```
0001  // REM   TI-TREK
0002  // REM  TEXAS INSTRUMENTS 4/14/80
0003  // ldom 10.19.2021
0004
0005  // user untils, label variables, build often.
0006
0007  CALL CLEAR
0008  randomize
0009
0010  // define variables
0011  //
0012  // Maybe a good way to go would be upcase for variables that shouldn't change, and
0013  //  lower case for that can change.
0014  TRUE=1                  // always 1, true
0015  A=1
0016  FALSE=0                 // always 0, false
0017  C=0                     // always 0, used to set keyboard unit.
0018  O=0                     // Speech True or Flase
0019  SPEECH=0
0020
0021  K1=100                  // used as a fixed value int 100 - never changes.
0022  C1=1000                 // used as a fixed value int 1000 - never changes.
0023  B=3                     // it appears B is always a constant of 3 and never changed.
0024  J=8                     // rows in the galaxy / also left arrow key char
0025  I=5                     // columns in galaxy / also used for sound
0026
0027                          // quadrants are 20x20 sectors
0028  DISKNUMBER$="4"         // god hates hard code paths.
0029  K$=""                   // input variable for speech
0030  CL=0                    // Column to accept command from.
0031  RW=0                    // Row to accept command from.
0032  DIF=0                   // Stores Difficulty Level (1-5).
0033  M$=""                   // Message string - sometimes "TORP" or "COMMAND?"
0034  DG=57.2957795           // 57.2957 Degrees .eq. one radian.
0035
0036  // M&N usually hold the x,y of a quadarent.
0037
0038  // Notable variables
0039  // tp - holds current number of TorPedos.
0040  // ep - holds Engine Power
0041  // f - Foes
0042  // tm - TiMe left.
0043  // U - is used as a pointer to a speech phrase.
0044  //   U is also used in many for next loops. - definitely not a constant.
0045
0046
0047
0048  // define functions
0049  DEF f_RM(Z)=A+INT(RND*Z)    // Generates a value between 1 and Z.
0050  DEF f_CT(Z)=INT(Z+.5)       // Returns either z or z+1 if z=z.6 or above.
0051  DEF f_MH(Z)=Z+SIN(D)        // Movement Horizontal
0052  DEF f_MV(Z)=Z-COS(D)        // Movement Vertical
0053  DEF f_L1(Z)=(Z>22)+(Z<B)    // Anything over 22 returns (-1)    ** Limit 1? **
```

```
0054  DEF f_L2(Z)=(Z>20)+(Z<A)    // Anything betwee 1-20 returns 0, all other return -1  ** Limit 2? **
0055  DEF f_T$=SEG$(K$,A,A)       // string, pos1, length 1
0056
0057
0058  // Print Header
0059  PRINT TAB(10);"99/4  BASIC": :tab(12);"TICodEd": :TAB(12);"TI-TREK": :TAB(7);"TEXAS INSTRUMENTS": :TAB(8);"COPYRIGHT  1980"
0060  PRINT
0061  PRINT tab(5);"*** UPPER CASE ONLY ***"
0062
0063  // check if they want speech.. put in a config file?
0064  PRINT tab(7);"USE SPEECH (Y/N) ";
0065  while status=0
0066    call key(5,k,status)    // 5=basic mode upper and lower case full keyboard
0067  endwhile
0068  if k=121 or k=89 then begin
0069     K$="Y"
0070     print "Y" end else begin
0071     print "N"
0072  end
0073  SPEECH=FALSE              // set inital state of speech to false
0074  IF f_T$<>"Y" THEN L390      // if no skip this block
0075  OPEN #1:"DSK"&DISKNUMBER$&".TREKSAY",RELATIVE,INTERNAL,INPUT,FIXED 255
0076  SPEECH=TRUE     // O=A - why of course I want speech
0077  U=1             // U is the record number of speech data from treksay
0078  GOSUB L5300     // load speech phrase u=record number.
0079  U=3             // get next record
0080  GOSUB L5300     // and speak! =)
0081
0082  //
0083  // Dimension arrays  --- why 20's?
0084  //   a quadrant is a 20x20 (400) grid - too much memory use,
0085  //       the most 'things' you will see in a single quadrant is
0086  //       a total of 20 things - which can consists of stars, a base,
0087  //       klingons, and the enterprise itself.
0088  //
0089  L390:
0090  // removed P(5) - never used.
0091  DIM a_S(20)       // Column of a thing
0092  DIM a_T(20)       // Row of a thing
0093  DIM a_R(20)       // The thing!
0094  DIM a_G(8,5)      // this holds are galaxy. row & col 0 are defined by not used.
0095                    // array returns a digit number ie.. 1213 - 2 klingons, 1 base and 3 stars. I think 1XXX means unknown to ship.
0096
0097  //
0098  // load our custom chars from data statements at the
0099  //    end of the program and call chars.
0100  FOR U=9 TO 14
0101    READ V,M$
0102    X=(U*J)+24            // 1st iteration (x=9*8 + 24) 96, next would be 104,112, 120,128,136
0103  //  print X,M$
0104    CALL CHAR(X,M$)
0105    CALL COLOR(U,V,2)     // select color for desired character set.
0106  NEXT U
0107
```

```
0108  // call my_hchar         // debug code to show all redefined characters.
0109
0110  call char(132,"5A8140EF243E815A")     // shielded enterprise
0111  call char(124,"0000000000000000")     // blank space
0112  call char(100, "0000000000000000")    // was K1, 100 looks nicer here.   Q? why two chars with same def?
0113  call char(140,"FFFFFFFFFFFFFFFF")     // filled space
0114
0115  //
0116  // select level of difficulty
0117  //
0118  L510:
0119  E=96          // 96 - Character for the enterprise.
0120  a_R(A)=E      // a_R(1)=96
0121  F=C           // F=1
0122  print tab(6);"LEVEL OF DIFFICULTY:"
0123  print tab(10);"1-EASY,5-HARD";
0124
0125  // need a number between 1 and 5
0126  L550:
0127  status=0               // keyboard status, no key being pressed
0128  while status=0
0129    call key(5,k,status)
0130  endwhile
0131  //  print k
0132  if k<49 or k>53 then goto L550
0133  DIF=k-48
0134  call clear
0135  print "Please Wait..."
0136
0137  //
0138  // Here we create the galaxy AND everything in it.
0139  //
0140  FOR U=A TO J       // 1 to 8 rows of quadrents
0141    FOR V=A TO I     // 1 to 5 columns of quadrents
0142      W=f_RM(12)     // f_RM(Z)=A+INT(RND*Z)
0143      W=W*ABS(W<(5+DIF))
0144      F=F+W
0145      a_G(U,V)=C1+(K1*W)+(1010*INT(.15+RND))+f_RM(9)    // a_G is the galaxy.
0146  //    print "W=";W;"F=";F;"U=";U;"V=";V;"a_GU,v)=";a_G(U,V)
0147  //    input xyz$
0148   NEXT V            // columns 5
0149  NEXT U          // rows 8
0150
0151  //
0152  //  Initialize and Display control panel (full only done once, per game)
0153  //
0154  U=29               // "The Klingon Says"
0155  GOSUB L5300        // call say
0156  CALL CLEAR
0157  TM=(11-DIF)*F      // TiMe left
0158  H=.5+RND
0159  GOSUB L4070        // initalize torpedos and power
0160  RW=12
0161  CL=24
```

```
0162  M$="TORP"
0163  GOSUB L2510         // print TORP at indicated R/C
0164  M=f_RM(J)
0165  N=f_RM(I)
0166  M$="PWR"
0167  GOSUB L5280         // inc row+2 then call L2510
0168  M$="FOES"
0169  GOSUB L5280         // inc row+2 then call L2510
0170  M$="TIME"
0171  GOSUB L5280         // inc row+2 then call L2510
0172
0173  L840:
0174  GOSUB L2010         // display the current quadrant 20x20 grid of sectors
0175
0176  L850:               // this code needs to be ran if something changes in the quadrant
0177  K=a_S(A)
0178  L=a_T(A)
0179  X=a_S(A)
0180  Y=a_T(A)
0181  IF f_T$="W" THEN L4460     // if the last command was W(arp) check if enemies lurk and allow the to attack!
0182
0183  // print "K";K;"L";L;"X";X;"Y";Y
0184  // BREAK
0185  // L851:
0186  // goto L851
0187
0188  L900:       // go here for no valid choice
0189  CALL HCHAR(21,A,32,128)
0190  M$="COMMAND?"
0191  RW=22
0192  GOSUB L2500   // print COMMAND? - one chacters at a time.
0193  O1=14
0194  GOSUB L2330   // get input!
0195
0196  //  let's print array and see if we can make some sense of what they do.
0197  //call my_wait
0198  //for zzz= 1 to 20
0199   //  print zzz;a_S(zzz);
0200   //  print zzz;a_T(zzz);
0201  //  print zzz;a_R(zzz);
0202  //next zzz
0203  //call my_wait
0204
0205  //
0206  // Main Decision Tree
0207  //
0208  //   Remember A always = 1, sometimes used for true.
0209  //       123456789
0210  U=A+POS("MDFTVWLSCQ",f_T$,A)
0211  ON U GOTO L900, L1150, L3870, L4890, L980, L4780, L3270, L2890, L1510, L2550, L5350
0212  //               M      D      F      T      V      W      L      S      C      Q
0213  // example L takes you to L2890
0214
0215
```

```
0216
0217  //
0218  // Torpedo - routine
0219  // drop in for torpedo - quanity of 1
0220  //
0221  L980:
0222  V=ABS(TP>C)     // two possible values returned -1=true statement and 0=false statement
0223
0224  L990:
0225  IF TP>C THEN L1110     // if tp=0 (or less) print no torpedos
0226  M$="NO TORPEDOS LEFT"
0227  RW=24
0228  GOSUB L2500     // print command
0229  GOTO L900       // return to command prompt
0230
0231  L1040:
0232  M$="ANGLE IN DEGREES?"
0233  RW=RW+A
0234  GOSUB L2500      // print command
0235  O1=22
0236  GOSUB L2330      // get input
0237  D=VAL(K$)/DG     // angel you gave (say 90) / DegreeRadian (57.29)
0238  RETURN
0239
0240  // now it's time to take action on the torpedo
0241  L1110:
0242  GOSUB L1040      // ask for angle in degrees ...
0243  IF V>A THEN L4820  // Ask of INC Angle.
0244  L1130:           // After Volley Input & Check(L4780) we end up back here.
0245  GOSUB L1560     // execute torpedo(1) or volley(max of 3)
0246  GOTO L4490      // give foes their turn
0247
0248  //
0249  // routine to move inter quadrent.
0250  //
0251  L1150:
0252  M$="MOVE HOW FAR?"
0253  RW=RW+A
0254  GOSUB L2500       // print prompt, usually 'Command?'
0255  O1=18
0256  GOSUB L2330       // get keyboard input
0257  ML=VAL(K$)
0258  GOSUB L1040       // Call get angel in degree command variable D is set.
0259  U=7               // "Ready to start"
0260  GOSUB L5300       // speech - say something!
0261  X=a_S(A)
0262  Y=a_T(A)
0263
0264  FOR W=A TO ML   // if move 5, then 1 to 5
0265  L1270:
0266    X=f_MH(X)
0267    Y=f_MV(Y)
0268    IF f_L1(f_CT(X))+f_L2(f_CT(Y))THEN L1400   // emergency stop
0269    CALL GCHAR(Y,X,V)
```

```
0270    IF V=E THEN L1270
0271    IF V<>K1 THEN L1400                 // if you hit something jump in to error routine, other wise at the end, jump over it.
0272    CALL HCHAR(L,K,K1)                  // put a blank space here.
0273    K=X
0274    L=Y
0275    CALL HCHAR(Y,X,a_R(A))              // put the enterprise at this new position.
0276   //  input aaa$  // moves one then stops
0277    CALL SOUND(-20,180,I)               // I always = 5, guess they needed to save a byte.
0278  NEXT W
0279
0280  GOTO L1450   // jump over error routine, because you didn't hit anything.
0281
0282  // error routine, you hit something.
0283  L1400:
0284  M$="EMERGENCY STOP!"    // you hit something
0285  ML=W
0286  CALL SOUND(-250,4000,4,-2,3)
0287  RW=22
0288  GOSUB L2500            // print message you hit something.
0289
0290  L1450:
0291  a_S(A)=f_CT(K)       // set Column in this quad to ..
0292  a_T(A)=f_CT(L)       // set Row in this quad to ..
0293  EP=EP-(10*W)         // Engine Power - MoveLength*10
0294  TM=TM-2*ML           // time - 2*MoveLength
0295  // print "TM=";TM;"W=";w
0296  // break
0297  GOSUB L4090          // update torpedo status
0298  GOTO L4490           // alert foes firing
0299
0300
0301  //
0302  // Shields Up 1/3 damage from attacks or radiation.
0303  //
0304  L1510:
0305  E=228-E                    // deduct energy for shields.
0306  a_R(A)=E                     // place current energy in ??
0307  CALL HCHAR(a_T(A),a_S(A),E)    // Row, Column, Character, Rep
0308  TM=TM-A                    // Subtract 1 from time (tm)
0309  GOTO L900                  // back to choices
0310
0311  //
0312  // Execute Torpedo or Volley
0313  //
0314  // Intesting note, you can torpedo stars & bases.
0315  L1560:
0316  TP=TP-V                    // TP is # of Torpedo - Volley is a max of 3.
0317  TM=TM-(A+V)                // TiMe = TiMe - (1+3)
0318  U=19                       // Set Speech String to 19 - "Photo Torpedos"
0319  GOSUB L5300                // Call Speech
0320  FOR W=A TO V               // V is 3, or the number of torpedo's you have left.
0321    X=a_S(A)                 // set base X for where the torpedo is coming from
0322    Y=a_T(A)                 // set base Y for where the torpedo is coming from
0323  L1630:
```

```
0324    U=700                    // base sound for torpedo moving
0325  L1640:
0326    X=f_MH(X)                // Calc next X position for torpedo
0327    Y=f_MV(Y)                // Calc next Y position for torpedo
0328    IF f_L1(f_CT(X))+f_L2(f_CT(Y))THEN L1950 // jump the klingon says..
0329    CALL GCHAR(Y,X,Z1)
0330    IF Z1<>K1 THEN L1740       // if current location not an empty sector
0331    CALL HCHAR(Y,X,128)        // make noise and see what is in the next sector....
0332    CALL SOUND(50,U,5)
0333    U=U*1.02
0334    CALL HCHAR(Y,X,K1)
0335    GOTO L1640
0336
0337  L1740:                       // Not an empty sector
0338    IF Z1=E THEN L1630         // Is it the E=96 (Enterprise) ?
0339    IF Z1=104 THEN L1790       // Is it a Klingon ?
0340
0341                               // O'darn... we just blew up a base or a star!!
0342    U=23                       // phrase uh oh
0343    GOSUB L5300                // speak
0344    EP=EP-INT(EP/(3+RND))      // Engine Power from 3 (min) to 4 (max) points.
0345
0346  L1790:                       // hit explosion, replaced with was there previously with a  blank.
0347    CALL HCHAR(Y,X,136)
0348    CALL SOUND(200,-7,A)
0349    CALL HCHAR(Y,X,124)
0350    CALL SOUND(50,-8,7)
0351    CALL HCHAR(Y,X,K1)
0352
0353  // Event has happened, update galaxy and say something.
0354    FOR Y1=A TO N1               // A=1 to N1 (Number of items in Quadrant)
0355      IF(a_S(Y1)=f_CT(X))+(a_T(Y1)=f_CT(Y))>-2 THEN L1930
0356      a_G(M,N)=a_G(M,N)-10^(2-(a_R(Y1)-104)/J)  // subtract one ?? from current quadrant
0357      F=F+(a_R(Y1)=104)          // F=F+0 unless a_R(Y1)=Klingon(104) then F=F-1
0358      a_R(Y1)=K1                 // make sector = 100 or empty.
0359      IF Z1<>104 THEN L1970      // it wasn't a klingon we blew up so jump to L1970
0360      U=21           // Klingon Destroyed
0361      GOSUB L5300    // call say
0362      GOTO L1970     // next W - do outer loop
0363  L1930:
0364    NEXT Y1
0365    STOP        // end program execution - maybe we never get here??
0366  L1950:
0367    U=25           // The klingon says..
0368    GOSUB L5300    // call say!
0369  L1970:
0370    D=D+T1/DG
0371  NEXT W
0372
0373  GOSUB L4090      // update torpedo status
0374  RETURN
0375
0376  //
0377  // Display the quadrent
```

```
0378  //
0379  L2010:
0380  a_G(M,N)=a_G(M,N)-C1*INT(a_G(M,N)/C1)  // maybe we insure the 1000 is removed so we know what's in these galaxies
0381  W=INT(a_G(M,N))
0382  FOR U=A TO 20
0383    CALL HCHAR(U,B,K1,20)
0384  NEXT U
0385  Y=A
0386  V=E
0387  GOSUB L2200  // goto display quadrant first pass through we get all blanks and the enterpris.
0388
0389  // this pass we display stars/bases/klingons.
0390  V=96
0391  FOR P1=2 TO C STEP-1    // 2 to 0 step-1
0392    U=INT(W/(10^P1))
0393    W=W-(10^P1)*U
0394    V=V+J
0395    FOR X=A TO U
0396      GOSUB L2190
0397    NEXT X
0398  NEXT P1
0399  GOTO L2270
0400
0401  L2190:
0402  a_R(Y)=V
0403
0404  L2200:
0405  a_S(Y)=4+INT(18*RND)
0406  a_T(Y)=2+INT(18*RND)
0407  CALL GCHAR(a_T(Y),a_S(Y),Z1)      // get char
0408  IF Z1<>K1 THEN L2200
0409  CALL HCHAR(a_T(Y),a_S(Y),V)
0410  Y=Y+A
0411  RETURN    // end of draw quadrent routine.
0412
0413  //
0414  // Display 'Quad' and your current location (5,1)
0415  //
0416  L2270:
0417  M$="QUAD("&STR$(M)&","&STR$(N)&")"
0418  RW=3
0419  CL=24
0420  GOSUB L2510   // execute print quad(3,3)
0421  N1=Y-A        // N1 = Y (number of iterations through display loop) - A(always 1)
0422                // N1 ~= number of things on screen, can be off by one, usually if
0423                //   more than 10 objects?
0424  RETURN    // end of quad and current location
0425
0426
0427
0428
0429  //
0430  // Keyboard Input Routine
0431  //
```

```
0432  L2330:
0433  R1=RW
0434  L2340:
0435  K$=""
0436  U1=C
0437
0438  L2360:
0439  CALL KEY(C,Q1,T1)       // Unit=0 - whole keyboard,
0440                          // Return Variable,
0441                          // Status 1=new, -1=same, 0=no key pressed.
0442  IF T1<=C THEN L2360     // no key and no auto repeat function
0443  IF Q1=32 THEN L2360     // space key pressed
0444  IF Q1<>J THEN L2420     // not left arrow...
0445  CALL HCHAR(R1,O1,32,U1)  // display key pressed
0446  GOTO L2340               // loop
0447
0448  L2420:
0449  IF Q1=13 THEN L2470      // user pressed enter.
0450  K$=K$&CHR$(Q1)
0451  CALL HCHAR(R1,O1+U1,Q1)
0452  U1=U1+A
0453  GOTO L2360
0454
0455  L2470:                  // what do we do, now that enter has been pressed?
0456  IF R1=19 THEN L2490       // ??? r1=RoW what's special about 19 - maybe because each command is one char??
0457  IF LEN(K$)=0 THEN L2340  // if enter and no input loop.
0458
0459  L2490:
0460  RETURN     // we should have good data and now are returning.
0461
0462  // Here we set the column to accept the command from.
0463  L2500:
0464  CL=4
0465  // Print "COMMAND?" - one char at a time, really need to do accept@
0466  // M$ = the message "Command" How far to move.
0467  L2510:                  // you can just jump here, if you don't want CL=4
0468  FOR I1=A TO LEN(M$)
0469    CALL HCHAR(RW,(CL+I1-1),ASC(SEG$(M$,I1,1)))
0470  NEXT I1
0471  RETURN
0472
0473  //
0474  // Chart the KNOWn gallaxy
0475  //  XYZ - X Klingons, Y Base either blank, zero or one., z - number of stars.
0476  //
0477  L2550:
0478  FOR Y=A TO J              // 1 to 8
0479    M$=""
0480    FOR X=A TO I            // 1 to 5
0481      ON A+INT(a_G(Y,X)/C1) GOTO L2590, L2640, L2620
0482  L2590:
0483      K$="00"&STR$(a_G(Y,X))    // no klingons, no bases, # of stars.
0484      K$=SEG$(K$,LEN(K$)-2,B)  // add in knowns.
0485      GOTO L2650
```

```
0486  L2620:
0487      K$="?1?"          //  There be a base here everything else is unknown at this time.
0488      GOTO L2650
0489  L2640:
0490      K$="   "          // Nothing known about this sector.
0491  L2650:
0492      M$=M$&K$&"!"      // ! is our quadrent divider.
0493    NEXT X
0494    RW=2*Y-A
0495    CL=B
0496    GOSUB L2510
0497    CALL HCHAR((2*Y),B,45,20)   // 45 is - (minus sign)
0498  NEXT Y
0499
0500  L2720:
0501  FOR W=17 TO 20              // clear out anything that might have been on lines 17 through 20
0502    CALL HCHAR(W,B,32,20)     // 32 (space)
0503  NEXT W
0504
0505  M$="PRESS ENTER"            // now print on that clean space "PRESS ENTER"
0506  RW=18
0507  GOSUB L2500                 // write M$
0508  R1=19
0509  O1=10
0510  GOSUB L2340                 // Get Input.
0511
0512  FOR W=A TO 20
0513    CALL HCHAR(W,B,K1,20)     // clear out chart K1 = 100 = 'd' - an empty space in the quadrent.
0514  NEXT W
0515
0516  FOR W=A TO N1
0517    IF a_R(W)=K1 THEN L2870
0518    CALL HCHAR(a_T(W),a_S(W),a_R(W))   // here lay the explination of the arrays!! 10.14.2021
0519  //  CALL MY_WAIT
0520  L2870:
0521  NEXT W
0522  GOTO L850
0523
0524  //
0525  // Long range scan L2890
0526  //
0527  L2890:
0528  Q1=M+(M<>A)
0529  R1=M-(M<>J)
0530  S1=N+(N<>A)
0531  T1=N-(N<>I)
0532  FOR W=B TO 22             // clear off main screen
0533    CALL VCHAR(A,W,32,20)
0534  NEXT W
0535
0536  CALL VCHAR(A,10,33,15)    // draw horizontal line  1
0537  CALL VCHAR(A,16,33,15)    // draw horizontal line 2
0538
0539  FOR W=5 TO 15 STEP 5      // draw verticle slashes
```

```
0540    CALL HCHAR(W,B,95,20)
0541  NEXT W
0542  //call my_wait
0543
0544  M$="LONG RANGE SCAN"
0545  RW=16
0546  CL=6
0547  GOSUB L2510
0548  M$=CHR$(E)&" "&STR$(M)&","&STR$(N)    // display enterprise and current quadrant number 2,4
0549  RW=6
0550  CL=11
0551  GOSUB L2510
0552
0553  FOR X=S1 TO T1                              // for values being displayed, look at notes
0554    CL=12+6*(X-N)                             // at bottom on how all things in a qradrant
0555    FOR Y=Q1 TO R1                            // are stored as a four digit number.
0556      a_G(Y,X)=a_G(Y,X)-C1*INT(a_G(Y,X)/C1)
0557      RW=6+5*(Y-M)
0558      U=96
0559      V=INT(a_G(Y,X))
0560         FOR W=2 TO C STEP-1
0561           A1=INT(V/(10^W))
0562           V=V-(10^W)*A1
0563           U=U+J
0564           M$=STR$(A1)&" "&CHR$(U)             // actually do it - a number and a symbol.
0565           RW=RW+A
0566           GOSUB L2510                         // call display routine.
0567         NEXT W
0568    NEXT Y
0569  NEXT X
0570  GOTO L2720                                 // clear out all prior messages
0571
0572  //
0573  // Warp
0574  //
0575  L3270:
0576  M$="DEST ROW?"
0577  RW=23
0578  GOSUB L2500     // print command
0579  O1=14
0580  GOSUB L2330     // get input
0581  S1=VAL(f_T$)
0582  IF(S1<A)+(S1>J)=0 THEN L3380   // check if row between 1 and 8
0583
0584  L3340:
0585  M$="OUTSIDE PATROL BOUNDARY"
0586  RW=22
0587  GOSUB L2500     // print command
0588  GOTO L900       // return to command prompt.
0589
0590  L3380:
0591  M$="DEST COL?"
0592  RW=24
0593  GOSUB L2500     // print command
```

```
0594  GOSUB L2330     // get input
0595  T1=VAL(f_T$)
0596  IF(T1<A)+(T1>I)THEN L3340    // test for column between 1 and 5.
0597
0598  IF(S1<>M)+(T1<>N)THEN L3490   // test for current row/col no need to move.
0599  M$="ALREADY HERE"
0600  RW=22
0601  GOSUB L2500                   // wait for enter
0602  GOTO L850                     // loop back to get command
0603
0604  // fall through, it's time to get a move on.
0605  L3490:
0606  W=SQR(((S1-M)^2)+((T1-N)^2))    // the beauty of mathmatical formula's is you
0607  EP=INT(EP-200*(A+RND)*W)        // don't have to understand them to use them.
0608  TM=TM-INT(5*W)
0609  M=S1
0610  N=T1
0611  GOSUB L4090                    // update status
0612  V=330
0613  U=27                           // going in to.
0614  GOSUB L5300                    // speak
0615
0616  FOR W=A TO 20
0617    CALL SOUND(-1000,V,4)
0618    CALL VCHAR(A,(2+W),140,20)
0619    V=V*1.02
0620  NEXT W
0621
0622  CALL SOUND(30,V,3,-3,1)
0623  CALL HCHAR(10,7,136,12)
0624  M$=" WARP "
0625  CALL SOUND(-200,4000,5,-6,2)
0626  CL=10
0627  RW=CL
0628  GOSUB L2510
0629  IF RND>.1 THEN L3830
0630  U=9                       // Uhoh Position Shift
0631  GOSUB L5300               // speak
0632  M$="MAGNETIC STORM"
0633  CL=6
0634  RW=12
0635  GOSUB L2510
0636  M$="BLOWN OFF COURSE"
0637  CL=I
0638  RW=14
0639  GOSUB L2510
0640  M=f_RM(J)
0641  N=f_RM(I)
0642
0643  L3830:
0644  CALL SOUND(10,-2,1)
0645  K$="W"
0646  H=.5+RND
0647  GOTO L840                 // redisplay quadrant
```

```
0648
0649  //
0650  //  Dock - but you must be adjacent to a base.
0651  //
0652  L3870:
0653  Q1=a_S(A)+(a_S(A)<>A)      // assume enterprise is at 17,19 this returns
0654  R1=a_S(A)-(a_S(A)<>22)     // 16,18 and 18,20 (enterprise in center)
0655  S1=a_T(A)+(a_T(A)<>A)      // then checks all positions around enterprise for
0656  T1=a_T(A)-(a_T(A)<>20)     // any that match 112 (base).
0657  FOR X=Q1 TO R1
0658    FOR Y=S1 TO T1
0659      CALL GCHAR(Y,X,Z1)
0660      IF Z1=112 THEN L4030 // if we have base 112 the jump to L4030 and do basey things.
0661    NEXT Y
0662  NEXT X
0663
0664  M$="NOT AT BASE"
0665  ROW=24              // this variable appears to never be used.
0666                      // normally this is CL (column) or RW (row)
0667  GOSUB L2500         // print
0668  U=13                // captain no base(star)?
0669  GOSUB L5300         // speak
0670  GOTO L850           //
0671
0672  // confirmed, we are at a base, lets top up.
0673  L4030:
0674  GOSUB L4070        // set TP and EP values to the MAX
0675  U=11               // enterprise at
0676  GOSUB L5300        // speak
0677  GOTO L850          // get command / main menu
0678
0679  L4070:            // values after you just top off at a base
0680  TP=20             // 20 Torpedos
0681  EP=8000           // 8000 Energy Points
0682
0683  L4090:                    // torpedo status
0684  RW=10
0685  CL=29
0686  CALL HCHAR(21,A,32,128)   // display number of TorPedo's remaining.
0687  M$=STR$(TP)
0688  GOSUB L5250
0689
0690  IF EP>C THEN L4160        // energy status.
0691  EP=C
0692  L4160:
0693  M$=STR$(EP)              // display Energy Points remaining.
0694  GOSUB L5250
0695  M$=STR$(F)               // display remaining Foes
0696  GOSUB L5250
0697
0698
0699  IF F>C THEN L4330       // if foes > 0 then continue. else all foes be gone!!!
0700  M$="CONGRATULATIONS!!"
0701  RW=22
```

```
0702  GOSUB L2500
0703  M$="ALL ENEMY SHIPS DESTROYED!"
0704  RW=23
0705  GOSUB L2500
0706  M$="YOU HAVE SAVED THE GALAXY!"
0707  RW=24
0708  GOSUB L2500
0709  U=35                    // Good Luck, you Win!
0710  GOSUB L5300             // Speak
0711  GOTO L5350              // Play Again(Y/N)
0712
0713  L4330:
0714  IF TM>C THEN L4350
0715  TM=C
0716
0717  L4350:                        // test for out of energy or time.
0718                                // called from warp (amongst others)
0719  M$=STR$(TM)
0720  GOSUB L5250
0721  IF EP<=C THEN L4690           // out of EnergyPower - end game
0722  IF TM>C THEN L4450            // still have TiMe keep playing.
0723  U=33                          // sorry time up
0724  GOSUB L5300                   // speak
0725  M$="TIME EXPIRED"
0726  RW=22
0727  GOSUB L2500                   // print
0728  GOTO L4740                    // end game - play again?
0729
0730  L4450:
0731  RETURN
0732
0733  L4460:
0734  IF a_G(M,N)/K1<A THEN L900    // Any klingons?
0735  U=5                           // red alert klingon
0736  GOSUB L5300                   // speak
0737
0738  L4490:
0739  W=INT(a_G(M,N)/K1)
0740  IF W<A THEN L900
0741  M$="ALERT! FOES FIRING!    "
0742  K$=""
0743  RW=24
0744  GOSUB L2500             // print message
0745  U=17                    // klingon firing.
0746  GOSUB L5300             // speak
0747  V=3-(E-96)/18
0748
0749  FOR U=1 TO W            // W is the number of klingons that are attacking.
0750    GOSUB L4650           // sound and graphics for phaser fire.
0751    CALL SOUND(100,-1,30)
0752  NEXT U
0753
0754  EP=EP-INT(W*H*V*(50+10*DIF))     // Deduct Energy Power
0755  GOSUB L4090                      // update status
```

```
0756  GOTO L850
0757
0758  //
0759  // graphics and sound for Fire phasers.
0760  //
0761  L4650:
0762  CALL SCREEN(7)              // red border
0763  CALL SOUND(400,-6,A)        // phaser sound
0764  CALL SCREEN(4)              // green border
0765  RETURN
0766
0767  //
0768  // result if you run out of EP.
0769  //
0770  L4690:
0771  M$="ENERGY GONE"
0772  RW=22
0773  GOSUB L2500           // print message
0774  U=31                  // The Enterprise
0775  GOSUB L5300           // play again?
0776
0777  L4740:
0778  M$="GAME OVER, YOU LOST"
0779  RW=23
0780  GOSUB L2500           // print message
0781  GOTO L5350            // play again?
0782
0783  //
0784  // Volley Torpedos
0785  //
0786  L4780:
0787  V=3                   // number of torpedos in a volley
0788  IF V<TP THEN L4810    // can't volley more than you got
0789  V=TP                  // else volley what you's got.
0790
0791  L4810:
0792  GOTO L990
0793
0794  L4820:
0795  M$="INC ANGLE?"
0796  RW=RW+A
0797  GOSUB L2500          // print command
0798  O1=18
0799  GOSUB L2330          // input
0800  T1=VAL(K$)           // T1 = Increment Angle
0801  GOTO L1130
0802
0803  //
0804  // Fire phasers, can bend around stars!
0805  //
0806  L4890:
0807  RW=23
0808  E=96
0809  a_R(A)=E
```

```
0810  CALL HCHAR(a_T(A),a_S(A),E)
0811  M$="HOW MUCH POWER?"
0812  GOSUB L2500          // print command
0813  O1=20
0814  GOSUB L2330          // get input
0815  W=VAL(K$)            // W - How much power to use on phaser power.
0816  TM=TM-B              // update TiMe
0817  EP=EP-W              // deduct power - you can use all of your power by putting in more
0818  //                      then you have, because the check afterwards show you have
0819  //                      no power and you loose.
0820  //                      Also, it's possible to spend enough power to only get some
0821  //                       of the enemies.
0822  V=INT(a_G(M,N)/K1)   // get enemies from array
0823  U=15                 // computer firing.
0824  GOSUB L5300          // call speech routine
0825  IF V<>0 THEN L5050   // There ARE enemies, do the fight routine.
0826  GOSUB L4650          // if no enemies, deduct power and display graphics and sound for fire phaser.
0827
0828  L5050:               // fight routine.
0829  FOR X=1 TO V         // 1 to # of enemies.
0830    Y=(50+50*DIF)*(H+(RND/2))
0831    IF Y>W THEN L5190     // Y=enemy strength, W (How much power) power not used in the attack
0832    W=W-Y
0833    Q=A
0834  L5100:
0835    Q=Q+A
0836    IF a_R(Q)<>104 THEN L5100       // if not a klingon skip this location.
0837    a_R(Q)=K1
0838    a_G(M,N)=a_G(M,N)-K1
0839    F=F-A                           // 1 foe down
0840    CALL HCHAR(a_T(Q),a_S(Q),136)   // explosion
0841    GOSUB L4650                     // do phaser graphics and sound
0842    CALL HCHAR(a_T(Q),a_S(Q),K1)    // put a blank space in that sector
0843  NEXT X
0844
0845  L5190:
0846  M$=STR$(INT(W))&" WASTED"
0847  RW=24
0848  CL=8
0849  GOSUB L2510          // print message
0850  GOSUB L4090          // display torpedo status
0851  GOTO L4490           // it's the klingons turn!
0852
0853  L5250:
0854  IF LEN(M$)>=4 THEN L5280     // long message go down two rows
0855  M$=" "&M$
0856  GOTO L5250                   // short message go down one row
0857
0858
0859  // this just says move down 2 rows and call print routine
0860  L5280:
0861  RW=RW+2
0862  GOTO L2510
0863
```

```
0864  L5300:                // load speech phrases
0865  IF O=C THEN return    // if speech=false, return
0866    INPUT #1,REC U:M$
0867    INPUT #1,REC U+1:K$
0868    CALL SAY("",M$,"",K$)
0869  RETURN
0870
0871  //
0872  // Quit
0873  //
0874  L5350:
0875  M$="PLAY AGAIN(Y/N)?"   // message to be displayed
0876  RW=22                   // row to display on
0877  GOSUB L2500             // call the display routine
0878
0879  status=0
0880  while status=0
0881    call key(5,k,status)
0882  endwhile
0883  if k=89 or k=121 then goto L510
0884
0885  L5430:
0886  // Now we quit
0887  call clear
0888  IF O<>C THEN close #1   // if speech close file.
0889  STOP
0890
0891  // Char defs
0892  // The Star Ship Enterprise.
0893     DATA 16,"0040EF243E000000"
0894  // Evil Klingon
0895     DATA  9,"0000037F3060F000"
0896  // Star Base
0897     DATA  6,"001899BDFF9918E7"
0898  // Star - don't run in to
0899     DATA 12,"00105428D6285410"
0900  // Photon Torpedo
0901     DATA 15,"0000001038100000"
0902  // Explosion?
0903     DATA  9,"815A3C66663C5A81"
0904
0905  //
0906  // sub programs begin here
0907  //
0908  // Not part of original program Dan created to help him understand
0909  // running code.
0910  //
0911  sub my_wait
0912    loop1:
0913      call key(5,k,s)
0914      if s=0 then loop1
0915  subend
0916
0917  //
```

```
0918  // since character defs is a combo of:
0919  //     data statements, made in to char, and direct call char statements, I needed a way to
0920  //     see everything at once. Hence this subroutine.
0921  //
0922  sub my_hchar                                        //  Character  ASCII Code    What
0923      call hchar(3,3,20)::display at(3,5):"20";       //
0924      call hchar(4,3,80)::display at(4,5):"80";       //      P          80          base?
0925      call hchar(5,3,96)::display at(5,5):"96";       // {grave symbol}  96       enterprise
0926      call hchar(6,3,100)::display at(6,5):"100";     //     d          100      blank space in quad
0927      call hchar(7,3,104)::display at(7,5):"104";     //     h          104         klingon
0928      call hchar(8,3,112)::display at(8,5):"112";     //     p          112        real base
0929      call hchar(9,3,120)::display at(9,5):"120";     //     x          120           star
0930      call hchar(10,3,124)::display at(10,5):"124";   //     |          124            |
0931      call hchar(11,3,128)::display at(11,5):"128";   //                128          torpedo
0932      call hchar(12,3,132)::display at(12,5):"132";   //                132     Shielded Enterprises
0933      call hchar(13,3,136)::display at(13,5):"136";   //                136     explosion - also recycled for warp graphics.
0934      call hchar(14,3,140)::display at(14,5):"140";   //                140     red square, used in WARP notice?
0935      call my_wait
0936  subend
0937
0938
0939  // C - Chart Galactic Sector
0940  // D - Dock with adjacent base
0941  // F - Fire Phases (Strength / power)
0942  // L - Long Range Scan
0943  // M - Move in Quadrent
0944  // Q - Quit
0945  // S - Shields
0946  // T - Torpedo One.
0947  // V - Three Torpedo's
0948  // W - Warp - go to another quadrent.
0949
0950  // To standardize terminology: The Galaxy holds 40 Quadrants which holds 400 sectors
0951  // starting to make sense the (20) arrays - the 'logical' way to do this would have been
0952  // a single dim(20,20,20) or 8000 bytes - then you could have kept track of every or everything
0953  // as is now, it's a clever trick - when you return to a quadrant, everything is moved around,
0954  // but you still have the same number of stars/bases and klingons
0955  // example Quadrant (5,4) Contains 1234 - 1 unknown to scanner, 2 Klingons, 3 Bases, 4 Stars
0956  // excellent clever hack.
0957
0958  // Galaxy 8x5 each space representing a quadrant.
0959  //      1  2  3  4  5
0960  //   1
0961  //   2
0962  //   3
0963  //   4
0964  //   5
0965  //   6
0966  //   7
0967  //   8
0968  // Coordinates always given in Column (8), Row (5)
0969
0970  // quadrants are a 20x20 matrix.
0971
```

```
0972  // a_G(x,y) - holds a four digit number, like: 1208 -
0973  // 1 - uncharted quadrant.
0974  // 2 - 2 klingons
0975  // 1 - Base always one or zero?
0976  // 8 - 8 stars - not in same place if you return.
0977
0978
0979  // maybe add a hard to kill romulan.
0980  // to fix dest row has no error checking.
0981  // when you warp out of a quadarnt and return, you and the stars are randomly placed, number of stars remain the same.
0982  // Q: Why does screen background color change to green? {Bug?}   - nope dan induced, border should be green, or red when
0983  //      under attach.
0984  // Q: Are shield suppose to go down when firing phasers & should you be able to warp with shields?
0985  // C: Klingons always face to the Right, Enterprise always faces to theh Left.
0986
0987  // How does that sin and cos thing work exactly?
0988  // https://www.raywenderlich.com/2736-trigonometry-for-game-programming-part-1-2
0989
0990  // rows (RW) verticle  (Up/Down)
0991  // columns (CL) horizontal (Left/Right) think 32 or 40 columns
0992
0993  // to do
0994  //    M(ove), and a How Far M - blows out with bad argument.
0995  //       need better error checking here.
0996
0997  // command returns.
0998  // M - L850
0999  // D - L850 - return logic is strange
1000  // F - L900
1001  // T - L900
1002  // V - then goes in to foes attacking routine.
1003  // W - either L900  or   L840
1004  //       anytime you enter a gradrant, everything must be redrawn,
1005  //       goto L840 involkes L2010 - display quadrent.
1006  //       everything else uses the quadrant already displayed.
1007  // L - L850
1008  // S - L890
1009  // C - L850
```