

```

{*****
* SoftScrp : Demonstrates soft scrolling on EGA and VGA cards.
*
**-----**
* Author      : Michael Tischer
* Developed on : 01/23/90
* Last update  : 02/20/92
*****}

program SoftScrp;

uses dos,                      { Add units }
    crt;

const SLOW      = 1;           { Speed constant for ShowScrlText() }
      MEDIUM   = 2;
      FAST      = 3;

const PCOLR     = $5E;        { Yellow from lilac palette }
      PCOLRL    = $5F;        { White from lilac palette }
      CWIDTH    = 8;          { Character width in pixels }
      CHEIGHT   = 14;         { Character height in scan lines }
      COLUMNS  = 216;        { Columns per row in video RAM }
      BANDSIZE  = 10800;      { Band size }
      BANDNUM   = 3;          { Number of bands }
      MAXLEN    = 61;         { Maximum number of characters }
      STARTR    = 6;          { Starting character row on screen }

      CrtAttr   = $3C0;        { CRT attribute controller register }
      CrtStatus = $3da;        { Status port }
      CrtAdr    = $3d4;        { Monitor address port }

type VRAM = array[1..BANDNUM,1..25,1..COLUMNS] of word; { Video RAM }
      VPTR = ^VRAM;          { Pointer to video RAM }

var vp          : vptr;        { Pointer to video RAM }

type CRDTYPE = ( EGA, VGA, NEITHERNOR ); { Video card type }

procedure CLI; inline( $FA ); { Disable interrupts }
procedure STI; inline( $FB ); { Enable interrupts }

{*****
* SetOrigin : Specifies the visible part of video RAM for
*             programming the video controller.
*
**-----**
* Input      : BAND      = Number of band to be displayed (1-5)
*             COLUMN,    = Number of columns and rows displayed in the
*             SCROW      upper-left corner of the screen (origin=0/0)
*             PIXX,      = Pixel offsets
*             PIXY
*****}

procedure SetOrigin( band, column, scrow, pixx, pixy : byte );

var offset : integer;          { Offset of video RAM start }
    ch : char;

begin
    offset := ( BANDSIZE div 2 ) * (band-1) + scrow * COLUMNS + column;

    {-- Execute vertical rescan and wait for end -----}

    repeat until port[CrtStatus] and 8 = 8;
    repeat until port[CrtStatus] and 8 = 0;

    {-- Write offset for start of video RAM to registers 0CH and 0DH,
    {-- after ensuring that next screen layout is valid
    }

    CLI;                          { Disable interrupts }
    portw[ CrtAdr ] := hi( offset ) shl 8 + $0c;
    portw[ CrtAdr ] := lo( offset ) shl 8 + $0d;
    STI;

    {-- While waiting for next rescan, write new -----}
    {-- pixel offset and set up new starting address -----}

    repeat until port[CrtStatus] and 8 = 8;

```

```

{-- Write pixel offsets in register 08H or -----}
{-- register 13H of the attribute controller -----}

CLI;
portw[ CrtAdr ] := pixy shl 8 + $08;
port[ CrtAttr ] := $13 or $20;           { Access attribute }
port[ CrtAttr ] := pixx;                 { controller by bytes }
STI;                                     { Enable interrupts }

                                           { ch := readkey; }

end;

{*****}
* PrintChar : Creates a character within the visible range of
*             video RAM.
*-----*
* Input      : THECHAR = Character to be created
*             BAND     = Band number (0-4)
*             COLUMN   = Column in video RAM at which the column
*                       should begin
* Info       : The character can be moved in the visible range of
*             the screen by calling SmoothLeft.
*             The character is created from a 14x8 matrix, based on
*             the EGA/VGA ROM font.
*-----*
*****}

procedure PrintChar( thechar : char; band, column : byte );

type FDEF = array[0..255,1..14] of byte;           { Font array }
TPTR = ^FDEF;                                       { Pointer to font }

var Regs : Registers;                               { Registers for interrupt call }
    ch   : char;                                     { Character pixels }
    i, k,                               { Loop counter }
    BMask : byte;                                   { Bit mask for character design }

const fptr : TPTR = NIL;                           { Pointer to ROM font }

begin
  if fptr = NIL then                                { Pointer to font already set? }
    begin                                           { No }
      Regs.AH := $11;                               { Call video BIOS function }
      Regs.AL := $30;                               { 11H, sub-function 30H }
      Regs.BH := 2;                                 { Get pointer to 8x14 font }
      intr( $10, Regs );
      fptr := ptr( Regs.ES, Regs.BP );               { Set pointers }
    end;
  {-- Generate character line by line -----}

  for i := 1 to CHEIGHT do
    begin
      BMask := fptr^[ord(thechar),i]; { Get bit pattern for one line }
      for k := 1 to CWIDTH do          { Set individual bits }
        begin
          if BMask and 128 = 128 then ch := #219 { Set pixel }
          else ch := #32; { Unset pixel }
          vp^[band, STARTR+i, (column-1)*CWIDTH+k] :=
            ord(ch) + PCOLR shl 8;
          BMask := BMask shl 1; { Process next bit }
        end;
      end;
    end;
  end;

{*****}
* IsEgaVga : Determines whether an EGA or a VGA card is installed.
*-----*
* Input      : None
* Output     : EGA, VGA or NEITHERNOR
*-----*
*****}

function IsEgaVga : CRDTYPE;

var Regs : Registers; { Processor registers for interrupt call }

begin
  Regs.AX := $1a00; { Function 1AH applies to VGA only }

```

```

Intr( $10, Regs );
if ( Regs.AL = $1a ) then           { Is the function available? }
    IsEgaVga := VGA
else
    begin
        Regs.ah := $12;             { Call function 12H, }
        Regs.bl := $10;             { sub-function 10H }
        intr($10, Regs);           { Call video BIOS }
        if ( Regs.bl <> $10 ) then IsEgaVga := EGA
        else IsEgaVga := NEITHERNOR;
    end;
end;

```

```

{*****
* ShowScrtext : Scrolls text on the screen. *
**-----**
* Input      : STEXT = String of text for scrolling *
*              SPEED = Scroll speed (see SLOW, FAST constants) *
*              VC    = Video card type (EGA or VGA) *
*****}

```

```

procedure ShowScrText( stext : string; speed : byte; vc : crdtype );

```

```

var band,                { Current band }
    column,              { Current column }
    index,               { Index to string to be created }
    len,                 { Length of text to be displayed }
    i, k : integer;      { Loop counter }
    step,                { Increment }
    uplimit : byte;      { Number of loop executions }
    Regs : Registers;    { Processor registers for interrupt call }

```

```

const steptable : array [EGA..VGA,1..3,1..10] of byte =
(
    (
        { EGA step values }
        ( 0, 1, 2, 3, 4, 5, 6, 7, 255, 255 ),
        ( 0, 2, 4, 6, 255, 255, 255, 255, 255, 255 ),
        ( 0, 4, 255, 255, 255, 255, 255, 255, 255, 255 )
    ),
    (
        { VGA step values }
        ( 8, 0, 1, 2, 3, 4, 5, 6, 7, 255 ),
        ( 8, 2, 5, 255, 255, 255, 255, 255, 255, 255 ),
        ( 8, 3, 255, 255, 255, 255, 255, 255, 255, 255 )
    )
);

```

```

begin
    vp := ptr( $B800, $0000 );      { Set pointer to video RAM }

```

```

{-- Fill entire video RAM with blank spaces -----}

```

```

for index := 1 to BANDNUM do
    for i := 1 to 25 do
        for k := 1 to COLUMNS do
            vp^[ index, i, k ] := PCOLR shl 8 + 32;

```

```

{-- Draw horizontal bands -----}

```

```

for k := 1 to BANDNUM do
    for i := 1 to COLUMNS do
        begin
            vp^[ k, STARTR-2, i ] := ord('í') + PCOLR1 shl 8;
            vp^[ k, STARTR + CHEIGHT + 2, i ] := ord('í') + PCOLR1 shl 8;
        end;

```

```

gotoxy( 1, 1 );      { Remove cursor when scrolling the screen }

```

```

{-- Set screen border color -----}

```

```

Regs.AH := $10;      { Function number: Set screen border color }
Regs.AL := $01;      { Sub-function number }
Regs.BH := PCOLR shr 4; { border color }
intr( $10, Regs );

```

```

{-- Place number of columns per row in video RAM from COLUMNS -----}

```

```

portw[ CrtAdr ] := ( COLUMNS div 2 ) shl 8 + $13;

```

```

}-- Place scrolling text in video RAM -----}

if length( stext ) > MAXLEN then len := MAXLEN
                                else len := length( stext );

column := 1;
band   := 1;
index  := 1;
while ( index <= len ) do
begin
    PrintChar( stext[index], band, column );      { Draw characters }
    inc( column );                                { in next column }
    inc( index );                                  { Process next character }
    if ( column > ( COLUMNS div CWIDTH ) ) then { Band change? }
    begin                                          { Yes }
        column := 1;                             { Restart in column 1 }
        inc( band );                             { Next band }
        dec( index, ( 80 div CWIDTH ) ); { Character one page back }
    end
end;

}-- Move scroll text from right to left on the screen -----}

column := 0;                                     { Start in column 0 with band 1 }
band   := 1;
for i := 1 to (len-( 80 div CWIDTH )) * CWIDTH do
begin
    k := 1;
    while ( steptable[vc, speed, k] <> 255 ) do
    begin
        SetOrigin( band, column, 0, steptable[vc, speed, k], 0 );
        inc( k );
    end;

    inc( column );                                { Next column }
    if ( column = COLUMNS - 80 ) then           { Band change? }
    begin                                          { Yes }
        column := 0;                             { Restart in column 0 }
        inc( band );                             { Increment band }
    end
end;

}-- Revert to 80 characters per row in video RAM -----}

portw[ CrtAdr ] := 40 shl 8 + $13;

}-- Reset border colors -----}

Regs.AH := $10;                                { Function number: Set border color }
Regs.AL := $01;                                { Sub-function number }
Regs.BH := 0;                                  { Black border }
intr( $10, Regs );

if ( vc = EGA ) then                           { Revert to default }
    SetOrigin( 1, 0, 0, 0, 0 )                  { for EGA }
else
    SetOrigin( 1, 0, 0, 8, 0 );                  { for VGA }
ClrScr;
end;

{*****}
{**                M A I N   P R O G R A M                **}
{*****}

var  ch : char; i : integer;
     vc : crdtype;

begin
    vc := IsEgaVga;                             { Get video card type }
    if vc = EGA then halt;
    if ( vc = NEITHERNOR ) then
    begin
        writeln( 'SOFSCRIP - (c) 1992 by Michael Tischer' );
        writeln( #13#10'Warning: No EGA or VGA card found' );
    end
    else
        ShowScrLText( '++ PC Intern.....Published by Abacus ++'+
            ' ', FAST, vc );
    while keypressed do

```

```
ch := readkey;  
  
end.
```