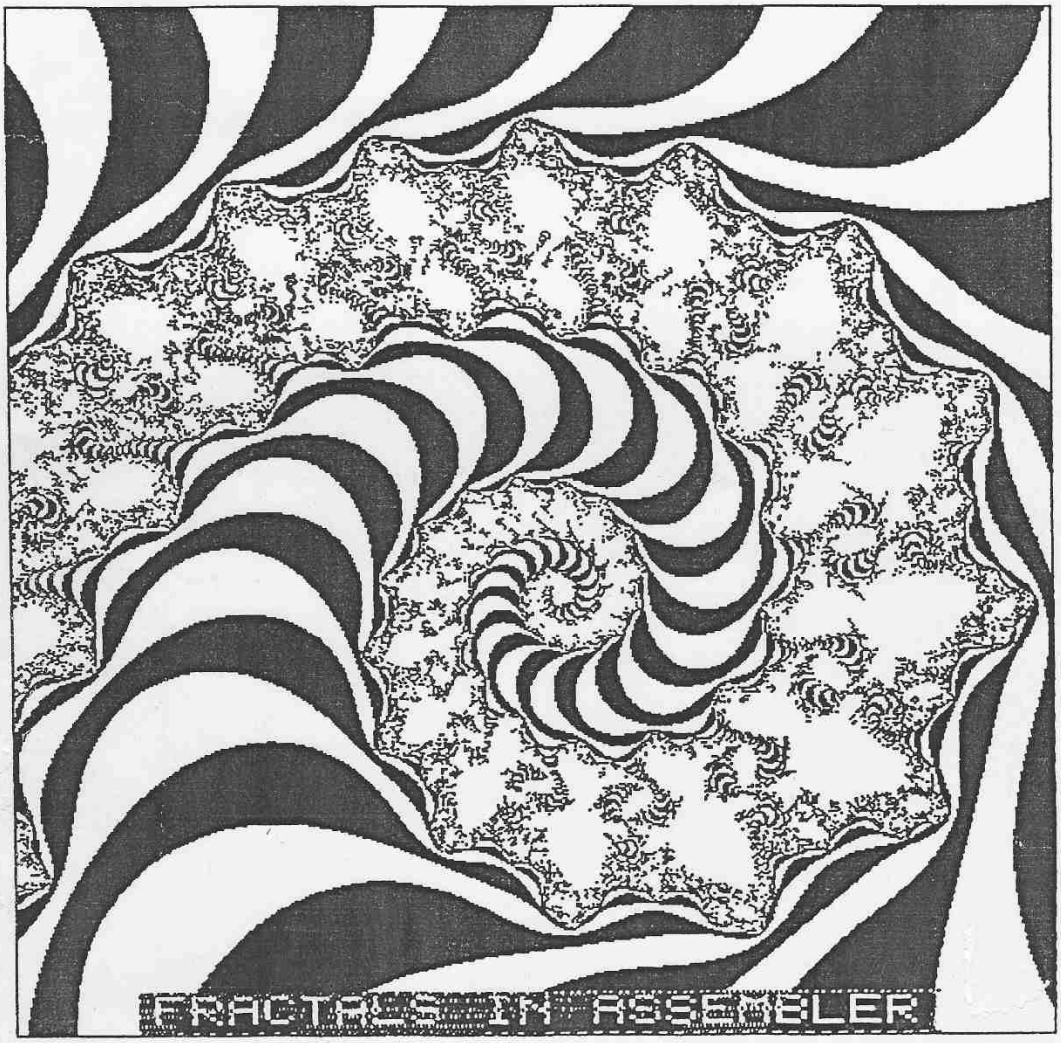
CompUser

INTERNATIONAL COMPUTING

ISSN: 0922-4203



Volume 1, Number 5, November 1988.

COMPUSER
International Computing

COMPUSER
Exchanging Computer Knowledge

CONTENTS VOLUME 1, NUMBER 5, NOVEMBER 1988.

TO SUBSCRIBE FOR 1989 SEND A CHEQUE OF HFL. 59,50 TO THE EDITORIAL OFFICE (EUROCHEQUE OR ON GIRO 841433 HFL. 50,=) BEFORE THE END OF 1988.
FOR COUNTRIES OUTSIDE EUROPE HFL. 119,50 ON BANKCHEQUE.
Remember number on backside !

COMPUSER is a magazine issued by:
COMPUSER International computing;
Exchanging computer knowledge.
ISSN: 0922-4203
Founded at Krimpen a.d. IJssel,
The Netherlands, by:
Willem L. van Pelt
Ir. Coen J. Boltjes.

Address all editorial correspondence to the editor COMPUSER:
Willem L. van Pelt
Jacob Jordaensstraat 15
NL - 2923 CK Krimpen a.d. IJssel.
The Netherlands.
Phone: (31) 01807 - 19881

Regular participants:
Fred. A. Behringer (W-Germany)
Hans Ebert (W-Germany)
Andrew Gregory (England)
Marc Lachaert (Belgium)
Iddy Oort (Holland)
Leif Rasmussen (Denmark)
Ronald van Vugt (Holland)

Drawings:
Herman Zondag (Holland)
Leo de Kok (Holland)

Translators:
Iddy Oort (Holland)
Cor Bergshoeff (Denmark)
Jan van Bakel (Holland)
Dirk Pickee (Holland)
Piet K. de Vries (Holland)

The articles to be published have to be written by the sender. All in English language.

The front page is completely designed by Leif Rasmussen, Denmark.

Fractals in assembler.

Leif Rasmussen

A complex number consist of a real part and an imaginary part, c = a + b*i where i per definition is the squareroot of -1. Benoit Mandelbrot found that C(n+1)=C(n)^2 + k where C(n) is the n.th iteration of the complex number C and k is a complex konstant. In the Mandelbrot set C(n) will never exceed 2. Whereas outside of the set it will quickly go towards infinity. The borderzone of the set will produce beautifull pictures if the real part of C is plotted along the x-axis and the imaginary part along the y-axis. A sort of visualising of the 1.5th dimension. Note that i is not needed in the computations (luckily).
A pseudo-code in fig. 1 show how simple the algorithm is. You can easily implement it on any computer in any language. Here is an example in 6502-assembly with a menu-prg. in Basic for EC65. This will produce a 512*512 pixel picture of the hole set from -2 to 1 real, -1.3 to 1.3 imag. in one hour. (I will put a 80386/87 machine on my list of Christmas wishes!)

Tips:

1) Do not dispair if you don't have a graphics card. You may still produce nice pictures on your matrix printer anyhow. See fig. 2

2) You may double the speed of this assembler prg. by making a separate arithmetic routine for each add, subtract and multiply to act on the numbers in situ, thus avoid all shuffle to and fro a special floating point area.

3) You may 'cheat a little to increase speed further by 'jumping over every second pixel when the iterations run through to maximum.

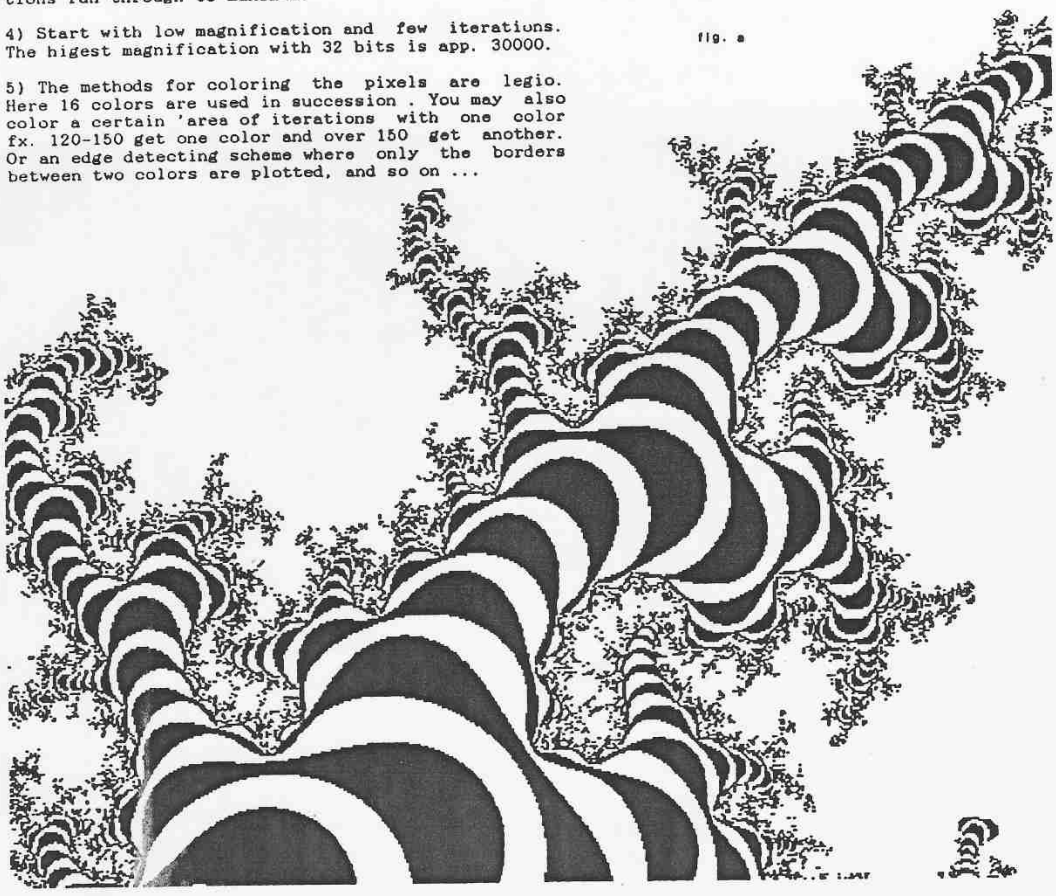
4) Start with low magnification and few iterations. The higest magnification with 32 bits is app. 30000.

5) The methods for coloring the pixels are legio. Here 16 colors are used in succession . You may also color a certain 'area of iterations with one color fx. 120-150 get one color and over 150 get another. Or an edge detecting scheme where only the borders between two colors are plotted, and so on ...

Lit. Peitgen & Richter,
"The beauty of Fractals"
Springer Verlag

Scientific American
Aug. 1985

fig. a



Pseudo-code for computing The Mandelbrot Set.

```
Define values for south-west corner of picture: Start.nr.real and Start.nr.imag
Define values for north-east corner of picture: End.nr.real and End.nr.imag
Compute horizontal increment/pixel as End-Start.nr.real/horizontal pixels
Compute vertical   increment/pixel as End-Start.nr.imag/vertical   pixels
Move pen to south-west corner of screen
FOR Current.nr.imag= Start.nr.imag
 .  Move pen to begin of line
 .     FOR Current.nr.real= Start.nr.real
 .       .  Temp.nr.real=Current.nr.real, Temp.nr.imag=Current.nr.imag
 .       .     FOR Count=0
 .       .       .  A=Temp.nr.real^2
 .       .       .  M=Temp.nr.imag^2
 .       .       .  Temp.nr.imag=2*Temp.nr.real*Temp.nr.imag+Current.nr.imag
 .       .       .  Temp.nr.real=A-M+Current.nr.real
 .       .       .  Count=Count+1
 .       .       Until Count=Maximum iterations or A+M>4
 .       .           If A+M>4 then Pencolor=Count
 .       .           Else Pencolor=black
 .       .  Paint pixel
 .       .  Current.nr.real=Current.nr.real+Horizontal.increment
 .       .  Move pen 1 pixel left
 .     Until pen reaches end of line
 .  Current.nr.imag=Current.nr.imag+Vertical.increment
 :  Move pen 1 line up
Until pen reaches topline
```

Fig 1.

"There are no straight lines
in the Universe...
only Chaos."

Fractals on a matrix printer:

```
Set up an array of fx. 256*256 bits
          in 32 lines of 256 bytes
          (put last pixelline in bit 0
           of arrays last byteline, put
           top pixelline in bit 7 of
           arrays top byteline)


proc. Init
          Bytepointer= 1.byte of 32.line
          Bitcounter = 0
          Bytecounter= 0
     Return

proc. Paint_Pixel
          If color= black then carry= 0
          else carry= 1
          Rotate carry right into byte
             pointed by Bytepointer.
          Inc. Bytepointer
          Inc. Bytecounter
          If Bytecounter= 256 then
           Bytecounter= 0 and
           Bytepointer to front of line
           Inc. Bitcounter
           If Bitcounter= 8 then
            Bitcounter= 0 and
            Bytepointer 1 line up
     Return

proc. Print
          Linecounter= 0
          Bytecounter= 0
          Bytepointer= 1.byte of 1.line

          Send cr,lf,Graph.mode,#chrs256
             Send 1 byte
             Until Bytecounter= 256
          Bytepointer 1 line down
          Until Linecounter= 32
     Return
```

Fig.2

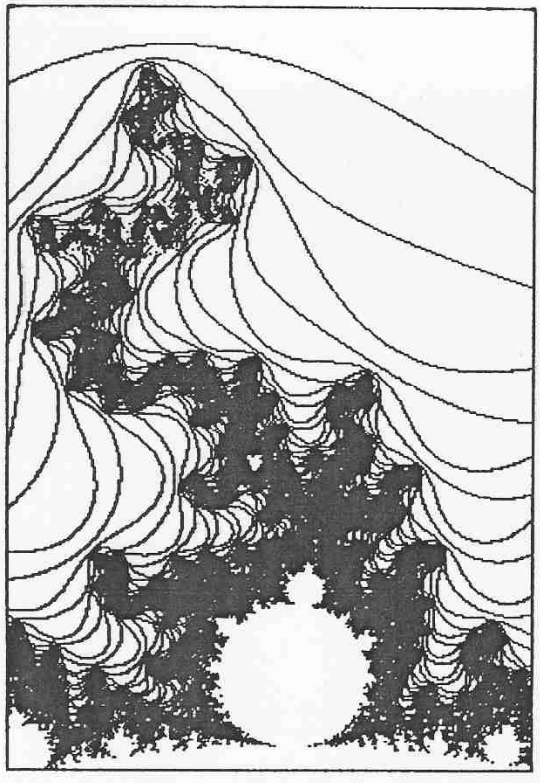


fig. b

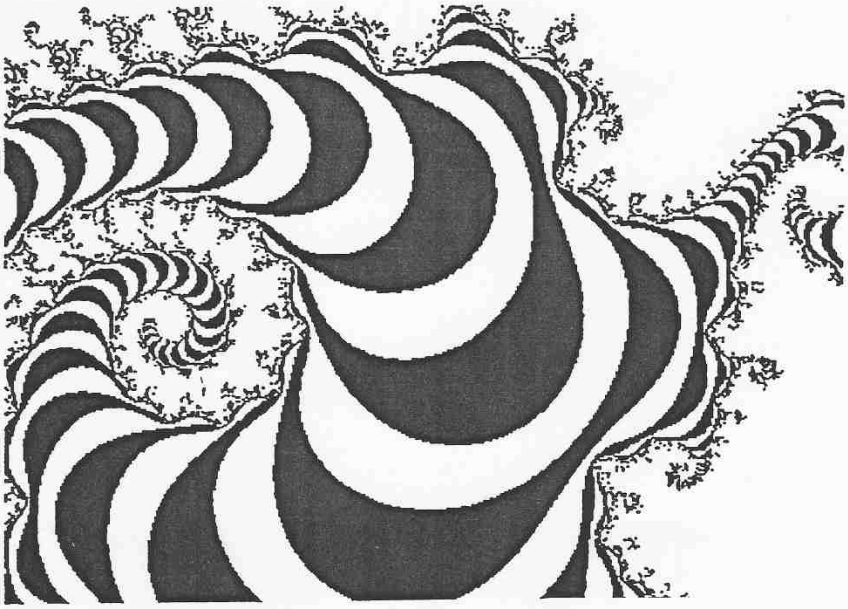


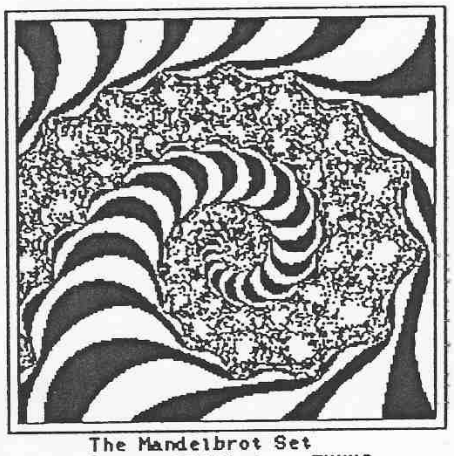
fig. e

```
    ------------------------------------------------
                    THE MANDELBROT SET
    a menu-program to set up the parameters for the
    machine-code program to compute the picture.
    ------------------------------------------------

PRINT#2,CHR$(18)"B14"              : clear graphic screen
                                   :
INPUT"Number_real_start : ";T      : Start value of complex nr. real part
  A=T : S=0                        : between -2 and 1
  GOSUB "MOV_PARAM"                :
INPUT"Number_imag_start : ";T      : Start value of complex nr. imaginary
  S=6                              : part, between -1.3 and 1.3
  GOSUB "MOV_PARAM"                :
INPUT"Number_real_end   : ";C      : End value of complex nr. real part
  T=ABS((C-A)/512)                 : Increment for 512*512 pixels
  S=12                             :
  GOSUB "MOV_PARAM"                :
INPUT"Iterations        : ";IT     : Nr. of iterations per pixel
  POKE O+55,IT                     : between 50 and 255
  POKE O+181,0                     :
DISK!"go 3A7E"                     : Compute picture
END                                :

LABEL "MOV_PARAM"
    ------------------------------------------------
    -   Move parameters to machine code program.    -
    -   Format of floating point binary numbers:    -
    -   Exponent (complemented)                     -
    -   Mantissa (msb) (uncomplemented)             -
    -   Mantissa                                    -
    -   Mantissa                                    -
    -   Mantissa (lsb)                              -
    -   Sign (0 if pos. 128 if neg.)                -
    -   Number_real_start = $00...$05  (flp.binary) -
    -   Number_imag_start = $06...$0b               -
    -   Increment         = $0o...$11               -
    -   Iterations        = $36...$37  (integer)    -
    ------------------------------------------------
  VS=PEEK(123)*256+PEEK(122)+2     : VARIABLES-POINTER
  O=12153                          : $2F79 = $0000 after page zero swab
 FOR I=0 TO 4                      :
  E=PEEK(VS+I)                     :
   IF I=1 AND E>=128 THEN POKE O+S+I+4,128      : If negative
   IF I=1 AND E<128 THEN E=E+128: POKE O+S+I+4,0: If positive
  POKEO+S+I,E                      :
 NEXT                              :
 RETURN                            :
```

```
┌──────────────────────────────────┐
|  MANDELBROT SET EC65 |
└──────────────────────────────────┘
```

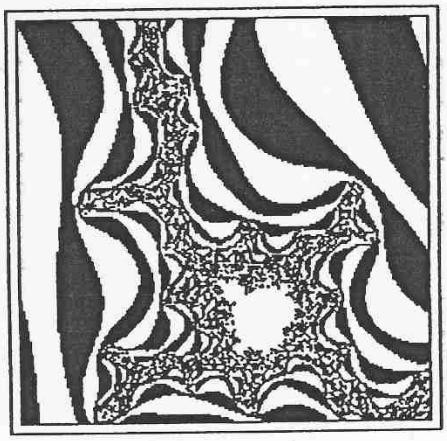
A program to compute the Mandelbrot set
in connection with a menu-program in
OS1 9KB Basic by Microsoft.
The Kolorator graphics card is set for
512*512 pixels.

```
3A7E                ORG   $3A7E

                    TTL   EC65 Mandelbrot set

                ┌──────────────────────────────────┐
                |  Page zero                       |
                └──────────────────────────────────┘
0000            ZERO    EQU   $00
0000            Xmin    EQU   $00     C_real_min
0006            Ymin    EQU   $06     C_imag_min
000C            DELTA   EQU   $0C     Increment
0012            X       EQU   $12     Current C_real
0018            Y       EQU   $18     Current C_imag
001E            A       EQU   $1E     (Temporary C_real)^2
0024            M       EQU   $24     (Temporary C_imag)^2
002A            B       EQU   $2A     Temporary C_real
0030            N       EQU   $30     Temporary C_imag
0036            COUNTER EQU   $36     Iteration counter
0037            ITERMAX EQU   $37       -       -    maximum

00AE            AFAC    EQU   $AE     floating point accumulator#1
00B6            BFAC    EQU   $B6       -       -       -     #2
00BC            SIGNCMP EQU   $BC     compare signs of #1 and #2
00BD            RNDBYTE EQU   $BD     rounding byte

                ┌──────────────────────────────────┐
                |  Basic floating point math       |
                └──────────────────────────────────┘
16C2            MINUS   EQU   $16C2   Basic float subtract
16DE            PLUS    EQU   $16DE     -     -   add
18F9            MULT    EQU   $18F9     -     -   multiply

                ┌──────────────────────────────────┐
                |  Graphic processor               |
                └──────────────────────────────────┘
E150            CMD     EQU   $E150   command register
E158            MSBX    EQU   $E158   pixel x-coordinate msb
E159            LSBX    EQU   $E159    -         -      lsb
E15A            MSBY    EQU   $E15A   pixel y-coordinate msb
E15B            LSBY    EQU   $E15B    -         -      lsb
E164            COLOR   EQU   $E164   pixel color


                ┌──────────────────────────────────┐
                |  FOR Y = Ymin                    |
                └──────────────────────────────────┘
3A7E A2 00      LAB1    LDXIM $00
3A80 8E 5A E1           STX   MSBY    y-coordinate = 0
3A83 8E 5B E1           STX   LSBY
3A86 B5 06      L11     LDAX  Ymin
3A88 95 18              STAX  Y
3A8A E8                 INX
3A8B E0 06              CPXIM $06
3A8D D0 F7              BNE   L11
                ┌──────────────────────────────────┐
                |  FOR X = Xmin                    |
                └──────────────────────────────────┘
3A8F A2 00      LAB2    LDXIM $00
3A91 8E 58 E1           STX   MSBX    x-coordinate = 0
3A94 8E 59 E1           STX   LSBX
```



The Mandelbrot Set
Real=-.74591 to -.74448
Imag=.11196 to .11339

fig. 1



The Mandelbrot Set
Real=-.1992 to -.1257
Imag=1.0148 to 1.0844

fig. 2

```
3A97 B5 00    L21     LDAX  Xmin
3A99 95 12            STAX  X
3A9B E8               INX
3A9C E0 06            CPXIM $06
3A9E D0 F7            BNE   L21
                ┌──────────────────────────────┐
                │  B = X : N = Y               │
                └──────────────────────────────┘
3AA0 A2 0C    LAB3    LDXIM $0C
3AA2 B5 11    L31     LDAX  X       -01
3AA4 95 29            STAX  B       -01
3AA6 CA               DEX
3AA7 D0 F9            BNE   L31
                ┌──────────────────────────────┐
                │  FOR COUNTER = 0 TO MAXIMUM  │
                └──────────────────────────────┘
3AA9 A9 00            LDAIM $00
3AAB 85 36            STA   COUNTER
                ┌──────────────────────────────┐
                │  A = B * B                   │
                └──────────────────────────────┘
3AAD A2 2A    LAB4    LDXIM B
3AAF 20 98 3B         JSR   MOVTO#1
3AB2 20 B1 3B         JSR   MOVTO#2
3AB5 20 F9 18         JSR   MULT
3AB8 A2 1E            LDXIM A
3ABA 20 D2 3B         JSR   MOVRESU
                ┌──────────────────────────────┐
                │  M = N * N                   │
                └──────────────────────────────┘
3ABD A2 30            LDXIM N
3ABF 20 98 3B         JSR   MOVTO#1
3AC2 20 B1 3B         JSR   MOVTO#2
3AC5 20 F9 18         JSR   MULT
3AC8 A2 24            LDXIM M
3ACA 20 D2 3B         JSR   MOVRESU
                ┌──────────────────────────────┐
                │  N = 2 * B * N + Y           │
                └──────────────────────────────┘
3ACD A2 2A            LDXIM B
3ACF 20 98 3B         JSR   MOVTO#1
3AD2 E6 AE            INC   AFAC    : 2*B
3AD4 A2 30            LDXIM N
3AD6 20 B1 3B         JSR   MOVTO#2
3AD9 20 F9 18         JSR   MULT    (2B)*N
3ADC A2 18            LDXIM Y
3ADE 20 B1 3B         JSR   MOVTO#2
3AE1 20 DE 16         JSR   PLUS    (2*B*N)+Y
3AE4 A2 30            LDXIM N
3AE6 20 D2 3B         JSR   MOVRESU
                ┌──────────────────────────────┐
                │  B = A - M + X               │
                └──────────────────────────────┘
3AE9 A2 24            LDXIM M
3AEB 20 98 3B         JSR   MOVTO#1
3AEE A2 1E            LDXIM A
3AF0 20 B1 3B         JSR   MOVTO#2
3AF3 20 C2 16         JSR   MINUS   A-M
3AF6 A2 12            LDXIM X
3AF8 20 B1 3B         JSR   MOVTO#2
3AFB 20 DE 16         JSR   PLUS    (A-M) + X
3AFE A2 2A            LDXIM B
3B00 20 D2 3B         JSR   MOVRESU
                ┌──────────────────────────────┐
                │  IF A + M > 4 THEN PAINT PIXEL │
                └──────────────────────────────┘
3B03 A2 1E            LDXIM A
3B05 20 98 3B         JSR   MOVTO#1
3B08 A2 24            LDXIM M
3B0A 20 B1 3B         JSR   MOVTO#2
3B0D 20 DE 16         JSR   PLUS    A+M
```

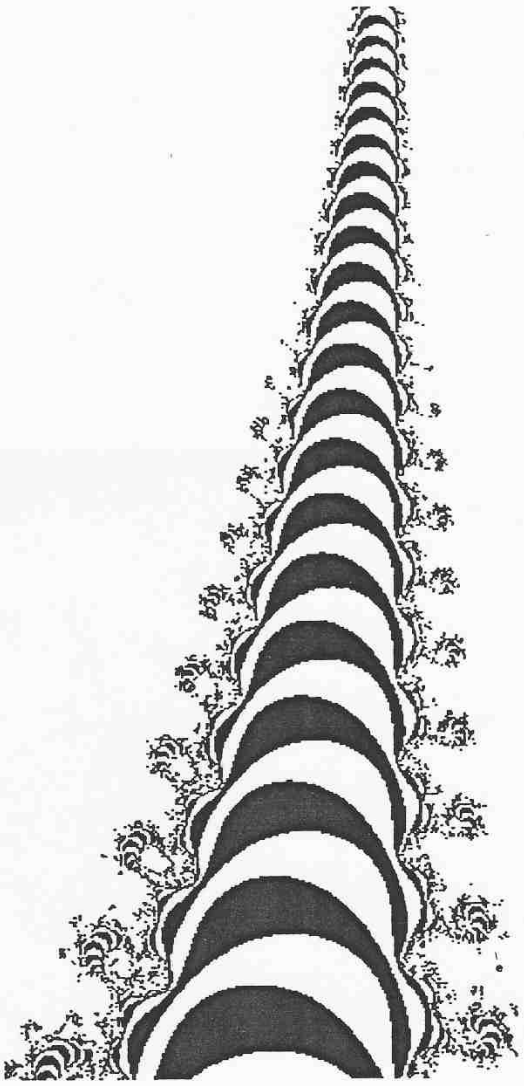


fig. c

--SAMSON 65--

```
Number_real_start : ? -.777
Number_imag_start : ? -.115
Number_real_end   : ? -.742
Iterations        : ? 90
```

```
3B10 A9 83          LDAIM $83     Decimal 4 in float-binary
3B12 85 B6          STA   BFAC
3B14 A9 80          LDAIM $80     +01
3B16 85 B7          STA   BFAC    +01
3B18 A9 00          LDAIM $00     +02
3B1A 85 B8          STA   BFAC    +02
3B1C 85 B9          STA   BFAC    +03
3B1E 85 BA          STA   BFAC    +04
3B20 85 BB          STA   BFAC    +05
3B22 85 BC          STA   SIGNCMP
3B24 85 BD          STA   RNDBYTE
3B26 20 C2 16       JSR   MINUS       4 - (A+M)
3B29 24 B3          BIT   AFAC    +05 if result negative
3B2B 30 0F          BMI   L41         then paint pixel

                ┌──────────────────────────────────┐
                │  INCREMENT ITERATIONS COUNTER    │
                └──────────────────────────────────┘
3B2D E6 36          INC   COUNTER
                ┌──────────────────────────────────┐
                │  UNTIL  COUNTER = MAX            │
                └──────────────────────────────────┘
3B2F A5 36          LDA   COUNTER
3B31 C5 37          CMP   ITERMAX
3B33 F0 03          BEQ   L40
3B35 4C AD 3A       JMP   LAB4
                ┌──────────────────────────────────┐
                │  IF COUNTER = MAX THEN BLACK PIXEL │
                └──────────────────────────────────┘
3B38 A9 0F    L40   LDAIM $0F     COLOR = BLACK
3B3A D0 02          BNE   L42
                ┌──────────────────────────────────┐
                │  PAINT PIXEL                     │
                └──────────────────────────────────┘
3B3C A5 36    L41   LDA   COUNTER COLOR = 1 OF 16
3B3E 29 0F    L42   ANDIM $0F
3B40 8D 64 E1       STA   COLOR
3B43 A9 80          LDAIM $80
3B45 8D 50 E1       STA   CMD
3B48 A9 04    WAIT  LDAIM $04     wait for GDF
3B4A 2D 50 E1       AND   CMD
3B4D F0 F9          BEQ   WAIT
                ┌──────────────────────────────────┐
                │  INCREMENT X COORDINATE          │
                └──────────────────────────────────┘
3B4F EE 59 E1 LAB5  INC   LSBX
3B52 D0 0A          BNE   L51
3B54 EE 58 E1       INC   MSBX
3B57 AD 58 E1       LDA   MSBX
3B5A C9 02          CMPIM $02     untill 512
3B5C F0 15          BEQ   LAB6
                ┌──────────────────────────────────┐
                │  X = X + DELTA                   │
                └──────────────────────────────────┘
3B5E A2 0C    L51   LDXIM DELTA
3B60 20 98 3B       JSR   MOVTO#1
3B63 A2 12          LDXIM X
3B65 20 B1 3B       JSR   MOVTO#2
3B68 20 DE 16       JSR   PLUS
3B6B A2 12          LDXIM X
3B6D 20 D2 3B       JSR   MOVRESU
3B70 4C A0 3A       JMP   LAB3
                ┌──────────────────────────────────┐
                │  INCREMENT Y-COORDINATE          │
                └──────────────────────────────────┘
3B73 EE 5B E1 LAB6  INC   LSBY
3B76 D0 0A          BNE   L61
3B78 EE 5A E1       INC   MSBY
3B7B AD 5A E1       LDA   MSBY
3B7E C9 02          CMPIM $02     untill 512
3B80 F0 15          BEQ   FINISH
```

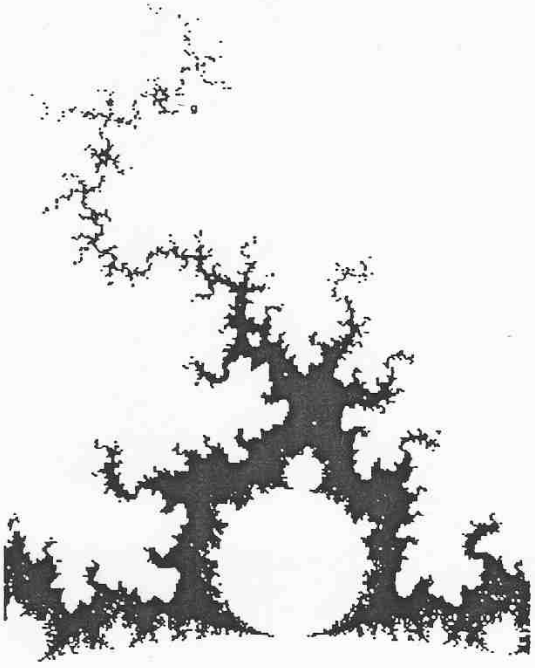


fig. b

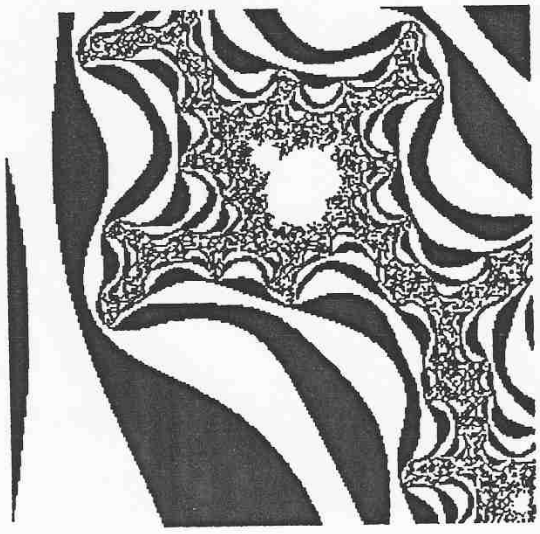


fig. g

```
                ┌──────────────────────────────────┐
                │  Y = Y + DELTA                   │
                └──────────────────────────────────┘
3B82 A2 0C    L61     LDXIM DELTA   ! note that here is
3B84 20 98 3B         JSR   MOVTO#1 DELTA_Y = DELTA_X !
3B87 A2 18            LDXIM Y
3B89 20 B1 3B         JSR   MOVTO#2
3B8C 20 DE 16         JSR   PLUS
3B8F A2 18            LDXIM Y
3B91 20 D2 3B         JSR   MOVRESU
3B94 4C 8F 3A         JMP   LAB2
                ┌──────────────────────────────────┐
                │  THE END                         │
                └──────────────────────────────────┘
3B97 60       FINISH  RTS


                ┌──────────────────────────────────┐
                │  MOVE NUMBER TO FLP-ACC #1       │
                └──────────────────────────────────┘
3B98 B5 00    MOVTO#1 LDAX  ZERO
3B9A 85 AE            STA   AFAC
3B9C B5 01            LDAX  ZERO    +01
3B9E 85 AF            STA   AFAC    +01
3BA0 B5 02            LDAX  ZERO    +02
3BA2 85 B0            STA   AFAC    +02
3BA4 B5 03            LDAX  ZERO    +03
3BA6 85 B1            STA   AFAC    +03
3BA8 B5 04            LDAX  ZERO    +04
3BAA 85 B2            STA   AFAC    +04
3BAC B5 05            LDAX  ZERO    +05
3BAE 85 B3            STA   AFAC    +05
3BB0 60               RTS
                ┌──────────────────────────────────┐
                │  MOVE NUMBER TO FLP-ACC #2       │
                └──────────────────────────────────┘
3BB1 B5 00    MOVTO#2 LDAX  ZERO
3BB3 85 B6            STA   BFAC
3BB5 B5 01            LDAX  ZERO    +01
3BB7 85 B7            STA   BFAC    +01
3BB9 B5 02            LDAX  ZERO    +02
3BBB 85 B8            STA   BFAC    +02
3BBD B5 03            LDAX  ZERO    +03
3BBF 85 B9            STA   BFAC    +03
3BC1 B5 04            LDAX  ZERO    +04
3BC3 85 BA            STA   BFAC    +04
3BC5 B5 05            LDAX  ZERO    +05
3BC7 85 BB            STA   BFAC    +05
3BC9 45 B3            EOR   AFAC    +05  compare signs
3BCB 85 BC            STA   SIGNCMP
3BCD A9 00            LDAIM $00
3BCF 85 BD            STA   RNDBYTE      clear rounding byte
3BD1 60               RTS
                ┌──────────────────────────────────┐
                │  MOVE RESULT BACK                │
                └──────────────────────────────────┘
3BD2 A5 AE    MOVRESU LDA   AFAC
3BD4 95 00            STAX  ZERO
3BD6 A5 AF            LDA   AFAC    +01
3BD8 95 01            STAX  ZERO    +01
3BDA A5 B0            LDA   AFAC    +02
3BDC 95 02            STAX  ZERO    +02
3BDE A5 B1            LDA   AFAC    +03
3BE0 95 03            STAX  ZERO    +03
3BE2 A5 B2            LDA   AFAC    +04
3BE4 95 04            STAX  ZERO    +04
3BE6 A5 B3            LDA   AFAC    +05
3BE8 95 05            STAX  ZERO    +05
3BEA 60               RTS
```

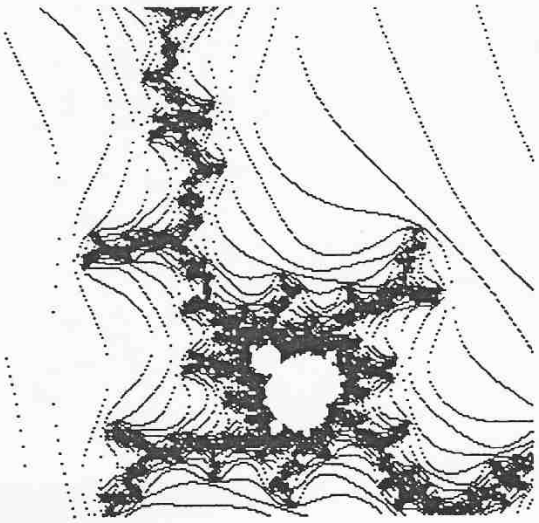


fig. g

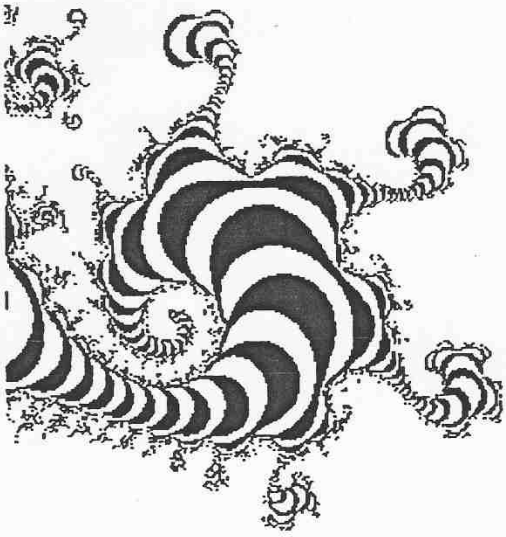
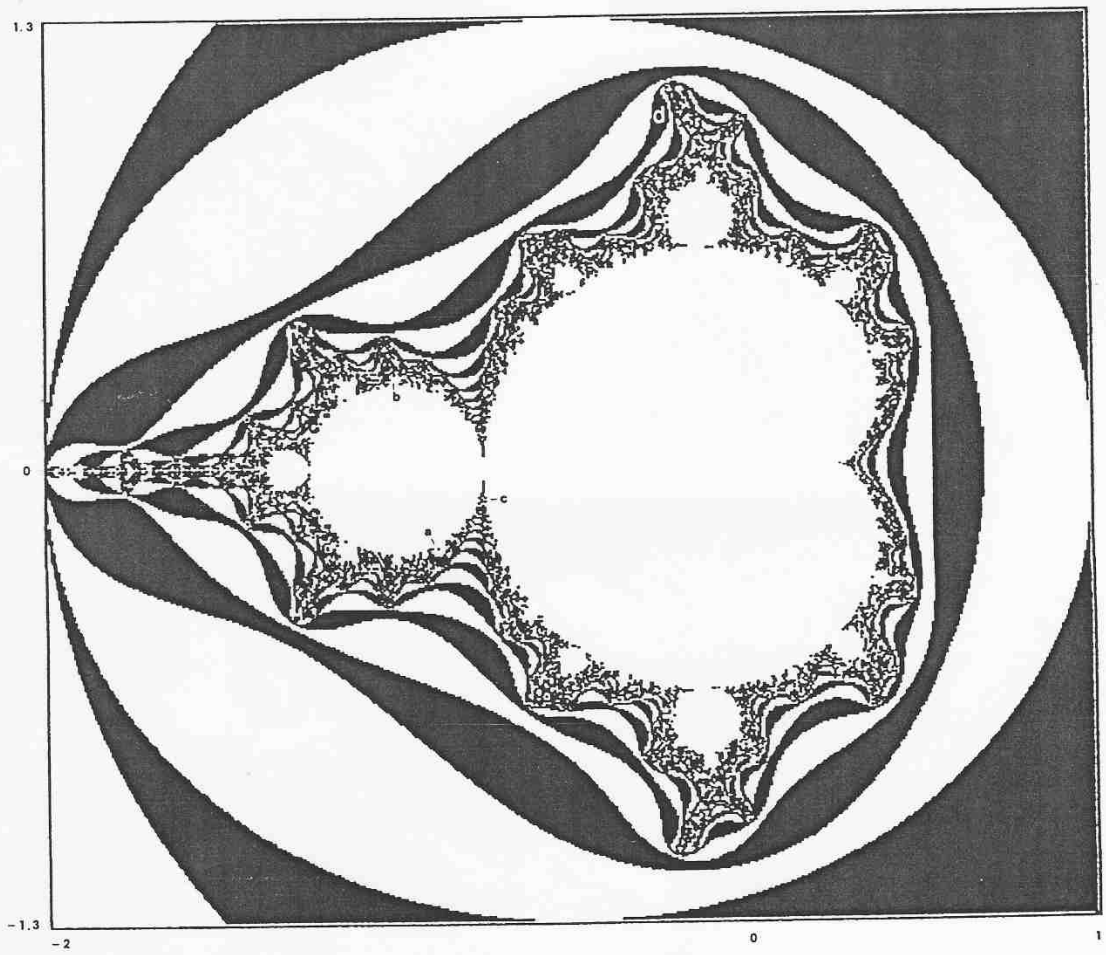
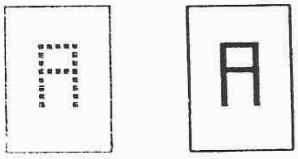


fig. e



Get
Near Laser Quality
on your matrixprinter

Hardcopy of graphics on a nine pin matrix printer is often little 'thin because the dots do not overlap. When printing the pictures of the Mandelbrot set I used a better technique.
The printer is set up for quadruple density = 240 dots/inch, and each pixel is send 3.25 times (3 pixels are send 3 times and the 4.th is send 4 times - to even out the difference in horizontal and vertical steps (1/240" versus 1/216")).
Since consecutive dots in a row are not printed in quadruple density each pixel is printed 2 times with 2/240" apart.
Then a linefeed of 1/216" is send and the same line is printed once again. So in all, each pixel is printed 4 times.
A linefeed of 16/216" is given before the next line.



Leif Rasmussen