Subject: mand.c
Posted by RJ[1] on Tue, 31 Dec 1985 15:51:12 GMT
View Forum Message <> Reply to Message

Article-I.D.: amiga.450
Posted: Tue Dec 31 10:51:12 1985
Date-Received: Wed, 1-Jan-86 04:38:36 EST
References:
Reply-To: rj@wizard.UUCP (Robert J. Mical)
Organization: Commodore-Amiga Inc., 983 University Ave #D, Los Gatos CA 95030
Lines: 707
Keywords: Amiga, Mandelbrot

The Mandelbrot Self-Squared Dragon Generator program
consists of the following files (which are about to be broadcast):

This file contains mand.c. The next 5 pieces of mail will
have the other files, one per mailing.

=== Directory ==
mand.c has the main loop and some auxiliary routines.
mand1.c has the graphics routines.
mand2.c has the menu data definitions.
mand3.c has control flow routines.
mand4.c has the information routines.
mand.h has the global definitions.

In the comments at the head of this file (mand.c) you will find
the correct compilation and link commands.

Congrats again to Robert French.

Brace yourselves ...

--- snip snip snip ----------

```
/*
                MAND.C - Command parsing
          Mandelbrot's Self-Squared Dragon Generator
                For the Commodore Amiga
                    Version 1.00

        Inspired by Scientific American, August/1985

          Corrections and improvements suggested by
                The Fractal Geometry of Nature
     By Benoit Mandelbrot, W.H. Freeman and Company, 1983
```

(Used to be Z=Z^2+C, now is Z=Z^2-u, etc.)

Copyright (C) 1985, Robert S. French
Placed in the Public Domain

Assorted Goodies and Intuition-stuff by =RJ Mical= 1985
Hope you appreciate them. I especially like the zoom.

This program may be distributed free of charge as long as the above
notice is retained.

Please see the accompanying documentation for more information.

Send ANY problems or suggestions to the address in the nformation
section. Thank you!

Programs should be compiled with:

```
1> lc -i:include/ -i:include/lattice/ mand.c
1> lc -i:include/ -i:include/lattice/ mand1.c
1> lc -i:include/ -i:include/lattice/ mand2.c
1> lc -i:include/ -i:include/lattice/ mand3.c
1> lc -i:include/ -i:include/lattice/ mand4.c
1> alink :lib/Lstartup.obj+mand.o+mand1.o+mand2.o+mand3.o+mand4.o to mand lib
:lib/lc.lib+:lib/amiga.lib
```

*/

/*
==========
=

==========
=
From: "french robert%d.mfenet"@LLL-MFE.ARPA

Well guys, you asked for it, and here it is. There are two parts to the
Mandelbrot Set Generator which must be linked together (instructions are
at the beginning of the file). I don't have any of the documentation
written yet, so you're on your own for a while. I'll upload it when I get
it finished. I can't send files over 16K thru the net, so the first
section is sent as two files...just JOIN them together. Right now, the
program seems to work without any problems, except for an occasional
mysterious crash and the problem with closing a window in the middle of
picture generation (see comments in MAND1.C). Please send me any comments
or suggestions.

By the way, on a historical note, the "Mandelbrot Set" as related in
Scientific American is a little different from the u-Map implemented
here (as discussed in Mandelbrot's "The Fractal Geometry of Nature").
The main difference is the use of Z -> Z 2+C in Scientific American vs.
Z -> Z 2-u in the book and this program. The primary effect is the reversal
of the picture across the Y axis.

Enjoy!

Robert French
French#Robert%d@LLL-MFE.ARPA


```
  ========================================================================
  ========================================================================
*/

#include "mand.h"


int MathBase,MathTransBase;


/*----------------------*/
/* Graphics definitions */

struct   GfxBase       *GfxBase;
struct   IntuitionBase *IntuitionBase;


/*---------------------------------*/
/* Miscellaneous Global Definitions */

union kludge {
   float f;
   int i;
} start_r,end_r,start_i,end_i;
int max_x,max_y,max_mem_y;
int max_count,color_inc,color_offset,color_set,color_mode,color_div;
int color_inset,func_num;

int v_starty,max_mem;
long v_offset;
UWORD *color_table,*v_mand_store;

int modified,want_read;

FILE *console,*v_fp = NULL,*redir_fp = NULL;
```

```
SHORT ZoomCenterX, ZoomCenterY, ZoomBoxSizeX, ZoomBoxSizeY;
SHORT ZoomBoxStartX, ZoomBoxStartY;

int cur_resource = NULL;


/*-------------*/
/* Here we go! */

main()
{
   FILE *fopen();
   char *fgets(),*stpblk();
   float cnvf();
   void anal_mand(),wait_close();

   int temp,y;
   char *cmd,inp_buf[80],secchar,*argpos;
   union kludge scale;
   FILE *fp;

   max_mem_y = MAXMY;

   color_table = (UWORD *)malloc(4096*sizeof(UWORD));
   if (color_table == NULL)
      abort("Can't allocate memory for color table");
   cur_resource |= F_COLORTAB;

   v_mand_store = (UWORD *)malloc(MAXX*max_mem_y*sizeof(UWORD));
   if (v_mand_store == NULL)
      abort("Can't allocate memory for set storage");
   cur_resource |= F_SETSTORE;

   MathBase = OpenLibrary("mathffp.library",0);
   if (MathBase < 1)
      abort("Can't open mathffp.library");
   cur_resource |= F_MATH;

   MathTransBase = OpenLibrary("mathtrans.library",0);
   if (MathTransBase < 1)
      abort("Can't open mathtrans.library");
   cur_resource |= F_MATHTRANS;

   GfxBase = (struct GfxBase *)OpenLibrary("graphics.library",0);
   if (GfxBase == NULL)
      abort("Can't open graphics.library");
   cur_resource |= F_GRAPHICS;
```

```
   IntuitionBase = (struct IntuitionBase *)OpenLibrary("intuition.library",0);
   if (IntuitionBase == NULL)
      abort("Can't open intuition.library");
   cur_resource |= F_INTUITION;

   console = fopen("con:0/0/640/200/Mandelbrot Commands","w+");
   if (console == NULL)
      abort("Can't open console window");
   cur_resource |= F_CONSOLE;

   fprintf(console,
      "Mandelbrot Self-Squared Dragon Generator - Version %s\n",VERSION);
   fputs("Copyright (C) 1985, Robert S. French\n",console);
   fputs("Power, Goodies, and Baubles by =RJ Mical= 1985\n",console);
   fputs("Placed in the public domain.\n",console);
   fputs("Type '?' or 'H' or  for help\n\n",console);

   SetPresets(0);

   for (EVER) {

command:

      fprintf(console,"Command: ");
      rewind(console);
      if (redir_fp)
         if (Chk_Abort()) {
            fputs("\nRedirected input aborted!\n",console);
            fclose(redir_fp);
            redir_fp = NULL;
         }
         else {
            cmd = fgets(inp_buf,78,redir_fp);
            if (cmd == NULL) {
               fputs("End of file encountered.\n",console);
               fclose(redir_fp);
               redir_fp = NULL;
               cmd = fgets(inp_buf,40,console);
            }
            else
               fputs(inp_buf,console);
         }
      else
         cmd = fgets(inp_buf,40,console);
      *(cmd+strlen(cmd)-1) = '\0';
      secchar = toupper(*(cmd+1));
      argpos = stpblk(cmd+2);
      rewind(console);
```

```
     switch (toupper(*cmd)) {
       case 'I':
         sscanf(argpos,"%d",&temp);
     Information(temp);
         goto command;
       case 'S':
         if (secchar == 'R') {
           sscanf(argpos,"%f",&start_r.f);
           start_r.i = SPFieee(start_r.i);
           if (v_fp) {
             fclose(v_fp);
             v_fp = NULL;
           }
           goto command;
         }
         if (secchar == 'I') {
           sscanf(argpos,"%f",&start_i.i);
           start_i.i = SPFieee(start_i.i);
           if (v_fp) {
             fclose(v_fp);
             v_fp = NULL;
           }
           goto command;
         }
         if (secchar == 'H') {
           fputs("Current settings:\n",console);
           fprintf(console,"MaxX=%d, MaxY=%d, SR=%f, ER=%f, SI=%f, EI=%f, F=%d\n",
             max_x,max_y,cnvf(start_r.i),cnvf(end_r.i),cnvf(start_i.i),
             cnvf(end_i.i),func_num);
           fprintf(console,"MaxC=%d, CI=%d, CO=%d, CS=%d, CM=%d, CD=%d, CT=%d
MM=%d\n",
             max_count,color_inc,color_offset,color_set,color_mode,color_div,color_inset,
max_mem_y);
           goto command;
         }
         if (secchar == 'A') {
           fp = fopen(argpos,"w");
           if (fp == NULL) {
             fprintf(console,"Cannot open file '%s'\n",argpos);
             goto command;
           }
           putc((char) 1,fp);
           fwrite(&start_r,sizeof(start_r),1,fp);
           fwrite(&start_i,sizeof(start_i),1,fp);
           fwrite(&end_i,sizeof(end_i),1,fp);
           fwrite(&max_x,sizeof(max_x),1,fp);
           fwrite(&max_y,sizeof(max_y),1,fp);
           want_read = TRUE;
```

```
    for (y=0;y 320 && !(color_mode & 4)) ||
         (temp > 640 && (color_mode & 4))) {
      fputs("Illegal parameter!\n",console);
      goto command;
    }
    max_x = temp;
    max_mem = max_mem_y * MAXX;
    max_mem /= max_x;
    if (v_fp) {
      fclose(v_fp);
      v_fp = NULL;
    }
    goto command;
  }
  if (secchar == 'Y') {
    sscanf(argpos,"%d",&temp);
    if (temp < 5 ||
         (temp > 200-STARTY && !(color_mode & 2)) ||
         (temp > 400-STARTY && (color_mode & 2))) {
      fputs("Illegal parameter!\n",console);
      goto command;
    }
    max_y = temp;
    if (v_fp) {
      fclose(v_fp);
      v_fp = NULL;
    }
    goto command;
  }
  if (secchar == 'C') {
    sscanf(argpos,"%d",&temp);
    if (temp * color_inc + color_offset > 4095) {
      fputs("More than 4096 colors!\n",console);
      goto command;
    }
    if (temp < 2) {
      fputs("MaxCount must be greater than 1\n",console);
      goto command;
    }
    max_count = temp;
    goto command;
  }
  if (secchar == 'M') {
    sscanf(argpos,"%d",&temp);
    if (temp < 5) {
      fputs("Number of lines must be >4!\n",console);
      goto command;
    }
```

```
      free(v_mand_store);
      v_mand_store = (UWORD *)malloc(MAXX*temp*sizeof(UWORD));
      if (v_mand_store == NULL) {
        fputs("Can't allocate that much memory for set storage",console);
        v_mand_store = (UWORD *)malloc(MAXX*max_mem_y*sizeof(UWORD));
        if (v_mand_store == NULL)
          abort("Can't reallocate memory!!!");
        goto command;
      }
      max_mem_y = temp;
      if (v_fp) {
        fclose(v_fp);
        v_fp = NULL;
      }
      max_mem = MAXX * max_mem_y;
      max_mem /= max_x;
      goto command;
    }
    ill_cmd();
    goto command;
  case 'L':
    if (v_fp) {
      fclose(v_fp);
      v_fp = NULL;
    }
    v_fp = fopen(stpblk(cmd+1),"r");
    if (v_fp == NULL) {
      fprintf(console,"Cannot open file '%s'\n",stpblk(cmd+1));
      goto command;
    }
    if (getc(v_fp) != 1) {
      fputs("File is not in proper format\n",console);
      fclose(v_fp);
      v_fp = NULL;
      goto command;
    }
    fread(&start_r,sizeof(start_r),1,v_fp);
    fread(&end_r,sizeof(end_r),1,v_fp);
    fread(&start_i,sizeof(start_i),1,v_fp);
    fread(&end_i,sizeof(end_i),1,v_fp);
    fread(&max_x,sizeof(max_x),1,v_fp);
    fread(&max_y,sizeof(max_y),1,v_fp);
    v_offset = 25L;
    modified = FALSE;
    v_starty = max_mem+1;
    want_read = TRUE;
    v_pos_line(0);
    goto command;
```

```
 case 'X':
   if (secchar == 'R') {
      sscanf(argpos,"%f",&scale.f);
      scale.i = SPFieee(scale.i);
      start_r.i = SPAdd(start_r.i,scale.i);
      end_r.i = SPAdd(end_r.i,scale.i);
      if (v_fp) {
         fclose(v_fp);
         v_fp = NULL;
      }
      goto command;
   }
   if (secchar == 'I') {
      sscanf(argpos,"%f",&scale.f);
      scale.i = SPFieee(scale.i);
      start_i.i = SPAdd(start_i.i,scale.i);
      end_i.i = SPAdd(end_i.i,scale.i);
      if (v_fp) {
         fclose(v_fp);
         v_fp = NULL;
      }
      goto command;
   }
   ill_cmd();
   goto command;
 case 'Z':
   if (secchar != 'R' && secchar != 'I' && secchar != 'B') {
      ill_cmd();
      goto command;
   }
   if (v_fp) {
      fclose(v_fp);
      v_fp = NULL;
   }
   if (secchar != 'I' ) {
      /* scale along the real axis */
      sscanf(argpos,"%f",&scale.f);
ZoomAlongDarling(SPFieee(scale.i), SPFlt(1));
   }
   if (secchar != 'R') {
      /* scale along the complex axis */
      sscanf(argpos,"%f",&scale.f);
ZoomAlongDarling(SPFlt(1), SPFieee(scale.i));
   }
   goto command;
 case 'C':
   if (secchar == 'I') {
      sscanf(argpos,"%d",&temp);
```

```
    if (max_count * temp + color_offset > 4095) {
      fputs("More than 4096 colors!\n",console);
      goto command;
    }
    if (temp < 1) {
      fputs("Increment must be greater than 0!\n",console);
      goto command;
    }
    color_inc = temp;
    goto command;
  }
  if (secchar == 'O') {
    sscanf(argpos,"%d",&temp);
    if (max_count * color_inc + temp > 4095) {
      fputs("More than 4096 colors!\n",console);
      goto command;
    }
    if (temp < 0) {
      fputs("Negative offset illegal!\n",console);
      goto command;
    }
    color_offset = temp;
    goto command;
  }
  if (secchar == 'S') {
    sscanf(argpos,"%d",&temp);
    if (temp < 0 || temp > 1) {
      fputs("Illegal color set!\n",console);
      goto command;
    }
    color_set = temp;
    init_colors();
    goto command;
  }
  if (secchar == 'M') {
    sscanf(argpos,"%d",&temp);
    if (temp < 0 || temp > 7 || ((temp&4) && !(temp&1))) {
      fputs("Illegal graphics mode!\n",console);
      goto command;
    }
    color_mode = temp;
    if (!(color_mode & 0x2))
      if (max_y > 200-STARTY)
        max_y = 200-STARTY;
    if (!(color_mode & 0x4))
      if (max_x > 320)
        max_x = 320;
    goto command;
```

```
    }
    if (secchar == 'D') {
      sscanf(argpos,"%d",&temp);
      if (temp < 1) {
        fputs("Divisor must be greater than 0!\n",console);
        goto command;
      }
      color_div = temp;
      goto command;
    }
    if (secchar == 'T') {
      sscanf(argpos,"%d",&temp);
      if (temp < 0 || temp > 4095) {
        fputs("Color must be between 0 and 4095!\n",console);
        goto command;
      }
      color_inset = temp;
      goto command;
    }
    ill_cmd();
    goto command;
  case 'F':
    sscanf(stpblk(cmd+1),"%d",&temp);
    if (temp < 0 || temp > 1) {
      fputs("Function number must be 0 or 1!\n",console);
      goto command;
    }
    func_num = temp;
    goto command;
  case 'P':
    sscanf(stpblk(cmd+1),"%d",&temp);
    SetPresets(temp);
    goto command;
  case 'D':
    if (v_fp == NULL) {
      fputs("Must enerate or oad first!\n",console);
      goto command;
    }
    if (disp_mand() == NULL)
      wait_close();
    goto command;
  case 'G':
    if (gen_mand() == NULL)
      wait_close();
    goto command;
  case 'A':
    if (v_fp == NULL) {
      fputs("Must enerate or oad first!\n",console);
```

```
      goto command;
    }
    if (!(color_mode & 1)) {
      fputs("Cannot be in hold and modify!\n",console);
      goto command;
    }
    anal_mand();
    goto command;
  case ';':
    goto command; /* Lattice will complain about this line! */
  case '
```