_blue fratricide_hits 0 0 0 0 0 0 0 0 0 0 10 0 0 0 0
*.....
*
*      termination parameters
*
*.....
termination_code?      4
flag_containment_range 10
containment_number      5
red_CM_to_BF_frac      .5

```

CNA

[14]

The "Penalty Weights (0-100)" section contains 15 columns of weights. There is one column for each of the 15 mission-fitness measures that ISAAC_FL can simultaneously keep track of. Each weight represents the relative degree of importance afforded to a particular mission objective "primitive" (see slide 8) and is a number between 0 and 100.

termination_code?: this software flag controls how the first mission-primitive (i.e. time to goal) is calculated. It can be assigned one of four integer values: 1, 2, 3 or 4. If *termination_code?* = 1, a run terminates when the first red ISAACA reaches the blue flag. If *termination_code?* = 2, a run terminates when the number of red ISAACAs within a range $R = \text{flag_containment_range}$ (see below) exceeds the threshold $N = \text{containment_number}$ (see below). If *termination_code?* = 3, a run terminates when the position of the red force's center-of-mass is closer to the blue flag than a threshold distance (defined by *red_CM_to_BF_frac*; see below). If *termination_code?* = 4, a run will terminate when the number of iterations $t = \text{max_time_to_goal}$.

flag_containment_range: sets a range around either the red or blue flags (depending on the values of other variables) which is used to count the number of ISAACAs near a flag. For example, if the relative weight for maximizing the number of red ISAACAs near the blue flag is nonzero (i.e., if the value of the mission primitive *w7_friendly_near_enemy_flag* > 0), the value of *flag_containment_range* sets the pertinent range from the blue flag.

containment_number: if the termination flag is set for terminating a run when the number of red ISAACAs within a range $R (= \text{flag_containment_range})$ exceeds a certain threshold N -- i.e., if *termination_code?* = 2; see above -- N is specified by the variable *containment_number*.

red_CM_to_BF_frac: if the termination flag is set for terminating a run when the position of the red force's center-of-mass is closer to the blue flag than a threshold distance D -- i.e., if *termination_code?* = 3; see above -- D is specified by the variable *red_CM_to_BF_frac*.

FIT_AVE.dat (Sample)

```
start date: 09/04/97
start time: 17:35:18

Associated files...

ISAAC input file:      isaac.dat
FITNESS input file:    fitness.dat
FITNESS AVE output file: fit_ave.dat
FITNESS DEV output file: fit_dev.dat
FITNESS HIST output file: fit_hist.dat

Fitness Landscape 'Sweep' Ranges...

      min    max    N
weight_w1/w3_AR/IR:  -100  100  5
weight_w2/w4_AR/IR:  -100  100  5
weight_w6_BF:         0   100  5
sensor_range:         1     7   3
combat_threshold:     -15    0   3
```

... (continued)

CNA

[15]

FIT_AVE.dat is ISAAC_FL's output data file that keeps track of each of the 15 mission-fitness measures defined in the "Penalty Weights" section of FITNESS.dat (see slides 13 and 14), averaged over $N = \text{num_initial_conds}$ number of initial conditions.

The file opens by recording the starting date and time and the name of all other input and output data files associated with this run. It also reminds the user of the parameter labels and ranges over which the run is sweeping over. Note that this is essentially a truncated version of the last 12 entries of the "Fitness Landscape Parameters" section of FITNESS.dat.

FIT_AVE.dat (Con't)

sample	N	w1	w2	w5	w6	rs	rf	ADV	CLS	COM	PS	RM	f1	f2	f3	...	f15
1	35	-100	-100	0	0	1	3	1	5	-15	0.0500	6	0.0000	1.0000	0.0000	...	0.7365
2	35	-100	-100	0	0	1	3	1	5	-10	0.0500	6	0.0000	1.0000	0.0000	...	0.7418
3	35	-100	-100	0	0	1	3	1	5	-5	0.0500	6	0.0000	1.0000	0.0000	...	0.7434
4	35	-100	-100	0	0	1	3	1	5	0	0.0500	6	0.0000	1.0000	0.0000	...	0.7421
5	35	-100	-100	0	0	3	3	1	5	-15	0.0500	6	0.0000	1.0000	0.0000	...	0.7424
6	35	-100	-100	0	0	3	3	1	5	-10	0.0500	6	0.0000	1.0000	0.0000	...	0.7421
7	35	-100	-100	0	0	3	3	1	5	-5	0.0500	6	0.0000	1.0000	0.0000	...	0.7422
8	35	-100	-100	0	0	3	3	1	5	0	0.0500	6	0.0000	1.0000	0.0000	...	0.7436
9	35	-100	-100	0	0	5	3	1	5	-15	0.0500	6	0.0000	1.0000	0.0000	...	0.7423
10	35	-100	-100	0	0	5	3	1	5	-10	0.0500	6	0.0000	0.9989	0.0000	...	0.7422
11	35	-100	-100	0	0	5	3	1	5	-5	0.0500	6	0.0000	1.0000	0.0000	...	0.7438
12	35	-100	-100	0	0	5	3	1	5	0	0.0500	6	0.0000	0.9989	0.0000	...	0.7419
13	35	-100	-100	0	0	7	3	1	5	-15	0.0500	6	0.0000	1.0000	0.0000	...	0.7434
14	35	-100	-100	0	0	7	3	1	5	-10	0.0500	6	0.0000	0.9989	0.0000	...	0.7436
15	35	-100	-100	0	0	7	3	1	5	-5	0.0500	6	0.0000	1.0000	0.0000	...	0.7429
...																	
3453	35	100	100	0	100	7	3	1	5	-15	0.0500	6	1.0000	0.4429	0.3812	...	0.6474
3454	35	100	100	0	100	7	3	1	5	-10	0.0500	6	1.0000	0.4286	0.4096	...	0.6467
3455	35	100	100	0	100	7	3	1	5	-5	0.0500	6	1.0000	0.4404	0.3804	...	0.6472
3456	35	100	100	0	100	7	3	1	5	0	0.0500	6	1.0000	0.4372	0.2876	...	0.6303
...																	
simulation completed																	
start date: 09/06/97																	
start time: 12:42:35																	
elapsed time: 155237 seconds																	

CNA

[16]

Following a short listing of pertinent values summarizing the scenario and input/output data files for a given run (see previous slide), FIT_AVE.dat next provides a complete record of the average mission-fitness values for each of the 15 measures defined in "Penalty Weights" section of FITNESS.dat (see slides 13 and 14).

Each *Sample* refers to one run with a given set of red parameter values, averaged over $N = \text{num_initial_conds}$ number of initial conditions. The column labels are self-explanatory: N = number of red ISAACAs, w1 = weight w₁ (=relative weight for moving toward alive red), ... ADV = ADVANCE constraint threshold, CLS = CLUSTER constraint threshold, COM = COMBAT constraint threshold, and so on.

A given run terminates either when the outcome of all parameter N-tuples are recorded, or when the user quits the program by hitting the 'Q' key.

FIT_HIST.dat (Sample)

Averages	bin	x_min	x_max	f: 1	2	3	4	5	...	15
	1	0.0000	0.0500	935	6	2032	654	360	...	0
	2	0.0500	0.1000	42	696	129	66	216	...	0
	3	0.1000	0.1500	44	18	24	0	0	...	0
	4	0.1500	0.2000	12	0	40	0	127	...	0
	5	0.2000	0.2500	17	0	106	444	787	...	0
	6	0.2500	0.3000	8	0	351	1052	402	...	0
	7	0.3000	0.3500	2	12	153	443	504	...	0
	8	0.3500	0.4000	6	250	239	221	1060	...	0
	9	0.4000	0.4500	5	626	342	501	0	...	0
	10	0.4500	0.5000	7	339	40	75	0	...	720
	11	0.5000	0.5500	10	59	0	0	0	...	0
	12	0.5500	0.6000	6	19	0	0	0	...	16
	13	0.6000	0.6500	7	17	0	0	0	...	952
	14	0.6500	0.7000	12	42	0	0	0	...	514
	15	0.7000	0.7500	5	38	0	0	0	...	1254
	16	0.7500	0.8000	15	52	0	0	0	...	0
	17	0.8000	0.8500	31	237	0	0	0	...	0
	18	0.8500	0.9000	42	112	0	0	0	...	0
	19	0.9000	0.9500	79	120	0	0	0	...	0
	20	0.9500	1.0000	2171	813	0	0	0	...	0

... (continued)

CNA

[17]

FIT_HIST.dat is ISAAC_FL's output data file that keeps track of the number of times each of the mission-fitness values ($f_1, f_2, \dots, f_{15}$) falls within a range of values (between 0 and a user-specified maximal value; see *max_value_for_AVE/DEV_histogram*). The number of bins is also user-selected (see slide 13).

FIT_HIST.dat (Con't)

Deviations	1	0.0000	0.0250	2791	725	1997	1501	2678	...	1908
	2	0.0250	0.0500	231	403	144	680	701	...	1304
	3	0.0500	0.0750	0	699	108	228	77	...	244
	4	0.0750	0.1000	84	551	648	831	0	...	0
	5	0.1000	0.1250	50	910	445	212	0	...	0
	6	0.1250	0.1500	47	162	112	4	0	...	0
	7	0.1500	0.1750	0	6	2	0	0	...	0
	8	0.1750	0.2000	38	0	0	0	0	...	0
	9	0.2000	0.2250	33	0	0	0	0	...	0
	10	0.2250	0.2500	39	0	0	0	0	...	0
	11	0.2500	0.2750	19	0	0	0	0	...	0
	12	0.2750	0.3000	17	0	0	0	0	...	0
	13	0.3000	0.3250	14	0	0	0	0	...	0
	14	0.3250	0.3500	12	0	0	0	0	...	0
	15	0.3500	0.3750	13	0	0	0	0	...	0
	16	0.3750	0.4000	4	0	0	0	0	...	0
	17	0.4000	0.4250	14	0	0	0	0	...	0
	18	0.4250	0.4500	9	0	0	0	0	...	0
	19	0.4500	0.4750	13	0	0	0	0	...	0
	20	0.4750	0.5000	28	0	0	0	0	...	0

CNA

[18]

The second part of FIT_HIST.dat consists of the number of times each of the mission-fitness deviations ($\delta f_1, \delta f_2, \dots, \delta f_{15}$) falls within a range of values (between 0 and a user-specified maximal value; see *max_value_for_AVE/DEV_histogram*). The number of bins is also user-selected (see slide 13).

F1A_TAG.dat (Sample)

Sample	F1	N	w1	w2	w5	w6	rS	rF	ADV	CLS	COM	PS	EN
17	1.0000	35	-100	-100	0	20	1	3	1	5	-15	0.0500	6
18	1.0000	35	-100	-100	0	20	1	3	1	5	-10	0.0500	6
19	1.0000	35	-100	-100	0	20	1	3	1	5	-5	0.0500	6
20	1.0000	35	-100	-100	0	20	1	3	1	5	0	0.0500	6
33	1.0000	35	-100	-100	0	40	1	3	1	5	-15	0.0500	6
34	1.0000	35	-100	-100	0	40	1	3	1	5	-10	0.0500	6
35	1.0000	35	-100	-100	0	40	1	3	1	5	-5	0.0500	6
36	1.0000	35	-100	-100	0	40	1	3	1	5	0	0.0500	6
37	0.9200	35	-100	-100	0	40	3	3	1	5	-15	0.0500	6
49	1.0000	35	-100	-100	0	60	1	3	1	5	-15	0.0500	6
50	1.0000	35	-100	-100	0	60	1	3	1	5	-10	0.0500	6
51	1.0000	35	-100	-100	0	60	1	3	1	5	-5	0.0500	6
52	1.0000	35	-100	-100	0	60	1	3	1	5	0	0.0500	6
55	0.9000	35	-100	-100	0	60	3	3	1	5	-5	0.0500	6
65	1.0000	35	-100	-100	0	80	1	3	1	5	-15	0.0500	6
66	1.0000	35	-100	-100	0	80	1	3	1	5	-10	0.0500	6
67	1.0000	35	-100	-100	0	80	1	3	1	5	-5	0.0500	6
68	1.0000	35	-100	-100	0	80	1	3	1	5	0	0.0500	6
70	0.9000	35	-100	-100	0	80	3	3	1	5	-10	0.0500	6

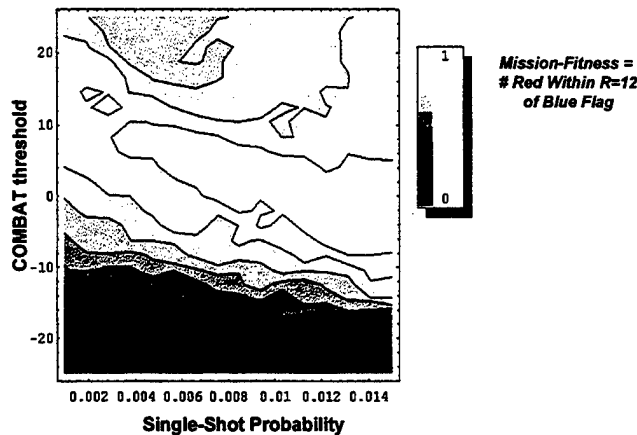
CNA

[19]

This slide shows a few entries from a sample F1A_TAG.dat output file. This file--which is open and written to only if the appropriate tag-flag is set (see *TAG_fitness_ave_flag*; slide 13)--records red ISAACA parameter values *only* if the average value of mission-fitness f_1 is within a user-specified window of values (that are defined by the values *fitness_AVE_min_for_TAG* and *fitness_AVE_max_for_TAG*; see slide 13). In this particular case, F1A_TAG.dat has "tagged" parameter values yielding an f_1 that is greater than 0.9.

ISAAC_FL can, in general, open up to 30 such "tag" files: 15 "average" tag files (F[X]A_TAG.dat, where X can be 1, 2, ..., 15) and 15 "deviation" tag files (F[X]D_TAG.dat, where X can be 1, 2, ..., 15).

Sample 2D Output from ISAAC_PM



CNA

[20]

This slide shows an example of the two-dimensional output of ISAAC_PM (see page 159 in [3]), which is to be contrasted with the figure produced with the output of ISAAC_FL, shown on the next slide. Both sides start out with 50 ISAACAs each, and red's mission objective is to get as many red ISAACAs within a distance $D=12$ of the blue flag as possible (each run is averaged over 50 initial conditions). Blue defends with a personality defined by $w=(0,10,0,10,0,0)$; i.e., blue "sees" only the enemy and does not distinguish between alive and injured reds. Blue's combat threshold is equal to negative 3 so that blue is fairly aggressive, and blue's sensor and fire ranges are $r_s = 4$ and $r_f = 3$, respectively. Red "attacks" with weight vector $w=(10,40,10,40,0,50)$. Red's sensor and fire ranges are equal to the blue forces'. Combat ensues for a maximum 125 iteration steps on a size 50-by-50 notional battlefield.

The x-coordinate for this slide is red's single-shot probability (with the value of x ranging from 0.001 to 0.015; note that $P_{\text{blue}} = 0.003$) and the y-coordinate is the "combat threshold" (with the value of y ranging from -25 to 25). The slide shows a contour plot of the fitness $f(x,y)$ as a function of (x,y) , in which lighter shades of gray represent high fitness (near $f \sim 1$) and darker shades represent low fitness (near $f \sim 0$).

The figure shows an interesting non-monotonic behavior. For single-shot probabilities $P_{\text{red}} > 0.003$, red performs this particular mission "best"--as defined by the lighter colored regions-- by being neither too aggressive (with large negative values of combat threshold) nor too timid (with large positive values of combat threshold).

Sample Mathematica Commands

READ IN DATAFILE:

```
dat7fla1:=ReadList["fla_1.txt",{Number, Number, Number, Number}]
```

PLOT FITNESS:

```
FitnessPlot3D[labelx_, labely_, labelz_, plotlabel_, boxsize_, thresh_,
data_ /; MatrixQ[data, NumberQ] && Length[First[data]]==4,
fun_, opts_] :=
Module[{res, coord},
coord=Map[Last, DeleteCases[data, {_,_,_,x_ /; x<thresh}]],
min=Min[coord],
max=Max[coord],
Show[Graphics3D[Map[{fun[(Last[#]-min)/(max-min)],
Cuboid[Take[#,3],Take[#,3]+{boxsize,boxsize,boxsize}]}],&,
DeleteCases[data, {_,_,_,x_ /; x<thresh}]],
Lighting->False,
DefaultFont->{"Helvetica-Bold",10}, PlotLabel->plotlabel,
Axes->True,AxesLabel->{labelx, labely, labelz},
opts]]
```

CNA

[21]

This slide shows two simple *Mathematica*¹ commands that can be used to read-in a truncated form of FIT_AVE.dat or FIT_DEV.dat (see slides 15-16) and render a three dimensional gray-scaled plot of either fitness averages or absolute deviations.

The ReadList command assumes that the user has created, from (FIT_AVE.dat or FIT_DEV.dat) a stand-alone file with the entries "x y z f" or "x y z df". This can be done easily with the *Filter AutoWizard* in Microsoft's Excel, for example, to isolate desired subsets of the ISAAC_FL's complete data output listing.

FitnessPlot3D function entries are as follows:

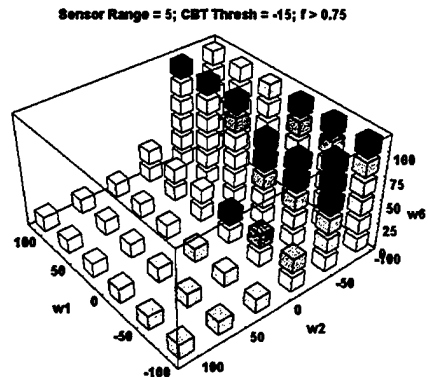
- **labelx** = the label for the plot's "x"-axis
- **labely** = the label for the plot's "y"-axis
- **labelz** = the label for the plot's "z"-axis
- **plotlabel** = the label for the entire plot
- **boxsize** = the size of each volume element $\Delta V(x,y,z)$ centered on (x,y,z)
- **thresh** = the threshold value that f must exceed in order to be plotted

The next slide shows the result of running this sample command:

```
FitnessPlot3D["w1","w2","w6","Sensor Range=5;CBT Thresh=-15;f >
0.75",15,.75,dat7f2a9,GrayLevel,ViewPoint->{-2.117,1.723,2.000}]
```

¹ Mathematica, A System for Doing Mathematics by Computer, Version 2.2.1, Wolfram Research Incorporated, 1991.

```
dat7f2a9:=ReadList["f2a_9.txt",{Number, Number, Number, Number}]
FitnessPlot3D["w1", "w2", "w6","Sensor Range = 5; CBT Thresh = -15; f > 0.75",
15,.75,dat7f2a9,GrayLevel,ViewPoint->{-2.117,1.723,2.000}]
```



CNA

[22]

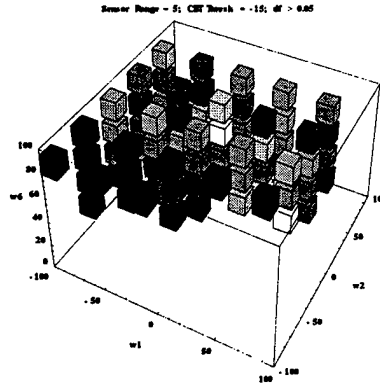
This slide shows an example of a three-dimensional plot that can be produced with the output of ISAAC_FL using the Mathematica code appearing on slide 21. The scenario is the same as for the previous slide, except that combat unfolds on a smaller lattice (40-by-40), there are 35 ISAACAs per side and red's mission objective is to minimize friendly casualties.

The plot generalizes ISAAC_PM's 2D density plot to 3D: each volume element $\Delta V(w_1, w_2, w_6)$ is gray-scaled according to the value of the mission-fitness of the center site of that element, $f(w_1, w_2, w_6)$. Lighter shades of gray represent high fitness (near $f \sim 1$) and darker shades represent low fitness (near $f \sim 0$). A given volume appears in the above plot only if the value of its associated mission fitness is greater than 0.75. Similar plots can, of course, be obtained for any combination of the parameter values that ISAAC_FL sweeps over.

Plots such as the one shown on this slide allow the user to tell, at a glance, which regions of ISAAC's fitness landscape have low or high fitness.

A Concise User's Guide to ISAAC_FL: ISAAC's Mission-Fitness Landscape "Mapper" Program

```
dat7f2d9:=ReadList["f2d_9.txt",{Number, Number, Number, Number}]
FitnessPlot3D [{"w1", "w2", "w6", "Sensor Range = 5; CBT Thresh = -15; df > 0.05",
15, .05, dat7f2d9, GrayLevel, PlotRange->{{-100,100},{-100,100},{0,100}}]
```



CNA

[23]

This slide shows a 3D gray-scaled plot of absolute deviations for the same scenario as in the previous slide. Both sides start out with 35 ISAACAs each, and red's mission objective is to get as many red ISAACAs within a distance $D=12$ of the blue flag as possible (each run is averaged over 50 initial conditions). Blue defends with a personality defined by $w=(0,10,0,10,0,0)$; i.e., blue "sees" only the enemy and does not distinguish between alive and injured reds. Blue's combat threshold is equal to negative 3 so that blue is fairly aggressive, and blue's sensor and fire ranges are $r_s = 4$ and $r_f = 3$, respectively. Red "attacks" with weight vector $w=(w_1, w_1, w_2, w_2, 0, w_6)$, where w_1, w_2 , and w_6 are the x, y and z axes, respectively. Red's sensor and fire ranges are equal to the blue forces'. Combat ensues for a maximum 50 iteration steps on a size 40-by-40 notional battlefield.

Each volume element $\Delta V(w_1, w_2, w_6)$ is gray-scaled according to the value of the mission-fitness *deviation* of the center site of that element, $\delta f(w_1, w_2, w_6)$. The deviation of mission-fitness f refers to *mean absolute deviation*

$$\delta f = N^{-1} \sum_j |f_j - \langle f \rangle|,$$

where f_i is the mission-fitness recorded for the i^{th} initial condition.

Lighter shades of gray represent high fitness deviations ($\delta f \gg 0$) and darker shades represent low fitness deviations ($\delta f \sim 0$). A given volume appears in the above plot only if the value of its associated fitness deviation is greater than 0.05. Similar plots can, of course, be obtained for any combination of the parameter values that ISAAC_FL sweeps over.

Plots such as the one shown on this slide allow the user to tell, at a glance, which regions of ISAAC's fitness landscape have small or large variability.

What Are the Next Steps?

- **User-Defined Mission Parameters**

- Link with data-collection routines

- **Coevolution**

- Red and blue fitness landscapes as functions of each other

- **Visualization**

- Animation

CNA

[24]

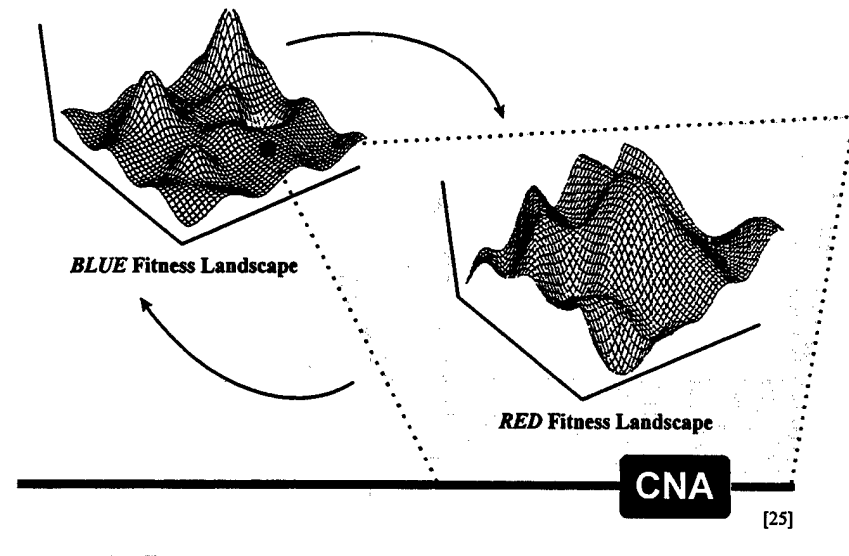
As has been repeatedly stressed in ISAAC's reference manual [3] and by the author in both formal and informal briefings, ISAAC is an interim version of a "work in progress." It represents but a skeletal fragment of what will eventually become the core engine of a much more sophisticated set of tools.

From a design perspective, planned future enhancements include more realistic offensive and defensive capabilities, an enhanced internal "value-system," added environmental realism, and endowing individual ISAACAs with both a memory of, and a facility to learn from, their past actions (using both neural-network and reinforcement learning techniques). A graphical-user-interface to facilitate interactive experimentation is also planned. Reference [3] provides a more thorough discussion of future enhancements to ISAAC's core-engine.

From a data collection perspective, three of the most important steps to be taken next are summarized on this slide:

1. To link ISAAC's growing built-in data-collection routines (that include force sizes, interpoint distance distributions, cluster-size distributions, estimates of spatial entropy and so on) with mission-objectives and fitness measures. The ultimate goal is to provide the user with a special "script" language so that the user can define specific mission objectives.
2. To provide measures of the *coevolution* of red and blue forces; that is, measures of the way in which the one side's fitness landscape deforms the other's (see next slide).
3. To enhance ISAAC's visualization tools. ISAAC_FL itself is but a "data collection" tool, designed to record certain measures of behavior on d-dimensional hypervolume subsets of ISAAC's N-dimensional phase space. What is needed is a set of visualization tools that would make it easier to detect emerging patterns. Various forms of color-coding, animation (in which a series of related 3D plots are rapidly projected on top of one another), perhaps even sound, can all be used to simultaneously render--and interpret--the information content of several ISAAC-dimensions.

Coevolution



ISAAC has heretofore been endowed with a purely static mission-fitness landscape. While programs such as ISAAC_PM (see pages 146-160 in [3]) and ISAAC_FL, described in this CAB, facilitate an exploration of portions of this landscape, the landscape itself is assumed to be fixed and unchanging. Once the blue personality is chosen it is clamped, and the fitness landscape represents only red's "ability" to perform specific missions against a specific blue force.

In reality, of course, combat forces must *coevolve*. Just as the fitness landscape of one species continually responds to the changing landscapes of other species in a natural ecology, so too in combat each side continually responds to how the other side responds to its actions. Blue responds to red's actions, red adapts to blue's response, blue responds to red's adaptation, and so on. Over time, co-evolution alters not just the species (or the personality mix of each combat force) but the manner in which species interact with another. (Appendix B in reference [3] discusses a future enhancement to ISAAC's genetic algorithm (GA) search capability that borrows from Hillis' co-evolving "host/parasite" GA strategy, and that may set up a perpetual "arms-race" between red and blue forces.)

This slide illustrates the fact that with each point in blue's fitness landscape is associated an entire fitness landscape for red. In order to really understand the dynamics of this complicated system, one must therefore first develop measures that quantify the relationship between the co-evolving red and blue fitness landscapes; i.e. develop measures that quantify how one landscape deforms the other. One can then address such important fundamental questions as "*What must blue do (i.e. where on its landscape should it move) in order to compel red to fall into a local minimum?*", and vice-versa.

Appendix A: Contents of Distribution Disk (ISAAC_FL:CNA 06-970271)

- ISAAC_FL
 - Version 1.0.3 of ISAAC's Mission-Fitness Landscape Mapper Program
- ISAAC.dat
 - Sample input data file defining red and blue ISAACA parameters
- FITNESS.dat
 - Sample input data file defining ISAAC_FL's run parameters

CNA

[26]

The distribution disk for ISAAC's *Mission-Fitness Landscape Mapper* program (ISAAC_FL) contains three files:

- ISAAC_FL.exe, which is version 1.0.3 of the DOS executable
- ISAAC.dat, which is a sample ISAAC input data file defining a simple scenario in which blue defends goal with $w = (0,10,0,10,0,0)$
- FITNESS.dat, which is a sample parameter input file for ISAAC_FL

Appendix B: Update to ISAAC_CE

- **Version 1.8.4b:**

- Fixes bug that did not allow interactive *increase* in number of red or blue ISAACAs
- Adds *line-of-sight (LOS)* calculation to terrain scenarios

- **Terrain LOS**

- Add parameter `LOS_flag` (= 1 if LOS used, else = 0) to GENERAL BATTLE PARAMETERS section of ISAAC.dat
- Use of LOS is indicated on lower-left of battlefield in main graphics display
- LOS option added to "Combat Parameters" option menu

CNA

[27]

Version 1.8.4b fixes one reported bug and adds an important feature.

The previous release (v.1.8.4) appears to "freeze" when the user interactively (i.e. via the "On-the-Fly Parameter Change" option; page 90 in [3]) *increases* the number of red or blue ISAACAs. Version 1.8.4b now permits this change without freezing.

The addition to ISAAC's core engine is a user-specified flag (=LOS_flag) that toggles an internal *line-of-sight* (LOS) calculation when terrain is present. An entry for this flag appears immediately after `terrain_flag` in the GENERAL BATTLE PARAMETERS section of ISAAC.dat (page 56 in [3]).

If `LOS_flag` = 0, the calculation proceeds as in earlier versions. In particular, this means that while terrain blocks ISAACA movement it does not hinder ISAACAs from "sensing" one another across the terrain.

If `LOS_flag` = 1, ISAAC_CE now updates moves by first determining if the line-of-sight between two ISAACAs is "blocked" by terrain. The algorithm is fairly simple, though it induces a small performance hit (roughly 5-10%). All ISAACAs that are within one lattice site of each other sense each other, regardless of terrain. If the line-of-sight between ISAACA X and Y intersects a terrain site, the X and Y do not sense each other. Communication (toggled via the `COMM_flag`; see page 75 in [3]) is not affected.

A line-of-sight (LOS) option has also been added to the "Combat Parameters" option menu (see page 92 in [3]).

References

- [1] *Land Warfare and Complexity, Part I: Mathematical Background and Technical Sourcebook*, CNA CIM-461, July 1996, Unclassified
- [2] *Land Warfare and Complexity, Part II: An Assessment of the Applicability of Nonlinear Dynamics and Complex Systems Theory to the Representation of Land Warfare*, CNA CRM-68, July 1996, Unclassified
- [3] *Irreducible Semi-Autonomous Adaptive Combat (ISAAC): An Artificial-Life Approach to Land Warfare*, CNA CRM 97-61.10, June 1997, Unclassified
- [4] *ISAAC Software v1.8.4*, CNA 06-970174, August 1997, Unclassified
- [5] *ISAAC_FL: ISAAC's Mission-Fitness Landscape Mapper Program v1.0.3*, CNA 06-970271, September 1997, Unclassified

CNA

[28]