MANDELBROT:
MANDELBR.PAS

MSX CLUB MAGAZINE 26

F.Scheffer Eindhoven

Scanned, ocr'ed and converted to PDFby HansO, 2001

Het programma MANDELBR.PAS is een TURBO PASCAL- programma waarin een aantal extra grafische procedures worden gebruikt. Deze procedures worden hieronder toegelicht.
Het programma dient gecompileerd te worden, dan heb je een programma dat je vanuit MSX-DOS kan opstarten.
Het SOURCE-programma vindt U terug onder de naam "MANDELBR.PAS", de COM-file heet "MANDELBR.COM".

procedure msxbios:
Als deze procedure wordt aangeroepen, wordt een CALL uitgevoerd naar de bios-ROM op de plaats van entry (zie magazine 21).

procedure msx2bios:
Deze procedure is gelijk aan die van hier boven, maar dan wordt een CALL uitgevoerd naar de extended-ROM in de MSX2.
Hierna volgen procedures die hetzelfde doen als de BASIC equivalenten. Ze maken gebruik van de bios-procedures.

procedure proces_point:
Deze procedure zet een kleur op een x-coördinaat om naar bytes zoals die in het videogeheugen staan en zet die in een buffer die een regel voorsteld. Die kleur bestaat uit 4 bits (O tot 15), een nibble. Voorx = 0 komt die nibble op de bits 7-4 van byte O terecht, voor x = 257 komt die nibble 3-0 van byte 128 terecht.

procedure moveline:
Deze procedure stuurt de buffer die met proces_point is gevuld, naar regel y in het videogeheugen. Dit gebeurt precies zoals in magazine 21 met de karakters:
- reken adres uit
- zet het write adres met CALL new_set_write in de BIOS
- stuur de bytes naar poort 98 (hexadecimaal)

het programma:
• Met de minimale en maximale x- of y-waarden wordt een venster over de fractal gelegd.
• Er wordt gevraagd hoeveel punten horizontaal worden uitgerekend, dan wordt het aantal vertikale punten uitgerekend.

• Er wordt gevraagd hoe diep er wordt gerekend.
• Als er gerekend wordt, dan kan het programma onderbroken worden met een toetsdruk.
plaatjes:
• het cijfer voor de punt geeft aan hoe diep er gerekend is, het cijfer achter de punt geeft aan welk scherm er gebruikt is, in dit geval 7.
• in mandel22 is het venster als volgt:
x van -1.5 tot 1, y van -1.5 tot 1.5, bij de andere is meer van de fractal zichtbaar.
mandel35 is nog niet klaar, dit gaat 14 uur duren, het venster is als volgt: x van -1.5 tot 1, y van -1.8 tot 1.8.

De fractal wordt in page 1 gemaakt, als de berekening af is of onderbroken wordt, dan wordt teruggeschakeld naar page 0.

Verlaat MSX-DOS als het programma beëindigd is. RUN daarna het volgende programmaatje:

```
10 screen 7: set page 1,1
20 bsave "naam.sc7",0,&hd3ff,s
30 set page 0,0
```

```
program mandelbrot;

label einde;

var

   regA, regBC, regDE, regHL, regF, regIX, regIY: integer;

   dppage: byte absolute $FAF5;
   acpage: byte absolute $FAF6;

{ procedure om MSX-bios aan te roepen (zie magazine 21) }

procedure msxbios(entry: integer);

begin
   inline(
      $3A/regA/                          { LD A ,(regA)  }
      $ED/$4B/regBC/                     { LD BC,(regBC) }
      $ED/$5B/regDE/                     { LD DE,(regDE) }
      $2A/regHL/                         { LD HL,(regHL) }
      $DD/$2A/entry/                     { LD IX,(entry) }
      $FD/$2A/$C0/$FC/                   { LD IY,EXPTBL  }
      $CD/$1C/$00/                       { CALL CALSLT   }
      $32/regA/                          { LD (regA) ,A  }
      $ED/$43/regBC/                     { LD (regBC),BC }
      $ED/$53/regDE/                     { LD (regDE),DE }
      $22/regHL/                         { LD (regHL),HL }
      $DD/$22/regIX/                     { LD (regIX),IX }
      $FD/$22/regIY/                     { LD (regIY),IY }
      $F5/                               { PUSH AF       }
      $E1/                               { POP  HL       }
      $22/regF/                          { LD (regF) ,HL }
      $AF/                               { XOR  A        }
      $32/regA+1/
      $32/regF+1/
      $FB)                               { EI }
end;

{ procedure voert call uit naar extended ROM in MSX2 }

procedure msx2bios(entry: integer);

begin
   inline(
      $3A/regA/                          { LD A ,(regA)  }
      $ED/$4B/regBC/                     { LD BC,(regBC) }
      $ED/$5B/regDE/                     { LD DE,(regDE) }
      $2A/regHL/                         { LD HL,(regHL) }
      $DD/$2A/entry/                     { LD IX,(entry) }
      $FD/$2A/$F7/$FA/                   { LD IY,EXBRSA  }
      $CD/$1C/$00/                       { CALL CALSLT   }
      $32/regA/                          { LD (regA) ,A  }
      $ED/$43/regBC/                     { LD (regBC),BC }
      $ED/$53/regDE/                     { LD (regDE),DE }
```

```
      $22/regHL/                          { LD (regHL),HL }
      $DD/$22/regIX/                      { LD (regIX),IX }
      $FD/$22/regIY/                      { LD (regIY),IY }

      $F5/                                { PUSH AF       }
      $E1/                                { POP  HL       }
      $22/regF/                           { LD (regF) ,HL }
      $AF/$32/regA+1/$32/regF+1/          { XOR  A        }
      $FB)                                { EI }
end;


procedure screen(mode: byte);
begin
   regA :=mode;
   msxbios($5f)
end;

procedure setpage(dp,ap: byte);
begin
   dppage:=dp;
   acpage:=ap;
   msx2bios($013D)
end;

procedure cls;
begin
   msxbios($C3)
end;


procedure kilbuf;
begin
   msxbios($156);
   write(chr(0))
end;


var x, x0, x1, x2, y, y0, y1, y2, z, zk, dx, dy: real;
    i, j, ni, n1, n2, s1: integer;
    k: byte;
    buffer: array [0..255] of byte;


{ procedure die een kleur van een pixel omrekent
  naar een byte in een schermregel.
}
procedure process_point(x: integer; cl: byte);
begin
   inline($11/ buffer/$2A/  x/$CB/$3C/$CB/$1D/$38/$0D/$19/$7E/$E6/
          $F0/$77/$3A/ cl/$E6/$0F/$B6/$18/$0D/$19/$7E/$E6/$0F/$77/
          $3A/ cl/$87/$87/$87/$87/$B6/$77)
end;

{ procedure die een omgerekende regel naar het scherm stuurt
}
procedure move_line(y:byte);
```

```
begin
   inline($21/ buffer/$ED/$5B/y-1/$1E/$00/$01/$00/$01/$DD/$21/$5C/
          $00/$FD/$2A/$C0/$FC/$CD/$1C/$00/$FB)
end;


begin
   clrscr;
   writeln;
   writeln('Geef minimale X-waarde'); readln(x1);
   writeln('Geef maximale X-waarde'); readln(x2);
   writeln('Geef minimale Y-waarde'); readln(y1);
   writeln('Geef maximale Y-waarde'); readln(y2);
   writeln('Geef de breedte van het schermvenster');readln(s1);
   writeln('Geef de maximale lus-grootte');readln(ni);
   kilbuf;
   screen(7);
   setpage(1,1);
   cls;
   n1:=s1 div 2;
   n2:=n1*53 div 128;
   dx:=(x2-x1)/s1;
   dy:=(y2-y1)/(2*n2);
   y0:=(y2+y1)/2;
   for k:=0 to 255 do buffer[k]:=0;
   for i:=0 to n2-1 do begin
      x0:=x1;
      for j:=-n1 to n1-1 do begin
         x:=x0; y:=y0;
         k:=0;
         repeat
            z:=x;
            x:=x*x-y*y+x0;
            y:=2*y*z+y0;
            k:=k+1;
            if keypressed then goto einde;
         until ((k>=ni-1) or ((x*x+y*y)>16));
         process_point(j+256,k);
         x0:=x0+dx
      end;
      { even funktie }
      move_line(106+i);
      move_line(106-i);
      y0:=y0+dy
   end;

einde:
   setpage(0,0);
   screen(0);
end.
```