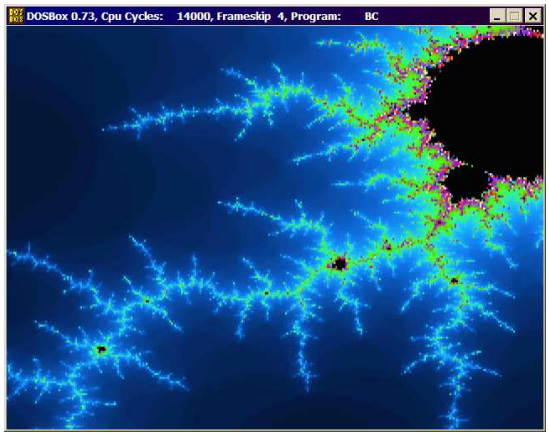
Mandelbrot boundary tracing example for Youtube – © Joel Yliluoma




```
#include <stdio.h>  /* for getchar */
#include <string.h> /* for memset */
#include <dos.h>    /* for outportb */
#include <math.h>   /* for sin,cos */

/* My custom graphics library begins here */
static void SetGFX()  { _asm { mov ax,0x13; int 0x10 } }
static void SetText()  { _asm { mov ax,0x03; int 0x10 } }
static unsigned char _far* VRAM = (unsigned char _far*) 0xA0000000UL;
static void PutPixel(unsigned x,unsigned y, int c) { VRAM[y*320u+x] = c; }
static void SetPalette(int index, int r,int g,int b)
    { outportb(0x3C8,index);
      outportb(0x3C9,r);
      outportb(0x3C9,g);
      outportb(0x3C9,b); }
/* End graphics library */

static const unsigned Width=320, Height=200, MaxIter=768;
static const double cre=-1.36022, cim=0.0653316, diam=0.035;
//static const double cre=-.5, cim=0, diam=3;
static const double minr = cre-diam*.5, mini = cim-diam*.5;
static const double maxr = cre+diam*.5, maxi = cim+diam*.5;
static const double stepr = (maxr-minr) / Width;
static const double stepi = (maxi-mini) / Height;

static int Iterate(double x,double y) /* c = (x,y) */
{
    int iter;
    double r=x, i=y;               /* z = (r,i) */
    for(iter=0; iter<MaxIter; ++iter)
    {
        double r2 = r*r, i2 = i*i;
        if(r2+i2 >= 4.0) break;     /* if |z| >= 2 */
        double ri = r*i;
        i = ri+ri + y;             /* z := z² + c */
        r = r2-i2 + x;
    }
    return iter;
}

enum { Loaded=1, Queued=2 };
static int          _huge Data[Width*Height] = {0};
static unsigned char _far Done[Width*Height] = {0};
static const unsigned QueueSize = (Width+Height)*4;
static unsigned Queue[QueueSize];
static unsigned QueueHead=0, QueueTail=0;

static void AddQueue(unsigned p)
{
    if(Done[p] & Queued) return;
    Done[p] |= Queued;
    Queue[QueueHead++] = p;
    if(QueueHead == QueueSize) QueueHead = 0;
}
static int Load(unsigned p)
{
   if(Done[p] & Loaded) return Data[p];
   unsigned x = p % Width, y = p / Width;
   int result = Iterate(minr + x*stepr, mini + y*stepi);
   PutPixel(x,y, result);
   Done[p] |= Loaded;
   return Data[p] = result;
}
static void Scan(unsigned p)
{
   unsigned x = p % Width, y = p / Width;
   int center = Load(p);
   int ll = x >= 1, rr = x < Width-1;
   int uu = y >= 1, dd = y < Height-1;
   /* ^These are booleans, but bc31 does
    * not support the "bool" type */
   /* If a neighbor color differs from
    * the center, scan the neighbor in turn */
   int l = ll && Load(p-1) != center;
   int r = rr && Load(p+1) != center;
   int u = uu && Load(p-Width) != center;
   int d = dd && Load(p+Width) != center;
   if(l) AddQueue(p-1);
   if(r) AddQueue(p+1);
   if(u) AddQueue(p-Width);
   if(d) AddQueue(p+Width);
   /* The corner pixels (nw,ne,sw,se) are also neighbors */
   if((uu&&ll)&&(l||u)) AddQueue(p-Width-1);
   if((uu&&rr)&&(r||u)) AddQueue(p-Width+1);
   if((dd&&ll)&&(l||d)) AddQueue(p+Width-1);
   if((dd&&rr)&&(r||d)) AddQueue(p+Width+1);
}

int main()
{
   SetGFX();

   /*for(unsigned y=0; y<Height; ++y)
      for(unsigned x=0; x<Width; ++x)
         PutPixel(x,y, x);*/

   /*for(unsigned y=0; y<Height; ++y)
      for(unsigned x=0; x<Width; ++x)
         PutPixel(x,y, Iterate( minr+x*stepr, mini+y*stepi ) ); */

    memset(VRAM, 2, Width*Height); /* clear screen */
    /* Set up an arbitrary but cool palette */
    for(unsigned c=0; c<256; ++c)
        SetPalette(c, 32-31*cos(c*.01227*1),
                      32-31*cos(c*.01227*3),
                      32-31*cos(c*.01227*5));

    /* Begin by adding the screen edges into the queue */
    for(unsigned y=0; y<Height; ++y)
    {
        AddQueue(y*Width + 0);
        AddQueue(y*Width + (Width-1));
    }
    for(unsigned x=1; x<Width-1; ++x)
    {
        AddQueue(0*Width + x);
        AddQueue((Height-1)*Width + x);
    }

    /* Process the queue (which is actually a ring buffer) */
    unsigned flag=0;
    while(QueueTail != QueueHead)
    {
        unsigned p;
        if(QueueHead <= QueueTail || ++flag & 3) {
            p = Queue[QueueTail++];
            if(QueueTail == QueueSize) QueueTail=0;
        } else p = Queue[--QueueHead];
        Scan(p);
    }

    /* Lastly, fill uncalculated areas with neighbor color */
    for(unsigned p=0; p<Width*Height; ++p)
        if(Done[p] & Loaded)
            if(!(Done[p+1] & Loaded))
                VRAM[p+1] = VRAM[p],
                Done[p+1] |= Loaded;

    getchar();

    SetText();
    return 0;
}
```

The End — written in March 2010.

P.S. The colors in this code are hints for me in directing the Youtube video. – Joel Yliluoma