This area is for the puzzle game called "Entrap" written by Harmonv as the prize winning entry in the LB summer 2006 contest and posted in Wikispaces with the author's permission.

In August, 2006, I added a "wizard" to Harmon's game. The wizard changes the cursor to a cross-hairs style when the player points at a cell which is an efficient move. In other words, it solves the puzzle for you. This should be made optional so the player can try without the "training wheels" of the cross-hair cursor. Other changes may be needed. Feel free to make them when you download the code.

To download the complete distribution of the original puzzle including sound effects, click here:

Entrap.zip

- Details
- Download
- 36 KB

The following code can be cut and pasted into the Entrap folder if you would like to have the sound effects with the wizard version.

Grahame King

The Zip file has been updated. The update includes the version with Grahame's wizard, named Entrapw.bas, plus an updated "readme" file and I've added a "New Game" button. Get it by downloading

Entrap11.zip

- Details
- Download
- 40 KB

. Enjoy!

```
' Program Entrap (with wizard)
' The first version of this program was written in machine
' language and used to demonstrate Digital Group equipment.
' MAXI-BASIC version by Hal Knippenberg, April 1979
```

```
' Liberty Basic version by Harmonv, July 2006
' Wizard added by Grahame King, August 2006
nomainwin
global w$, cr$, move, blu, yelo
blu = 1 : yelo = 0 : gwin = 0
cr$ = chr$(13)
dim dist(511,9) ' array to store information about game positions

' the 1st index represents the postion as sum bit(i)*2^cell(i) over 9
cells
    ' where bit(i) is set for blue and off for yellow
    ' the value of an element of this array is,
    ' when 2nd dim = 0 :-
        ' 0 - no information

' n - positive integer represents the least number of moves to win+1
            ' so the won game postion has value 1
        ' -1 - for the lost position (all yellow)   - not implimented
    ' when 2nd dim, k = 1 to 9, representing the cell clicked :-

' the position code of the position where that click in the kth cell t
akes you
on error goto [educateWizard]
open "wiz.txt" for input as #wiz
on error goto [quit]
' read in wizardry
for i = 1 to 511
    for j = 0 to 9
        input #wiz, dist(i,j)
    next j
next i
close #wiz
goto [newgame]

[educateWizard]
    ' start at won postion and trace back to earlier positions
    ' storing the number of moves in dist()
    for i = 1 to 9
        if i<>5 then
            b(i) = 1
        else
            b(i) = 0
        end if
    next i
    dist(posnCode(),0) = 1
    moves = 1
```

```
    notice
"Wizard text not found - the wizard will rewrite it.  This takes time!
"
    textbox #mainWiz.stxt  20,20,WindowWidth/2,WindowHeight/4
    stylebits #mainWiz, 0 ,0, _WS_EX_TOPMOST, 0
    open "Wizard at work!" for window_popup as #mainWiz
    #mainWiz "trapclose [done]"
    #mainWiz.stxt "Wizard at work!"+chr$(13)+chr$(10)+_
                  "This is a one-off operation."+chr$(13)+chr$(10)+_
                  "Please wait."
    cursor hourglass
    call trainWiz moves

    [done]
    open "wiz.txt" for output as #wiz
    for i = 1 to 511
      for j = 0 to 9
        print #wiz, dist(i,j);
        if j<>9 then print #wiz, ","; else print #wiz,""
      next j
    next i
    playwave "yeehaw.wav", async
    close #wiz
    close #mainWiz
    cursor normal

[newgame]
  if gwin=1 then close #g
  tm = 0  ' total moves
  call boardsetup
  UpperLeftX = 40 : UpperLeftY = 20
  WindowWidth = 500 : WindowHeight = 350
  menu #g, "File", "New Game", [newgame],_
                   "Options", [options],_
                    |, "Exit", [quit]
  menu #g, "Help", "Entrap Rules", showrules, "About Entrap", about
  button #g,b1, "New Game", [newgame], UL, 320, 220
  button #g.b2, "Done", [quit], UL, 340, 250 ' {, width, height}
  statictext #g.s1, "Entrap", 320, 40, 160, 60
  statictext #g.s2, "Want the rules?"+cr$+
"Check Help menu.", 300, 120, 190, 80
  open "Entrap" for graphics_nsb as #g
  gwin=1  ' set window flag to active
  print #g, "color black ; down"
  print #g.s1, "!font Arial 14"
  print #g.s2, "!font Arial 14"
```

```
  print #g, "flush"
  print #g, "trapclose [quit]"
  print #g, "when leftButtonDown [clicked]"
  #g "when mouseMove [mouseOver]"
  gosub [showgboard]
wait

' start of main loop
[clicked]
  gosub [getmousemove]
  if b(move)=yelo then
    #g.s2, "  Illegal move."+cr$+"  Try Again."
    wait
  else
    #g.s2, ""
  end if
  tm = tm + 1
  call updateboard move
  gosub [showgboard]
  call winorlose
  if w$<>"" then [endgame]
  viewcell = 0 : move = 0
' so mouseover will be activated for cell clicked on
wait

[showgboard]
  #g.s1, "Total Moves"+cr$+"      ";tm
  for i = 1 to 9
    x = 10+90*((i-1) mod 3)
    y = 10+90*int((i-1)/3)
    #g, "place ";x;" ";y
    if b(i)=yelo then
      #g, "backcolor yellow"
    else
      #g, "backcolor blue"
    end if
    #g, "boxfilled ";x+80;" ";y+80
    #g, "flush"
  next i
return

[getmousemove]
  move = 0
  if abs(MouseX- 50)<40 and abs(MouseY- 50)<40 then move = 1
  if abs(MouseX-140)<40 and abs(MouseY- 50)<40 then move = 2
  if abs(MouseX-230)<40 and abs(MouseY- 50)<40 then move = 3
```

```
  if abs(MouseX- 50)<40 and abs(MouseY-140)<40 then move = 4
  if abs(MouseX-140)<40 and abs(MouseY-140)<40 then move = 5
  if abs(MouseX-230)<40 and abs(MouseY-140)<40 then move = 6
  if abs(MouseX- 50)<40 and abs(MouseY-230)<40 then move = 7
  if abs(MouseX-140)<40 and abs(MouseY-230)<40 then move = 8
  if abs(MouseX-230)<40 and abs(MouseY-230)<40 then move = 9
return

[mouseOver]
    gosub [getmousemove]
    if move = viewcell then wait ' no need to check same move twice
    if move  then
        viewcell = move
        if b(move) then ' if legal move
            if winningMove() then
                cursor crosshair
            else
                cursor normal
            end if
        else
                cursor normal
        end if
    else
        cursor normal
        viewcell = 0
    end if
wait

[endgame]
  if w$="won" then
    playwave "yeehaw.wav", async
    #g.s2, " ** You WON!! **"+cr$+"Congratulations!!"
  end if
  if w$="lost" then
    playwave "wlaugh.wav", async
    #g.s2, "   -- You lost --"+cr$+"Better luck next time."
  end if
  wait

[options]
'#optionsDialog "!show"
wait

[quit]
if gwin=1 then close #g
end
```

```
' ----- end of main routine -----

sub boardsetup
  ok = 0
  do
    for i = 1 to 9
      t = rnd(1)
      if t<0.666 then
        b(i) = yelo
      else
        b(i) = blu
        ok = 1
      end if
    next i
  loop until ok=1
end sub

sub winorlose
  sum = 0
  w$ = ""
  for i = 1 to 9
    sum = sum + b(i)
  next i
  if sum=0 then w$="lost"
  if (sum=8) and (b(5)=yelo) then w$="won"
end sub

sub updateboard move
  select case move
    case 1: s=1245
    case 2: s=123
    case 3: s=2356
    case 4: s=147
    case 5: s=24568
    case 6: s=369
    case 7: s=4578
    case 8: s=789
    case 9: s=5689
  end select
  for i = 1 to len(str$(s))
    t = s mod 10 : b(t) = 1 - b(t)
    s = int(s/10)
  next i
end sub

sub showrules
```

```
  playwave "help.wav", async
  n$ = space$(10)+"Entrap Instructions"+cr$+cr$
  n$=n$+ "This game is played on a 3-by-3 grid. "+cr$
  n$=n$+ "When the game starts the board will be "+cr$
  n$=n$+ "filled with yellow and blue squares. "+cr$+cr$
  n$=n$+ "To change the board, click on any of the "+cr$
  n$=n$+ "blue squares. "+cr$+cr$
  n$=n$+ "To win, make the center square yellow  "+cr$
  n$=n$+ "and all the other squares blue."+cr$+cr$
  n$=n$+ "If all the squares turn yellow, you lose. "+cr$+cr$
  n$=n$+ "As you pick squares, the board will change "+cr$
  n$=n$+ "based on these rule(s): "+cr$+cr$
  n$=n$+ "If you pick a corner square, the 4 squares "+cr$
  n$=n$+ "in the area will change."+cr$+cr$
  n$=n$+ "Pick one in the middle of an edge and all  "+cr$
  n$=n$+ "three squares along the edge will change. "+cr$+cr$
  n$=n$+ "Pick the center square and the center plus "+cr$
  n$=n$+ "the 4 middle edge squares will change.  "+cr$+cr$
  n$=n$+ "To end the game, click the [Done] button. "+cr$+cr$
  n$=n$+ "Good Luck !"+cr$
  notice n$
end sub

sub about
  n$="About Entrap"+cr$
  n$=n$+ "The first version of this program was "+cr$
  n$=n$+ "written in machine language and was used "+cr$
  n$=n$+ "to demonstrate Digital Group equipment. "+cr$+cr$
  n$=n$+ "MAXI-BASIC version by Hal Knippenberg, Apr 1979 "+cr$+cr$
  n$=n$+ "Liberty Basic version by HarmonV, July 2006 "+cr$
  n$=n$+ "Wizard by Grahame King, August 2006 "+cr$
  notice n$
end sub

function winningMove()

' returns 1 plus number of moves to win for the selected move (global)
 in current posn
    ' returns 0 if move is poor
    pc = posnCode()
    if move then
            posnext = dist(pc,move)
            valnext = dist(posnext,0) ' value of posn one move ahead
        ' compare to other legal moves if any
        test = valnext
        asgoodasanyother = test
```

```
        for i  = 1 to 9
            if b(i) and i<>move then
                test = dist(dist(pc,i),0)
                if test>0 then
 asgoodasanyother = min(asgoodasanyother,test)
            end if
        next i
        if valnext>asgoodasanyother then
            winningMove = 0
        else
            winningMove = valnext
        end if
    end if
end function
function losingMove()
    ' stub - you have to protect yourself from wiping out
    losingMove = 0
end function
sub trainWiz moves
    ' recursive subroutine

' exhaustively searches for all winnable positions by regressive moves
    ' working back from the target position
    tpc = posnCode()
'print tpc, posnUncode$(tpc)
    for i = 1 to 9

' if cell is yellow it was a possible legal move (i.e. it was blue)
        if b(i)=yelo then
            call updateboard i  ' regressive move
            moves = moves+1
            pc = posnCode()
            if dist(pc,0)=0  then
                dist(pc,0) = moves
                dist(pc,i) = tpc
                call trainWiz moves ' delve further
            else
                ' position has already been found  and recorded
                if moves<dist(pc,0) then ' store the shorter route
                    dist(pc,0) = moves
                    dist(pc,i) = tpc
                    call trainWiz moves ' delve further
                end if
                if moves=dist(pc,0) then
                    dist(pc,i) =
 tpc  ' store possibly new equally good move
```

```
                end if
            end if
            call updateboard i  ' take move back
            moves = moves-1
        end if
    next i
end sub
function posnCode()

' "binary" encoding function to find index of position for storing inf
o
    for i = 1 to 9
        posnCode = posnCode+b(i)*2^(i-1)
    next i
end function
function posnUncode$(pc)
' converts the position code to a string of 9 bits representing the bl
ue/yellow cells
' may be useful in debugging
cell = 0
for cell = 1 to 9
    b= pc mod 2
    pc = (pc-b)/2
    if cell>1 then posnUncode$ = posnUncode$+","
    posnUncode$ = posnUncode$+str$(b)
next cell
end function
```