

# CS442/ECE432: Project 1

February 26, 2008

## General

- This project is **due March 25, 2008**.
- Submit via e-mail to [riesen@cs.unm.edu](mailto:riesen@cs.unm.edu) (Mail it before class on the 25th.) Your subject line must say: "Project 1 Submission".

## 1 Mandelbrot Set

### 1.1 Introduction

A Mandelbrot set is a set of points in the complex plane that are quasi-stable (i.e. will not exceed a certain threshold) when computed by iterating a function. The function we will compute is:

$$z_{k+1} = z_k^2 + c$$

where  $c$  is a complex number representing a point in the complex plane. In order to compute the  $k^{\text{th}}$  iteration, we need the value of  $c$  during the  $(k - 1)^{\text{th}}$  iteration. In other words, we can start computing  $c$  at  $k = 0$  and iterate until we reach a desired  $k$ .

When we do that, some  $c$  will head towards infinite values very quickly. Those are the values that do not belong to the Mandelbrot set. We cannot iterate forever to determine whether a point belongs to the Mandelbrot set or not. Therefore, we have to terminate iteration at some predefined value.

You can learn more about the Mandelbrot set at [http://en.wikipedia.org/wiki/Mandelbrot\\_set](http://en.wikipedia.org/wiki/Mandelbrot_set) and many other web sites.

### 1.2 Task

Create a parallel program called `mandel` that computes the Mandelbrot set. Your program should take five parameters in this order:

```
mandel LleftR LleftI UrightR UrightL Iter Xres Yres
```

For example:

```
mandel -1.5 -1.0 1.0 1.0 1000 800 800
```

Would produce the famous view of the set. The parameters have the following meaning:

**LleftR** The real part of the number represented by the lower left pixel

**LleftI** The imaginary part of the number represented by the lower left pixel

**UrightR** The real part of the number represented by the upper right pixel

**UrightL** The imaginary part of the number represented by the upper right pixel

**Iter** Number of iterations per pixel

**Xres** X-axis resolution in pixels

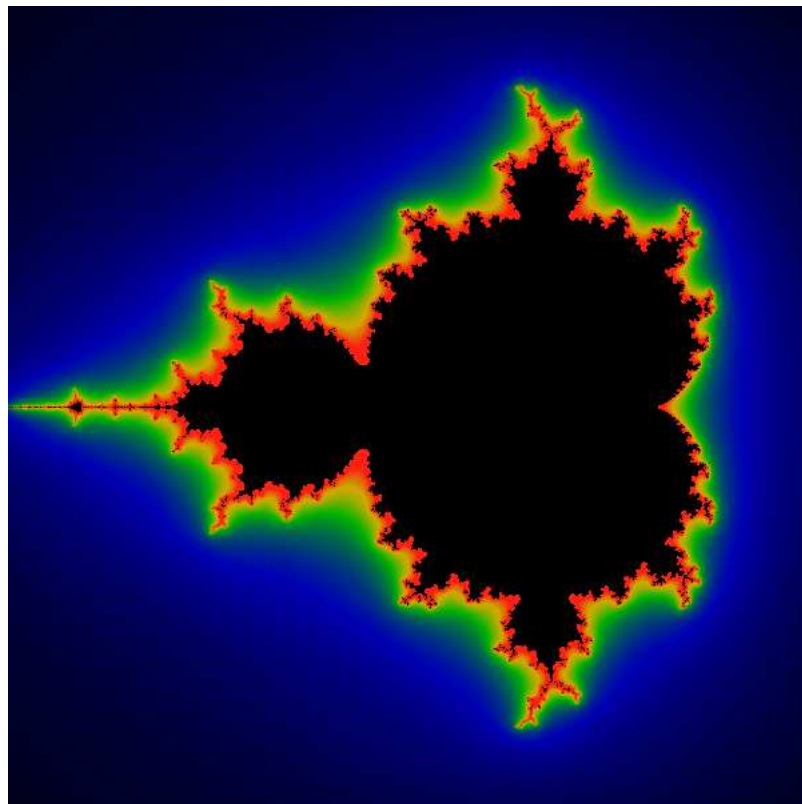
**Yres** Y-axis resolution in pixels

The program must produce an image in PPM format (described in the next section) with the resolution set on the command line. The region of the complex plane bounded by the values on the command line is divided into a grid with the width and height specified on the command line. Calculate  $c$  at the center point of each grid cell.

Your program must run on  $\geq 1$  processors. You are not allowed to incorporate code from any existing Mandelbrot or PPM program. Each rank should be responsible for about the same number of pixels, as any other node.

The program must accept an optional `-b` argument. If `-b` is set, then mark a pixel that belongs to the set black, and all others white. A pixel belongs to the set, if after `Iter` iterations it's value is  $< 2$ . **Do this task first!**

Without the `-b` option, your program should produce a color picture. The color should represent how quickly a point is escaping from the set. (The points in the set remain black.) There are several ways to do this, and <http://linas.org/art-gallery/escape/smooth.html> gives some suggestions. Here is an example:



### 1.3 Output

Your program must output a picture in the plain PPM format (as opposed to the more common raw PPM format). The plain PPM format is extremely simple (and inefficient) and therefore easy to produce. The output file can then be converted by a variety of available programs to formats such as PNG and JPG.

You can learn more about the format at [http://en.wikipedia.org/wiki/Portable\\_Pixmap\\_File\\_Format](http://en.wikipedia.org/wiki/Portable_Pixmap_File_Format) and from the ppm man page (man ppm or <http://www.martinreddy.net/gfx/2d/PPM.txt>). Briefly, the file is an ASCII file and contains the following header:

- The string “P3”
- The width and height in pixels; e.g., “600 400”
- The maximum color value; e.g., 255
  - We use one byte each for red, green, and blue
  - Therefore, we use values 0...255 and the max value is 255
- Width  $\times$  height number of RGB values in order from left to right in each row, and top to bottom row.
- Each RGB value consists of three decimal numbers between 0 and 255, separated by white space.
- All these items could be on a single line separated by white space
  - However, no individual line should be more than 70 characters.
  - Therefore, it is easier to terminate each item described above with a newline
- The # starts a comment

Here is an example:

```
P3
# Rolf's handcrafted example
4 6
255
10 110 213   20 110 213   30 10 13   140 110 213
10 100 213   20 110 203   30 10 13   140 110 213
10  90 213   20 110 193   30 10 13   140 110 213
10  80 213   20 110 183   30 10 13   140 110 213
10  70 213   20 110 173   30 10 13   140 110 213
10  60 213   20 110 163   30 10 13   140 110 213
```

After enlargement and conversion to Postscript, it looks something like this:



## 1.4 Grading

- 40 For a Mandelbrot program that works as specified and produces a black and white picture (-b option) where pixels in the set are black and pixels outside are white
- 30 For a version that produces color output
- 10 For a good description of how your program works. In a paragraph or two, describe how you divide the work among the available processors, and how you generate the output file with the data from all ranks.