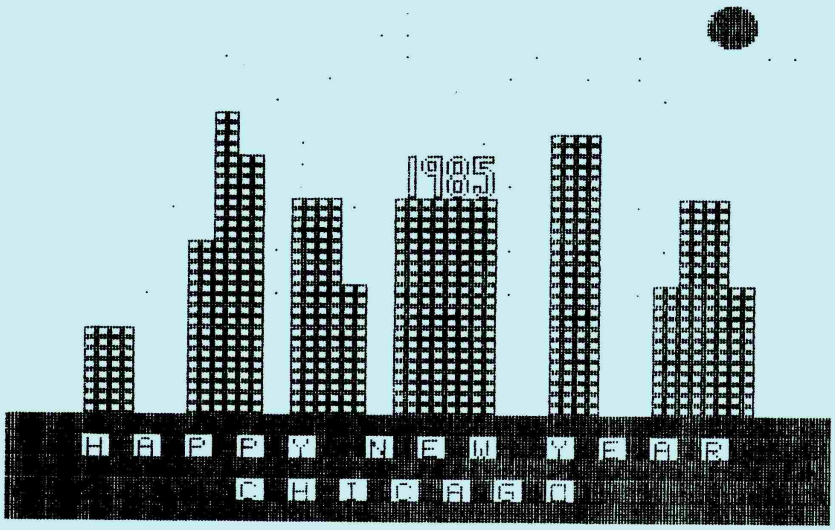
Chicago TImes

THE NEWSLETTER OF THE CHICAGO-AREA TI-99/4(A) USER'S GROUP, VOLUME 4 #5

BASIC TRAINING BEGINS!

EDITOR: Carole Goldstein 30 DEC. 1984



THE JAN MEETING

will be held of Saturday, Jan. 12, 1984, from 1:00 to 3:00 PM in the Fireside Lounge at Triton College. This meeting will feature a demonstration of three kinds of printers, the ink jet, the dot matrix and the letter quality.

IN THIS ISSUE...

The Business Sig will meet this month at 11:00.

Basic Training will begin after the meeting at 3:00.

Please note that if you wish to attend both and the meeting you could be at Triton all day!

Please remember to renew your membership!!!!

The FORTH CHAPTER column will return next month.

THE DISASSEMBLY: Dave Wakely

Last month I closed this column by noting that the group has some interesting plans for the new year. Those plans begin with the Jan. 12th meeting, when there will be a demonstration of three different types of printers which can be interfaced to the TI-99/4A. The three types of printers are dot matrix, ink jet, and "fully-formed" character. The demonstrators will be Rich Klein, Bill Hoffman, and Bob Lee, respectively. Each will put his printer through its paces and explain the various strengths and weaknesses of the machine. They will also be available to answer questions about printing in general. How will you be able to see a printer in operation if you are sitting in one of the back rows in the Fireside Lounge? We will have the usual monitor setup, but on the monitors will be showing the output from a video camera pointed at the printers, that's how.

In the past we have given presentations or programs aimed at many of the interests of the TI user. Games, system utilities, even new peripherals have all been demonstrated (and often given away) at our meetings, and we will continue to do so in the future. On occasion, however, it has been pointed out to us that the novice TI user can easily be left behind in the technical esoterica one hears and sees at the meetings from the more experienced or even "professional" users. One result of this was evident in the user survey we recently completed, when one responder noted that he was probably just one of many group members too embarrassed to admit that they did not know BASIC. Well, all that is about to change, as the group is now offering a three-part course in TI BASIC, taught by Sam Pincus. What will these sessions, which will be held after the main presentation of the meeting, cost the group member? Nothing. The cost is being paid for by the group, but there is one catch-you MUST bring your own TI console and some type of display (TV or monitor). The course is definitely a "hands-on" tutorial, and without these you will most likely be lost. This approach is both the most effective way to learn as well as less demanding of materials which must be reproduced.

Each session will run up to 90 minutes, and will usually start between 2:30 and 3:00, so be prepared to stay as late as 4:30. The classes will begin at this meeting, Jan. 12, and continue on Feb. 9 and Mar. 9. Sam Pincus has taught TI BASIC for Texas Instruments itself as well as for various colleges in the Chicago area. Don't forget to bring the required TI console, power supply, and display. If you bring a small TV, don't forget the RF modulator. Just another service of your TI user's group.

Have you ever been at the point of running an Extended BASIC program you just loaded into memory when it occurs to you that it requires a file which may or may not be on the same disk? Sure, you can run the program, and if the file is not there it may do no more damage than ending the program with an error message, but then how do you find that file? What I used to do was load a disk catalog program-thereby losing my Extended BASIC program-check which disk had the file, then reload the XB program once again. In other words, a waste of time.

The above is what I used to do, but not anymore. I just hit CTRL C instead and I have the information I want. What was that? Let me go back:

At the Faire, someone was showing me some of the purchases he had made when a particular program caught my attention. I worked my way over to the Software III booth and picked up a copy of "PFKEYS" by Jim Kryzak (I assume Mr. Kryzak is one of the III, as he was there in person). What I had noticed is that PFKEYS is a "background" type program. That is, no matter what Extended BASIC program is in memory, the functions of PFKEYS are always there in the background, waiting for a few key presses to activate them.

According to the title screen, PFKEYS stands for "Preset Function Keys", actually something of a misnomer. On most computers, such as the IBM PC, the function keys initiate certain actions, such as RUN, LOAD, etc. On the 99/4A, however, the FCTN-N sequence usually performs some line-oriented editing function such as the FCTN-2 (delete character) or FCTN-3 (erase). What PFKEYS does is to use the CTRL key in conjunction with the numbers and certain letters to access functions. I suppose it should have been titled "PCKEYS" (Preset Control Keys), but that's a minor point.

PFKEYS comes on a disk which contains two files. There is an assembly language object file named "pfk" (yes, lowercase), and a program named LOAD. If the disk is in drive 1 when you boot Extended BASIC, the LOAD program will ask if you want to load PF. If you do, there will be some wait while PF loads, as it apparently uses the Extended BASIC loader and is 53 disk sectors long, hence some patience is required. When finished, a title screen appears, acknowledging Mr. Kryzak's copyright. Then, however, the usual Extended BASIC
prompt (>) reappears at the bottom of the screen, and XB functions as usual. Programs can be loaded and run, and the continued existence of PFKEYS is not evident.

The simple, but sufficient 2 pages of instructions which accompany the disk explain that the following functions (controls?) are available in Command mode: 1-RUN, 2-LIST, 3-NUM, 4-RES, 5-OLD DSK1., 6-RUN"DSK1.LOAD, 7-SAVE "DSK1., 8-DELETE"DSK1., 9-LIST"PIO", 0-SAVE INSTRUCTIONS. That is, if from the XB prompt you press CTRL 1, the word RUN will appear at the bottom of the screen. Press enter and whatever program is in memory will run, just as if you had typed it. One complaint about these functions is that I wish for numbers 5, 7 and 8 the cursor would appear at the end of the line so that I could just fill in the rest of OLD DSK1. (for example), instead of at the beginning of the line, which necessitates using FCTN D to move the cursor right to enter the required program name. Also, the functions do not always appear on the first attempt to invoke them. Sometimes several key presses were needed before the word would appear. And while I appreciate that some of the PFK functions are user definable, perhaps most annoyingly PFKEYS sometimes appears "too sensitive". Fairly often when inputting the number "3" (I have no idea why this key rather than any other) PFK would prompt me for the number I wished to save under (Save as # 8,9, or 0 ?), when I didn't want to save anything! However, this did no discernable damage to any program in memory at the time, and my input of "3" was always accepted by the Extended BASIC program running. Within these last few sentences I have about exhausted any complaints I might have about this program.

PFKEYS apparently functions by using the interrupt capabilities of the 99/4A. Interrupts can be seen as a way of setting "breakpoints" dependent on the clock cycles of the 9901 chip in the console. I would

refer the interested reader to one of the group's fine assembly language programmers, noting also that such technical esoterica is not required to operate PFKEYS. What is required is Extended BASIC, a 32K expansion, and optionally an Epson compatible printer.

Among the letter commands available, which are even more powerful than the numbers, is a "text or graphics" screen dump program. Having an Okidata 82A printer, which is definitely not Epson compatible, I cannot report on this function. I can report that attempts to use the CTRL P to access the dump with my printer caused it do some very unusual things, many of which I had never seen it do before, but none of which was the fault of PFKEYS, the instructions of which clearly state the Epson-compatible requirement. There is also a CTRL M command, which changes the XB screen from its usual black on cyan colors to white on black, presumably for the benefit of those with monochrome monitors. PFKEYS can be turned off, and back on again, whenever you want, and some of the functions can be written into programs of your own.

Now back to the problem which started this review. PFKEYS can access the disk drive and will put a disk directory on your screen at almost any time. It gives you the usual directory information such as program name, type, size, etc. and most important it will do so while another program is in memory. This all appears to be possible because Extended BASIC programs reside in the upper portion of the 32K, while PFK resides in the lower portion and is activated when the appropriate key press initiates a vectored interrupt which causes the...(oops, more technical stuff. Sorry.). Suffice it to say that the CTRL C command causes PFK to function much like the "SD" (Show Directory) command in TI-Writer, only you return in this case to Extended BASIC.

I have on occasion been at in input prompt within an Extended BASIC program which called for me to input the name of some file. This is no longer a problem. I can now execute FCTN 4 (break), hit CTRL C and have the PFKEYS directory help me find the file, then type >CONTINUE and pick up my program and its variables intact. And yes, PFKEYS can detect disk errors (like no disk in the drive) without crashing and will notify you on screen if it finds one. I have even changed modules and when I went back to XBASIC sometimes found that PFKEYS was still there (but not always). One guide to the usefullness of a program is how often it is actually run. How often do you really boot up all those space shoot-em-ups? I use PFKEYS a lot. It is $15 and available from group member Jim Kryzak, 443 Perrie Dr. #302, Elk Grove Village, IL. Perhaps JJK will bring a few copies to the next meeting. Of course, I am also looking forward to PFKEYS II, which perhaps will additionally feature a full-function calculator, a notepad, transferrable variables, etc., etc...

Well, I can't believe it. After several months of promising I finally delivered a product review. To paraphrase former NFL coach John Madden, "Hey, wait a minute! (I wrote a review!)". As those of you who were at the last meeting know, I am stepping down as group president as of the January meeting, and so far it appears that Sam Pincus is running unopposed for the spot (as did I last year). Now that the Faire and its related duties are completed, I must get back to several other projects and must reduce the time I can give to the group. For now I plan to try to continue the column on a monthly basis, and of course I will see you all at the meetings. Thank you for all your support over the last year, it certainly has been an interesting one.

If you want to know who was responsible for the successful year we had, check the names mentioned in almost any of my columns of the past. Those people are all volunteers, and you should be too. As Texincia Lubbock once said, "At one time we all knew the same thing about computers, nothing." The way to learn, folks, is to jump in. Happy holidays!

BASICALLY YOURS: Rich Klein

This month the column will address only one inquiry. This is because of the lack of time for this newsletter to go to press. Our printers are going on vacation over the holidays and if we're going to get a newsletter this month, we've got to get it to them before they leave. By way of compensation, I will include a Mini-tutorial on nested loops. This will include a short listing and explanation to illustrate.

Now, to our mailbox...

Q: In the technical material (Extended Basic manual..ed.) I received when I got Extended Basic I was told to refer to the Speech Editor manual for more information on use of CALL SAY and CALL SPGET with my Speech Synthesizer. What would I learn by doing this? Where could I borrow a copy of a Speech Editor manual?

By the way, what does CALL VERSION do?.....W.E. (No address Given)

A: I don't own Speech Editor or know of anyone who does, and consequently can be of little help. However, we can ask anyone the would like to help you out to put a message to that effect up on the chalkboard at the next meeting. Then maybe the two of you can compare notes.

CALL VERSION merely reports the version of Extended Basic you are using. To use this statement you would type the statement:

```
nnnn CALL VERSION(V)::PRINT V
```

In case you were not aware of it, there are two versions of Extended Basic. The first was made available when there was only the TI99/4 Home Computer and supports only upper case characters and does not support the Auto Repeat feature of the TI99/4A console. One other item to note, the way Subprograms are supported is slightly different although not documented in the manual. Obviously you know which version you have, but you are not sure, use the above statement in the Command Mode. If it shows 100 then you have the old version. If you have the new version, you will see 110 displayed. I wonder what the new third party Extended basic module due for release soon will report.

Here's a short routine that demonstrates the usefulness of "Nested Loops". Nested loops are a series of For-Next Loops within one another. When they are nested, each loop is initialized from the outer loop to the inner loop . Then the inside loop is performed either until some condition is met within the loop causing control to pass to some other part of the program, or until the limit set when the loop was initialized is met or exceeded. Then control is passed to the next

innermost loop and commands are executed until the same conditions are met as above. This process continues from the innermost loop to the outermost loop. When the outer loop is completed, control will pass to the next statement or if none are present, the program will end.

The listing which follows is simply a counter. It does nothing else. Think of the inner loop as counting units while the next innermost loop will count tens and so on. If you type "TRACE" before you "RUN" the program, you will note that once the loops are initialized, the computer will not "read" that line again while that loop is executing. Instead, when it reads the "NEXT" statement the corresponding variable is incremented (or decremented). The use of the "TRACE" statement can provide valuable insight to some statements and how they are executed. By the way, to stop the program from "TRACE"ing, type the command "UNTRACE". These are both COMMANDS and can be entered in the Command Mode or included in a program listing and can be a valuable aid in debugging programs and testing logical branches in a program.

One thing to remember is that if you initialize nested loops, the first "NEXT" statement encountered must correspond to the innermost loop; the second "NEXT" statement encountered must refer to the second innermost loop and so on. See the listing that follows if you're not clear on this.

LISTING:

```
100 CALL CLEAR
110 FOR A=0 TO 9
120 FOR B=0 TO 9
130 FOR C=0 TO 9
140 FOR D=0 TO 9 <------------------------¦
150 FOR E=0 TO 9 <----------¦              ¦
160 CALL HCHAR(12,17,E+48) ¦--INNER LOOP  ¦----SECOND INNER LOOP
170 NEXT E <---------------¦              ¦
180 CALL HCHAR(12,16,D+48)                ¦
190 NEXT D <------------------------------¦
200 CALL HCHAR(12,15,C+48)
210 NEXT C
220 CALL HCHAR(12,14,B+48)
230 NEXT B
240 CALL HCHAR(12,13,A+48)
250 NEXT A
```

EXPLANATION:

100 Clears screen.
110 Initializes the variable A to the outside loop.
120-150 Initializes the rest of the loops.
160 Calls HCHAR routine to place character on screen. Requires Row,Col,ASCII value of character to be placed on screen. ASCII value of ZERO is 48 so 48 must be added to value of each loop variable to display proper number.
170 NEXT statement for inner loop. Increments the variable E and passes control to Line 160 until loop is done. Then control goes to the next line.
180 Calls HCHAR to place next digit on screen.
190 NEXT statement for D loop. Increments variable D and passes control to Line 150 and starts loop E again. Each time loop E is

finished, the new value of D is displayed and D is incremented until loop D is done. 200-250 Displays values for each loop and increments each loop as needed. USE "TRACE" TO SEE THE WAY THIS WORKS!!!

Well, I hope this little tutorial and example helps explain how Nested Loops work as well as loops in general. Perhaps with a little imagination, you can incorporate this into a program of your own.

HAVE A HAPPY HOLIDAY!!!

TRADING TImes:

Brian Bennett still has a few things left. He has a console and a PES for sale. Call him at 338-5883. Larry Keck has a mini-mem for sale for $50. You can reach him at 860-7075. Ronald Shafer has for sale the following: console, ex basic, personal record keeping, Early Learning Fun, Video Graphs, Music Maker, TE II, Video Chess, Teach Yourself Basic, Haunted House, Early Math, Typwriter, Tiny Logo, Joysticks, cables adaptors manuals etc. Best offer takes everything. 878-1346.

PRACTICAL PROGRAMMING PRACTICES:

Our first guest programmer this month is Dave Punke. The programs he has included here will also appear on the group bulletin board in the download section.

The program that follows is very long. To load it you will need to do a CALL FILES(1) and NEW. To run it you may need to turn off the expansion box with a CALL LOAD(-3188,215).

The object of the game is to guess where marbles have been hidden in a grid by shooting 'rays' into the box and noting where (or if) they exit. Points are scored on each shot. Low score wins.

One or two persons may play. In a one player game, the computer hides the marbles and the human guesses. In a two person game, the players take turns hiding and guessing. the one hiding may choose where to hide the marbles, or may let the computer do it.

Hiding Marbles-

The cursor (dark square) appears at the upper left of the grid and moves right and then down each time a key is pressed. Press 'Y' to place a marble at the location of the cursor. Press any other key to skip that square. You may hide up to 5 marbles. The cursor must be moved past the lower right hand square to exit. you get more chances to hide if you make a mistake.

Shooting Rays -

At the prompt "shoot, Guess or Call?", type "s" and ENTER. At the prompt "from?", type the number at the margin of the grid at the point

at which you want to shoot a ray into the grid, and press ENTER. Be patient, it will take about 30 seconds for a response.

A shot will have one of 4 results:

Miss - the ray passed thru the grid in a straight line. No marbles were encountered in its path, or on either side of its path. Miss markers are placed at the entry and exit points. Score 2 points.

Hit - The ray hit a hidden marble directly in its path and was absorbed. A Hit marker is placed at the entry point. Score 1 point.

Detour - The ray was deflected by one or more marbles which lay one squareon either side of its course. Detour markers are placed at the entry and exit points. Score 2 points

Reflection - The ray was deflected by one or more marbles in such a way as o cause it to exit at the entry point. A Reflection marker is placed at the entry point. Score 1 point.

If you are confused about the motion of the rays, try a two player game and hide the marbles yourself. Mark the spots on a piece of paper so you remember where they are, and then shoot away, noting the results until you've mastered the motion.

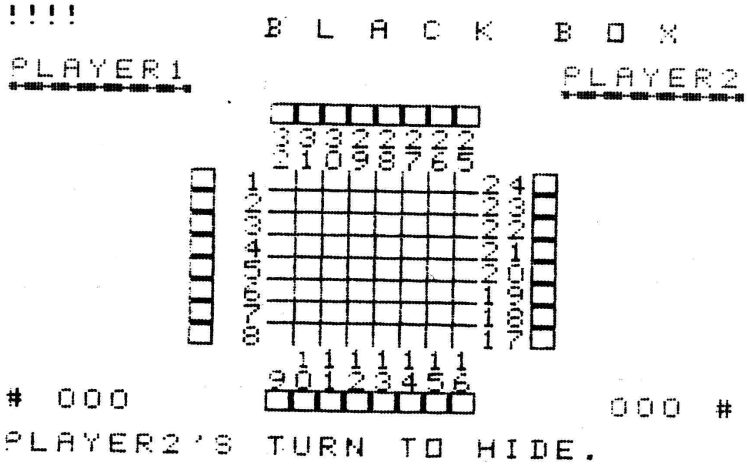
Guessing -

After a few shots, or when ever you think you know where a marble is hidden, enter "g" at the "Shoot, Guess or Call" prompt. Enter your guesses in the same way you hide the marbles. If you are placing some guesses on the grid when you already have some there, you must press "y" where they already exist, or the previous guess will be erased.

Calling the Game -

When you think you know where all the marbles are and have placed all of your guesses on the grid, enter "c" at the "shoot, Guess or Call" prompt. The computer will check your guesses. A star will replace each correct guess. If there is really no marble in a spot in which you guessed, an "X" will appear and 5 points will be added to your score. If you have placed a guess in a square that contains a hidden marble, a marble will appear and again you get 5 super bonus points.

Good Luck!!!!!!!!!!!



Note added 2022: Missing credit:
This is the game of BLACK BOX invented by Dr Eric Solomon.
It was marketed in the UK by Waddingtons and in the USA by Parkers.
In the UK Waddingtons and Dr Solomon authorised a commercial computer simulation for the TI99/4A released by Stainless Software.
Waddingtons also licenced the game to Acornsoft for the BBC Micro.
Most computer simulations failed to give proper credit and did not obtain consent.

Dr Solomon stated that the game was inspired by the CAT scanner.

```
100 REM BLACK BOX
101 REM VERSION 1
102 REM 9/29/83
103 GOTO 148
104 GOTO 149
105 REM DIMS & DEFS
106 GOTO 158
107 REM MAIN CONTROL
108 GOTO 344
109 REM INITIALIZE MATRIX
110 GOTO 362
111 REM GET INIT VALS
112 GOTO 395
113 REM ATTEMPT MOVE
114 GOTO 490
115 REM HIDE MARBLES
116 GOTO 506
117 REM GUESS MARBLES
118 GOTO 543
119 REM DEFINE BOX CHARS
120 GOTO 578
121 REM DRAW BOX
122 GOTO 618
123 REM SUBRTN FOR DRAW BOX
124 GOTO 634
125 REM ERASE A SQUARE
126 GOTO 647
127 REM PUT X IN SQR
128 GOTO 651
129 REM PUT STAR IN SQR
130 GOTO 655
131 REM PUT CIRCLE IN SQR
132 GOTO 671
133 REM CLEAR GRID & MRKRS
134 GOTO 676
135 REM CLEAR GRID
136 GOTO 683
137 REM INPUT RTN
138 GOTO 705
139 REM PUT AN OUTSIDE MRKR
140 GOTO 723
141 REM CHECK GUESSES
142 GOTO 743
143 REM PUT CURRENT SCORE
144 GOTO 750
145 REM CLEAR ANSWR LINE
146 GOTO 534
147 REM RANDOM HIDE
148 REM START OF PROGRAM
149 DIM M(9,9),PN$(1),PS(1),
MRK$(32)
150 DEF OUTMAT(X)=(INT(X/100
)<>1)
151 DEF OCCUPIED(X)=("1"=SEG
$(STR$(X),3,1))*("1"=SEG$(ST
R$(X),1,1))
152 DEF GUESSED(X)=("1"=SEG$
(STR$(X),2,1))*("1"=SEG$(STR
$(X),1,1))
153 DEF MISS=(SW1=0)
154 DEF HIT=(SW1=1)
155 DEF REF=(SW1=2)
156 DEF DET=(SW1=3)
157 DEF ERROR=(SW1=9)
158 REM MAIN CONTROL
159 RANDOMIZE
160 CALL CLEAR
161 X$="030328WELCOME TO BLA
CK BOX."
162 GOSUB 624
163 X$="040328HOPE YOU ENJOY
 THE GAME."
164 GOSUB 624
165 X$="060328DO I HAVE 1 OR
 2 PLAYERS?"
166 GOSUB 624
167 X$="062901N"
168 GOSUB 683
169 IF (X<1)+(X>2)THEN 167
170 NPS=INT(X)
171 IF NPS=2 THEN 174
172 X$="080328WHAT'S YOUR NA
ME?"
173 GOTO 175
174 X$="080328PLAYER 1'S NAM
E?"
175 GOSUB 624
176 X$="082110A"
177 GOSUB 683
178 PN$(0)=X$
179 IF NPS=1 THEN 185
180 X$="090328PLAYER 2'S NAM
E?"
181 GOSUB 624
182 X$="092110A"
183 GOSUB 683
184 PN$(1)=X$
185 X$="110328PLEASE WAIT A
MINUTE WHILE"
186 GOSUB 624
187 X$="120328I DRAW THE BOX
."
188 GOSUB 624
189 FOR DELAY=1 TO 500
190 NEXT DELAY
191 HPTR=0
192 GPTR=0
193 IF NPS=1 THEN 195
194 HPTR=1
195 SCORE=0
196 GOSUB 344
197 GOSUB 543
198 GOSUB 578
199 IF NPS=1 THEN 212
```

```
200 X$=XMSG$&PN$(HPTR)&"'S T
URN TO HIDE."
201 GOSUB 624
202 FOR DELAY=1 TO 750
203 NEXT DELAY
204 X$=XMSG$&"SHOULD I HIDE
THE MARBLES?"
205 GOSUB 624
206 X$="220303A"
207 GOSUB 683
208 GOSUB 750
209 IF SEG$(X$,1,1)="Y" THEN
 212
210 IF SEG$(X$,1,1)="N" THEN
 214
211 GOTO 204
212 GOSUB 534
213 GOTO 218
214 X$=XMSG$&"DONT' LET "&PN
$(GPTR)&" PEEK."
215 GOSUB 750
216 GOSUB 624
217 GOSUB 490
218 X$=XMSG$&"FIND THE MARBL
ES "&PN$(GPTR)&"."
219 GOSUB 624
220 FOR DELAY=1 TO 750
221 NEXT DELAY
222 X$=XMSG$&"SHOOT, GUESS O
R CALL?"
223 GOSUB 624
224 X$="220310A"
225 GOSUB 683
226 GOSUB 750
227 IF SEG$(X$,1,1)="S" THEN
 231
228 IF SEG$(X$,1,1)="G" THEN
 305
229 IF SEG$(X$,1,1)="C" THEN
 308
230 GOTO 224
231 X$=XMSG$&"SHOOT FROM?"
232 GOSUB 624
233 X$="220302N"
234 GOSUB 683
235 GOSUB 750
236 IF (X<1)+(X>32)THEN 233
237 IF LEN(MRK$(X))=0 THEN 2
43
238 X$=XMSG$&"YOU'VE ALREADY
 TRIED THAT!"
239 GOSUB 624
240 FOR DELAY=1 TO 300
241 NEXT DELAY
242 GOTO 222
243 X$=XMSG$&"ONE MOMENT PLE
ASE"
244 GOSUB 624
245 GOSUB 362
246 GOSUB 362
247 LCNT=0
248 SW1=0
249 SW2=0
250 GOSUB 395
251 LCNT=LCNT+1
252 IF SW2=0 THEN 250
253 FOR DELAY=LCNT0 TO 40
00
254 NEXT DELAY
255 IF MISS THEN 261
256 IF HIT THEN 273
257 IF DET THEN 282
258 IF REF THEN 294
259 PRINT "OOPS"
260 GOTO 260
261 X$=XMSG$&"MISS"
262 GOSUB 624
263 X=121
264 S=ES
265 GOSUB 705
266 S=EE
267 GOSUB 705
268 SCORE=SCORE+2
269 GOSUB 743
270 MRK$(ES)="M"&STR$(EE)
271 MRK$(EE)="M"&STR$(ES)
272 GOTO 302
273 X$=XMSG$&"HIT"
274 GOSUB 624
275 X=112
276 S=ES
277 GOSUB 705
278 SCORE=SCORE+1
279 GOSUB 743
280 MRK$(ES)="H"
281 GOTO 302
282 X$=XMSG$&"DETOUR TO "&ST
R$((M(R,C)/100-INT(M(R,C)/10
0))0)
283 GOSUB 624
284 X=120
285 S=ES
286 GOSUB 705
287 S=EE
288 GOSUB 705
289 SCORE=SCORE+2
290 GOSUB 743
291 MRK$(ES)="D"&STR$(EE)
292 MRK$(EE)="D"&STR$(ES)
293 GOTO 302
294 X$=XMSG$&"REFLECTION"
295 GOSUB 624
296 X=111
297 S=ES
298 GOSUB 705
299 SCORE=SCORE+1
```

```
300 GOSUB 743
301 MRK$(ES)="R"
302 FOR DELAY=1 TO 750
303 NEXT DELAY
304 GOTO 222
305 REM GUESS
306 GOSUB 506
307 GOTO 222
308 REM CALL
309 GOSUB 723
310 PS(GPTR)=PS(GPTR)+SCORE
311 X$=SEG$(STR$(INT(GAME)+1
06),2,2)&SEG$(STR$(106+21*GP
TR),2,2)&"02"
312 X$=X$&SEG$(" "&STR$(SCOR
E),LEN(STR$(SCORE)),5)
313 GOSUB 624
314 X$="18"&SEG$(STR$(105+21
*GPTR),2,2)&"03"&SEG$("  "&S
TR$(PS(GPTR)),LEN(STR$(P S(GPT
R))),5)
315 GOSUB 624
316 SCORE=0
317 GOSUB 743
318 IF NPS=1 THEN 325
319 IF HPTR=0 THEN 323
320 HPTR=0
321 GPTR=1
322 GOTO 325
323 HPTR=1
324 GPTR=0
325 IF GAME>8 THEN 338
326 IF NPS=2 THEN 328
327 GAME=GAME+.5
328 GAME=GAME+.5
329 X$=XMSG$&"ANOTHER GAME?"
330 GOSUB 624
331 X$="220310A"
332 GOSUB 683
333 GOSUB 750
334 IF SEG$(X$,1,1)="N" THEN
 338
335 GOSUB 671
336 GOSUB 344
337 GOTO 199
338 X$=XMSG$&"I'VE HAD ENOUG
H - GOODBYE."
339 GOSUB 624
340 FOR DELAY=1 TO 1000
341 NEXT DELAY
342 CALL CLEAR
343 STOP
344 REM INITIALIZE MATRIX
345 FOR R=1 TO 8
346 FOR C=1 TO 8
347 M(R,C)=100
348 NEXT C
349 NEXT R
350 FOR R=1 TO 8
351 M(R,0)=R+200
352 M(R,9)=25-R+200
353 NEXT R
354 FOR C=1 TO 8
355 M(0,C)=33-C+200
356 M(9,C)=8+C+200
357 NEXT C
358 FOR I=1 TO 32
359 MRK$(I)=""
360 NEXT I
361 RETURN
362 REM GIVEN X=EDGE NUMBER
363 REM  RETURN R,C,DR,DC
      AND ERROR SWITCH
364 SW1=0
365 IF X<>INT(X)THEN 393
366 IF X<1 THEN 393
367 IF X>32 THEN 393
368 ES=X
369 ON INT((X-1)/8)+1 GOTO 3
70,375,380,385
370 R=X
371 C=0
372 DR=0
373 DC=1
374 GOTO 390
375 R=9
376 C=X-8
377 DR=-1
378 DC=0
379 GOTO 390
380 R=25-X
381 C=9
382 DR=0
383 DC=-1
384 GOTO 390
385 R=0
386 C=33-X
387 DR=1
388 DC=0
389 GOTO 390
390 SR=R
391 SC=C
392 RETURN
393 SW1=9
394 RETURN
395 REM ATTEMPT A MOVE
396 IF (DR=0)*(DC=1)THEN 402
397 IF (DR=-1)*(DC=0)THEN 41
1
398 IF (DR=0)*(DC=-1)THEN 42
0
399 IF (DR=1)*(DC=0)THEN 429
400 PRINT "LOGIC ERROR 1"
401 STOP
402 LDR=-1
403 LDC=1
```

```
404 LTDR=-1
405 LTDC=0
406 RDR=1
407 RDC=1
408 RTDR=1
409 RTDC=0
410 GOTO 438
411 LDR=-1
412 LDC=-1
413 LTDR=0
414 LTDC=-1
415 RDR=-1
416 RDC=1
417 RTDR=0
418 RTDC=1
419 GOTO 438
420 LDR=1
421 LDC=-1
422 LTDR=-1
423 LTDC=0
424 RDR=-1
425 RDC=-1
426 RTDR=1
427 RTDC=0
428 GOTO 438
429 LDR=1
430 LDC=1
431 LTDR=0
432 LTDC=1
433 RDR=1
434 RDC=-1
435 RTDR=0
436 RTDC=-1
437 GOTO 438
438 AR=R+DR
439 AC=C+DC
440 LR=R+LTDR
441 LC=C+LTDC
442 RR=R+RTDR
443 RC=C+RTDC
444 RAR=RR+DR
445 RAC=RC+DC
446 LAR=LR+DR
447 LAC=LC+DC
448 IF OUTMAT(M(AR,AC))THEN
449 ELSE 455
449 REM OUT MATRIX
450 R=AR
451 C=AC
452 SW2=1
453 EE=M(R,C)-200
454 RETURN
455 IF OCCUPIED(M(AR,AC))THE
N 456 ELSE 461
456 REM HIT
457 SW2=1
458 EE=ES
459 SW1=1
460 RETURN
461 IF OCCUPIED(M(LAR,LAC))*
OCCUPIED(M(RAR,RAC))THEN 464
462 IF OUTMAT(M(R,C))*(OCCUP
IED(M(LAR,LAC))+OCCUPIED(M(R
AR,RAC)))THEN 464 ELSE 4
68
463 REM REFLECTIONS
464 SW2=1
465 EE=ES
466 SW1=2
467 RETURN
468 IF OCCUPIED(M(LAR,LAC))T
HEN 469 ELSE 477
469 REM  TURN RIGHT
470 SW1=3
471 R=RR
472 C=RC
473 DR=RTDR
474 DC=RTDC
475 IF OUTMAT(M(R,C))THEN 45
2
476 RETURN
477 IF OCCUPIED(M(RAR,RAC))T
HEN 478 ELSE 486
478 REM   TURN LEFT
479 R=LR
480 C=LC
481 DR=LTDR
482 DC=LTDC
483 SW1=3
484 IF OUTMAT(M(R,C))THEN 45
2
485 RETURN
486 REM STRAIGHT AHEAD
487 R=AR
488 C=AC
489 RETURN
490 REM HIDE MARBLES
491 SW1=0
492 GOSUB 512
493 X$=XMSG$&"IS THIS ARRANG
EMENT OK?"
494 GOSUB 624
495 X$="220310A"
496 GOSUB 683
497 GOSUB 750
498 IF SEG$(X$,1,1)="Y" THEN
 504
499 GOSUB 344
500 GOSUB 671
501 X$=XMSG$&"OK, TRY AGAIN.
"
502 GOSUB 624
503 GOTO 491
504 GOSUB 671
505 RETURN
506 REM GUESS MARBLES
```

```
507 X$=XMSG$&"GUESS AWAY"
508 GOSUB 624
509 SW1=1
510 GOSUB 512
511 RETURN
512 REM COMMON HIDE
513 HC=0
514 FOR R=1 TO 8
515 FOR C=1 TO 8
516 CALL HCHAR(BR+2+R,BC+2+C
,100)
517 IF HC>4 THEN 521
518 CALL KEY(0,X,STAT)
519 IF STAT=0 THEN 518
520 IF X=89 THEN 525
521 GOSUB 634
522 IF SW1=0 THEN 531
523 M(R,C)=100+VAL(SEG$(STR$
(M(R,C)),3,1))
524 GOTO 531
525 IF SW1=1 THEN 528
526 M(R,C)=101
527 GOTO 529
528 M(R,C)=110+VAL(SEG$(STR$
(M(R,C)),3,1))
529 HC=HC+1
530 GOSUB 655
531 NEXT C
532 NEXT R
533 RETURN
534 REM RANDOM PLACE
535 FOR HC=1 TO 5
536 R=INT(8*RND)+1
537 C=INT(8*RND)+1
538 IF OCCUPIED(M(R,C))THEN
536
539 M(R,C)=101
540 NEXT HC
541 RETURN
542 REM I THINK LINE NO GOOD
 IF OCCUPIED(M(RAR,RAC))THEN
 1401 ELSE 1410
543 REM DEF CHARACTERS
544 CALL CHAR(96,"0000000000
0000FF")
545 CALL CHAR(97,"8080808080
8080FF")
546 CALL CHAR(98,"8080808080
808080")
547 CALL CHAR(99,"FF81818181
8181FF")
548 CALL CHAR(100,"FFFFFFFFF
FFFFFFF")
549 CALL CHAR(101,"000000000
0000000")
550 CALL CHAR(104,"001C3E3E3
E1C00FF")
551 CALL CHAR(105,"809CBEBEB
E9C80FF")
552 CALL CHAR(106,"809CBEBEB
E9C8080")
553 CALL CHAR(107,"001C3E3E3
E1C0000")
554 CALL CHAR(109,"0000E7FFE
7000000")
555 CALL CHAR(111,"FFFFFFFFF
FFFFFFF")
556 CALL CHAR(112,"FF8181899
99181FF")
557 CALL CHAR(40,"C163361C18
3663FF")
558 CALL CHAR(41,"C1E3B69C98
B6E3FF")
559 CALL CHAR(42,"C1E3B69C98
B6E3C3")
560 CALL CHAR(43,"C163361C18
3663C1")
561 CALL CHAR(120,"FF9999FFF
F9999FF")
562 CALL CHAR(121,"FF8181FFF
F8181FF")
563 CALL CHAR(114,"082A1CFF1
C2A08FF")
564 CALL CHAR(115,"88AA9CFF9
CAA88FF")
565 CALL CHAR(116,"88AA9CFF9
CAA8888")
566 CALL CHAR(117,"082A1CFF1
C2A0808")
567 CALL CLEAR
568 CALL SCREEN(16)
569 CALL COLOR(10,9,1)
570 CALL COLOR(11,8,1)
571 CALL COLOR(12,11,1)
572 CALL COLOR(2,14,1)
573 CALL COLOR(9,2,4)
574 FOR I=3 TO 8
575 CALL COLOR(I,13,1)
576 NEXT I
577 RETURN
578 REM DRAW BOX
579 BR=5
580 BC=10
581 XMSG$="200329"
582 X$="010916B L A C K  B O
 X"
583 GOSUB 624
584 GOSUB 671
585 FOR I=1 TO 8
586 X$=STR$(M(0,I))
587 GOSUB 618
588 CALL HCHAR(BR+1,BC+2+I,X
1)
589 CALL HCHAR(BR+2,BC+2+I,X
2)
```

```
590 X$=STR$(M(I,0))
591 GOSUB 618
592 CALL HCHAR(BR+2+I,BC+1,X
1)
593 CALL HCHAR(BR+2+I,BC+2,X
2)
594 X$=STR$(M(I,9))
595 GOSUB 618
596 CALL HCHAR(BR+2+I,BC+11,
X1)
597 CALL HCHAR(BR+2+I,BC+12,
X2)
598 X$=STR$(M(9,I))
599 GOSUB 618
600 CALL HCHAR(BR+11,BC+2+I,
X1)
601 CALL HCHAR(BR+12,BC+2+I,
X2)
602 NEXT I
603 X$="030310"&PN$(0)
604 GOSUB 624
605 X$="03"&STR$(31-LEN(PN$(
1)))&"01"&PN$(1)
606 GOSUB 624
607 CALL HCHAR(4,3,109,7)
608 IF NPS=1 THEN 610
609 CALL HCHAR(4,24,109,7)
610 X$="241308## 00 ##"
611 GOSUB 624
612 X$="180308# 000"
613 GOSUB 624
614 IF NPS=1 THEN 617
615 X$="182308   000 #"
616 GOSUB 624
617 RETURN
618 REM GET CHARS FOR BOX
619 X1=ASC(SEG$(X$,2,1))
620 IF X1>48 THEN 622
621 X1=32
622 X2=ASC(SEG$(X$,3,1))
623 RETURN
624 REM PRINT ROUTINE
625 ZR=VAL(SEG$(X$,1,2))
626 ZC=VAL(SEG$(X$,3,2))
627 ZM=VAL(SEG$(X$,5,2))
628 ZL=LEN(X$)-7
629 CALL HCHAR(ZR,ZC,32,ZM)
630 FOR ZX=0 TO ZL
631 CALL HCHAR(ZR,ZC+ZX,ASC(
SEG$(X$,7+ZX,1)))
632 NEXT ZX
633 RETURN
634 REM ERASE SQUARE
635 IF (C=1)*(R<>8)THEN 640
636 IF (R=8)*(C<>1)THEN 642
637 IF (C=1)*(R=8)THEN 644
638 ZX=97
639 GOTO 645
640 ZX=96
641 GOTO 645
642 ZX=98
643 GOTO 645
644 ZX=101
645 CALL HCHAR(BR+2+R,BC+2+C
,ZX)
646 RETURN
647 REM PUT X
648 ZX=40
649 GOSUB 659
650 RETURN
651 REM  PUT STAR
652 ZX=114
653 GOSUB 659
654 RETURN
655 REM PUT CIRCLE
656 ZX=104
657 GOSUB 659
658 RETURN
659 REM COMMON PUT
660 IF (C=1)*(R<>8)THEN 665
661 IF (R=8)*(C<>1)THEN 666
662 IF (C=1)*(R=8)THEN 668
663 ZX=ZX+1
664 GOTO 669
665 GOTO 669
666 ZX=ZX+2
667 GOTO 669
668 ZX=ZX+3
669 CALL HCHAR(BR+2+R,BC+2+C
,ZX)
670 RETURN
671 REM CLEAR GRID & MRKRS
672 CALL HCHAR(BR,BC+3,99,8)
673 CALL HCHAR(BR+13,BC+3,99
,8)
674 CALL VCHAR(BR+3,BC,99,8)
675 CALL VCHAR(BR+3,BC+13,99
,8)
676 REM CLEAR GRID
677 FOR R=1 TO 8
678 FOR C=1 TO 8
679 GOSUB 634
680 NEXT C
681 NEXT R
682 RETURN
683 REM INP NUM RTN
684 XI$=""
685 ZN=0
686 ZR=VAL(SEG$(X$,1,2))
687 ZC=VAL(SEG$(X$,3,2))
688 ZM=VAL(SEG$(X$,5,2))
689 CALL HCHAR(ZR,ZC,32,ZM)
690 CALL SOUND(150,440,10)
691 CALL KEY(0,X,STAT)
692 IF STAT=0 THEN 691
693 IF X=13 THEN 701
```

```
694 IF X=7 THEN 684
695 IF ZN=ZM THEN 691
696 IF ((X<48)+(X>57))*(SEG$
(X$,7,1)="N")THEN 691
697 XI$=XI$&CHR$(X)
698 CALL HCHAR(ZR,ZC+ZN,X)
699 ZN=ZN+1
700 GOTO 691
701 IF SEG$(X$,7,1)<>"N" THE
N 703
702 X=VAL(XI$)
703 X$=XI$
704 RETURN
705 REM PUT MARKERS
706 IF S<25 THEN 710
707 R=BR
708 C=BC+35-S
709 GOTO 721
710 IF S<17 THEN 714
711 C=BC+13
712 R=BR+27-S
713 GOTO 721
714 IF S<9 THEN 718
715 R=BR+13
716 C=BC+S-6
717 GOTO 721
718 REM
719 R=BR+S+2
720 C=BC
721 CALL HCHAR(R,C,X)
722 RETURN
723 REM CHECK GUESSES
724 X$=XMSG$&"CHECKING"
725 GOSUB 624
726 FOR R=1 TO 8
727 FOR C=1 TO 8
728 IF OCCUPIED(M(R,C))THEN
730
729 IF GUESSED(M(R,C))THEN 7
37 ELSE 740
730 IF GUESSED(M(R,C))THEN 7
31 ELSE 733
731 GOSUB 651
732 GOTO 740
733 GOSUB 655
734 SCORE=SCORE+5
735 GOSUB 743
736 GOTO 740
737 GOSUB 647
738 SCORE=SCORE+5
739 GOSUB 743
740 NEXT C
741 NEXT R
742 RETURN
743 REM PUT CURRENT SCORE
744 X$=STR$(SCORE)
745 IF LEN(X$)>1 THEN 747
746 X$=" "&X$
747 X$="241602"&X$
748 GOSUB 624
749 RETURN
750 REM CLEAR ANSWER LINE
751 CALL HCHAR(22,1,32,32)
752 RETURN
753 END
```

- - - - - - - - - - - -

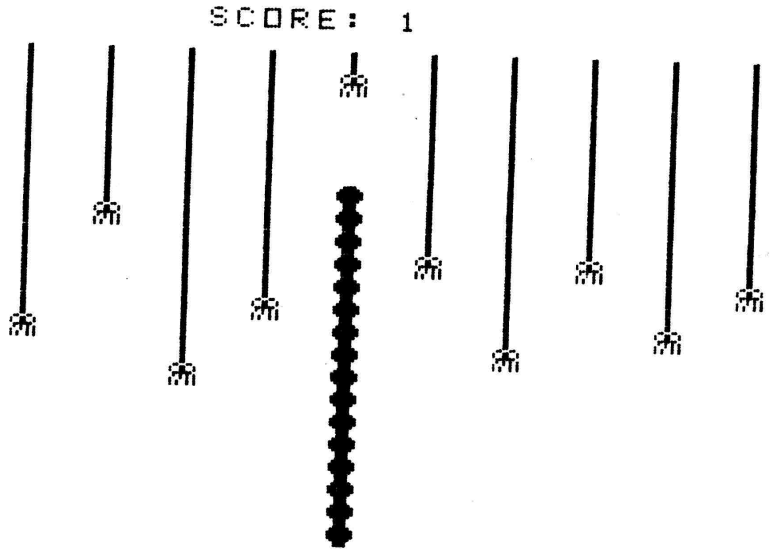
Our second guest programmer is John Behnke . If you've called the group's bulletin board lately you've seen a lot of his work up there as J.B. FREEWARE and J.B. SOFTWARE.

The first program is called SPIDER-BOP. It was written as a childrens program but it is pretty addicting for adults also. Enjoy it. The second program is for 1985.

```
100 REM SPIDER BOP 12/19/84
110 REM BY JOHN BEHNKE
120 REM BASIC OR X-BASIC
130 REM DISK DRIVE REQUIRED
140 REM FOR HIGH SCORES
150 CALL CLEAR
160 RANDOMIZE
170 A=0
180 B=12
190 C=0
200 CALL CHAR(96,"1818181818
181818")
210 CALL CHAR(104,"6699997E5
595A5A5")
220 CALL CHAR(112,"3C7EFFFFF
F7E3C3C")
230 CALL CHAR(113,"3C3C7EFFF
F7E3C3C")
240 CALL CHAR(97,"000000FFFF
")
250 FOR E=1 TO 8
260 CALL COLOR(E,16,1)
270 NEXT E
280 CALL SCREEN(2)
290 CALL COLOR(9,16,1)
300 CALL COLOR(10,14,1)
310 CALL COLOR(11,3,1)
320 FOR E=1 TO 10
330 D(E)=3
340 NEXT E
350 A$="SPIDER BOP"
360 F=4
370 G=11
380 GOSUB 950
390 F=22
400 G=5
410 A$="USE <ARROW KEYS> TO
MOVE"
420 GOSUB 950
430 F=8
440 G=3
450 A$="WHAT ARE YOUR INITIA
LS? ___"
460 GOSUB 950
470 NAME$=""
480 FOR E=27 TO 29
490 CALL KEY(0,KEY,ST)
500 IF ST=0 THEN 490
510 CALL HCHAR(8,E,KEY)
520 CALL SOUND(10,500,0)
530 NAME$=NAME$&CHR$(KEY)
540 NEXT E
550 F=10
560 A$="SET UP NEW HIGH SCOR
E FILE?N"
570 GOSUB 950
580 CALL KEY(0,KEY,ST)
590 IF ST=0 THEN 580
600 CALL SOUND(10,500,0)
610 IF KEY=13 THEN 630
620 CALL HCHAR(10,30,KEY)
630 CALL GCHAR(10,30,TE)
640 IF TE<>89 THEN 660
650 GOSUB 2120
660 F=24
670 G=3
680 A$="PRESS <ENTER> TO BOP
 SPIDERS"
690 GOSUB 950
700 A$="PUNCH POWER:"
710 F=12
720 G=10
730 GOSUB 950
740 A$="(H)I (M)ED (L)OW"
750 F=14
760 G=8
770 GOSUB 950
780 CALL KEY(0,H,I)
790 IF I=0 THEN 780
800 IF (H<>72)*(H<>77)*(H<>7
6)THEN 780
810 IF H<>72 THEN 840
820 J=1
830 GOTO 890
840 IF H<>77 THEN 870
850 J=2
860 GOTO 890
870 IF H<>76 THEN 890
880 J=3
890 GOSUB 990
900 F=1
910 G=17
920 GOSUB 1100
930 GOSUB 1530
940 GOTO 920
950 FOR E=1 TO LEN(A$)
```

```
960 CALL HCHAR(F,G+E-1,ASC(S
EG$(A$,E,1)))
970 NEXT E
980 RETURN
990 F=1
1000 CALL CLEAR
1010 G=10
1020 A$="SCORE: 0"
1030 GOSUB 950
1040 FOR E=1 TO 10
1050 CALL HCHAR(3,E*3,96)
1060 CALL HCHAR(4,E*3,104)
1070 NEXT E
1080 CALL HCHAR(24,B,112)
1090 RETURN
1100 A=INT(10*RND)+1
1110 IF D(A)+J>23 THEN 1170
1120 D(A)=D(A)+J
1130 CALL SOUND(-50,110+50*R
ND,0)
1140 CALL VCHAR(D(A)-J,A*3,9
6,J)
1150 CALL HCHAR(D(A),A*3,104
)
1160 RETURN
1170 CALL HCHAR(D(A),A*3,32)
1180 B=B+1
1190 FOR E=A*3 TO B STEP SGN
(B-A*3)
1200 CALL HCHAR(24,E,104)
1210 CALL HCHAR(24,E,32)
1220 NEXT E
1230 CALL HCHAR(24,E,104)
1240 FOR E=1 TO 400
1250 NEXT E
1260 CALL CLEAR
1270 F=1
1280 G=12
1290 A$="GAME OVER"
1300 GOSUB 950
1310 GOSUB 1870
1320 OPEN #1:"DSK1.SPIDERFIL
E",INTERNAL,VARIABLE
1330 FOR I=5 TO 1 STEP -1
1340 INPUT #1:A1,A1$
1350 F=5+I*2
1360 G=13
1370 A$=A1$
1380 GOSUB 950
1390 G=17
1400 A$=STR$(A1)
1410 GOSUB 950
1420 NEXT I
1430 CLOSE #1
1440 F=20
1450 G=3
1460 A$="PRESS <ENTER> TO PL
AY AGAIN."
1470 GOSUB 950
1480 CALL KEY(0,H,I)
1490 IF I=0 THEN 1480
1500 IF H=13 THEN 150
1510 CALL CLEAR
1520 END
1530 CALL KEY(0,H,I)
1540 IF I=0 THEN 1700
1550 CALL SOUND(-10,-3,5)
1560 IF H<>68 THEN 1620
1570 CALL HCHAR(24,B,32)
1580 IF B<30 THEN 1600
1590 B=0
1600 B=B+3
1610 CALL HCHAR(24,B,112)
1620 IF H<>83 THEN 1680
1630 CALL HCHAR(24,B,32)
1640 IF B>3 THEN 1660
1650 B=33
1660 B=B-3
1670 CALL HCHAR(24,B,112)
1680 IF H<>13 THEN 1700
1690 GOSUB 1710
1700 RETURN
1710 FOR K=24 TO 5+J*2 STEP
-1
1720 IF K=D(B/3)THEN 1760
1730 CALL HCHAR(K,B,112)
1740 NEXT K
1750 GOTO 1830
1760 CALL SOUND(100,-5,0)
1770 CALL VCHAR(K-4+J,B,32,4
)
1780 CALL HCHAR(K-5+J,B,104)
1790 D(B/3)=K-5+J
1800 C=C+1
1810 A$=STR$(C)
1820 GOSUB 950
1830 FOR E=K TO 23
1840 CALL HCHAR(E,B,32)
1850 NEXT E
1860 RETURN
1870 REM HIGH SCORE
1880 OPEN #1:"DSK1.SPIDERFIL
E",INTERNAL,VARIABLE
1890 FOR I=1 TO 5
1900 INPUT #1:HS(I),NAM$(I)
1910 NEXT I
1920 IF HS(1)>C THEN 2100
1930 HS(1)=C
1940 NAM$(1)=NAME$
1950 FOR I=1 TO 4
1960 FOR J=I+1 TO 5
1970 IF HS(I)<=HS(J)THEN 204
0
1980 CHANGE=HS(I)
1990 NN$=NAM$(I)
2000 HS(I)=HS(J)
```

```
2010 NAM$(I)=NAM$(J)
2020 HS(J)=CHANGE
2030 NAM$(J)=NN$
2040 NEXT J
2050 NEXT I
2060 RESTORE #1
2070 FOR I=1 TO 5
2080 PRINT #1:HS(I),NAM$(I)
2090 NEXT I
2100 CLOSE #1
2110 RETURN
2120 REM SET UP NEW FILE
2130 OPEN #1:"DSK1.SPIDERFIL
E",INTERNAL,VARIABLE
2140 FOR I=1 TO 5
2150 PRINT #1:0,"___"
2160 NEXT I
2170 CLOSE #1
2180 RETURN
```



The following listing is depicted on the front cover of this newsletter. We join with John in wishing everyone a very happy, healthy and prosperous 1985.

CHARACTER \ IS OBTAINED WITH FCTN C

```
100 RANDOMIZE 100
110 RAN1=1.5 :: RAN2=2
120 CALL CLEAR
130 CALL SCREEN(2)
140 FOR I=3 TO 8 :: CALL COL
OR(I,16,5,I+6,16,2):: NEXT I
150 GOTO 410
160 CC=INT(175*RND)+50 :: SP
=INT(2*RND)+10 :: DIR=INT(9*
RND)-4
170 GOSUB 760
180 CALL SPRITE(#1,40,16,160
,CC,-SP,DIR)
190 FOR I=1 TO 4 :: GOSUB 76
0 :: NEXT I
200 GOSUB 340
210 CALL MOTION(#1,0,0)
220 CALL POSITION(#1,R,C)
230 GOSUB 760
240 CALL SPRITE(#2,40,16,R,C
,#3,40,16,R,C,#4,40,16,R,C,#
5,40,16,R,C)
250 CALL SPRITE(#6,40,16,R,C
,#7,40,16,R,C,#8,40,16,R,C)
260 GOSUB 760
270 CALL MOTION(#1,-6,-6,#2,
-6,0,#3,-6,6,#4,0,-6,#5,0,6,
#6,6,-6,#7,6,0,#8,6,6)
280 GOSUB 760
290 GOSUB 760
300 CALL DELSPRITE(#1,#2,#3,
#4,#5,#6,#7,#8)
310 GOSUB 760
320 GOSUB 340
330 GOTO 160
340 GOSUB 760
350 IF 137-ROW<=57 THEN CALL
 COLOR(10,5,1):: GOTO 390
360 GOSUB 760
370 ROW=ROW+4
380 CALL LOCATE(#25,ROW+57,1
37,#26,137-ROW,137)
390 GOSUB 760
400 RETURN
410 REM SCREEN
420 CALL CHAR(40,"0000001"R
PT$("0",55))
430 CALL CHAR(96,"FF9999FFFF
9999FFFFFFFFFFFFFFFF")
440 CALL CHAR(104,"FF9999FFF
F9999FF")
450 CALL CHAR(128,"071F3F7F7
FFFFFFFFFFFF7F7F3F1F07")
460 CALL CHAR(130,"E0F8FCFEF
EFFFFFFFFFFFFFEFEFCF8E0")
470 CALL CHAR(132,"0E0A0A0A0
A0A0A0A0A0A0A0A0A0E")
480 CALL CHAR(134,"3E42BAAAA
ABA423A0A0A0A0A0A0A0E")
490 CALL CHAR(136,"3844BAAAA
AAABA4444BAAAAAAABA4438")
500 CALL CHAR(138,"EEAAAAAAA
AAABA82FA0A0A0A0A0A0A0E")
510 CALL CHAR(140,"3844BAAAA
AAABA4444BAAAAAAABA4438")
520 CALL CHAR(142,"FE82BEA0B
884F20A0A0A0AEAAABA4438")
530 CALL SCREEN(2)
540 CALL COLOR(2,16,1,9,5,1,
10,5,12,13,16,2)
550 FOR I=1 TO 20
560 CALL HCHAR(RND*5+1,31*RN
D+1,40)
570 NEXT I
580 DISPLAY AT(6,7):"ˢ":" (
   ˢ  (      (  ˢˢ"
590 DISPLAY AT(8,1):"(     ˢ
ˢ (         ˢˢ":"  (   ˢh
      ( ˢˢ"
600 DISPLAY AT(10,1):"    (
ˢˢ ˢˢ( ˢˢˢˢ  hˢ ( ˢh
hˢ ˢˢ  ˢˢhˢ  ˢˢ   ˢˢ"
610 DISPLAY AT(12,1):" (   ˢ
ˢh(ˢh  ˢhˢˢ( ˢˢ  (ˢˢ       h
ˢˢ ˢˢ  ˢˢˢˢ  ˢˢ   ˢˢ"
620 DISPLAY AT(14,1):"   ( ˢ
ˢˢ(ˢˢˢ ˢhˢˢ( hˢ (ˢˢˢh      ˢ
ˢˢ ˢˢˢ ˢˢˢˢ  ˢˢ  hˢˢˢ"
630 DISPLAY AT(16,1):" ˢˢ  ˢ
hˢ hˢˢ ˢˢˢˢ  ˢˢ  ˢˢhˢ  ˢˢ  ˢ
ˢˢ ˢˢˢ ˢˢˢˢ  ˢˢ  ˢˢˢˢ"
640 DISPLAY AT(18,1):" hˢ  h
ˢh ˢhh ˢˢˢˢ  ˢh  ˢˢhh  ˢˢ  ˢ
ˢˢ hˢˢ ˢˢˢˢ  ˢˢ  ˢˢˢˢ"
650 CALL HCHAR(20,1,97,160)
660 FOR I=10 TO 13 :: CALL S
PRITE(#I,65,1,137,255):: NEX
T I
670 DISPLAY AT(21,1):"aHaAaP
aPaYaaNaEaWaaYaEaAaRaa"
680 DISPLAY AT(23,1):"aaaaaa
aCaHaIaCaAaGaOaaaaaaaa"
690 CALL HCHAR(8,16,132):: C
ALL HCHAR(8,17,134)
700 CALL HCHAR(9,16,133):: C
ALL HCHAR(9,17,135)
710 CALL SPRITE(#25,136,16,5
7,137,#26,140,16,137,137)
720 CALL MAGNIFY(3)
730 CALL HCHAR(1,28,128):: C
ALL HCHAR(1,29,130)
740 CALL HCHAR(2,28,129):: C
ALL HCHAR(2,29,131)
750 GOTO 160
760 READ LE,NT
```

```
770 IF LE=0 THEN RESTORE ::
RAN1=INT(2*RND)+1 :: RAN2=IN
T(2*RND)+1 :: GOTO 760
780 CALL SOUND(LE*.9,NT,0,NT
/RAN1,2,NT/RAN2,2)
790 RETURN
800 DATA 600,392
810 DATA 700,523
820 DATA 400,494
830 DATA 500,523
840 DATA 800,659
850 DATA 600,587
860 DATA 400,554
870 DATA 900,587
880 DATA 600,659
890 DATA 600,523
900 DATA 400,523
910 DATA 400,659
920 DATA 800,784
930 DATA 1000,880
940 DATA 600,880
950 DATA 600,784
960 DATA 400,659
970 DATA 500,659
980 DATA 600,523
990 DATA,600,587
1000 DATA 500,554
1010 DATA 600,587
1020 DATA 400,659
1030 DATA 500,587
1040 DATA 600,523
1050 DATA 500,440
1060 DATA 600,440
1070 DATA 600,392
1080 DATA 1000,523
1090 DATA 600,880
1100 DATA 800,784
1110 DATA 500,659
1120 DATA 600,659
1130 DATA 600,523
1140 DATA 600,587
1150 DATA 500,554
1160 DATA 800,587
1170 DATA 600,880
1180 DATA 600,784
1190 DATA 500,659
1200 DATA 600,659
1210 DATA 600,784
1220 DATA 1200,880
1230 DATA 500,1047
1240 DATA 600,784
1250 DATA 500,659
1260 DATA 600,659
1270 DATA 600,523
1280 DATA 600,587
1290 DATA 500,554
1300 DATA 700,587
1310 DATA 400,659
1320 DATA 400,587
1330 DATA 700,523
1340 DATA 400,440
1350 DATA 600,440
1360 DATA 700,392
1370 DATA 900,523
1380 DATA 900,40000
1390 DATA 0,0
```

2022 note:

On the 4A keyboard the character \ is obtained with FCTN C

966-2342: Irwin Goldstein

As of December 1, 1984 the TI-99/4A Users Group Bulletin Board has been moved to Niles and along with the new location comes a new SYSOP. It is my pleasure to have taken over this position as the enjoyment of managing this BBS far outweighs the work involved. We have met a number of very interesting people both young and old and have had some very good "chats" with many of them.

Those of you who frequent the board may have noticed some changes such as new programs for downloading or updated information on the meetings and a current updated newsletter. You may have also noticed that the board is now up and running 99% of the time. Many more changes both bigger and better are soon to be beaming their way onto your screens. Some ideas for improving the Board are to eliminate those awful color changes constantly going on while we try to use the system. We are considering going to a consistant yellow backgroung with black letters. This has been proven to be the best combination in terms of visability and lowest eye strain. We are also considering opening up the message base once more so that messages may be left by all who wish to do so but a password will still be required to do downloading and other features contained in the private section of the system. This step may be taken to allow for a more free and open message base which can only be of benefit to all who use the BBS; especially those who call long distance from out of state. This will also be in keeping with the philosophy that the TI USERS GROUP is open to all who wish to participate in our organization, bar none.

I would like to take this opportunity to give a few hints to those less seasoned veterans of the BBS as to the best way to log-on. sist is super important that one types his name in all CAPS or else the BBS will hang up in a less than courtious manner. Also it is imperative that one does NOT type a space prior to the first letter of his name. For some unexplained reason not known to me a good many callers will leave a blank space just prior to the first letter of their name at log-on. Again the BBS will rudely hang up with not even a good-bye.

Again we would like to welcome all who wish to call to please do so. The new number is 966-2342, 24 hours a day,7 days per week. To abtain a password simply go into the help section and follow the instructions. ee you soon on the system and enjoy the changes.

VIDEO MODES: Jim Ness

Way back when the TI99/4A computer was designed, it was understood that a low buck computer had to provide entertain- ment as much, or more than, computing power. So it was necessary for such a machine to have decent color graphics capability, as well as math and text handling capabilities.

TI's response to this was the TMS9901 video display processor. This chip has the ability to display text in 8 x 8 dot size or 8 x 6 dot size (32 column vs. 40 column), or non-text graphics using 4 x 4 dot squares (multi-color mode), or even single dot graphics (bit-map mode). Actually, there are two versions of this chip. The early one does not have bit-map mode capability.

This video display processor (VDP) mode is set by an 8-bit register in the processor itself. There is no access to this register from TI-Basic or Xbasic, but you can set it from TI-Forth, Pascal and Assembly Language.

GRAPHICS MODE
****

The default upon power-up of the com-puter is Graphics mode, which is the 32 column mode we are all familiar with from Basic. In this mode the processor keeps track of 768 squares, each of them an 8x8 dot matrix. There are 256 possi-ble configurations of each of these squares, with the information on each configuration stored in VDP Ram. The standard TI99/4A chars are loaded upon start up of the machine (Basic only allows characters 32 through 159 to be tampered with).

Another section of VDP Ram stores which character is currently in each of the 768 squares. And a third section keeps track of which color each character configuration is supposed to be. Colors in Graphics mode are assigned to configurations in groups of 16 consecutive character numbers. Graphics mode does not set enough Ram aside to let you set each character to a particular color. Remember that your Basic program is stored in VDP Ram, so the processor has to leave 16k of VDP ram open for that.

TEXT MODE
****

Another VDP mode you are familiar with is Text, or 40 Column mode. This is the mode used for text-based programs such as the Terminal Emulator, TI-Writer, Editor/Assembler, and some of the Adventure games. Use of the VDP Ram is simi- lar to Graphics mode, except that since there are 24 lines of 40 columns, there are now 960 8x6 dot squares on the screen. Keeping track of almost 200 extra positions in VDP Ram uses up space normally used to keep track of color sets in Graphics mode. That fact, plus the fact that colors are not as important in text programs, limits your color choices to just picking foreground and background colors to be used for all characters.

****

Future chapters will deal with the other two modes, Multi-color mode and Bit-map mode. Also there will be tips on how to use these modes and I will write and upload to TI-WEST (766-2797) an assy lang program that will demonstrate each of the modes, to go along with this article.

LEARNING TI ASSEMBLY: NICK IACOVELLI

Well here is my first in a long series of articles on TI Assembler. The first article will be on the most important thing-getting a character to be display on the screen.

The first thing I must tell you is '*' character this is the TID-Assembler rem character.If you put this character in the first column of your assembly code it will ignore what is printed to the right of it. You can take this article off the screen using option 2

on the TE-2 module and dump it to your disk drive. Then reassemble and it will execute. Use a ctrl 0 to turn off the the machine or you will lose the the file.

THE FIRST STEP

1. Put into disk drive one the E/A disk part 1

2. Hit 1 on the E/A cartridge then hit 2 to edit. This is the area that you type your source code.

3. After you type your source code hit function 9 twice hit 3 to save and 'Y' for variable format and save your code under the filename 'source'.

THE COLUMNS

LABELS-Any name or marker you desire for jump and bl must start in column one

OPERATION FIELD-The first position following a space character. terminate a label field no label then it start in the second column.

OPERAND FIELD- The first position after a space following the OPERATION FIELD

EXAMPLE

```
12345678901234567890123456789O

   TEST    LI   R1,>0000
```

The LABEL is TEST
The OPERATION FIELD is LI
The OPERAND FIELD is r1,>0000

THE FIRST PROGRAM

The following program will show you how to put one character on the screen.

The TI computer screen is divided into 768 squares. These square start at the top at 0 and work down.TI has built in utilities to move one character to a location on the screen. These utilities are VMBW,VSBW,VMBR,VSBR.

VMBW-write more than one character to your screen

VSBW-write one character to your screen

VMBR-read more than one character off the screen

VSBR-read a single character off the screen

ALL THESE UTILITIES USES REGISTER 0-2

Lets define the start of the program and put it in the label field

```
      DEF START
      REF VSBW
* NEED TO REF THE
UTILITY
      WE WILL USE
START LI R0,400
      LI R1,>4100
      BLWP @VSBW
      END
```

LI R0,400 400 IS THE SPOT ON SCREEN WHERE I WANT TO PLACE MY CHARACTER. LI R1,>4100 THE FIRST BYTE >41='A' THAT IS THE CHARACTER I WANT WRITTEN ON THE SCREEN IN HEX. CHANGE THAT AND YOU COULD HAVE ANOTHER CHARACTER WRITTEN ON THE SCREEN. THE BLWP @VSBW DOES THE ACTUAL WRITE TO THE SCREEN

NEXT SAVE THE PROGRAM,LOAD THE ASSEMBLER TYPE DSK1.SOURCE THEN TYPE DSK1.OBJECT HIT ENTER THEN TYPE RC AND IT WILL ASSEMBLE. AFTER IT ASSEMBLE WITH NO ERRORS HIT 3 FOR LOAD AND RUN. TYPE DSK1.OBJECT WHEN THE CURSOR COME BACK HIT ENTER THEN TYPE START AND HIT ENTER AGAIN

In the next article I will show you how to write more than one character to the screen and move it forward or backward.The use of a KEYSCAN and answer to any questions you may have.

A Fifth of Forth: Stephan Meyers

It has come to my attention over the past few months that a whole lot of people have bought FORTH, but not a lot of people know what they are going to do with it. I have received two (written) questions to that effect. The first was given anonymously to our illustrious Carole Goldstein. The young lady in question wanted to know, in essence, "What can FORTH do for me? What can FORTH do that BASIC can't do?". The other question comes from R. J. Schweikert in North Riverside. He writes "Do you think you could write a FORTH article taking the novice STEP-BY-STEP?". Well, I am finally getting down to it. Everybody seems to need a helping hand getting started.

First, we shall assume that you have your TI-99 in front of you, all switched on. To run TI FORTH, you will need the following: your faithful TI-99, 32K (or higher) memory expansion, at least one disk drive, the EDITOR/ASSEMBLER command module, and the TI FORTH system disk.

The first thing you need to do is back up your system disk. The TI FORTH system disk is a wondrous and fragile thing, thanks to the incomplete nature of this implementation (i.e. T.I. wasn't quite done with it-see chapter 0, page 1 of the TI FORTH manual). Always use the backup, and never take the write-protect tab off of the original. I have blown enough system disks to have learned this as a matter of course. You can duplicate the disk through Disk Manager or any of the

"Quick-copy" type disk duplicator programs available commercially. A good one for single-sided disks, named MASSCOPY is in the public domain; I have heard about a double side version, but I haven't seen it myself.

Next, insert the EDITOR/ASSEMBLER module and put the DUPLICATE system disk in drive one. Select EDITOR/ASSEMBLER (2) from the menu screen. Type 3 to select LOAD AND RUN. When the module prompts you for a filename, type DSK1.FORTH and press enter. The screen will clear, then show "BOOTING..." in the upper left hand corner, then it will display something like:

```
-SYNONYMS   -EDITOR     -COPY

-DUMP       -TRACE      -FLOAT

-TEXT       -GRAPH1     -MULTI

-GRAPH2     -SPLIT      -VDPMODES

-GRAPH      -FILE       -PRINT

-CODE       -ASSEMBLER  -64SUPPORT

-BSAVE      -CRU



TI FORTH

ok
```

It won't look exactly like that, but it will be very similar. The words preceeded by dashes refer to options available to you for use in your FORTH programs. Each is a FORTH WORD, one that will load a set of subroutines. They are all adequately explained in the manual, chapter 1, page 5. You may not understand all of it, but explaining it all right now would take too much space and would not be very productive at this time. There is one, however that merits closer attention: -64SUPPORT.

This word loads the necessary program for using the TI FORTH 64 column editor. You heard me right: not 28, not 32, not 40 columns, but a full 64 columns of text. This is something that cannot be accomplished in BASIC, at least not with the ease of FORTH (prove me wrong, "Mr. Bit-map"!). This mode is known as the bit-map mode, which allows you to access each pixel on the screen individually. A pixel is a dot on the screen, the same size as the individual dots in a TI BASIC character definition. For neophytes, this is the same size as the dot in an exclamation point or question mark in almost anything TI has put out (i.e. BASIC, TI-WRITER, etc...). We are going to load this editor program and use it to modify your system disk copy. More on that later.

Digression #1: If anybody out there has a TI-99/4, the old kind, with the 'Chiclet' keys, as opposed to a TI-99/4A, with the full-strike keys, give up now. Your computer does not support bit map mode, unless

you had the chips switched. If this is the case, you know what you're doing and don't need me to help you. Otherwise, you will have to use the 40-column editor. This also applies to anyone whose sight "Isn't what it used to be...". Read the manual, chapter 3, page 4-6 for the differences (very slight) in usage between the two editors.

Digression #2: Although TI FORTH can (easily) access standard files, it "prefers" to work in what we call SCREENs. A screen is a perfect one K (1024 bytes) of disk memory, 64 columns by 16 lines (hence the 64-column editor). there are 90 screens (3 sectors each) on a single-sided, single-density disk. These screens may be thought of as sort of like a row of numbered cubbyholes to store information in. Subsequent disk drives pick up the numbering where the previous drive left off. i.e. Screen one on drive 2 (referred to as drive 1 usually: drive 1 becomes drive 0 in FORTH-you'll understand why eventually) is screen 91 (unless you have double sided, etc... not too important at this point).

O.K., now you understand (I hope) how FORTH stores information. Now you're going to get to work with some information-namely, your system disk. TI, true to form, put the TI FORTH language into the public domain at entirely the wrong moment-too late to save the TI, too early to be a finished product. Thus, there is a very simple typographical error ON THE SYSTEM DISK. This becomes an advantage here, because it gives me something to have you do with the editor. Load the 64 column editor by typing:

```
-64SUPPORT
```

following it with the enter key, as usual. You probably already know that the enter key lets the computer know when you're done typing. The disk drive should whir, and you should get a hopeful, lower case "ok" on the screen. Next type:

```
72 EDIT
```

And enter. Incidentally, if you have a write-protect tab on your COPY of the system disk, take it off. Also, if you are obstinately keeping the master of the system disk in the drive, switch the computer off, think about how much you forked over to the group for it (or should have) and make a copy. The phrase "72 EDIT" (and it must be in capitals) will let you edit screen number 72 on the system disk. In FORTH, the number a routine acts on comes before the routine-the explanation for that should come next month.

You should be pleasantly surprised to see the screen clear, with a black patch on top and a blue patch on the bottom, showing you screen 72 in all of it's 64-column living glory. Play around with the cursor keys (Function S, D, E, X, as in TI-WRITER). What fun, eh? Be vary careful at this point not to type over anything, and if you do, type it back to what it was. The back of the manual lists the screens on the system disk. The pages aren't numbered, but poke around, you'll find it.

Now, look at the beginning of the fifth line down. You will see a word spelled: "PAB_ADDR". Well, that's not the way it's supposed to be. Move over to that point and type a dash ("-") over the underscore ("_"). Now press FCTN-9 to leave the edit mode. You will see the

cursor blinking at the bottom of the screen (if you don't, try FCTN-9 again, or if you're adventurous, type the space bar once or twice). Type:

FLUSH

to save your changes to the disk. This is enough terminal work for today. After the disk stops rotating, type:

MON

to return to the title screen.

There, that wasn't so hard now, was it? OK, so it was, but tough. On to the part of the article where I extol the virtues of FORTH and make you wonder why anybody programs in anything else.

Think for a while about what makes a good programming language. Nick Iacovelli, don't do this near dry leaves or twigs: thinking is not something you're structured for, and there's no sense in starting a brush fire from an overload. I'll wait while you go away and make a list of what you want from a programming language. (pause)

Oh, back so soon? Let me see your list (hold it up close to the page). OK, very good! High speed. Ease of programming. Not having to wait for a compiler or assembler. Not having the language take up so much room there's no-where for the program. Easy access of ALL machine facilities. Compatible with other computers. Easy implementation of structured programming (a comp. sci. person in the audience!). Nicki, did you write this one:"Has to make good pizza?". Well, taking them all one by one:

High speed: This is FORTH's best point. The November issue of MICROpendium ran a comparison of BASIC and FORTH. a program that took 209.4 seconds to run took 7.2 seconds in TI FORTH (don't bug me, I'm know I'm ignoring the Wycove FORTH stuff!). Thus, FORTH takes only 3.4% of the time BASIC for this program. FORTH seems to be about 25 times faster than basic. However, assembler is faster than FORTH, most estimates at about 5 times faster. Anybody who has tried to write assembler should agree that the difference is worth it.

Ease of programming. In 99'er magazine, November, 1982 (this is the issue that boldly proclaims "We're Now Monthly"-need I say more?) there was a "comparison of languages for the TI home computer", rating languages on, among other things, ease of use. Assembly rated a 1 (hardest) and TI PILOT a 10 (easiest). I would rate FORTH a 5, easier than Pascal, but harder than BASIC. FORTH has some difficult concepts, such as the use of the stack (next month), but once these basic concepts are understood and one can think in reverse polish notation, FORTH becomes as easy as "1 1 + 2 =" ! (a little FORTH humor). Although Leo Brodie and other FORTH proponents suggest that FORTH is a good beginner's language, I still think that BASIC, with all it's shortcomings, is still the best to start with. I would say LOGO, but I am not as well versed in this magnificent language as I ought to be.

FORTH is also easier to work in than many languages because it is an interpreted language. You can enter commands and programs directly

from the keyboard, but you don't have to wait for the computer to interpret/parse/scan everything (dontcha just hate waiting for BASIC to finish scanning a long program!).

Not having to wait for the compiler or assembler. This is what I hate most about Pascal and Assembler. You have to load the editor, load your program, edit the program, save it again, leave the editor, load the compiler, compile the program, and run it, just to make a one-line change. The great thing about FORTH is that the compiler is always in memory and can compile the longest of FORTH programs in a trice. If you like to customize your system to make things go faster, you can. As you will see next month, each program you write becomes an integral part of the FORTH system until you end your session!

Not having the language take too much room. The FORTH compiler is probably the most compact in all of programming. The TI Basic interpreter takes up 24 K of ROM (8K ROM, 18K GROM) and Extended Basic takes 36K. FORTH takes well under 8K, most of which is written in FORTH. Unfortunately, FORTH has to eat into your memory expansion (PROM'S, anyone?), and you can't store FORTH in the video ram, as you do with Basic. However, compare this to Extended Basic, where there is a full 8K that you can't use for anything but assembly language! Furthermore, FORTH programs are VERY compact, often oven more compact than assembly. The memory you've got goes a lot further with FORTH than most other languages. Incidentally, I would like to see a firmware version of FORTH. If anyone has access to a PROM programmer, I would like to try and modify FORTH to work in a module.

Easy access of ALL machine facilities. This is where FORTH wins hands down. FORTH allows you to access all four screen modes (Basic allows one, Pascal allows 3, and LOGO 1). There is nothing that you can do in assembly that you cannot do in FORTH, although you may be able to do it faster in assembly. In case you would like to be impressed, go into the 64 column editor. This is a special mode known as SPLIT2, and exists only in FORTH. It is a specialized version of the bit-map mode. Counting the special modes included on the system disk, FORTH allows 6 different modes-2 more than the machine has! How's that for squeezing blood out of a rock! While you are in the bottom part of the screen, type:

And a line, one pixel wide, will go from the top left to the bottom right of the graphics window. After slaving away, trying to work with bit-map in Assembler, here comes FORTH which lets you type commands right there and watch them work! FORTH also let's you have speech, sound, anything, although you may have to write it yourself. This would, of course, be easier in FORTH than any other language. I have written a crude SOUND routine in FORTH. If any more knowledgeable FORTH programmers would suggest how such a routine should handle the stack, let me know: my version only supports voice A.

Compatible with other machines. FORTH is one of the most transportable languages around, when it is written PROPERLY. There is a simple BREAKOUT game in FORTH in a back issue of BYTE for the TRS-80. One of these days I'll convert it, because I will only have to change the one or two routines that use graphics and sound. Such a conversion would be nearly impossible between TRS-80 Basic and TI Basic.

Easy implementation of structured programming: For the computer science types (anybody else just nod and smile) FORTH is an entirely untyped language (although you can build in type restrictions, if you really want them). However, the structure of FORTH forces you to write clear, concise code, if you are inclined to do so. In the words of Charles H. Moore, the inventor of FORTH:

"... FORTH is an amplifier. A good programmer can do a fantastic job with FORTH; a bad programmer can do a disastrous job... There are some visible characteristics of good FORTH, such as very short definitions (many of them). Bad FORTH often takes the form of one definition per block-big, long, and dense"

John S James says:

"FORTH enforces extreme modularity, so the decomposition of each task into component parts is critical."

Briefly put, FORTH allows the programmer to modularize more easily than even Pascal, and allows modules to be tested interactively from the terminal. Put that in your compiler and smoke it!

Furthermore, FORTH is filled with a bevy of program flow control structures tailor made for the computer science type. There is a DO..LOOP (like FOR..NEXT) with a way to exit prematurely and increment by any amount (like FOR..TO..STEP). There is an IF..THEN..ELSE structure (actually, it's IF..ELSE..THEN or IF..ELSE..ENDIF). There is a BEGIN..UNTIL and BEGIN..WHILE..REPEAT and BEGIN..AGAIN for infinite loops. There is even a CASE..OF construct that nobody seems to use for some unknown reason. There is, however, no GOTO statement. Don't grumble, Basic fans. You might not believe it, but programming theory says that anything you can write with GOTO can be written more clearly in another manner. Pascal has GOTO as a concession to those too lazy to find the proper way to do things, and BASIC has it because it doesn't have all the other things. This is why BASIC is not encouraged as a first language by computer science types, because it discourages the use of proper program modules and encourages sloppy programming, for lack of a better way to do things. Not so in FORTH...

No, FORTH does not make very good pizza, at least not TI FORTH. However, if someone would build a good house-controlling interface, such as the Radio Shack one, this would be an ideal application for FORTH. In fact, FORTH's "first job" was controlling a radio telescope. FORTH's high speed, interactive nature, easy use of interrupts, and machine-level I/O make it ideal for real-time control jobs.

FORTH's greatest weakness comes in it's number-crunching abilities. It lacks any kind of array structure: however, FORTH's most powerful commands allow you to design your own data structures (and even control structures) in ways that would put Pascal to shame. You can make any kind of data type you want, such as the STRING type that I published in a previous newsletter. Just remember that you have to write your own routines to handle anything you come up with. Well, that's about all we have time for now. Next month, we shall begin to work with the stack and colon-definition. That is, writing your own programs. Class dismissed

You can call me with any questions you may have at (312) 271-1586. If

you have a modem, type them up in TI-WRITER format so you can download them to me for easy use in the newsletter. If not, you can catch me at the meetings or mail to me C/O the group P.O. Box.

REMarks: Carole Goldstein

Once again we can thank the US Mail for getting our newsletter to us in time. Only three people showed up on the wrong date for the December meeting. To alleviate this problem I am including the dates for the rest of the meetings before the summer break when as you know, the officers of this group take a well deserved break. Please make a note of the following meeting dates:

JAN 12	FEB 9
MAR 9	APR 6
MAY 11	JUNE 1

A couple of letters to the editor came in the mail this month. This warning from Hans-Joachim Szipronat. "This is to Caution our members not to take loaded disks into C.T.A. trains. If the programs are not getting erased they become extremely retarded or erratic and too much damaged for further use. This is caused by the magnetic field of the D.C. motors driving the trains. The damage to the programs(s) will occur not only when the briefcase is placed on the floor, but also if it is kept in the lap or the disk is carried in the coat pocket."

Also before I end for this month I want to give special recognition to John Behnke for all his help with the programs this month. Not only has he contributed an enormous amount of programs to the group but he also wrote the program that prepares the listings so they can be put into this newsletter in a compact and easily readable format. John is willing to write to the best of his ability (and his ability seems endless) anything that anybody wants to see done for the TI-99/4A Thanks John, you don't realize how much you are appreciated.

Again, don't forget that dues are payable now for 1985 membership. We will soon have to delete those who have not paid from the mailing list. Please, to make sure you do not miss and issue of the CHICAGO TImes please make sure your membership is paid up!

CHICAGO TI USER'S GROUP
PO BOX 578341
CHICAGO, IL 60657

FIRST CLASS MAIL
DATED MATERIAL
DO NOT DELAY

Tidings % Mr. Stephen Shaw

Englan