SpiBus

SPI's Newsletter of Microcomputer/Conflict Simulations Applications ☐ Nr. 001

Edited by Dave Robertson and Redmond Simonsen

SpiBus is a newsletter designed to serve those simulation gamers interested in microcomputer application and those microcomputer hobbyists who have an interest in simulations. We intend to give SpiBus its first public exposure in the pages of MOVES *in order to interact with the audience before casting it in its final form. Any comments or suggestions will be welcomed and should be addressed to SpiBus, care of SPI. —RAS*

First Things First

Some may wonder "why computers?" The answer to the question comes from the readers of *Moves* and *S&T* — most people in the wargaming hobby are (a) fascinated by computers and (b) plan to buy a personal microcomputer. Nearly everyone in the hobby is interested in the effect that small computers will have on game design and development. Even now SPI is using its own microcomputer to aid the development of some new games.

Now that microcomputers are generally available to the public for less than a thousand dollars, there has been a rising demand for these machines. For example, Radio Shack has sold several hundred thousand of its TRS-80 system. Most new purchases of a microcomputer will soon tire of the mindless games that have been devised for the initiate to computers. Some people will even tire of the sameness of a Startrek simulation. Consequently, there is demand for new blood in the realm of computer games.

On the other hand, the wargamer may be looking for a way to speed up a tedious game or find a way to add realism to a game without rendering it unplayable. How many gamers have tired of repeated die rolls or accidently missed a crucial step in the sequence of play? Is there a way to have a self-prompting aid to play or to avoid the tiring recording of dozens of die rolls?

The needs of both these groups can be satisfied by the microcomputer. It's just a matter of getting the word out to people so they can share the know-how and build up the state of the art of mating the computer to the manual simulation wargames. This is where *SpiBus* comes in. This is the forum where the state of the art can be developed.

Decisions Decisions

Before starting this column, Simulation Publications made some decisions based on the feedback from *S&T* and *MOVES*. The decisions were as follows:

1. All computer programs sponsored by SPI would be in BASIC, the most commonly understood computer language.

2. The machine for which they would be designed would be the Radio Shack 16-K Level II TRS-80.

3. The wargame programs would be game assistance programs — i.e., they would help the players play the game manually — rather than games run exclusively on and in the machine. This means that the map, counters, and extensive rules still are alive and well, and that all the games can be played "by hand."

4. The programs themselves would exactly model the manual games — that is the odds, steps, sequence of play, and general feel of the game would be just the same for the manual and machine versions.

5. The ideas, plans, and techniques of the programs would be freely available for all to see — no secrets folks.

Why

If these points seem arbitrary, think a little and you can see why these decisions were made. First, BASIC is the best language for several reasons. Most people know it or can learn it in a short time. All the microcomputers worth their salt have a version of BASIC in ROM (it comes built in, so to speak). The facility for coping with text (called "strings") in BASIC is quite good. Indeed, the biggest selling personal computer has BASIC.

BASIC is also important because its widespread acceptance assures that whoever has a program can inspect the programming techniques, complain, offer suggestions, and thereby improve the state of the art. This is the big feature — everyone can learn what can be done with wargames together.

The Radio Shack TRS-80 Level II was the choice of a particular machine because it is the most widely owned personal computer in the United States (and probably overseas too). This machine is also fairly good in terms of the various things it can do. We have not talked to the people at Radio Shack or been paid to say so, we just believe that for now, anyway, it is the best all around machine to use for wargaming purposes. In any event, the programs written for wargames on the TRS-80 will most likely run — with some slight modification — on such other machines as PET or APPLE II.

The programs that SPI will sponsor will be game assistance programs. They will prompt the players so nothing will be forgotten. The programs will do tedious record keeping functions — so all the second counters in *Stonewall* could be omitted. Game assistance programs will not render the games obsolete or make them inaccessible to someone without a computer. This also lets SPI take the games it has and write programs so those with computers will have some help in playing what they already have. Also, it will interest computer owners in wargames and, hopefully bring some fresh new faces into the wargaming hobby. The maps, counters, and rules will stay much as they are now — the graphics really are special playing aids anyway. The result is that the games will become more playable with a computer.

The programs must model the games and not distort the game in any way. This is the cardinal rule for the time being. It will assure that the game plays the same by hand as by machine. The key is developing the techniques to do this and this is one of the main topics of this column, where we hope to give a lot of practical advice on how to write your own game assistance programs to help you play the wargames you already own. Resolving combat, keeping track of the current casualities and unit strengths, victory points, and other factors is where a computer can do all the work to help you enjoy the game more. It hardly needs to be said but here it is — the computer can speed up play in some instances by eighty percent. Yes, a four hour game can play in one hour (sounds like a T.V. ad).

Money for You

Right now, SPI has only a couple of programs to sell in the immediate future, but it has plans to expand the number available. In fact it is willing to pay money to anyone who can write acceptable programs for resale to the gaming public. The present design criteria have been outlined by Redmond Simonsen, and his specifications are set forth at length at the end of this column. The key to these specifications is *quality* — we're looking for really good programs that have some unique ways of doing the job. Also they must pass the usual SPI testing procedure for its games — they have to work under actual game playing conditions. The key here is "actually works" — this is not simple. The important thing to remember is that people make mistakes, and when they do machines must be slightly forgiving. This means that when a player makes a mistake typing in some command to the computer, then the computer program must allow the stupid or careless a chance to correct the error so he will not have to be stuck with the consequences of the mistake for the rest of the game session. In fact, though, this is not too difficult to do, but it has to be done for any kind of mistake. This omission can be very frustrating when a single error cannot be recovered and the time invested in the game becomes a sad tragedy for the player who has inadvertently put himself out of ammunition or supply. A later edition of this column will discuss some of the ways to insure error recovery in spite of any screw up.

Range Algorithm

A number of gamers with personal computers have written SPI to learn if there is a simple way to have the computer compute the range between the hexes on an SPI style map. We have told them that we were coming up with it and now this dark mystery can be revealed for all to see.

The problem here is that Pythagoras's technique does not work too well on a hex grid map when the distance you are looking for is the number of hexes between two given hexes counting one hex but not counting the other hex. In the first place what numbers would you start with if you did take the square root of the squares of the difference of the rectangular coordinates (which is the way to do it on a Cartesian coordinate system like we all learned in plane geometry or algebra). The next problem is the fact that a hexagonal map is *not* a Cartesian coordinate system anyway — it has a different purpose (making the distances between hexes easy to count when moving on the diagonal). There is a way that is not too straightforward, but it works every time without fail. For anyone playing a game like *Tank, UpScope,* or *Mech War 77,* this simple program finds the range by inputting only the hex numbers printed on official SPI maps. The algothrim is this:

```
10 INPUT "ENTER THE TWO HEX NUMBERS,
WITH A COMMA BETWEEN THEM LIKE
THIS—1134, 1122"; A,C
20 B = INT(A/2):D = INT(C/2);
E = (ABS(A – C) + ABS(B – D) + ABS
(A + D – B – C))/2
30 E = (E + ABS(A + D – B – C) + ABS
(E – ABS(A + D – B – C)))/2
40 PRINT "THE RANGE IS";E;"HEXES"
50END
```

The possibilities for this little algorithm are profound if you have a fettish for double blind search games and you find dummy counters, and other traditional ways of dealing with the problem, give away too much since you know the enemy is at least one of the dummies. Naval and search games can become really interesting now because there is one feature of naval warfare that no manual game has ever been able to duplicate — unknown damage. With this algorithm you can search in the blind and fire without knowing what's there or whether you hit it provided you can get a combat results table procedure into the machine too.

Agincourt

SPI has made a game assistance program for its colorful *Agincourt* game published last year. The program overcomes many of the problems of the game which can be somewhat tedious at times. For example, the program prompts the players to perform every phase in the sequence of play so that nothing is left out. Also the program prompts for every bit of information in the combat sequences so that careless errors (a frequent cause of arguments) cannot accidently occur. The program keeps track of victory points, game turns, and losses that are recorded as step reductions by a set of forty counters on a chart — thus rendering the chart unnecessary. The game takes about five hours to play by hand, but seems to play out in two hours or so with the game assistance program. The map, movement, counters and all rules are in effect and used as normal — the machine rolls the die, does the bookkeeping, and prompts the players for every step in the game.

Instead of describing the program further, we have reproduced the first draft of the instruction leaflet that will accompany the computer tape soon available for purchase from SPI.

Agincourt Specs
Outline of Program

Lines:	Function:	Subroutines
1-10	Initialize arrays and matrices	
11	French movement of fugitives	137
12-27	French movement and English archery — recycles for each impulse and asks for fire per unit; asks for Target, Range, and #Volleys — records losses (if any), resolves die roll cross references table, and reduces arrow supply (see note on unit names)	136,90, 102,93, 94,88,145
28-29	French Morale Phase — standing on dead, etc. for each unit so that no step gets left out and rolls on appropriate column of the Morale Table and prints results and tells how many fugitives to put on the map — this is idiot-proof	144,125
30-35	French Leader Combat Phase — user inputs name and strength of leaders and then inputs strategy for each one for each of the 5 phases of leader combat — it rolls the die and prints the results by referencing the leader CRT in the LC(X,Y,Z) matrix	86,87 145,85, 107
36-43	French Melee Phase — user inputs units names and the machine asks about the various facts that it does not know that affect melee, and then prints the results and records the loses	102,104, 106,95 141,142
44-45	English Fatigue Phase— the units that were meleed this turn are forced to undergo a morale check that folows the sequence of lines 28-29 and subroutines for melee	
47-65	English movement and French archery fire — like lines 12-27 except uses subroutines for short bows etc.	91,90,104 93,94,88 145
66-68	English Morale Phase — like lines 28-35	
69-73	English Leader Combat Phase — like 30-35	
74-78	English Melee Phase	
79-81	French morale check for fatigue like 44	
82	Army Morale Phase (every 5 game turns)	
84	Game turn change	

Significant Variables

A$ — Inquiry variable (yes or no, unit names, etc.)

DI — Die roll result from RND (100) statements

EC — Cumulative English line losses for victory points

FI — Unit "number" identification counter

FL — Cumulative French casualty line losses for victory (like EC)

IM — Impulse number for movement Phases counter

IL — Strength of inferior leader in leader combat

MD — Melee strength in melee combat phase

MA — Melee strength of unit in melee phase

SL — Strength of superior leader in leader combat (like IL)

U$ — Unit number counter in arrow fire phases

X — Loop variable

Z$ — A null string of several hundred characters to erase with ? @

Arrays and Matrices

U$(45) — The unit names read from data in lines 146-148

LB(2,6,6,4,)* — Long bow CRT (8.7)

SB(1,5,4,3)* — Short bow CRT

CB(1,4,2,2)* — Cross bow CRT

MC(7,11) — Melee CRT (9:5)

MT(10,4) — Unit Morale Table (10.4) (the "0" element of this matrix is used)

F$(2) — East and West text for output of fugitive movement phase

AS(10,2) — Long bow arrow supply and variable supply counter

S(5,2) — Short bow arrow supply

C(3,2) — Cross bow arrow supply

L(45) — Current line strength

LC(5,5,3) — Leader CRT (13.3)

M(45) — Units melee strength (as printed and per rules)

FS(8) — French arrow supply

MU(10) — Names of units that defended in melee for fatigue use

*These matrices are set up as follows — (TARGET TYPE, RANGE, #VOLLEYS, ODDS). Target Type: 1 = men at arms, 2 = bowmen, 3 = archers. After TT, R, and V are in the last item referenced to die roll to get result of strikes "S" in subroutine.

Agincourt's User's Leaflet

A. This is the user's leaflet for the *Agincourt* game assistance program. It is written for the Radio Shack TRS-80 Level II 16-K microcomputer. It will run on a larger machine with say 32-K of RAM, of course.

B. Loading. The file name for the program is "A" using the CLOAD command. After loading, try executing a print memory command ?MEM. You should have 4079 bytes of free memory after loading. If this is not what you have left, try reloading the program with a different volume setting.

C. Running. If you are satisfied that the program is properly loaded (try listing it to see that no disc system commands showed up through load error, then you are ready to try a test run to become familiar with how the program can help speed up the accuracy and pace of the game. Do not try to play until you have tried all of the steps in the program and have a feel for just what will happen and the kind of mistakes you are apt to make when you try to use the program.

D. Initial Wait. When you enter RUN, nothing will happen as far as you can tell for several minutes. Actually, the machine is recording all the tables and unit information from DATA statements written at the end of the program. When all the data is read and the various matrices are set up in RAM, there are only a few hundred of bytes left. If you listed the program first you probably noticed the DEFINT A-Z statement so you know that there is integer precision only—which means saving memory and allowing fastest execution time.

French Fugitive Movement Phase. Soon the screen of the CRT will clear and the first phase will be prompted. This is the French Fugitive Movement Phase. The machine will ask you to enter the number of fugitives to move. (On the first game turn enter "0" of course.) Here you must look at the map and count the number of French fugitive counters present and enter that number. The machine will print the direction (east or west) that each counter must move during this phase (there's a ½ chance either way). When the fugitives have been moved, press ENTER to signal the computer that you have executed this phase.

E. Operation. In order to operate the program only a couple things may throw a user off the track of successful operation. The way to avoid problems is simple: when a unit name is asked for be sure to type in at least the first 5 letters of the unit name and the last letter of the unit name (as printed on the screen). If the unit is a cavalry or archer unit, that is type "B" or "C" when the case arises. This is important because some units on both sides have the same name except for the fact that the unit is a cavalry or archer unit (e.g., BAR and BAR-C). As far as the questions that are to be answered "yes" or "no" are concerned, they can be answered with a "y" or "n" since the machine is only concerned with the first letter.

F. Error recovery. Error Recovery can be accomplished fairly easily, but the best practice is to avoid input errors. Some errors may be corrected without hitting "BREAK". For example when an improper archery fire target is input enter "0" for the number of volleys. If a simple error cannot be recovered in this manner then hit "BREAK" and note the line number. If the line number is greater than 84, enter "RETURN" to exit the present subroutine and proceed to enter null strings until the phase ends. Then re-enter that phase by hitting "BREAK" and then "GOTO" the line number where that phase began. If a line strength for a unit or an arrow supply is incorrect then a more cumbersome procedure must be used. Hit "BREAK" and then enter the new values for the arrow strength or line strength for the unit by directly entering a statement such as "L(22) = 1" (without the quotes of course). Then hit "CONT", and things should be all right. To find the particular element of the L(x) matrix consult the roster printed below to find which unit number to use in making the changes. *Never* enter RUN after the program starts because that will clear all the variables and strings. Use the GOTO for moving around in the program to clear up any input errors.

G. Theory of operation. The guts of a program like *Agincourt* is its ability to pick out the input text of the unit names and make adjustments (bookkeeping entries) to the various rosters and tables in its memory. This is done by comparing the input text names to a list of the unit names stored in the matrix of one dimension set up in U$(x). Each unit name has a numbered location corresponding to "x". When the machine uses the two search loops in the subroutines at lines 102 and 104, it will store the number in the U$(x) which is corresponding to the input unit name. The machine then uses this number to reference the melee strengths at M(x) and line strength at L(x) or (after a little arithmetic) the arrow supply at AS(x,1). After the operation the machine (which knows the unit's identification) prints the name U$(x) back at the user with the result for example. The last character of the names of the bowmen and calvary units ("B" or "C") is used a marker for whether the unit is men at arms, cavalry, or bowmen in the Arrow Fire Phase, the Morale or Melee Phase. The "3" at the end of some of the French unit names is used to let the machine shift two columns on the Morale Table (10.4). The Line Strength Matrix (L(x)) is shifted one value so that a line strength of 1 is handled on the "0" element of the MT(10,x) — which is why X runs from 0 to 4 rather than 1 to 5 (there are five columns on the morale table). ■■

GENERAL SPECIFICATIONS FOR SPI GAME ASSISTANCE PROGRAMS
As of Jan 79

General Description: Game Assistance Programs (GAP's) are to be designed to "computerize" as much as is practicable, some of the more complex and/or bookkeeping-intensive SPI games. They should not be thought of as replacements for the games themselves, but rather as the ultimate player-aid. The target system is the RADIO SHACK TRS-80.

Language and Memory: Programs are to be written in *Radio Shack Basic, LEVEL II* to run on a *16K machine.* Programs should *not* depend upon the use of diskettes, printers or other peripherals.

A. The program should *run* within 16 K

B. The program should contain no machine language subroutines.

Documentation: The program should be fully documented in outline form. Documentation should be typewritten, double-spaced. All operating features must be fully explained in the context of the game and the program (what, where, when, and why). Programs must be referenced by line number and games rules by case number. Note that this documentation (and the program itself) is the basis for the plain-language brochure that will be written to accompany the program to instruct the user.

First Submission: The software designer will submit a written outline of his intended program before actually writing any code. This outline will specify what the program *will* do as well as what it will *not* do (vis-a-vis the game functions). This outline must be complete enough to indicate to the project manager the scope and depth of the treatment. This submission is done on speculation unless otherwise indicated by the project manager.

A number of SPI games are candidates for Game Assistance Programs. *AirWar* and *Agincourt* are currently in work. The following is a list of games up for grabs (you should be extremely well versed in any game you are attempting to program.):

Next War; War in Europe; War in the Pacific; Atlantic Wall.

Please feel free to suggest any others (so long as they're SPI games).

Program Submission: Consists of the following:

1. Program on high quality magnetic tape cassette
2. Program documentation
3. Test data used to de-bug program

The submitted program is subject to acceptance by SPI. It should be free of bugs and ready to run. SPI reserves the right to turn back unacceptable work for correction at no additional fee.

[continued on page 22]

both a function of professional pride and a simple desire to add to the joys of one of the most stimulating and enjoyable hobbies extant. We at SPI are aware of how many people share our view of what we are doing. We feel keenly the desire of the people to take a more active part in the seminal aspects of the hobby and herewith hope to provide them with a direct means of doing so.

REQUESTS FROM THE SPIRIT WORLD

In the past few weeks things have been hectic around SPI's new offices at 257 Park Avenue South. Steve "Donny" Donaldson has started a remake of the old *East is Red* which will probably appear in *S&T*. He is interested in the OBs and relative effectiveness of the Chinese, Mongolian, Vietnamese, Indian, Korean, Japanese, and Taiwanese forces, and in the logistical and amphibious capabilities of both Chinas. Please *do not* send information that can be easily taken from publications like the *Military Balance.*

John Butterfield and Joe Balkoski need blindtesters in April or May to playtest *Stalingrad* and *Trafalgar* respectively.

Dave Isby has been working on *Naval War.* Dave is looking for someone who has practical experience on warships to blindtest. In conjunction with further developments on *Air War,* Dave is looking for pilots or aeronautical engineers who have information on Saab A37 Viggen, Saab A35 Draken, F-100 Super Sabre, North F-104 Starfighter, F-106 Delta Dart, F-8 Crusader, A-6 Intruder, Su-15 Flagon-E, Mig 19 Farmer, Harrier, Mirage F1, Lightning, Hunter, Meteor. Accompanying missiles are: Red Top, AA-5, Falcon, and Swedish RH.

In the next year, SPI will rework *PanzerArmee Afrika.* The new game will use the same system with all state of the art innovations added in, a correct and improved Order of Battle, and superior graphics. Blindtesters are solicited so that the Victory Conditions may be properly balanced. It is the opinion of the SPI staff that this game is an unpolished gem, and we intend to give it wider exposure. Send all applications in care of *SPIRIT.*

World War 3 and *Global War*

These two games suffer from unfinished development, but remain on the bestseller list. We would like to update these games by giving them a thorough overhaul. Therefore, we are looking for a serious individual willing to rewrite the rules and change the system so that the games will be playable and balanced. The counters and the map may *not* be changed, but some charts may. This will require a substantial commitment of time on the part of the person(s) who accepts this task, but there will be cash remuneration. *World War 3* has priority over *Global War,* so if the respondent is undecided, he should go with *World War 3.* Write to "World War 3" or "Global War" in care of *SPIRIT.*

Game Questions and Errata

SPI has an extremely large line of games, and wishes to be able to answer questions and update errata on all products it has published. However, because staff inevitably turns over with the passage of time, we are looking for people who are interested in particular games to answer game questions and compile errata. Any person who wishes to apply for the position of "official" answer man should be forewarned that this requires thorough knowledge of the game system in question. All errata will be sent to SPI, in charge of the Games Questions editor, but all Games Questions will be sent by the Games Question editor to the answer man.

To apply for the position of answer man, send a letter to SPI and mark it "*Attention:* Eric Goldberg, Games Question Editor" to determine whether the game you wish to work on is indeed available. If the game of your choice is on the list below (or you are informed by return mail that we need a person for the position), a sample errata should be collated so the Editor may determine your ability. Remuneration will be in the form of game credit.

The following games are in immediate need of an answer man:

- **World War I Module**
- **Raid!**
- **October War**
- **Red Sun Rising**

Attention Fantasy Role-Playing Gamers

In the upcoming months, SPI will be endeavoring to produce a fantasy role-playing game, to be known as *Dragonslayer.* Due to the nature of the beast, the game will require extensive testing, starting as early as the conceptual stage. We need several good groups of people who will be willing to input into the design and development process. Requirements for this project include a commitment of about ten hours a week, reliability (in keeping in touch with SPI), and several friends with whom the game will be played.

If you are interested in working on this project, please respond to the following questions, and mail the responses to *SPIRIT,* in care of SPI. We will respond to those people we wish to use, by June 30 at the latest.

1. Have you ever played fantasy role-playing games before?
2. Have you ever worked with SPI on a game before publication?
3. If your answer to number 1 was yes, how many hours per week do you devote to fantasy role-playing games?
4. How many reliable role-playing people do you have access to on a once a week basis?
5. Do you prefer to be a player or a moderator (gamesmaster) when playing role-playing games?
6. What do you think is most important in a role-playing game (try to be specific—e.g., playing a role over game systems?
7. Would you prefer to test individual systems (such as magic or combat), or would you prefer to tackle the game as a whole?

Any additional comments would be appreciated, because the responsibility of the designer will be to select those who will do the best testing. The more we know about your preferences and perceptions, the better. However, we do not need long essays on your opinions on role-playing games. Thanks!

SpiBus *[continued from page 20]*

Rights:. SPI acquires *all* rights to all material upon final acceptance. The software designer will attest that the material submitted to SPI is essentially original and specifically developed for SPI and that no competing use or sale will be made by the software designer of the material developed by SPI.

Support: The software designer will remain in contact with SPI and provide answers to any questions SPI may have concerning the program and the impact of any changes in game material. The period of support obligation will extend for six months from the first sale of program to public.

General Structure of Program

A. The program will follow the Sequence of Play of the SPI game without embellishment or modification.

B. The user will be prompted with explicit print statements and interrogatories. The only permissable abbreviations are those used in the manual version of the game.

C. It should be possible for the user to interrupt and recover without having to restart the program from scratch.

In programs that have bookkeeping functions as well as resolution and procedural functions, it should be possible to review data at any time during the SOP.

E. The parts of the Sequence of Play should (when feasible) be treated as virtually independent, modularized "sub-programs" so that each major part of the sequence can be dealt with separately both in the editing and using process.