fractal (z, $\Delta_x z$, $\Delta_y z$, k, x, y, Δx, Δy), tc, prgrmachine, M)

tc(i) i de 0 à k

W)
Y) FRACTAL : BSR WCXFP (30) 150 décode nb complexe → FP0, FP1
LEA CMANDZ0, A0
FMOVE.X FP0, (A0)+ ; CMANDZ0
FMOVE.X FP1, (A0)+ ; ——— Z1

BSR WCXFPV (30) 150 , $\Delta x z$
FMOVE.X FP0, (A0)+ ; CMANDX0
FMOVE.X FP1, (A0)+ ; CMANDX1

BSR WCXFPV (30) 150 , $\Delta_y z$
LEA CMANDY0, A1
FMOVE.X FP0, (A1)+ ; CMANDY0
FMOVE.X FP1, (A1)+ ; CMANDY1
MOVEQ #~~16~~ 100, D3
BSR WVGAD [, K] défaut k = ~~16~~ 100
MOVE D3, D6 ← ~~AND #$3FF, D6~~ ~~limité à ≈ 1000~~ ← MOVE
SUBQ #1, D6 ← AND #$7FFF, D6 limité à 2^{15}

BSR WVGAZ (3) 89 [, x] défaut = 0
MOVE D3, (A0)+ ; CMANDX
BSR WVGAZ (3) 89 [, y] défaut = 0
MOVE D3, (A0)+ ; CMANDY

× MOVE , D3 ↙ screen_px
BSR WVGAD [, Δx]
MOVE D3, (A0)+ ; CMANDPX
MOVE D3, D4

× MOVE , D3 screen_py
BSR WVGAD
MOVE D3, (A0)+ ; CMANDPY
MOVE D3, D5

× MOVE , D0 nb de couleurs
SUBQ #1, D0

{ MOVE , D0 } screen_pc (×2)
ADD D0, D0
MOVE D0, (A0)+ ; CMAND2PC
MULU D3, D0
ASR.L #4, D0
d4 = Δx MOVE D0, (A0)+ ; CMAND L
nb d'octets / ligne
d5 = Δy

```
BSR   WVGAZ        [x]   défaut x
MOVE  D3,(A0)+     ;CMAND X

BSR   WVGAZ        [,y]  défaut y

~~LEA   CMANDY,A1~~
MOVE  D3,(A1)+     ;CMANDY

x  (MOVE RESOLPX,D3     screen_px
    AND #$1FFF,D3
    MOVE  D3,(A0)+ ; CMANDXM

x  MOVE RESOLPC,D0      screen_pc
   AND #$1FFF,D0
   ADD   D0,D0          (x2)
   MOVE  D0,CMAND2PC    2 nb de plans
   MULU  D3,D0
   ASR.L #4,D0

   MOVE  D0,CMANDL      nb d'octets par ligne

   BSR   WVGAD    [,Δx]
   MOVE  D3,(A0)+ ; CMANDPX

x  MOVE RESOLPY,D3      screen_py
   AND #$1FFF,D3
   BSR   WVGAD    [,Δy]

   BSR   MAND7          vérif et divise Δyz par Δx

   MOVE.L  A1,A0
   BSR   MAND7          //   Δyz par Δy
```

MOVE D3,(A1)+;CMANDYM

MOVE D3,(A1)+;CMANDPY

```
30      lecture table de couleurs
(X)     LEA   MAND1,A4
        BSR   DECCRV      (5)
X       BNE   \8          → table par défaut
        MOVE  D6,-(SP)    conserve d6 = k-1
répète  LEA   CMANDT,A0   début } table de 40 octets
YDET    LEA   CMANDTF-10,A2  fin }

        BRA   \32
\30:    CMP.L A2,A0
        BCC   ERRVL       q non trop long
        MOVE.B D0,(A0)+
        ADDQ  #1,A5
\32:    BSR   DECTMN      (3)21
X       BEQ   \33         → fin si (0) ou (1)
        CMP.B #")",D0     fin si ) ou (5)
        BEQ   \33
        CMP.B #",",D0
        BNE   \30
\33:    MOVE  #"(§",(A0)+
        MOVE.L A5,A1
        MOVEQ #0,D0
        ADDQ  #1,D6       boucle d6=0 à k
\35:    MOVEM.L D0/A0,-(SP)
        BSR   PINTA
        MOVE  #")"*256,(A0)
        LEA   CMANDT,A5
        BSR   WADR        (3)89 → d3
        MOVEM.L (SP)+,D0/A0
        MOVE  D3,-(SP)
        BSR   VERAC
        ADDQ  #1,D0
        CMP   D0,D6
(X)→    BCC   \35
        MOVE.L A1,A5
        BSR   DECCRV      (5)
        BNE   \36
        BSR   WADRP       (3)89 décode adresse point
        CMP.L #$65100ADE,(A0)+  magic
        BNE   ERRMAG      → erreur "
        MOVE.L A0,A4

\36:    FMOVE.W #100,FP5  M
        JSR   DECCRV      (5)
        BNE   \360
        BSR   WCXFP       (150)
        FMOVE FP0,FP5
\360
```

```
\36: MOVE  RESOLI,D0
     AND   #$1FFF,D0
     SUBQ  #1,D0          nb de couleurs - 1

     LEA   1(A6),A0

\37: MOVE  (SP)+,D1
     AND   D0,D1
     MOVE.B D1,(A0)+
     DBRA  D6,\37
     MOVE.B D1,(A6)       valeur de couleur TC(0)
     MOVE.L (SP)+,A5
     MOVE  (SP)+,D6
     BRA   \39
```

```
\12: LEA   MAND1,A4
     BSR   DECCRV
     BNE   \39
     BSR   WADRP          décode adresse paire
     CMP.L #$65100ADE,(A0)+
     BNE   ERRMAG         → illégal  erreur magic
     MOVE.L A0,A4
\39: MOVE.W #100,FPS      M
     BRA   \37
```

abcde
dec
dec

→ 1526

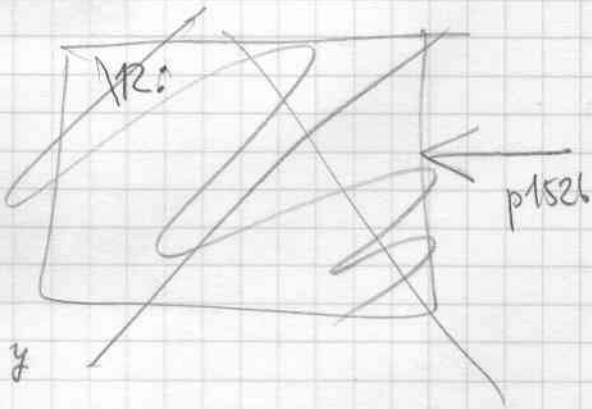
```
ERRMAG: moveq #12,d0
        TRAP  #15
```

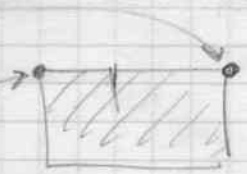
30

```
\18: Crée table des couleurs en A6
     MOVE RESOLI,D0   nb de couleurs
     SUBQ #1,D0       <- AND #$1FFF,D0
     MOVE.L A6,A0
     MOVE D6,D1
     MOVE.B D0,(A6)+
     MOVEQ #0,D2
\10: AND d0,d2
     MOVE.B d2(A6)+
     ADDQ #1,d2
     DBRA d1,\10

\10: MOVEQ #1,D2
     ADD d1,D2
     AND d0,d2
     MOVE.B d2,(A6)+
     DBRA d1,\10

     BSR VERA6
     MOVE.L A6,A6

     LEA CMANDX0,A0
     FMOVE.X (A0),FP0
     FDIV.W D4,FP0
     FMOVE.X FP0,(A0)+
     FMOVE.X (A0),FP0
     FDIV.W D4,FP0
     FMOVE.X FP0,(A0)+
     FMOVE.X (A0),FP0
     FDIV.W D5,FP0
     FMOVE.X FP0,(A0)+
     FMOVE.X (A0),FP0
     FDIV.W D5,FP0
     FMOVE.X FP0,(A0)+
\39: FMOVE.W #100,FP5     M
     LEA MAND1,A4         progr
\39: MOVE CMANDX,D0
     MOVE CMANDY,D1       y
     ADD CMANDPX,D0       x+Δx
     BSR MAND6
     MOVE.L A0,A3
     MOVE.L D1,D3
     MOVE CMANDX,D0       x
     MOVE CMANDY,D1       y
     BSR MAND6            ) A0,D1
```

pas suivant x

pas suivant y



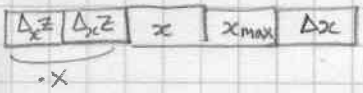


Trace les CMANDPY lignes


```
      BSR HIDECMB                ⊗
\20 : LEA  CMANDZ0,A1
      FMOVE.X (A1),FP6     Re(z0)
      FMOVE CMANDY0,FP0    Re(Δy)
      FADD  FP6,FP0
      FMOVE.X FP0,(A1)+    nouvelle valeur
repite
      FMOVE.X (A1),FP7     Im(z0)
      FMOVE CMANDY1,FP0
      FADD  FP7,FP0
      FMOVE.X FP0,(A1)+    nouvelle valeur

      MOVEM.L D1/A0,-(SP)
      BSR MAND4                  trace la ligne
      MOVEM.L (SP)+,D1/A0
      MOVE CMANDL,D0
      ADD D0,A0
      ADD D0,A3
      SUB #1,CMANDPY       ⊗
      BNE \20              ⊗
      RTS
      BRA SHOWCMB          ⊗
```

30 Vérif table A_0 (A_0^e)

détruit D0-D2 / A0
sortie D0 = Δx (recalculé)




```
MAND7: MOVE  -(A0),D0
       MOVE  -(A0),D1   max
       MOVE  -(A0),D2
       CMP   D1,D2          0 ≤ x < xmax  sinon x ← 0
       BCS   \1
       MOVEQ #0,D2
\1:    MOVE  D2,(A0)
       ADD   D2,D0          x' = x + Δx
       CMP   D1,D0          il faut 0 ≤ x' ≤ xmax  sinon Δx = xmax - x
       BCS   \2
       MOVE  D1,D0
\2:    SUB   D2,D0
       BNE   \3             →
       MOVEQ #1,D0
\3:    MOVE  D0,4(A0)       nouveau Δx
       FMOVE.X -(A0),FP0
       FDIV.W  D0,FP0
       FMOVE.X FP0,(A0)
       FMOVE.X -(A0),FP0
       FDIV.W  D0,FP0
       FMOVE.X FP0,(A0)
       RTS
```

Met adresse α,δ du pixel (x,y)
A0,D1.L ↑D0 ↑D1

démit D0
conserve D2-D7
A1-A7

```
MANDG :MOVEM.L D0-D2/A1/A2,-(SP)
       MOVE #3,-(SP)
       TRAP #14
       ADDQ #2,SP
       MOVE.L D0,A0
       MOVEM.L (SP)+,D0/D1
       MULU  CMAND1L,D1      nb_octets/ligne * y
       ADD.L D1,A0
       MOVEQ #15,D1
       AND   D0,D1
       LSR #4,D0
       MULU  CMAND2PC,D0     2 nb plans * x\16
       ADD.L D0,A0
       MOVEM.L (SP)+D2/A1/A2
       RTS
```

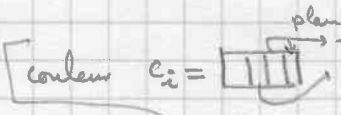
Trace la ligne A0,D1 à A3,D3

détruit
FP0-FP4
D0-D2
A0-A1

```
>BSR ESCAPE
MAND4:MOVEQ  #16,D2
      :CMP.L  A3,A0      ⊗
       BGT    \12        → fin
       BNE    \10
       MOVE.L D3,D2
\10:BSR   MAND3
×      ADD  CMAND2PC,A0  ⊗  2x16 de plans couleurs

       BRA   MAND4
\12:RTS
```

Entrées $c_1 = FP6 + i\,FP7$, $FP5 = M$, $D6 = k-1$ $A4 = MAND1$, (ou autre)

$\Delta z = CMANDX0 + i\,CMANDX1$ (par pixel)

A6 : table des $k+1$ couleurs 0, 1 ... K couleur $c_i =$ 

A0 : adresse sur l'écran pointe sur mot du plan 0

D1.L $\in [0,15]$ 1er pixel à traiter est décalé de D1 / A0

D2.L $\in [0,16]$ 1er pixel à ne pas traiter D2 ⩾ D1

si 4 plans : $c_i \in [0,15]$
si 2 plans $c_i \in [0,3]$
si 1 plan $c_i = 0$ ou 1

Effet : Trait les pixels D1 → D2 (exclu)
Avance c_1 jusqu'à sa valeur pour D2
En sortie D0 = 0

détruit FP0 – FP4
D0/D1
A1

```
MAND3: CMP.L   D2,D1
       BGE     \16          → fin
       BSR     MAND2
       MOVE.B  +1(A6,D5.W),D0
       MOVE.L  A0,A1

       BEQ     \14          → fin
\10:   LSR.B   #1,D0
       BCC     \12
       BFSET   (A1),D1,1
\12:   ADDQ    #2,A1
       TST.B   D0
       BNE     \10
\14:   ADDQ.L  #1,D1
       FADD.X  CMANDX0,FP6
       FADD.X  CMANDX1,FP7
       BRA     MAND3        \4 ⊗
\16:   MOVEQ   #0,D1
       RTS
```

```
MAND2: MOVE    D3,-(SP)
       MOVE    CMAND2PC,D3   2 nb de plans
       MOVE.L  D2,D0
       SUB.L   D1,D0
       MOVE.L  A0,A1
\2:    BFCLR   (A1),D1,D0
       ADDQ    #2,A1
       SUBQ    #2,D3
       BGT     \2
       MOVE    (SP)+,D3
\4:    CMP.L   D2,D1
       BGE     \16
```

Entrée $c = FP6 + i\,FP7$, $FP5 = M = 100$, $D6 = K-1$ nb d'itérations max

Sortie : $D5 \in K-1, K-2, \ldots$ ou -1

$|z_1|^2 > M$ $|z_2|^2 > M$ ... $|z_k|^2 > M$ ... $|z_K|^2 \leq M$

A4 : (SP) calcule $z_{k+1} = f(z_k, c)$; z_k : FP0, FP1 ; c : FP6, FP7 ; peut détruire FP2-FP4/D0

détruit FP0–FP4 / D0

```
MAND2: MOVE   D6,D5
       FMOVECR #$F,FP0   } z0 = 0
       FMOVECR #$F,FP1
\10 :  JSR    (A4)        z1 = c
                          zk-1 → zk

\5:    FMOVE  FP0,FP2     ici  k = D6 - D5 = 1, 2, ....
       FMOVE  FP1,FP3
       FMUL   FP2,FP2
       FMUL   FP3,FP3
       FADD   FP3,FP2     FP2 = |zk|^2
       FCMP   FP5,FP2
                          → |zk|^2 > M
       FDBGT  D5,\10
          ↓ sort |zk|^2 > M et D5 = K-k
       RTS    sort k = K et D5 = -1
```

```
MAND1: BSR    CPCMUL1     zk^2
       FADD   FP6,FP0
       FADD   FP7,FP1     zk^2 + c  ≡ zk+1
       RTS'
```

→ A4

30 Entrée $z = FP0_a + i\,FP1_b$ $c = FP6 + i\,FP7$

Calcule $z^2 + c = FP0_s + i\,FP1_s$

reprise de FMULQ

```
MAND1:FMOVE   FP0, FP2     a
      FMUL    FP1, FP2     ab
      FMUL    FP0, FP0     a²
      FMUL    FP1, FP1     b²
      FSCALE.N #1, FP2     2ab
      FSUB    FP1, FP0     a²-b²
      FMOVE   FP2, FP1     2ab
      FADD    FP6, FP0     } z²+c
      FADD    FP7, FP1     }
      RTS
```