Fractals

New Mexico

Supercomputing Challenge

Final Report

April 1, 2009

Team number: 64

Los Alamos Middle School

Team Members:

Johnny Jacobs

Thomas Henderson

Teachers:

Bob Dryja

Project Mentor:

Michael Henderson

Executive Summary

We used Java to write a program that plots the Mandelbrot set. In mathematics, the Mandelbrot set, named after Benoît Mandelbrot, is a set of points in the complex plane, the boundary of which forms a fractal[1]. Our program first computes which numbers are in the Mandelbrot set using the equation: $z_{n+1} = z_n^2 + c$. Then it plots the numbers on the complex plane using a drawing program that our mentor helped us to create.

Problem

The problem we are investigating is using Java to make fractals. Fractals are repeating patterns that use both real and imaginary numbers so that no matter how far in you zoom, they will appear very similar to the original fractal. This was our first time using Java and we thought making fractals would help us to start learning code.

Method

We used an integrated development environment with visualizations for improving software comprehensibility to program in Java. It is called jGrasp[2]. We started with the equation: $z_{n+1} = z_n^2 + c$. Then we ran it through several iterations using different numbers to determine which numbers were in the Mandelbrot set. After we got our program running to compute whether or not a number was in the Mandelbrot set, we needed to create a drawing program to plot the set of points in the complex plane. We then combined the two programs to make one larger working program that draws the Mandelbrot set. Our mentor helped us create these programs. During the course of our project, we learned how to work with imaginary and complex numbers. We also researched fractals and had meetings to discuss our progress.

Results

First, we determined which numbers were in the Mandelbrot set. If numbers, when put into the equation, went to infinity, then they were not in the Mandelbrot set. However, if they had a repeating pattern of values then they were in the Mandelbrot set. We were able to make a program that creates a fractal with fairly good graphics. The program allows us to change colors, zoom in on certain points, and change the size of the image and screen. We have gotten a solid background in Java programming and code.

Conclusion

The conclusion we reached by carefully analyzing our results is that it is interesting to work in an imaginary coordinate system and useful to visualize imaginary and complex numbers in a graphical environment.

---

[1] http://en.wikipedia.org/wiki/Mandelbrot_set
[2] http://www.jgrasp.org/

Achievement

Our most significant achievement was succeeding in writing a working program that did what we set out to make it do. It was fun to learn about fractals and complex numbers.

Acknowledgements

For our acknowledgements we would like to thank our mentor, Michael Henderson who helped us with a lot of the programming specifically the graphics programming. We would also like to thank the other parents, Elaine Jacobs and Nancy Henderson, who helped us with this project. Also, the judges at our presentation had very helpful comments.

Code

```
Ïimport java.awt.BasicStroke;
ÏÏÏimport java.awt.BorderLayout;
ÏÏÏimport java.awt.Color;
ÏÏÏimport java.awt.Dimension;
ÏÏÏimport java.awt.Graphics;
ÏÏÏimport java.awt.Graphics2D;
ÏÏÏimport java.awt.Point;
ÏÏÏimport java.awt.RenderingHints;
ÏÏÏimport java.awt.geom.Ellipse2D;
ÏÏÏimport java.awt.geom.Arc2D;
ÏÏÏimport java.awt.geom.Line2D;
ÏÏÏimport java.awt.geom.Rectangle2D;
ÏÏÏimport java.awt.geom.RoundRectangle2D;
ÏÏÏimport javax.swing.JComponent;
ÏÏÏimport javax.swing.JFrame;
ÏÏÏimport javax.swing.UIManager;
ÏÏÏimport java.awt.Image;
ÏÏÏimport java.awt.image.BufferedImage;


ÏÕÖ×public class Mandelbrot extends JFrame
ÏÏ§{
ÏÏ§
ÏÏ§
ÏÏ§
ÏÏ§
ÏÏ§ÏÞßàpublic Mandelbrot()
ÏÏ§ÏÏ§{
ÏÏ§ÏÏ¨¹¹ÏsetLayout(new BorderLayout());
ÏÏ§ÏÏ¨¹¹ÏgetContentPane().add(new DrawComponent(),
BorderLayout.CENTER);
ÏÏ§ÏÏ©}
ÏÏ§
```

```
ÏÏ§ÏÕÖ×public static class DrawComponent extends JComponent
ÏÏ§ÏÏ§{
ÏÏ§ÏÏ§ÏÞßàprivate int makeARGB( int alpha, int red, int grn,
int blu ) {
ÏÏ§ÏÏ§Â¹Ä¹¹Ïreturn( alpha<<24 | red<<16 | grn<<8 | blu );
ÏÏ§ÏÏ§ÏÏ©}
ÏÏ§ÏÏ§ÏÞßàpublic void paintComponent(Graphics g)
ÏÏ§ÏÏ§ÏÏ§{
ÏÏ§ÏÏ§ÏÏ¨¹íÏdouble xmin, xmax, xinc, x, xmid;
ÏÏ§ÏÏ§ÏÏ¨¹íÏdouble ymin, ymax, yinc, y, ymid;
ÏÏ§ÏÏ§ÏÏ¨¹íÏint nj, j;
ÏÏ§ÏÏ§ÏÏ¨¹íÏint ni, i;
ÏÏ§ÏÏ§ÏÏ¨¹íÏint n, val, nmax;
ÏÏ§ÏÏ§ÏÏ¨¹íÏdouble cr;
ÏÏ§ÏÏ§ÏÏ¨¹íÏdouble ci;
ÏÏ§ÏÏ§ÏÏ¨¹íÏdouble zr, new_zr;
ÏÏ§ÏÏ§ÏÏ¨¹íÏdouble zi, new_zi;
ÏÏ§ÏÏ§ÏÏ¨¹íÏdouble zabs, dx, dy;
ÏÏ§ÏÏ§ÏÏ§
ÏÏ§ÏÏ§ÏÏ¨¹íÏint XSIZE = 750;
ÏÏ§ÏÏ§ÏÏ¨¹íÏint YSIZE = 750;
ÏÏ§ÏÏ§ÏÏ§
ÏÏ§ÏÏ§ÏÏ§
ÏÏ§ÏÏ§ÏÏ¨¹¹Ïnmax  = 1024;
ÏÏ§ÏÏ§ÏÏ§
ÏÏ§ÏÏ§ÏÏ¨¹íÏBufferedImage myImage = new BufferedImage(XSIZE,
YSIZE, BufferedImage.TYPE_INT_ARGB);
ÏÏ§ÏÏ§ÏÏ¨¹íÏGraphics2D g2 = myImage.createGraphics();
ÏÏ§ÏÏ§ÏÏ¨¹íÏGraphics2D g2d = (Graphics2D)g;
ÏÏ§ÏÏ§ÏÏ¨¹¹Ïg2d.setRenderingHint(RenderingHints.KEY_ANTIALIASI
NG,
ÏÏ§ÏÏ§ÏÏ§ÏÏÏÏÏÏRenderingHints.VALUE_ANTIALIAS_ON);
ÏÏ§ÏÏ§ÏÏ¨¹¹Ïg2d.setStroke(new BasicStroke(3.0f));
ÏÏ§ÏÏ§ÏÏ§
ÏÏ§ÏÏ§ÏÏ¨¹íÏColor newcol1 = new Color (0.8f, 0.2f, 0.25f,
0.5f);
ÏÏ§ÏÏ§ÏÏ¨¹íÏColor newcol2 = new Color (0.25f, 0.85f,
0.15f,0.25f);
ÏÏ§ÏÏ§ÏÏ§// ----------------------------------------------------
---------------
ÏÏ§ÏÏ§ÏÏ§// Put your implementation here
ÏÏ§ÏÏ§ÏÏ§// ----------------------------------------------------
---------------
ÏÏ§ÏÏ§ÏÏ§
ÏÏ§ÏÏ§ÏÏ§   //good ranges for whole mandelbrot set
ÏÏ§ÏÏ§ÏÏ¨¹¹Ïxmin = -3.0;
ÏÏ§ÏÏ§ÏÏ¨¹¹Ïxmax =  1.0;
ÏÏ§ÏÏ§ÏÏ¨¹¹Ïymin = -2.0;
```

```
ÏÏ§ÏÏ§ÏÏ¨¹¹Ïymax =  2.0;
ÏÏ§ÏÏ§ÏÏ§
ÏÏ§ÏÏ§ÏÏ§   // interesting zoomed in region
ÏÏ§ÏÏ§ÏÏ¨¹¹Ïxmin = -0.6818;
ÏÏ§ÏÏ§ÏÏ¨¹¹Ïxmax =  -0.6782;
ÏÏ§ÏÏ§ÏÏ¨¹¹Ïymin = -0.31575;
ÏÏ§ÏÏ§ÏÏ¨¹¹Ïymax =  -0.31305;
ÏÏ§ÏÏ§ÏÏ§
ÏÏ§ÏÏ§ÏÏ§   // interesting zoomed in region
ÏÏ§ÏÏ§ÏÏ¨¹¹Ïxmin = -1.4195;
ÏÏ§ÏÏ§ÏÏ¨¹¹Ïxmax = -1.4159;
ÏÏ§ÏÏ§ÏÏ¨¹¹Ïymin = -0.0023;
ÏÏ§ÏÏ§ÏÏ¨¹¹Ïymax =  0.0004;
ÏÏ§ÏÏ§ÏÏ§
ÏÏ§ÏÏ§ÏÏ§
ÏÏ§ÏÏ§ÏÏ§
ÏÏ§ÏÏ§ÏÏ¨¹¹Ïdx = xmax-xmin;
ÏÏ§ÏÏ§ÏÏ¨¹¹Ïdy = ymax-ymin;
ÏÏ§ÏÏ§ÏÏ§
ÏÏ§ÏÏ§ÏÏ¨¹¹Ïxmid = (xmin+xmax)/2.0;
ÏÏ§ÏÏ§ÏÏ¨¹¹Ïymid = (ymin+ymax)/2.0;
ÏÏ§ÏÏ§ÏÏ§
ÏÏ§ÏÏ§ÏÏ§
ÏÏ§ÏÏ§ÏÏ¨¹¹Ïdx = dx;
ÏÏ§ÏÏ§ÏÏ¨¹¹Ïdy = dy;
ÏÏ§ÏÏ§ÏÏ§
ÏÏ§ÏÏ§ÏÏ¨¹¹Ïxmin = xmid - 0.5*dx;
ÏÏ§ÏÏ§ÏÏ¨¹¹Ïxmax = xmid + 0.5*dx;
ÏÏ§ÏÏ§ÏÏ¨¹¹Ïymin = ymid - 0.5*dy;
ÏÏ§ÏÏ§ÏÏ¨¹¹Ïymax = ymid + 0.5*dy;
ÏÏ§ÏÏ§ÏÏ§
ÏÏ§ÏÏ§ÏÏ§
ÏÏ§ÏÏ§ÏÏ§
ÏÏ§ÏÏ§ÏÏ§
ÏÏ§ÏÏ§ÏÏ§
ÏÏ§ÏÏ§ÏÏ§
ÏÏ§ÏÏ§ÏÏ§
ÏÏ§ÏÏ§ÏÏ§
ÏÏ§ÏÏ§ÏÏ¨¹¹Ïni=XSIZE;
ÏÏ§ÏÏ§ÏÏ¨¹¹Ïnj=YSIZE;
ÏÏ§ÏÏ§ÏÏ¨¹¹Ïxinc= (xmax -xmin)/(double) ni;
ÏÏ§ÏÏ§ÏÏ¨¹¹Ïyinc= (ymax -ymin)/(double) nj;
ÏÏ§ÏÏ§ÏÏ§
ÏÏ§ÏÏ§ÏÏ¨¹¹Ïx = xmin + xinc / 2.0;
ÏÏ§ÏÏ§ÏÏ¨¹¹±for (i =0; i < ni; i++){
ÏÏ§ÏÏ§ÏÏ§ÏÏ5
ÏÏ§ÏÏ§ÏÏ§ÏÏ7¹¹Ïy = ymin + yinc /2.0;
ÏÏ§ÏÏ§ÏÏ§ÏÏ7¹¹±for (j =0; j < nj; j++){
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ7¹¹Ïzr = 0.0; zi = 0.0;
```

```
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ7¹¹Ïcr = x;   ci = y;
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ7¹¹Ïzabs = 0.0;
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5   //System.out.println("c = " + x + " " + y +
"i");
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5   // (z_r + z_i i)*(z_r + z_i i) + (c_r + c_i
i)
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5   //  = (z_r^2 - z_i^2 + c_r) + (2 z_i + c_i)
i
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5   // do a max of nmax iterations
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ7¹¹±for (n = 0; n <=nmax; n++ ){
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5ÏÏ7¹¹Ïnew_zr = zr*zr-zi*zi + cr;
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5ÏÏ7¹¹Ïnew_zi = 2.0*zi*zr+ci;
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5ÏÏ7¹¹Ïzabs = Math.sqrt(new_zr*new_zr +
new_zi*new_zi);
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5ÏÏ7¹¹Ïzr = new_zr;
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5ÏÏ7¹¹Ïzi = new_zi;
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5ÏÏ7¹³´if ( zabs > 2.0 ) {
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5ÏÏ5Ï6§   // if magnitude (distance from origin)
ever gets
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5ÏÏ5Ï6§   // above 2 -- then it will always
increase forever
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5ÏÏ5Ï6§   // break out of the loop if this
happens
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5ÏÏ5Ï6§   // the n value we got to will then be
an indicator
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5ÏÏ5Ï6§   // of "how close to being in the set"
this point is.
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5ÏÏ5Ï6§   // If n is high its close to being in
the set (it lasted
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5ÏÏ5Ï6§   // a long time.)
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5Â¹ÇÏ6¾¹¹Ïbreak;
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5ÏÏ5Ï¶Ï}
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5ÏÏ°}
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5   //System.out.println("z = " + zr + " " + zi
+ "i");
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5   //System.out.println("n = " + n);
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ7¹³´if (n >= nmax) {
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5Ï6¾¹¹Ïval = 0; // we'll call these "in the set"
and set to zero (black)
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5Ï6Ï}
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5Ïö´else {
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5Ï¸¾¹¹Ïval =
(int)((double)n/(double)nmax*255.0); // scale these from 0-
>255
```

```
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5ÏÈÏ}
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5   // set the color of the pixel. The four vals
are alpha, red, grn, blu.
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5   // They can be in the range 0->255. alpha=0
means transparent alpha=255
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5   // means opaque. having red=grn=blu gives a
grayscale image.
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ7¹¹ÏmyImage.setRGB( i, j, makeARGB(255, 0, val,
0) );
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ5
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ7¹¹Ïy = y+yinc;
ÏÏ§ÏÏ§ÏÏ§ÏÏ5ÏÏ°}
ÏÏ§ÏÏ§ÏÏ§ÏÏ7¹¹Ïx = x+xinc;
ÏÏ§ÏÏ§ÏÏ§ÏÏ°}
ÏÏ§ÏÏ§ÏÏ§
ÏÏ§ÏÏ§ÏÏ§   // Draw the image to the screen
ÏÏ§ÏÏ§ÏÏ¨¹¹Ïg2d.drawImage( myImage, 0, 0, null );
ÏÏ§ÏÏ§ÏÏ§
ÏÏ§ÏÏ§ÏÏ§
ÏÏ§ÏÏ§ÏÏ§
ÏÏ§ÏÏ§ÏÏ§
ÏÏ§ÏÏ§ÏÏ§
ÏÏ§ÏÏ§ÏÏ§
ÏÏ§ÏÏ§ÏÏ§
ÏÏ§ÏÏ§ÏÏ§
ÏÏ§ÏÏ§ÏÏ©}
ÏÏ§ÏÏ©}
ÏÏ§
ÏÏ§ÏÞßàpublic void center()
ÏÏ§ÏÏ§{
ÏÏ§ÏÏ¨¹íÏDimension screenSize = getToolkit().getScreenSize();
ÏÏ§ÏÏ¨¹íÏDimension frameSize = getSize();
ÏÏ§ÏÏ¨¹íÏPoint location = new Point((screenSize.width -
frameSize.width) / 2,
ÏÏ§ÏÏ§ÏÏÏÏÏÏ(screenSize.height - frameSize.height) / 2);
ÏÏ§ÏÏ¨¹¹ÏsetLocation(location);
ÏÏ§ÏÏ©}
ÏÏ§
ÏÏ§ÏÞßàpublic static void main(String[] args) throws Exception
ÏÏ§ÏÏ§{
ÏÏ§ÏÏ¨¹¹ÏUIManager.setLookAndFeel(UIManager.getSystemLookAndFe
elClassName());
ÏÏ§ÏÏ¨¹¹ÏJFrame.setDefaultLookAndFeelDecorated(true);
ÏÏ§ÏÏ§
ÏÏ§ÏÏ¨¹íÏMandelbrot f = new Mandelbrot();
ÏÏ§ÏÏ¨¹¹Ïf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
ÏÏ§ÏÏ¨¹¹Ïf.setSize(new Dimension(750, 750));
ÏÏ§ÏÏ¨¹¹Ïf.center();
ÏÏ§ÏÏ¨¹¹Ïf.setVisible(true);
ÏÏ§ÏÏ©}
ÏÏ©}
```