```
1   .TITLE  Chaotic Dusty Curls Generation Using VAX Macro
2
3   ;++
4   ; This program will produce a set of coordinate and value tuples
5   ; for a window on a Mandelbrot set to the standard output.
6   ;
7   ; References:
8   ;       VAX Macro and Instruction Set Reference Manual
9   ;       VMS Run-Time Library (LIB$) Reference Manual
10  ;       VMS System Services Reference Manual
11  ;       VMS Run-Time Math Library (MTH$) Reference Manual
12  ;       Sara Baase 'VAX Assembler Language Programming'
13  ;
14  ; Author        : Mark Wickens
15  ; Date          : 11-Jul-2009
16  ;--
17  ;    ALGORITHM:  Calculation of chaotic dusty curls
18  ;          Variables:  rz, iz = real, imaginary component of complex number
19  ;                      i = iteration counter
20  ;                      u, z = complex numbers
21  ;          Note:       Choose one of the three different tests for divergence.
22  ;
23  ;          u = -.74 + .11 i;
24  ;          DO rrz = -1 to 1 by 0.001;
25  ;            DO iiz = -1 to 1 by 0.001;
26  ;              z = cplx(rrz, iiz);
27  ;              InnerLoop: DO i = 1 to 100;
28  ;                z = z**2 + u;
29  ;                rz = real(z); iz = imag(z);
30  ;                if sqrt(rz**2 + iz**2) > 2 then leave InnerLoop;
31  ;              END;
32  ;              color = i;
33  ;              if convergence_test = 1 then
34  ;                if rz**2 + iz**2 > 4 then PRINTDOT(rrz,iiz,color);
35  ;              if convergence_test = 2 then
36  ;                if ((abs(rz)<2) & (abs(iz)<2)) then PRINTDOT(rrz,iiz,color);
37  ;              if convergence_test = 3 then
38  ;                if rz**2+iz**2>4 & mod(i,2) = 0 then PRINTDOT(rrz,iiz,color);
39  ;            END;
40  ;          END;
41  ;--
42
43  ; Define symbols, constants, etc. used in this module
44
45  ; Debugging aids
46  ; Debug the inner loop by outputting each value calculated for sqrt(rz**2 + iz**2)
47          DEBUG_INNER     = 0
48
49  ; Offset in bytes in real/imaginary pair of the real and imaginary value
50          COMP_REAL       = 0
51          COMP_IMAG       = 4
52
53          .PSECT MBRT1_RWDATA     WRT,NOSHR,NOEXE,PAGE
54          .ALIGN LONG
55
56  ; Real and imaginary values for constant 'u'
57  U:       .F_FLOATING    0.74, 0.11
58
59  ; Convergence test 1, comparison value
60  CONV1_MAX: .F_FLOATING  6.0
61
62  ; Real axis dimensions and increment
63  RRZ_POINTS:     .BLKL   1
64  RRZ_MIN: .F_FLOATING    -0.45
65  RRZ_MAX: .F_FLOATING    -0.425
66  RRZ_INC:        .BLKF   1
67  RRZ_ITER:       .LONG   0
```

```
68  IMAGE_SIZE:     .LONG   1000
69
70  POW2:    .LONG          2
71
72  ; Imaginary axis dimensions and increment
73  IIZ_POINTS:     .BLKL   1
74  IIZ_MIN:        .F_FLOATING     -0.625
75  IIZ_MAX:        .F_FLOATING     -0.60
76  IIZ_INC:        .BLKF   1
77  IIZ_ITER:       .LONG   0
78
79  TWOF:   .F_FLOATING     2.0
80
81  ; Number of iterations to perform in convergence loop
82          LOOP_MAX        = 99
83
84  ; Convergence test to use
85          CONV_TEST_VALUE = 1
86
87  ; Output buffer maximum size
88          MAX_SIZE        = 132
89
90  ; Fortran E Format buffer max size
91          FEF_MAX_SIZE    = 16
92
93  ; Setup a text buffer for use as an output line
94
95  OUT_BUFFER:     .BLKB MAX_SIZE
96
97  OUT_DESC:       .WORD MAX_SIZE
98                  .BYTE DSC$K_DTYPE_T
99                  .BYTE DSC$K_CLASS_S
100                 .ADDRESS OUT_BUFFER
101
102 FEF_BUFFER:     .BLKB FEF_MAX_SIZE
103
104 FEF_DESC:       .WORD FEF_MAX_SIZE
105                 .BYTE DSC$K_DTYPE_T
106                 .BYTE DSC$K_CLASS_S
107                 .ADDRESS FEF_BUFFER
108 FEF_VAL:        .F_FLOATING 0.0
109
110 ASC_SPACE:      .ASCII  / /
111         NULL    = 00
112 ASC_NULL:       .ASCII  <NULL>
113
114 ; Allocate a longword to store any failure status value we might see
115
116 ERROR_CODE:     .BLKL 1
117
118 ; Real axis loop counter
119 RRZ:            .BLKL 1
120
121 ; Imaginary axis loop counter
122 IIZ:            .BLKL 1
123
124 ; Z Complex value (real/imaginary pair)
125 Z:              .BLKF 2
126
127 ; Complex temporary
128 COMP_TEMP:      .BLKF 2
129
130 RZ:             .BLKF 1
131 IZ:             .BLKF 1
132 RZPOW2:         .BLKF 1
133 IZPOW2:         .BLKF 1
134 SQRTZ:          .BLKF 1
```

```
135 ZPOW2:          .BLKF 1
136 COLOUR_F:       .BLKF 1
137 COLOUR_L:       .BLKL 1
138
139 ; Convergence test
140 CONV_TEST:      .BYTE   CONV_TEST_VALUE
141
142 ; Netpbm Stuff
143 MAGIC_NUMBER:   .ASCID /P2/     ; Plain PGM format magic number
144 PACKED_LONG:    .ASCII /        /
145                 .ASCII <NULL>
146
147 ; Conversion between long and leading separate numeric via packed decimal
148 LSN:            .BLKB   7       ; Leading separate numeric
149 PKD:            .BLKB   3       ; Packed decimal
150
151 LONG_SIZE:      .LONG   4       ; number of characters in output number
152 MAX_GRAY:       .ASCID  /256/
153
154 COUNT:          .LONG 0
155 MARKER:         .ASCID /+/
156
157         .PSECT MBRT1_CODE NOWRT,SHR,EXE,PAGE
158         .ALIGN PAGE
159 ;******************************************************************************
160 ; Program Entry Point
161 ;******************************************************************************
162
163         .ENTRY MBRT1,^M<R2,R3,R4,R5,R7>
164
165 ; Initialize
166
167 START:
168         MOVB    #SS$_NORMAL, ERROR_CODE         ; error code to normal
169         MOVB    #CONV_TEST_VALUE, CONV_TEST     ; convergence test to use
170
171 ; Output magic number
172         PUSHAQ  MAGIC_NUMBER
173         CALLS   #1, G^LIB$PUT_OUTPUT
174
175 ; Calculate width and height of image
176
177 ; rrz_inc = (rrz_max - rrz_min) / image_size
178         SUBF3   RRZ_MIN, RRZ_MAX, R2
179         CVTLF   IMAGE_SIZE, RRZ_INC
180         DIVF3   RRZ_INC, R2, RRZ_INC
181
182         PUSHL   IMAGE_SIZE
183         CALLS   #1, G^LONG2STR
184         PUSHAQ  OUT_DESC
185         CALLS   #1, G^LIB$PUT_OUTPUT
186
187 ; iiz_inc = (iiz_max - iiz_min) / image_size
188         SUBF3   IIZ_MIN, IIZ_MAX, R2
189         CVTLF   IMAGE_SIZE, IIZ_INC
190         DIVF3   IIZ_INC, R2, IIZ_INC
191
192         PUSHL   IMAGE_SIZE
193         CALLS   #1, G^LONG2STR
194         PUSHAQ  OUT_DESC
195         CALLS   #1, G^LIB$PUT_OUTPUT
196
197 ; Output max gray
198         PUSHAQ  MAX_GRAY
199         CALLS   #1, G^LIB$PUT_OUTPUT
200
201         MOVF    RRZ_MIN, RRZ                    ; init real axis value
```

```
202
203 RRZ_LOOP:
204         MOVL    #0, IIZ_ITER
205         MOVF    IIZ_MIN, IIZ                    ; init imaginary axis value
206
207 ;       PUSHAQ  MARKER
208 ;       CALLS   #1, G^LIB$PUT_OUTPUT
209 ;       PUSHL   COUNT
210 ;       CALLS   #1, G^LONG2STR
211 ;       PUSHAQ  OUT_DESC
212 ;       CALLS   #1, G^LIB$PUT_OUTPUT
213 ;       INCL    COUNT
214
215 ; Set value of complex variable Z, stored in R2
216 IIZ_LOOP:
217         MOVAF   Z, R2
218         MOVF    RRZ, COMP_REAL(R2)
219         MOVF    IIZ, COMP_IMAG(R2)
220
221 ; Initialize inner loop counter, stored in R3
222         MOVL    #1, R3
223
224 ; Within loop
225 LOOP:
226 ; z**2
227         PUSHL   POW2
228         PUSHL   COMP_IMAG(R2)
229         PUSHL   COMP_REAL(R2)
230         CALLS   #3, G^OTS$POWCJ
231         MOVAF   COMP_TEMP, R4
232         MOVF    R0, COMP_REAL(R4)
233         MOVF    R1, COMP_IMAG(R4)
234
235 ;R4 now contains address of z**2, COMP_TEMP
236 ;; z = z**2 + u
237
238         MOVAL   U, R5
239         ADDF3   COMP_IMAG(R4), COMP_IMAG(R5), COMP_IMAG(R2)
240         ADDF3   COMP_REAL(R4), COMP_REAL(R5), COMP_REAL(R2)
241
242 ; rz = real(z)
243         MOVF    COMP_REAL(R2), RZ
244 ; iz = imag(z)
245         MOVF    COMP_IMAG(R2), IZ
246
247         MULF3   RZ, RZ, RZPOW2
248         MULF3   IZ, IZ, IZPOW2
249         ADDF3   RZPOW2, IZPOW2, R5
250
251 ; R5 = rz**2 + iz**2
252         MOVF    R5, ZPOW2
253         PUSHAF  ZPOW2
254         CALLS   #1, G^MTH$SQRT
255         MOVF    R0, SQRTZ
256
257 ; R0 = SQRT(rz**2 + iz**2)
258
259         .IF     DEFINED DEBUG_INNER
260         PUSHAL  SQRTZ
261         CALLS   #1, G^FLOAT2STR
262         .ENDC
263
264         CMPF    SQRTZ, TWOF
265         BGTR    EXIT_LOOP
266
267         CMPL    #LOOP_MAX, R3
268         BEQLU   EXIT_LOOP
```

```
269
270         ADDL    #1, R3
271         JMP     LOOP
272
273 EXIT_LOOP:
274         CVTLF   R3, COLOUR_F
275         MOVL    R3, COLOUR_L
276
277         JSB     CONVERGENCE_TEST
278
279 ; IIZ = IIZ + IIZ_INC
280         ADDF3   IIZ, IIZ_INC, IIZ
281         ADDL2   #1, IIZ_ITER
282 ; IF IIZ_ITER < IMAGE_SIZE THEN GOTO iiz_loop
283         CMPL    IIZ_ITER, IMAGE_SIZE
284         BGEQ    EXIT_IIZ_LOOP
285         JMP     IIZ_LOOP
286
287 EXIT_IIZ_LOOP:
288         ADDF3   RRZ, RRZ_INC, RRZ
289         ADDL2   #1, RRZ_ITER
290 ; if rrz < rrz_max then goto rrz_loop
291         CMPL    RRZ_ITER, IMAGE_SIZE
292         BGEQ    EXIT_RRZ_LOOP
293         JMP     RRZ_LOOP
294
295 CONVERGENCE_TEST:
296         CMPF    R5, CONV1_MAX
297         BLEQ    NO_CONVERGENCE
298         JSB     PRINTPIXEL
299 ;       JSB     PRINTDOT
300         JMP     CONVERGENCE_TEST_END
301
302 NO_CONVERGENCE:
303         MOVL    #0, COLOUR_L
304         JSB     PRINTPIXEL
305
306 CONVERGENCE_TEST_END:
307         RSB
308
309 EXIT_RRZ_LOOP:
310 ; Normal Exit
311         MOVL    #SS$_NORMAL, ERROR_CODE
312         BRB     COMMON_EXIT
313
314 FAILURE_EXIT:
315         MOVL    R0, ERROR_CODE
316         MOVW    #MAX_SIZE, OUT_DESC
317
318         PUSHAQ  OUT_DESC
319         PUSHAW  OUT_DESC
320         PUSHAL  ERROR_CODE
321         CALLS   #3, G^LIB$SYS_GETMSG
322
323         PUSHAQ  OUT_DESC
324         CALLS   #1, G^LIB$PUT_OUTPUT
325
326 COMMON_EXIT:
327         $EXIT_S ERROR_CODE
328
329 ;-------------------------------------------------
330 PRINTDOT:
331         PUSHAL  RRZ
332         CALLS   #1, FLOAT2STR
333
334 ; Append string to the output buffer
335         MOVC3   #FEF_MAX_SIZE, FEF_BUFFER, OUT_BUFFER
```

```
336         MOVC3   #1, ASC_SPACE, OUT_BUFFER+16
337
338         PUSHAL  IIZ
339         CALLS   #1, FLOAT2STR
340         MOVC3   #FEF_MAX_SIZE, FEF_BUFFER, OUT_BUFFER+17
341         MOVC3   #1, ASC_SPACE, OUT_BUFFER+33
342
343         PUSHAL  COLOUR_F
344         CALLS   #1, FLOAT2STR
345         MOVC3   #FEF_MAX_SIZE, FEF_BUFFER, OUT_BUFFER+34
346         MOVC3   #1, ASC_NULL, OUT_BUFFER+50
347
348 ; Set the length
349         MOVW    #50, OUT_DESC
350         RSB
351
352 ;--------------------------------------------------
353 PRINTPIXEL:
354         MOVL    COLOUR_L, R2
355         MOVL    LONG_SIZE, R3
356         MOVAB   PACKED_LONG, R4                 ; R4 contains ascii destination
357         BSBW    CVTLS
358         MOVC3   LONG_SIZE, PACKED_LONG+1, OUT_BUFFER
359         MOVC3   #1, ASC_NULL, OUT_BUFFER+5
360
361         MOVW    LONG_SIZE, OUT_DESC
362         PUSHAQ  OUT_DESC
363         CALLS   #1, G^LIB$PUT_OUTPUT
364
365         RSB
366
367 ;--------------------------------------------------
368 ; Convert min,max,inc long to string
369         .ENTRY  TUPLE2STR,^M<R2,R3,R4>
370
371         MOVF    12(AP), R2              ; MAX
372         MOVF    8(AP), R4               ; MIN
373         MOVF    4(AP), R3               ; INC
374         SUBF    R4, R2
375         DIVF3   R3, R2, R4
376
377         CVTFL   R4, R2                          ; R2 contains width in pixels
378         MOVAB   PACKED_LONG, R4                 ; R4 contains ascii destination
379         BSBW    CVTLS
380         MOVC3   LONG_SIZE, PACKED_LONG+1, OUT_BUFFER
381         MOVC3   #1, ASC_NULL, OUT_BUFFER+5
382
383         MOVW    LONG_SIZE, OUT_DESC
384         RET
385
386 ;--------------------------------------------------
387 ; Convert long to string
388         .ENTRY  LONG2STR,^M<R2,R4>
389
390         MOVL    4(AP), R2
391         MOVAB   PACKED_LONG, R4
392         BSBW    CVTLS
393         MOVC3   LONG_SIZE, PACKED_LONG+1, OUT_BUFFER
394         MOVC3   #1, ASC_NULL, OUT_BUFFER+5
395
396         MOVW    LONG_SIZE, OUT_DESC
397         RET
398
399 ;--------------------------------------------------
400 ; Convert float to string
401         .ENTRY  FLOAT2STR,^M<>
402
```

```
403         MOVL    @4(AP),FEF_VAL
404
405         MOVW    #FEF_MAX_SIZE, FEF_DESC
406         PUSHL   #6
407         PUSHAQ  FEF_DESC
408         PUSHAQ  FEF_VAL
409         CALLS   #3, G^OTS$CNVOUT
410         RET
411
412 ;--------------------------------------------------
413 ; Convert long to string
414 ;       R2 contains long value to convert
415 ;       R3 contains number of digits in output
416 ;       R4 contains address to store leading separate numeric value
417 CVTLS:  CVTLP   R2,LONG_SIZE,PKD
418         CVTPS   LONG_SIZE,PKD,LONG_SIZE,(R4)
419         RSB
420
421 ;--------------------------------------------------
422 ; Output string to terminal
423         .ENTRY  PRTSTR,^M<>
424
425         PUSHAQ  OUT_DESC
426         CALLS   #1, G^LIB$PUT_OUTPUT
427         RET
428
429         .END MBRT1
430
```