

# POTATOHEAD - A MISSING LINK TUTORIAL

By Harry Wilhelm

The program "POTATOHEAD" is a computerized version of the "MR. POTATO HEAD" game that many of us grew up with. And although I've never seen the program, it is probably similar to the program called "FACEMAKER". POTATOHEAD was written to provide a tutorial for those interested in using some of the more advanced features of "THE MISSING LINK" (sold by Texaments) in their own programming. You will see that "POTATOHEAD" does things that are absolutely impossible to accomplish in standard Extended BASIC. But even if you're not interested in programming, it's great fun for kids (and adults) to play with!!

## USING THE GAME

The first step is to enter the program into your computer. I would advise using the checksum program for the sake of accuracy. Save to disk when finished. If you used checksum you will then have to convert the program to IV254 format as detailed on pages 23-24 of the TML manual. Load "THE MISSING LINK" and select 1 disk file and the 16 color mode. Load and RUN "POTATOHEAD".

A menu is displayed on the left hand side of the screen, and the right hand part of the screen is gray. This latter area is where the face will be drawn. The menu gives you the choice of drawing the HEAD, LEFT EYE, PUPIL, RIGHT EYE, PUPIL, NOSE, MOUTH, LEFT EAR, RIGHT EAR, or HAIR, and of PRINTing the screen or REDOing the picture.

The menu choice currently being offered is highlighted in the colors black/white. When choosing from the menu options, press the space bar to move to the next option. Press "Enter" to select the option being offered.

Usually you would start a face by drawing the head shape, so the program offers that as the first option. Press "Enter" and the menu option is now highlighted in white/blue to show that you're now in the "select graphic" mode. An ellipse is shown in the middle of the drawing area. This is the first of the proposed shapes for the head. Press the space bar to see the rest - there are seven in all. When you see a shape you like, press "Enter" and it is printed in the current pen color. Then the program returns to the menu.

Underneath the menu options is a black box. This shows the current color of the pen. The pen color can be changed at any time by pressing the "C" key, which advances the color of the box through all the colors available on the TI computer. The colors are grouped in a logical order - i.e. dark blue to light blue, dark green to light green, etc.

After creating the head, the next menu option to be highlighted is LEFT EYE. Push "Enter" to make an eye. The menu option turns white/blue to show that you're in the "select graphic" mode, and the first of the proposed eyes appears under the menu. Push the space bar to see the rest of the possible eyes. When you see one you like, press "Enter". The menu option turns to black/red to show that you're ready to position the eye. The eye appears in the center of the drawing area where it flashes on and off. To position the eye, press <FCTN> plus the W,E,R,S,D,Z,X, or C keys. As these keys are held down, the eye begins to move more rapidly. Let up on the key for a moment if the eye starts moving too fast. When the eye is positioned where you want it, press "Enter", and it is printed on the screen.

The rest of the options, from PUPIL to HAIR function in exactly the same manner. Remember that you can change the pen color at any time by pressing the "C" key.

Choosing the PRINT option sends a screen dump to your printer, which needs to be turned on. Selecting REDO clears the screen and starts the program over again.

## UNDERSTANDING "THE MISSING LINK"

Probably the biggest problem beginners have when programming for "The Missing Link" is that they are still mentally geared up for the standard Graphics Mode. Because the Bit-Mapped mode offers so much more, it is worth investing some time in learning to use it. To help you, here are some simple programs that illustrate important principles used in "POTATOHEAD". Take a few minutes to enter these programs, then study them until you understand them. And don't be afraid to experiment - it's the best way to learn!!!

Load The Missing Link, select 1 disk file and 16 colors, then enter and RUN the following programs.

```
100 ! FILL DEMO
110 FOR I=1 TO 80 STEP 3 ::
CALL LINK("CIRCLE",96,I+1,I)
:: NEXT I
120 CALL LINK("PENHUE",9,16)
:: CALL LINK("PU"):: CALL LI
NK("FILL",50,1,80,135)
130 !CALL LINK("PENHUE",2,8)
:: CALL LINK("PR"):: CALL LI
NK("FILL",1,50,192,80):: FOR
  I=1 TO 500 :: NEXT I :: GOT
O 130
140 GOTO 140
```

Line 110 generates a pattern on the screen by repeatedly drawing circles. Line 120 sets the penhue to red/white, sets the pen condition to "penup", then fills in a rectangle from row 50, column 1 to row 80, column 135. Line 140 freezes the program so that you can see your handiwork. On the screen is a horizontal rectangle that has had the color changed from black/cyan to red/white. Try out other colors for the PENHUE, then try PD, PE, and PR to see what results you get.

**LESSON ONE:** The colors of the rectangle being filled are always changed to the current PENHUE. If the pen condition is PU, only the color will be changed. If it is PD then all the pixels in the rectangle will be set to the foreground color. If it is PE then all the pixels will be erased so they appear in the background color. If it is PR then all the pixels are reversed or set to inverse video.

Now activate line 130 by removing the exclamation point. This line sets the penhue to black on cyan, sets the pen to "penreverse", then fills a rectangle from row 1, column 50 to row 192, column 80. This "paints" a vertical rectangle that is black on cyan and in inverse video. Finally there is a short delay loop before line 140 is repeated. You will see the filled rectangle alternately flashing between inverse and regular video.

**LESSON TWO:** When the pen is set to penreverse, performing the same graphic operation TWICE restores the screen to its original appearance.

Now change the PENHUE in line 130 to 1,1 and RUN the program again. You will see that the colors of the previously filled horizontal black/red rectangle aren't changed.

**LESSON THREE:** Use a penhue of 1,1 to avoid changing existing screen colors.

Enter and RUN the following program:

```
100 ! CHAR DEFINITION DEMO
110 CALL LINK("CHSIZE",8,8)
120 FOR I=1 TO 3 :: READ A$
:: CALL LINK("CHAR",122,A$):
: CALL LINK("PRINT",I*20,I*2
0,"z"):: NEXT I
```

```

130 GOTO 130
140 DATA FF818181818181FF
150 DATA 8142241818244281
160 DATA 00007C0810207C00

```

Line 120 reads character definitions from DATA statements and uses them to redefine the lower case "z", which it then prints in three places on the screen. Although only the "z" was printed, the screen displays a box, an X, and a Z.

**LESSON FOUR:** You can think of the character definitions as stencils. Once a stencil is used to "paint" graphic information on the screen, it can be changed to a different pattern without effecting previously placed characters.

Enter and RUN the following program:

```

100 ! WINDOW DEMO
110 FOR I=1 TO 80 STEP 3 ::
CALL LINK("CIRCLE",96,I+1,I)
:: NEXT I
120 WIDTH=3 :: HEIGHT=8 :: C
ALL LINK("CHSIZE",8,8)
130 ROW=60 :: COL=120
140 !CALL LOAD(11110,16,0,40
)
150 CALL LINK("WINDOW",ROW,C
OL,ROW+1+HEIGHT*8,COL+1+WIDT
H*8)
160 !FOR I=1 TO 2
170 CALL LINK("PRINT",1,1,"A
BCDEFGHIJKLMNOPQRSTUVWXYZ")
180 !NEXT I
190 COL=COL+1 :: GOTO 150

```

Line 110 uses circles to draw a pattern on screen. Line 120 sets the WIDTH to 3 and the HEIGHT to 8, then sets the character size to 8x8. Line 130 is self explanatory. Line 150 sets the window so that (assuming 8x8 characters) there is just enough room for a block 3 characters wide by 8 characters high. Line 170 prints a string starting in the upper left hand corner of the window. Only the letters that fit into the window will be displayed. Line 190 increments the column and goes back to line 150, which moves the window over by one pixel. You will see a block of letters marching across the screen one pixel at a time. Try changing the width and/or height in line 120 to see what happens.

**LESSON FIVE:** The WINDOW subroutine can be used to force a block of letters of any desired size and shape to be printed on the screen wherever you want.

One problem is that as the letters move across the screen they obliterate the existing pattern of circles. Activate lines 140,160 and 180 to get around this problem. The CALL LOAD in line 140 modifies the PRINT routine so that the background isn't blanked and it also sets the pen to penreverse. This is documented on page 11 of the TML manual. The FOR/NEXT loop in lines 160 and 180 causes the print operation to be performed twice, the same principle described in lesson two above. Now the characters move across the screen, yet the background isn't disrupted.

**LESSON SIX:** CALL LOAD(11110,16,0,40) will set the PRINT routine to "penreverse" and avoid blanking the background. Then the principles detailed in lesson two and lesson three above can be applied.

## HOW POTATOHEAD WORKS:

Potatohead can be divided into three parts: menu selection; graphic selection; and graphic placement. Following are line by line comments on the program.

## **MENU SELECTION:**

160-170 Put all 15 colors into an orderly array called HUE(n).  
180 Generate two strings used later when moving the graphic patterns and when printing in the window.  
190 Select double size sprites. Define the first sprite pattern as a solid square. Define characters 92 to 94 as left arrow, right arrow, and a blank.  
200-210 Read the menu selections and print them on the screen. When done MO will be the number of options available in the menu.  
220 Initialize HN (hue number) and X which is the current position in the menu. Draw a box on the screen, then go to subroutine in line 1160 to display the current penhue.  
230 Use FILL to clear the drawing area on the screen and again to highlight the first menu selection in black/white  
240 Set the character size to 8x8. Print the title.  
250 Scan the keyboard. If "C" pressed then change the penhue. Otherwise go to 270 unless the key was the space bar.  
260 Space bar pressed. Unhighlight current menu selection and highlight the next one, then continue scanning the keyboard.  
270 If "Enter" not pressed then keep scanning the keyboard.  
280 "Enter" pressed so highlight menu selection in white/blue. Set penhue to 1,1 and the pen condition to "PR".  
290 X corresponds to the current position in the menu. Go to the appropriate line based on X.

## **DRAW ELLIPSES FOR HEAD**

320-390 Use partial arcs of circles to draw pseudo ellipses in a variety of proportions. Let the user choose the ellipse with the space bar, then print it and return to the menu by pressing "Enter".

## **SELECT FACIAL FEATURE**

All the routines for selecting facial features work identically, so we'll only look at the one for selecting the eyes. (To shorten POTATOHEAD, both eye drawing routines come here. It would easily be possible to have separate routines for the left and right eyes.)

These routines rely on DATA statements containing two numbers and a string. The numbers are the width and height (in 8x8 blocks) of the window needed to print the feature. The string is the character definition of the graphics. The final entry should be a zero.

420 Restore pointer to beginning of graphic information. Go to a subroutine that lets the user select from all the possible features. Once in this sub, the only way to return is to press "Enter" or else press the space bar when you're at the last DATA statement. If the space bar was pressed then the last feature was just displayed so restore the pointer to the start of the graphic information and continue. Otherwise "Enter" was pressed so move on to position the feature.  
880 Read the width. If it's zero then return, otherwise read the height and the character definition.  
890 If more than nine characters are to be redefined then read the rest of the char definition. Combine the two definitions.  
900 When printing text don't blank background. Set pencondition to PR.  
910 Set the window to the proper size. Define the characters and print them in the window.  
920 Scan the keyboard. If "C" then update the penhue. If neither space bar or Enter then keep reading.  
930 Either Space or Enter was pressed. Print graphics a second time to erase them. If space bar pressed then go to line 880 to read the next set of DATA statements. If "enter" then return.

## **POSITION FEATURE ON FACE**

960 Highlight menu selection in white/red. Set R and C so feature is positioned in center of face.  
970 Put the window in center of face.  
980 Display the feature, read the keyboard, display the feature again to erase it. If no key pressed then speed equals

one. Keep reading keys.

990 If "C" then update penhue. Keep reading keys.

1000 A key was pressed, so find out if it's contained in X\$. Branch to the appropriate line.

1010-1050 Move R and C up, down, right or left as necessary.

1060 Because key is held down, increase speed (SPD) and continue.

1070 Enter had to be pressed to get here. The CALL LOAD selects PD (but doesn't blank the background). Set the penhue and print the feature on the face

1080 Restore normal print operation with this CALL LOAD. Go back to offer the next menu selection.

1100-1150 Should be self explanatory.

1170 Increments the huenumber (HN) and displays a sprite to show the current penhue. Delay a moment, then return.

1190 HILITE Subprogram. The current position in the menu, foreground and background colors are passed. CALL LINK("WINDOW") makes the entire screen the window. With PU the letters won't get disrupted. Set the penhue and fill the area to be highlighted. Set the penhue to 1,1 and return.

```
100 ! POTATOHEAD
110 ! by Harry Wilhelm
120 ! Requires THE MISSING L
INK - 1 disk file, 16 colors
130 !
140 !
150 !
160 DIM HUE(15):: FOR I=1 TO
  15 :: READ HUE(I):: NEXT I
  :: BKG=15
170 DATA 2,5,6,8,13,3,4,11,1
  2,7,9,10,14,15,16
180 X$="~"&CHR$(11)&"["&CHR$(
  8)&CHR$(9)&"\"&CHR$(10)&"`"
&CHR$(13):: FOR I=95 TO 126
  :: P$=P$&CHR$(I):: NEXT I
190 CALL MAGNIFY(2):: CALL L
INK("CHAR",1,RPT$("F",16))::
  CALL LINK("CHAR",92,"103070
F0703010002030383C3830200000
")
200 READ A$ :: IF A$<>" " THE
  N MO=MO+1 :: CALL LINK("PRIN
  T",MO*8-6,2,A$):: GOTO 200
210 DATA HEAD,\EYE,PUPIL,^EY
  E],PUPIL,NOSE,MOUTH,\EAR,^EA
  R],HAIR,PRINT,REDO," "
220 HN=0 :: X=1 :: CALL LINK
  ("BOX",MO*8+7,8,MO*8+24,25):
  : GOSUB 1160
230 CALL LINK("PE"):: CALL L
INK("PENHUE",2,BKG):: CALL L
INK("FILL",1,49,192,240):: C
ALL HILITE(X,2,16)
240 CALL LINK("CHSIZE",8,8):
  : CALL LINK("PRINT",185,100,
  "POTATOHEAD")
250 CALL KEY(0,K,S):: IF K=6
  7 THEN GOSUB 1160 :: GOTO 25
  0 ELSE IF K<>32 THEN 270
260 CALL HILITE(X,2,1):: X=X
  +1+MO*(X=MO):: CALL HILITE(X
  ,2,16):: GOTO 250
270 IF K<>13 THEN 250
280 CALL HILITE(X,16,5):: CA
  LL LINK("PENHUE",1,1):: CALL
  LINK("PR")
290 ON X GOTO 310,410,500,41
  0,500,560,630,680,740,800,11
  10,1140
300 !
310 !HEAD
320 DIST=-48 :: MA=120 :: CR
  =96 :: CC=144
330 R1=MA/2-ABS(DIST):: R2=R
  1+SQR(2*DIST*DIST):: IF DIST
  <0 THEN XX=R1 :: R1=R2 :: R2
  =XX
340 GOSUB 380
350 CALL KEY(0,K,S):: IF K=6
```

```

7 THEN GOSUB 1160
360 IF K=32 THEN GOSUB 380 :
: DIST=DIST+16+112*(DIST=48)
:: GOTO 330
370 IF K<>13 THEN 350 ELSE C
ALL LINK("PD"):: CALL LINK("
PENHUE",HUE(HN),BKG):: GOSUB
380 :: GOTO 260
380 CALL LINK("CIRCLE",CR,CC
-DIST,R1,63):: CALL LINK("CI
RCLE",CR-DIST,CC,R2,250)
390 CALL LINK("CIRCLE",CR,CC
+DIST,R1,207):: CALL LINK("C
IRCLE",CR+DIST,CC,R2,245)::
RETURN
400 !
410 ! LEFT AND RIGHT EYES
420 RESTORE 430 :: GOSUB 880
:: IF K=32 THEN 420 ELSE GO
TO 960
430 DATA 3,2,000106081020408
07E8100000000000000806010080
4020180402010080601000000000
00000817E0102040810608
440 DATA 2,3,030C10102020404
0C03008080404020240808080808
0804002010101010101024040202
010100C030202040408083080
450 DATA 3,3,00030F3F7F7C300
07EFFFFFFF8100000000C0F0FCFE3
E0C0000010608102040807E81000
0000000000080601008040201804
020100806010000000000000817
E0102040810608
460 DATA 3,3,000106081020204
07E8100000000000000806010080
4040240808080808080400000000
0000000000201010101010102402
0201008060100000000000000817
E0204040810608
470 DATA 0
480 !
490 ! PUPIL
500 RESTORE 510 :: GOSUB 880
:: IF K=32 THEN 500 ELSE GO
TO 960
510 DATA 1,1,0000183C3C18
520 DATA 1,1,3C7EFFFFFFFFF7E3
C
530 DATA 2,2,0F3F7F7FFFFFFFFF
F00C0E0E0F0F0F0F07F7F3F0F000
00000E0E0C
540 DATA 0

```

```

550 !
560 ! NOSE
570 RESTORE 580 :: GOSUB 880
:: IF K=32 THEN 570 ELSE GO
TO 960
580 DATA 2,3,040404040404040
4202020202020202004040424488
8889020202024121111098080404
020100C0301010202040830C
590 DATA 3,2,00030C182041838
37E8100000000818100C03018048
2C1C183834120180C03008181000
00000817EC1C182041830C
600 DATA 0
610 !
620 ! MOUTH
630 RESTORE 640 :: GOSUB 880
:: IF K=32 THEN 630 ELSE GO
TO 960
640 DATA 4,2,C0FC7F7F773B3C1
F0000E0FFFFFFFFF3F000007FFFFF
FFFFC033FFEFEEEDC3CF80F07030
100000000C7F8FFFFFFF7F3F07E31
FFFFFFFFFEFCE0F0E0C080
650 DATA 4,2,000000000103070
F073F7FFFFFFFFF8C7E0FCFEFFFFFF
F1FE30000000080C0E0F01F3C3B7
77F7F7C003FFFFFFFFFE00000FCF
FFFFFFFFF070000F83CDCEEFEFE3E
660 DATA 0
670 !
680 ! LEFT EAR
690 RESTORE 700 :: GOSUB 880
:: IF K=32 THEN 690 ELSE GO
TO 960
700 DATA 2,3,010204081122244
4E01804E21904020248888888484
4241200000708080807000904040
20101000000828C7001028C70
710 DATA 2,4,000101020204040
9804040202010904809111212242
42448482424120A0905024848484
8484848240106090808070000242
41212090402010003040932C418E
720 DATA 0
730 !
740 ! RIGHT EAR
750 RESTORE 760 :: GOSUB 880
:: IF K=32 THEN 750 ELSE GO
TO 960
760 DATA 2,3,071820479820404
080402010884424220000E010101
0E00012111111122224480041310

```

```

E8040310E90202040808
770 DATA 2,4,010202040408091
2008080404020209012242448509
0A04090884848242424128060901
010E000001212121212122400C
020904C231807242448489020408
780 DATA 0
790 !
800 ! HAIR
810 RESTORE 820 :: GOSUB 880
  :: IF K=32 THEN 810 ELSE GO
  TO 960
820 DATA 3,3,000004040424242
2109292929292929200004040404
8488822129292514949295454545
4555555558890929214242428259
494524A2B1F1F39BABABAFEFFFF0
148525294A4A8F0F0
830 DATA 7,2,000000000002020
A00020A2AAAAAAA0AAAAAAA
AAAAAAAAAAAAAAAAAAAA0AAAA
AAAAAAAA0080A0A8AAAAAAA000
00000008080A0
840 DATA 0A2A2AAAABBFEE0AAA
AAAAFFFF80000AAAABFFFE000000
OAAAAFFFF00000000AAAAFAFF0F0
00000AAAAAAEAF3F0000A0A8A8A
AAAF0FE0F
850 DATA 0
860 !
870 ! SUBROUTINE TO DISPLAY
  FEATURES ONE AT A TIME
880 READ W :: IF W=0 THEN RE
  TURN ELSE READ H,A$
890 IF W*H>9 THEN READ T$ ::
  A$=A$&T$
900 CALL LOAD(11110,16,0,40)
  ! DON'T BLANK BACKGROUND; PE
  NREVERSE
910 CALL LINK("WINDOW",MO*8+
  29,1,MO*8+30+H*8,2+W*8):: CA
  LL LINK("CHAR",95,A$):: CALL
  LINK("PRINT",1,1,P$)
920 CALL KEY(0,K,S):: IF K=6
  7 THEN GOSUB 1160 :: GOTO 92
  0 ELSE IF K<>13 AND K<>32 TH
  EN 920
930 CALL LINK("PRINT",1,1,P$
  ):: IF K=32 THEN 880 ELSE RE
  TURN
940 !
950 ! PUT FEATURE ON FACE &

```

```

MOVE AROUND
960 CALL HILITE(X,16,9):: R=
  96-H*4 :: C=144-W*4
970 CALL LINK("WINDOW",R,C,R
  +8*H+1,C+8*W+1)
980 CALL LINK("PRINT",1,1,P$
  ):: CALL KEY(0,K,S):: CALL L
  INK("PRINT",1,1,P$):: IF S=0
  THEN SPD=1 :: GOTO 980
990 IF K=67 THEN GOSUB 1160
  :: GOTO 980
1000 TX=1+POS(X$,CHR$(K),1):
  : ON TX GOTO 980,1010,1030,1
  020,1010,1020,1010,1030,1020
  ,1070
1010 C=C-SPD :: GOTO 1030
1020 C=C+SPD
1030 R=R+SPD*((TX<5)-(TX>6))
1040 IF C<49 THEN C=49 ELSE
  IF C>240-W*8 THEN C=240-W*8
1050 IF R<1 THEN R=1 ELSE IF
  R>192-H*8 THEN R=192-H*8
1060 SPD=SPD*1.4 :: GOTO 970
1070 CALL LOAD(11112,224)::
  CALL LINK("PENHUE",HUE(HN),B
  KG):: CALL LINK("PRINT",1,1,
  P$)
1080 CALL LOAD(11110,64,72):
  : GOTO 260
1090 !
1100 ! SCREEN DUMP
1110 CALL LINK("DUMP"):: GOT
  O 260
1120 !
1130 ! CLEAR FOR NEW FACE
1140 CALL HILITE(X,2,1):: GO
  TO 220
1150 !
1160 ! CHANGE COLOR
1170 HN=HN+1+15*(HN=15):: CA
  LL LINK("SPRITE",1,1,HUE(HN)
  ,MO*8+8,9):: FOR I=1 TO 50 :
  : NEXT I :: RETURN
1180 !
1190 SUB HILITE(X,F,B):: CAL
  L LINK("WINDOW"):: CALL LINK
  ("PU"):: CALL LINK("PENHUE",
  F,B):: CALL LINK("FILL",X*8-
  7,1,X*8+1,30):: CALL LINK("P
  ENHUE",1,1):: SUBEND

```