August 1970

The Use of Interactive Graphics to Reduce and Eliminate Errors

Judith W. Soley
University of Pennsylvania

Follow this and additional works at: http://repository.upenn.edu/cis_reports

Recommended Citation
http://repository.upenn.edu/cis_reports/800


This paper is posted at ScholarlyCommons. http://repository.upenn.edu/cis_reports/800
For more information, please contact repository@pobox.upenn.edu.
The Use of Interactive Graphics to Reduce and Eliminate Errors

Abstract
Intuitively, it would be acceptable that through examination of data and various verification procedures, a human being can eliminate erroneous input readings or calculations. This report is concerned with investigating this aspect through the use of interactive graphics. It is proposed to investigate this aspect through a case study in a competitive game situation.

When taking sonar readings one can anticipate that periodically erratic readings will occur. Thus the data displayed may appear to be inconsistent when visually compared to other historical data. This report proposes a method for eliminating this erroneous data, and the submarine vs. task force game in the Moore School is the vehicle for investigation.

Specific displayed information is made light sensitive. Then, by using the light pen, the player eliminates that information considered to be inconsistent. Thus, a more reliable estimate is created.

Comments

This technical report is available at ScholarlyCommons: http://repository.upenn.edu/cis_reports/800
TECHNICAL REPORT

THE USE OF INTERACTIVE GRAPHICS TO REDUCE AND ELIMINATE ERRORS

by

Judith W. Soley

August 1970

Submitted to the
Office of Naval Research
Naval Applications & Analysis Division
Arlington, Virginia 22217

under
Contract N00014-67-A-0216-0013
Research Project NR 276-007

This document is subject to special export controls and each transmittal to foreign governments or foreign nationals may be made with prior approval of ONR Code 462.

Moore School Report No. 71-01
THE USE OF INTERACTIVE GRAPHICS TO REDUCE
AND ELIMINATE ERRORS

Abstract

Intuitively, it would be acceptable that through examination of data and various verification procedures, a human being can eliminate erroneous input readings or calculations. This report is concerned with investigating this aspect through the use of interactive graphics. It is proposed to investigate this aspect through a case study in a competitive game situation.

When taking sonar readings one can anticipate that periodically erratic readings will occur. Thus the data displayed may appear to be inconsistent when visually compared to other historical data. This report proposes a method for eliminating this erroneous data, and the submarine vs. task force game in the Moore School is the vehicle for investigation.

Specific displayed information is made light sensitive. Then, by using the light pen, the player eliminates that information considered to be inconsistent. Thus, a more reliable estimate is created.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2.0 Description of Use of INCOM - With Illustrations</td>
<td>3</td>
</tr>
<tr>
<td>3.0 Description of the Program INCOM</td>
<td>14</td>
</tr>
<tr>
<td>Bibliography</td>
<td>16</td>
</tr>
<tr>
<td>Appendix A - The Equipment Configuration</td>
<td>17</td>
</tr>
<tr>
<td>Appendix B - The MULTILANG Statement ELISA</td>
<td>20</td>
</tr>
<tr>
<td>Appendix C - Flowcharts</td>
<td>22</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Display of Eight Sonar Readings</td>
<td>5</td>
</tr>
<tr>
<td>Figure 2</td>
<td>An Enlargement of Figure 1 With the Seconds Included in the Times</td>
<td>6</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Three Estimates Based on the Sonar Readings of Figure 1</td>
<td>7</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Display of Twelve Sonar Readings</td>
<td>9</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Two Estimates Based on the Sonar Readings Shown in Figure 4</td>
<td>10</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Display of Sonar Readings Obtained While Ship Was Turning</td>
<td>11</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Two Estimates Based on the Sonar Readings Shown in Figure 6</td>
<td>12</td>
</tr>
<tr>
<td>Figure A.1</td>
<td>Equipment Configuration</td>
<td>18</td>
</tr>
</tbody>
</table>
THE USE OF INTERACTIVE GRAPHICS TO REDUCE AND ELIMINATE ERRORS

1.0 INTRODUCTION

Intuitively, it would be accepted that through examination of data and various verification procedures, a human being can eliminate erroneous input readings or calculations. This report is concerned with investigating this aspect through the use of interactive graphics. It is proposed to investigate this aspect through a case study in a competitive game situation.

The submarine vs. task force game in the Moore School is proposed as a vehicle for investigation.

The game is played by two individuals, each of whom is assigned specific goals, not necessarily known by the other, and such that they must compete to attain their preassigned goals. Normally, these goals will conflict. The attainment of the goals of one individual will cause the failure of the other to attain his. The players, one acting as captain of a submarine and the other as commander of an opposing task force, control the movements of their respective ships through the interaction of an on-line computer system.

The course of the ship being tracked by sonar is displayed by our OPPonent's HISTory program (OPHIS) and which also has the ability to display the following additional options:

1) display the scale factor
2) box the area being displayed
3) indicate the latitude and longitude of the center of the display area
4) the distance in miles of $\frac{1}{2}$ the side of the box
5) display a grid
6) put the direction indicator on the screen
7) indicate the time for each marked position.

We therefore can selectively choose for display any or all of the display options calculated under OPHIS. As under true battle conditions, the real life environment, one can anticipate that periodically erratic sonar readings and/or calculations will occur. Thus the data displayed may reveal itself to be inconsistent when visually comparing it to other historical data displayed. In developing the proper tactics and strategies one might therefore wish to use only a subset of the readings available, that is: that information believed to be valid.

It is therefore readily apparent that the deletion of selected pieces of information, possibly noise and erroneous sonar readings, will necessitate a generation of new ESTimate based on Active sonar (ESTAC) records which should be more reliable in estimating the opponent's course, speed, range, and bearing. This in turn would give the evaluator revised data upon which to reassess the situation and to react accordingly. This is a further example of the benefits of the on-line system.

The Moore School submarine vs. task force game was given the additional ability to eliminate the apparent inconsistent readings and then update the calculations by creating new ESTAC records. An evaluation of the revised available information enhances the players' ability to correctly maneuver his ship and assists the evaluator in his ability in problem solving. This new ESTAC record is now stored in the display file, and can be requested for display at any time.
2.0 DESCRIPTION OF USE OF INCOM - With Illustrations

The question arises: what happens if the data displayed reveals itself to be inconsistent when visually comparing it to the other historical data displayed. When this occurs on shipboard, the evaluator immediately eliminates this reading and makes any necessary decision based on the assumed reliable readings. The program should also have this capability. It is just that ability that it has now been given along with the ability to send the corrected information back to the computer to recalculate the course and speed of the ship and thus the player can now base his new tactics and strategy on this updated information.

The programs have been altered to implement the above. The light pen and two additional pushbuttons have been utilized for this. The specific readings on the display (not the connecting lines) have been made light-sensitive. These readings, when touched with the light pen, blink. This is done to indicate which readings the player intends to eliminate. When making a reading blink, a bit is set to indicate which reading is temporarily eliminated from the new calculation. If the player wishes to change his mind or has eliminated a reading by mistake, he has the ability to include a reading that he has just temporarily eliminated. This is accomplished by pressing pushbutton "7" and then touching any of the blinking readings with the light pen. This will cause any of the blinking readings that are touched with the light pen to stop blinking. These readings will then be included in any new estimate and the total number of readings will be updated. If one touches a reading that was not blinking, nothing will happen until pushbutton "7" is turned off. One can eliminate and
put back readings as often as one pleases. The purpose of the blink is to indicate to the player which readings have been temporarily eliminated.

The following example may serve to better illustrate how the player uses pushbuttons "7" and "8" along with the light pen. Figure 1 is a picture of the actual sonar readings. The same picture is illustrated in Fig. 2 but the scale is greater and the seconds have been included in the times for clarity of the following discussion.

For the following example only these times are to be included in an estimate: 161347, 161432, 161447, 161552. Making sure that pushbutton "7" is off, one takes the light pen and touches all the other readings, specifically, 161317, 161332, 161417, 161452, 161417. If a reading that is to remain in the estimate is touched by mistake, it will be evident, as it will blink. This can be included again in the estimate by turning pushbutton "7" on, taking the light pen and touching the blinking reading which is desired in the estimate. This reading then stops blinking and will now be included in an estimate. When sending this information back to the computer for recalculation one will receive a new estimate based on the chosen four readings. The information is sent to the computer by depressing pushbutton "8". Only those readings not blinking are used in the calculation. The particular points chosen created the ESTO record WA@# (see Fig. 3).

As the original sonar record is 8 readings, there are 256 different possible combinations of those points, using as few as none of them and as many as all of them. For another example, one could choose the first and last points, 161317 and 161552. This gives ESTO record WA@#. 
Figure 1

Display of Eight Sonar Readings
Figure 2: An Enlargement of Figure 1 With the Seconds Included in the Times
ELISA/WHITE/=1615°2,=1614°47,=1614°32,=1614°17,=1614°2,=1614°7,=1613°2,=1613°17
RANGE 28.°7 BEARING 119 TIME 1615°2
COURSE 187 SPEED 08
ESTO RECORD WAS CREATED

ELISA/WHITE/=1615°2,=1614°47,=1614°32,=1614°7
RANGE 28.°7 BEARING 119 TIME 1615°2
COURSE 153 SPEED 11
ESTO RECORD WAS CREATED

ELISA/WHITE/=1615°2,=1613°17
RANGE 28.°7 BEARING 119 TIME 1615°2
COURSE 013 SPEED 014
ESTO RECORD WAS CREATED

OPHIS/WA°8:WA°8:WA°9/ALL/INCL/=1613/1616

Figure 3

Three Estimates Based on the Sonar Readings of Figure 1
For a third estimate, one might choose all 8 points for comparison. This yields ESTO record WA$@W$.

Figure 3 shows the three ESTO records in one picture. One can see how the individual estimates differ. The estimate of 4 readings is 153°, that of the two readings is 013°, and that of all the readings is 187°. In each instance the speed is also different, 11 knots, 4 knots, and 8 knots respectively. Thus the estimated course and speed are dependent on the readings. Since erroneous readings can affect the ESTO record, any tactics or strategies based on an incorrect ESTO record would in turn be an error. For example a weapon firing based on an incorrect ESTO record could easily miss the ship although fired exactly as it was intended.

Additional examples can be seen in Figures 4 through 7. Which readings were used for the estimates can be seen in Figures 5 and 7.

Once having decided which readings we wish to eliminate we turn on pushbutton "8". This sends the updated information back to the computer by causing the communications program INCOM (INteractive COMmunications) to be called in from disk. This automatically creates the necessary program calls which, now using the chosen sonar readings, will calculate a more reliable estimate. Once the information is sent to the computer, control is returned to CONSOL and a player may call upon this new estimate or may continue the game with any of the other available functions.

The player also has the option of not deleting any information. He can continue immediately with the game. This can be accomplished in two ways: he can hit pushbutton "8" without deleting any information; or, as in the original OPHIS, he can return to CONSOL by
Figure 4

Display of Twelve Sonar Readings
Figure 5

Two Estimates Based on the Sonar Readings Shown in Figure 4
Figure 6

Display of Sonar Readings Obtained While Ship Was Turning
Figure 7

Two Estimates Based on the Sonar Readings Shown In
Figure 6
depressing the "interrupt" button. Thus it is not necessary to delete information; this is only an additional feature of the game.
3.0 DESCRIPTION OF THE PROGRAM INCOM

The program Interactive Communication (INCOM) is the communications program between the DEC 338 and the IBM 794. When pushbutton "8" is turned on a routine in OPHIS reads INCOM in from the disk and starts the program as soon as it has been entered.

INCOM begins by transferring the desired information calculated under OPHIS to a buffer NEWEST and then checking if the information will be one or two lines in length. The dataphones are then activated and the interrupt facility is turned on. The main program runs until an interrupt occurs at which point the interrupt is turned off and a JMS $\phi$ instruction is automatically executed. This causes the address of the next instruction to be performed in the main program to be stored in location $\phi$ and the instruction at location 1 to be executed, which is JMP INTRPT. The accumulator and the link are saved and the following possible flags are checked: transmit flag, receive flag, keyboard flag, teleprinter flag, and receive flag. When the interrupt is found, a JMP to a specific routine which handles that flag is executed and the flag is cleared. Upon completion of the service routine, a JMP INTRPT is performed which restores the link and the accumulator, turns the interrupt back on and then executes a JMP* $\phi$ which causes a return to the main program.

Although the interrupt routine checks for five different flags, INCOM is really only interested in two of them: the transmit flag and the receive flag. Associated with these flags are two special locations which are set up for the dataphone communication. These locations are SNDCH, send character, which is associated with the transmit flag and RCVCH, receive character, which is for the receive flag.
For sending information, the location SNDCH contains a pointer to a location which performs a dual purpose. SNDCH contains a pointer to a location which is an entry point for sending a character and it is also an indicator. In order to keep the dataphones active, a character must be transmitted over the dataphone lines every 3 1/3 ms. If SNDCH contains then a null character is transmitted. Therefore the program first checks to see if the location contains . If it does, the null character is transmitted; otherwise control is transferred to the location specified by the contents of SNDCH and the 8 bit character is transmitted. The return occurs the next time a transmit flag interrupt occurs. In the meantime the main program flow continues independently.

In order to receive information, the location RCVCH is utilized. This location contains a pointer to a location which is also used both as an indicator and a subroutine entry point for receiving a character. When there is an incoming character the dataphone receive flag is set. The interrupt routine checks the contents of RCVCH and treats it as an indicator. If this location contains the character is ignored, but if the location is non-zero control is transferred to the location specified by the contents of RCVCH. Bits 4-11 of the accumulator then contain the character received.

When there is no more data transmission, control is returned to CONSOL.
BIBLIOGRAPHY


APPENDIX A

THE EQUIPMENT CONFIGURATION

Figure A.1 illustrates the arrangement of the equipment used to play the game. The system includes an IBM 7040, which acts as the main computer, and a Digital Equipment Corp. (DEC) PDP-8 general purpose computer, which acts as the satellite computer, to monitor the Teletypes.

The main processor (IBM 7040) is a general purpose scientific machine with 32768 words of magnetic core memory. The following units are attached to the 7040 through appropriate Data Channels (IBM 7904's):
a 1402 Card Reader/Punch; 1403 Line Printer; 1301-2 Disk File; 6 type 729II magnetic tape units; and the PDP-8 satellite processor.

The PDP-8 is a one address, fixed word length parallel computer using 12-bit two’s complement arithmetic. One of the two PDP-8’s used in the system acts as the satellite processor and the other as the buffering unit for a DEC 338 CRT display.

Two types of Dataphones are used in the system. Low speed Dataphones are used between the ASR-33 Teletypes and the PDP-8 satellite processing unit and a high speed Dataphone is used between the DEC 338, PDP-8 buffered display and the satellite unit.

A player's command to the system is passed from his Teletype to the satellite processor via a Dataphone and proper interface (DEC 637 or LTO8). After performing simple text handling operations, the satellite sends the information through an interfacing unit (DEC DM03) to the main processor (IBM 7040). After processing, the system's reply to the player travels the same route. The information is transmitted from the 7040 through the interface to the PDP-8, where simple text
Figure A.1  Equipment Configuration
handling may be performed, and then over telephone lines to the player.
APPENDIX B

THE MULTILANG STATEMENT ELISA

The form of the MULTILANG statement is:

ELISA/COLOR//=HHMMSS,,=HHMMSS,== . . .

where COLOR is replaced by the name of the vessel. We use RED or WHITE to indicate the different ships. The start of the individual times is indicated by "/=" and the remaining times are separated by ",=". The "HHMMSS" are the times where each letter indicated a different time digit. For example if the time was 14:04:12 the HHMMSS would be: 0104 0004 0102 and would be three separate words as indicated because the PDP-8 handles only 12-bit words.
THE DISPLAY LIST

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>check sum</td>
<td>7777</td>
</tr>
<tr>
<td>check code</td>
<td></td>
</tr>
<tr>
<td>display options to be used</td>
<td></td>
</tr>
<tr>
<td>parameter to give scale size</td>
<td></td>
</tr>
<tr>
<td>the center latitude</td>
<td></td>
</tr>
<tr>
<td>the center longitude</td>
<td></td>
</tr>
<tr>
<td>1/2 box length</td>
<td></td>
</tr>
<tr>
<td>pointer to the end of the display file</td>
<td></td>
</tr>
<tr>
<td>start of the display file which is a set of commands to the DEC 338 display</td>
<td></td>
</tr>
<tr>
<td>POP</td>
<td></td>
</tr>
<tr>
<td>The end of the display file</td>
<td></td>
</tr>
<tr>
<td>total number of times displayed</td>
<td></td>
</tr>
<tr>
<td>the number of words before the &quot;/=&quot;</td>
<td></td>
</tr>
<tr>
<td>ELISE/WHITE or ELISA/RED</td>
<td></td>
</tr>
<tr>
<td>time list,</td>
<td></td>
</tr>
<tr>
<td>/=</td>
<td></td>
</tr>
<tr>
<td>HH</td>
<td></td>
</tr>
<tr>
<td>MM</td>
<td></td>
</tr>
<tr>
<td>SS</td>
<td></td>
</tr>
<tr>
<td>,=</td>
<td></td>
</tr>
<tr>
<td>HH</td>
<td></td>
</tr>
<tr>
<td>MM</td>
<td></td>
</tr>
<tr>
<td>SS</td>
<td></td>
</tr>
<tr>
<td>,=</td>
<td></td>
</tr>
<tr>
<td>etc.</td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX C

FLOWCHARTS

light pen interrupt

Were pushbuttons 7 or 8 hit?

Yes

Was pushbutton 7 hit?

Yes

Was this point blinking?

Yes

Reset the bit in the time list. Turn the blinker off.

No

Stop the display. Call program INCOM.

Was pushbutton 8 hit?

Yes

Set the bit in the time list. Turn the blinker on.

No

No

No

Return from interrupt

Flowchart of Pushbuttons "7" and "8"
INCOM

Calculate:
Pointer to the end
of buffer: BEND
pointer to the start
of buffer: BSTART
length of information:
COUNT

Are there one
or two lines of
information?

one line

LIN1
-(number of words in line) →
NEWCOUNT

two lines

LIN2
NEWCOUNT[-32
-(number of words in the second line) → NEW2
TWOLNS[NOP

MCOUNT
Move the information
to be sent to the 7040
from where it was in
OPHIS to NEWEST (and
NEW2)

COMM
Activate the communica-
tions. Turn the
interrupt on.

BEGIN
BEGIN

Prepare to send CLEAR INPUT (C I)

CALL SENDWT

Setup:
Pointer to command string: SNDCPT
Number of characters on the line: NOCHLN
Number of the information line: NINFLN
Update the return: SNDCOM

CALL SEND

Return to main program

CALL SENDWT

STATE=1 (filling buffer)
Count of lines during fill: FILNCT
Character count during fill: FLCHCT
Pointer to buffer: FILPFT
Command to append: FBUFF
NOCHLN[1 (one command character)

Return to main program

APP
Pointer to where the information begins: XRL
Update APP
APPCNT-(length of message)
update APP

FILLW[ pointer to message word

CALL FILLW

Update APPCNT

Does APPCNT= \emptyset

No

Yes

CALL FILLFL

one line

one or two lines

two lines

CALL FILLFL

CALL SNDLMO

CALL SENDWT

EXIT:
RETURN TO TCON
Two
lines

Pointer to where the second line of information begins: XR1
Update APP
APPCNT[ - (length of line)
Update APP

FILLW[ pointer to message word

CALL FILLW

Update APPCNT

Does APPCNT = \emptyset ?

Yes

CALL FILLDL

CALL FLPILL

CALL SNDCOM

CALL SENDWT

EXIT: RETURN TO TCON

No
**INTRPT (Interrupt)**

Location \( \text{箸} \) has a pointer to the location where the program was when the interrupt occurred.

**INTERRUPT**

Save the AC
Save the LINK

**Is the transmit flag up?**

- **Yes**
  - **TRFLG**
    - Send the character clear SNDCHC
  - **No**
    - **RCFLG**
      - Set the receive flag to zero
      - RCFLGC[ contents of receive buffer
      - **Does SNDCH = 0?**
        - **Yes**
          - Go to the subroutine indicated by SNDCH
        - **No**
          - Call INTRET

- **No**
  - **Is the receive flag up?**
    - **Yes**
      - With the character in the AC go to the subroutine indicated by RCVCH.
    - **No**
      - **Does RCVCH = 0?**
        - **Yes**
          - Clear receive active
        - **No**
          - Call INTRET
INTRPT (continued)

- Call ITSRREC

- With the clear receive character in the AC, go to active the subroutine indicated by RCVCH.

- If the keyboard flag is up:
  - No: Is the teleprinter flag up?
    - No: Is the receive end flag zero?
      - Yes: Call INTRET
      - No: RCFSEC[1's complement of AC]
        - HLT (the interrupt was not found)
        - Clear flag
  - Yes: Call INTRET
  - Clear flag

- If the teleprinter flag is up:
  - Yes: Call INTRET
  - No: Does RCVCH=0?
    - Yes: Clear receive active
    - No: With the character in the AC, go to the subroutine indicated by RCVCH
      - Call INTRET
INTRET

- Restore the LINK.
- Restore the AC.
- Turn the interrupt on.
- Return to program (i.e. JMP*∅)

RETURN
SENDWT

SENDWT

AC[STATE]

Does the state = \emptyset

No

RETURN
SNDCOM

Call SENDWT

Setup:
Pointer to command string:
SNDCPT
Update SNDCOM
Number of characters on the line:
NOCHLN
Update SNDCOM
Number of information lines:
NINFLN

Call SEND

RETURN
SEND

SEND

Call SENDWT
(the first time it also checks if STATE=1)

Set STATE=2 (waiting for POLL)
BUFFER\[ pointer to the command string (or to FBUFF)
RCVCH[ pointer to POLL3
Update SEND

RETURN
**STFILL**

- Call SENDWT
- **STATE=1** (filling buffer)
  - Count of lines during fill: FLLNCT
  - Character count during fill: FLCHCT
  - Pointer to buffer: FILLPT
  - Command to append: FBUFF
  - NOCHLN[1] (one command character)

**RETURN**
FLFILL

FLFILL

FILLEL

Are there any more lines?

No

STATE = ∅

RETURN

Yes

Update NINFLN

Call SEND

RETURN
FILLW

FILLW

FILLT[ pointer to word

Update FLCHCT

Yes

Does FLCHCT=Φ

No

Update FLCHCT

Yes

Does FLCHCT=Φ

No

FILLET
FILLET[ pointer to line

Yes

Does the character count =0

No

Were there no characters?

Yes

Store the character count: FILLET

No

FILLE2
Update FILLPT
FLCHCT[-65

Yes

Are there any more lines?

No

RETURN

FILLPT*
pointer to the word

Yes

Does FILLPT=0

No

FILLPT*[0

CALL SEND

NINFLN*[0

CALL STFILL
Call RCVCH Check for the order of the POLL characters

No Is there an SOH? Yes

Call RCVCH

Check for the order of the POLL characters

Was correct POLL character received? No Yes

STATE[2] (waiting for POLL) Stop receiving RCVCH[0] increment STATE SENDPT[0] pointer to SENDL Clear parity

Yes Is it finished? No

CALL SENDWP Increment pointer SENDPT

Does SENDPT[0]?

SEND2 LINCT[2] - number of information lines - initialises the line number

SEND 3
SEND 3

Increment LINENO

Does LINENO = ∅?

No

Increment LINECT

Does LINECT = ∅?

No

LINENO[∅] the next line that will be sent will be line 12

Call SENDWP

SEND 4
BUFFPTR[ pointer to first word in indicated line (32 x LINENO + BUFFER)
Initialize BYTE CHARCT[-(number of characters)]

Yes

SEND 5
SEND 5

Increment CHARCT

Does CHARCT=∅?

No

SEND 6
Check to see if first or second character in word

AC[2 bytes of 7040 BCD]

CALL SENDWP

Yes

AC[DC3=223 (end of line)]

CALL SENDWP

Does LINECT=∅?

No

SEND 3

AC[ETX=293 (end of block)]

CALL SENDWP

Yes

AC[-(right most 7 bits of parity)]

CALL SNDCH

SNDCH[∅ (Stop sending) increment state (wait for reply)]
CHARCT[−6 (give 6 chances to start returning proper reply)]

REPLY ∅
SENDWP

CHARAC[AC
AC[CHARAC

CALL SNDCH

Exclusive or of parity & character
⇒ PARITY

RETURN

SNDCH

SNDCH[ pointer to return
SNDCHC[AC

CALL INTRET

RCVCH

RCVCH[ pointer to return

CALL INTRET
Intuitively, it would be acceptable that through examination of data and various verification procedures, a human being can eliminate erroneous input readings or calculations. This report is concerned with investigating this aspect through the use of interactive graphics. It is proposed to investigate this aspect through a case study in a competitive game situation.

When taking sonar readings one can anticipate that periodically erratic readings will occur. Thus the data displayed may appear to be inconsistent when visually compared to other historical data. This report proposes a method for eliminating this erroneous data, and the submarine vs. task force game in the Moore School is the vehicle for investigation.

Specific displayed information is made light sensitive. Then, by using the light pen, the player eliminates that information considered to be inconsistent. Thus, a more reliable estimate is created.
<table>
<thead>
<tr>
<th>KEY WORDS</th>
<th>LINK A</th>
<th>LINK B</th>
<th>LINK C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interactive graphics</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sonar</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remote access</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interface</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multilang</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multilist</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Competitive war games</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Information display</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Light sensitive display</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>On-line computer systems</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>