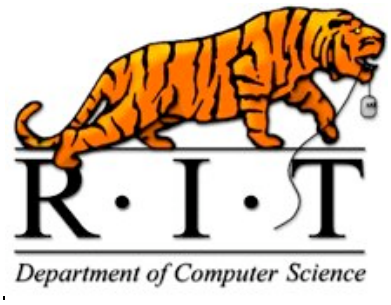


Project 1
GBA Point Drawing Algorithms

Computer Graphics 1

ICSS 4003-570-01, 4003-761-01

Winter Quarter 20062

Due Dates

Project 1 is due on Saturday, 12/23/2006.

Introduction

In this project, you will implement some of the drawing routines that we have been discussing in class for the GBA. This will help improve your understanding of these algorithms, and introduce you to console programming on the GBA.

Here are the requirements for this project:

- You must program your solutions entirely in C (and/or assembly).
- You are not allowed to use any API's that come with the GBA development library you are using.
- Once control is passed to your main routine, you should set up the GBA for Mode 3 drawing. Your program must then only draw colored points directly to the video buffer using the API listed below.
- You must follow the drawing API's listed below exactly, so that I can recreate and test your project on my system. Do not change the header files!

Important Links:

- http://www.ngine.de/ - Ham development toolkit for Windows/Linux.
- http://sourceforge.net/projects/devkitpro/ - DevKitAdv package for Mac/Windows/Linux.
  - http://mycourses.rit.edu - A skeleton tar file for command line building of the project can be found under the "News" section.
- http://www.cs.rit.edu/usr/local/pub/sps/courses/cg1/project1/ - The reference source files for this project:
  - gba.h – Interface to the GBA drawing routines you will implement. **NO MODIFY.**
  - gba.c – Implementation of required GBA drawing routines. **WRITE FROM SCRATCH.**
  - fractal.c – Client program for rendering fractal image (activity 1). **MODIFY.**
  - bresenham.c – Client program for drawing lines (activity 2). **NO MODIFY.**
  - wu.c – Client program for drawing anti-aliased lines using Wu's algorithm (activity 3). **NO MODIFY.**
  - circle.c – Client program for drawing circles using the Midpoint line algorithm (activity 4). **NO MODIFY.**
- http://freespace.virgin.net/hugo.elias/graphics/x_wuline.htm - Writeup for Wu's anti-aliasing algorithm.

Routines to Implement

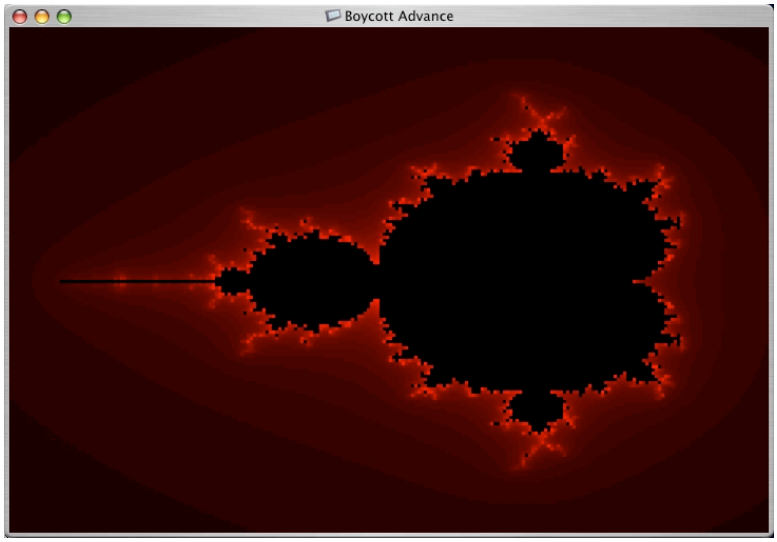
You are to implement the following routines using the GBA video buffer as a basis. A sample header file can be found at http://www.cs.rit.edu/usr/local/pub/sps/courses/cg1/project1/gba.h. The implementation of these routines should go in a file called `gba.c`.

Function	Description
`void GBA_init()`	Initialize the GBA hardware to do mode 3 drawing with background 2 enabled.
`void GBA_drawPixel(int x, int y, int r, int g, int b)`	Draw a pixel at `(x,y)` with color `(r,g,b)`.
`void GBA_drawLine(int x0, int y0, int x1, int y1, int r, int g, int b, int alias)`	Draw a line from `(x0,y0)` to `(x1,y1)` with pixel

	color `(r,g,b)` using Bresenham's line drawing algorithm. It should call `GBA_drawPixel()` to render each point. If `alias` is nonzero, the line should be drawn using Wu's anti-aliasing algorithm
`void GBA_drawCircle(inr radius, int cx, int cy, int r, int g, int b)`	Draw a circle centered at `(cx,cy)` with a radius of `radius` and color `(r,g,b)`.

Part 1 – Rendering the Mandelbrot Set

Your first program will test the `GBA_init()` and `GBA_drawPixel()` functions. It should render the popular fractal image for the Mandelbrot set by only drawing points. The image must be in color and should follow the general algorithm listed below.



The `main` skeleton can be found at http://www.cs.rit.edu/usr/local/pub/sps/courses/cg1/project1/fractal.c.

The reference image can be found at http://www.cs.rit.edu/usr/local/pub/sps/courses/cg1/project1/fractal.jpg.

The mathematician Benoit Mandelbrot is considered the grandfather of fractal geometry. He created visual imagery out of mathematical sets using complex numbers. In the image above, the black area represents points that are part of the set. The colored points represent how quickly those points "escape" from the set during a simple iterative calculation:

$z = z^2 + C$

The algorithm is as follows:

```
loop hy over each scan line
      cy is a point on the complex plane: hy/screenHeight * scaleFactorY
      loop hx over each pixel per scan line
            cx is a point on the complex plane: hx/screenWidth * scaleFactorX
            reset x and y
            while the point is still in the set and maximum iterations has not been hit
                  newx=  x^2 - y^2 + cx;
                  y = 2xy + cy;
                  x = newx
                  if x^2 + y^2 is greater than 100 the point is not in the set
            if the point escaped, draw it using the iteration count as a color index
            otherwise the point should be black
```

Notes:

- The space in which the fractal exists is roughly in the area of `(-2, 2)` to `(2, -2)`, with the center of the image at `(0,0)`. These coordinates need to get transformed to the GBA coordinates when plotting. When choosing the values for `hx` and `hy`, some find it easier to go from `-120` to `120`, and `-80` to `80` respectively when computing `cx` and `cy`. When it actually comes time to plot the pixel on the GBA, `(120,80)` are used as offsets.
- You will have to experiment with the scale factors used when calculating `cx` and `cy` to find the suitable range which will display the image above. As long as the entire fractal is displayed it is acceptable (scaling does not have to be perfect).
- When debugging, it is helpful to use a large scale factor to get a general idea of the final image without spending too much time computing.
- It is a good idea to iterate between `0` and `31` to determine whether the point is in the set or not, because it yields an immediate color value to use when plotting. The larger this value is, the longer it will take your program to run.
- This program will run fairly slow on the GBA. You can make your screen region smaller to test with. You should also check if your emulator can run in "fast mode".
- You can use whatever color values/scheme you want when plotting.

Part 2 – Bresenham's Line Drawing Algorithm

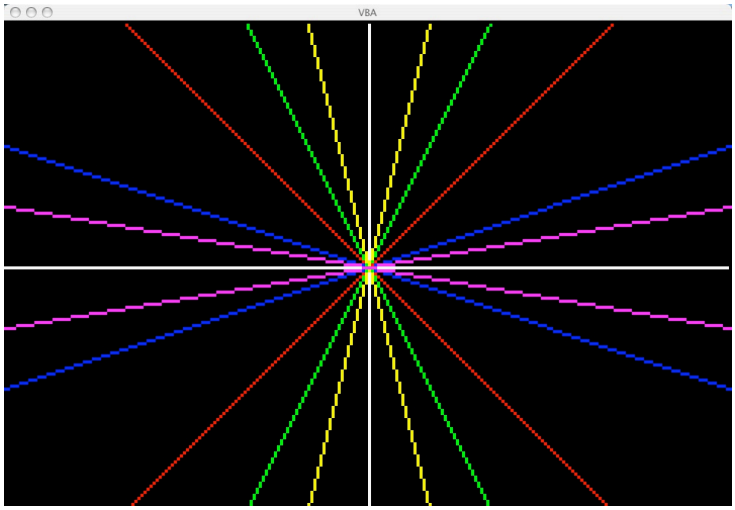
Your second program will additionally test your `GBA_drawLine()` function without anti-aliasing. Your program should be able to draw lines of varying slopes and directions using Bresenham's line drawing algorithm discussed in class. The routine should work entirely with integers and will contain no floating point calculations whatsoever.

Notes:

- It's best to always draw from left to right. Re-order your endpoints if this is not the case.
- The `abs()` routine from the C library is useful for computing an integer absolute value.
- There is only one entry point for line drawing in the next two activities. I recommend that you implement two local functions, based on whether the line is to be drawn using bresenham's or wu's.

The `main` program can be found at http://www.cs.rit.edu/usr/local/pub/sps/courses/cg1/project1/bresenham.c.

The reference image can be found at http://www.cs.rit.edu/usr/local/pub/sps/courses/cg1/project1/bresenham.jpg.



Part 3 – Drawing Wu Anti-aliased Lines

Your third program will additionally test your `GBA_drawLine()` function with anti-aliasing. Your program should smooth the jaginess of the lines using Wu's anti-aliasing algorithm discussed in class. For this algorithm, you are allowed to use floating point calculations. A link to the Wu algorithm can be found here:

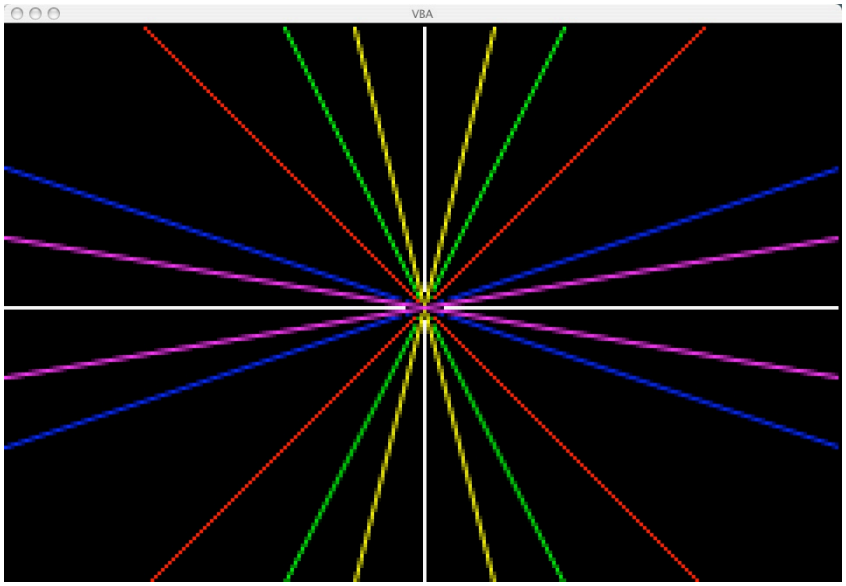
http://freespace.virgin.net/hugo.elias/graphics/x_wuline.htm

The `main` program can be found at http://www.cs.rit.edu/usr/local/pub/sps/courses/cg1/project1/wu.c.

The reference image can be found at http://www.cs.rit.edu/usr/local/pub/sps/courses/cg1/project1/wu.jpg.

Notes:

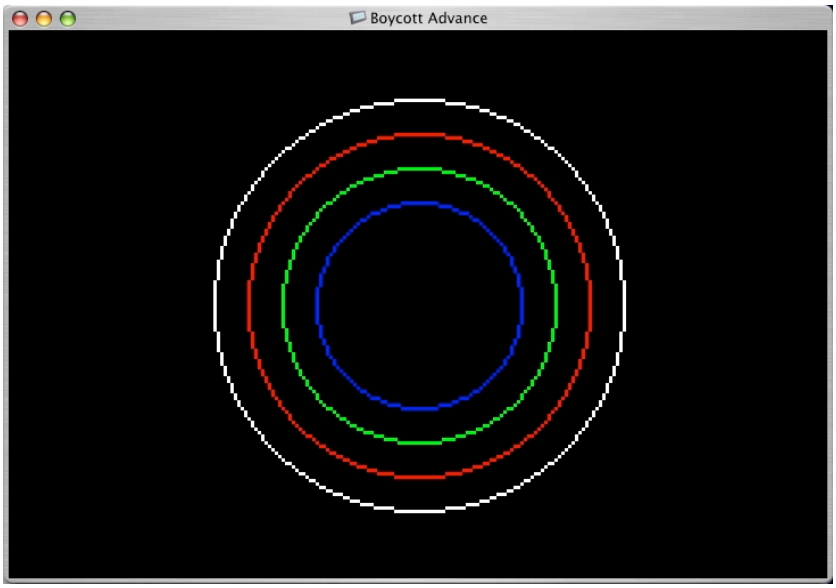
- There are routines in the C math library for returning the integral and fractional parts of a floating point value. They are *modf()* and *modff().*



Part 4 – Drawing Circles with the Midpoint Algorithm

Your fourth program will additionally test your `GBA_drawCircle()` function. It should draw circles using the Midpoint Circle Algorithm with eight way symmetry, as discussed in class. The routine should work entirely with integers and cannot contain any floating point calculations whatsoever.

Use the notes from class as a guideline when implementing this algorithm.



The `main` program can be found at http://www.cs.rit.edu/usr/local/pub/sps/courses/cg1/project1/circle.c.

The reference image can be found at http://www.cs.rit.edu/usr/local/pub/sps/courses/cg1/project1/circle.jpg.

What to Submit

We will be using a dropbox on mycourses for submitting this project. **Make sure you do not submit all the files individually**. You should zip/tar a top level folder which contains all the code required for each of the activities. That single archived file is what you will then submit through mycourses.

You must include a `README` in your archive. It should include your name, indicate what platform/toolkit you built your program with, along with any special build instructions for re-creating your executable.

The minimum archived files that are necessary to submit are:

- `fractal.c` – Your mandelbrot implementation.
- `gba.c` – Your implementation of all the drawing routines.
- `README` – A text file explaining your project.

Grading

When grading, all the reference files will be used except for `fractal.c` and `gba.c`. This is why it is very important that you do not modify any other source files. Your code will be built under Linux using DevKitPro. If you are using HAM, do not put any specific HAM calls anywhere in your code.

Your grade will be based on your re-implementation of the required routines and their usability with the supplied test programs. Use the supplied sample gradesheet as a reference.

Most of the credit for this project is based on the correct functionality of your programs. If you do not submit working code you cannot receive credit for the assignment.

Common Problems / General Notes

The following is a list of common problems people run into when working on this project. Most of them are related to the correct use of the C language:

- Initialize floats and doubles correctly (where it is allowed):

```
float x;

double d;

x = 0.0f;

d = 1.2;
```

- If you use the `abs()` routine to compute the absolute value of a number, you must `#include <stdlib.h>`, otherwise it is an implicit declaration with `gcc`. Windows plays nice and automatically gives you an implementation of `abs()`, but it won't on other platforms.
- Make sure you understand how `#include` works:
  - `#include <math.h>`: Includes `math.h` from standard headers.
  - `#include "gba.h"`: Include `gba.h` from the current directory.
- You must declare functions before defining them.
- A lot of common errors can be resolved by recompiling your code with warnings enabled:
  - `$(CCC) -Wall -pedantic -ansi`
- Most C is still using the C90 standard, so:
  - C++ style comments are not allowed (i.e. `// comment`). You must use: `/* comment */`.
  - All local variables must be declared at the top of the block they appear in. You are not allowed to mix declarations in with other statements.
  - `bool` is not a type. Use a `int` or `char` instead.