Parallel Design Patterns: Exercise 1

Lawrence Mitchell

January 11, 2012

Introduction

In this series of exercises, you will be exploring some of the design patterns covered in the course. To do this, we need some code to start from. The first exercise is to produce this code.

You will be implementing an algorithm which generates a picture of the Mandelbrot set (figure 1). There is a template code from which you can start which includes routines for initialising and outputting the image, however, you will have to implement the code to compute the Mandelbrot set yourself.

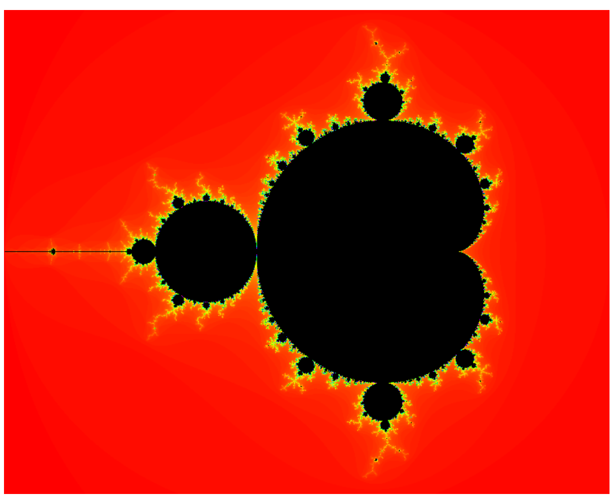

Figure 1: The Mandelbrot set

The Mandelbrot set

Consider the recursion

$$z_{n+1} = z_n^2 + c \tag{1}$$

with $z_0 = 0$. The Mandelbrot set, M, is the set of complex numbers $M \subset \mathbb{C}$ such that $\lim_{n\to\infty} |z_n|$ is bounded. For example, if $c = 0$ then $\lim_{n\to\infty} |z_n| = 0$ and so $c = 0$ is in the Mandelbrot set. It turns out that we do not need to check if $|z_n|$ escapes to infinity, $c \in M$ if and only if $\lim_{n\to\infty} |z_n| \leq 2$.

To compute the Mandelbrot set, rather than explicitly evaluating the infinite limits, we use a heuristic to decide if a point is in the set. We limit the number of iterations to some finite value n_{max}. To check if a complex number c is in the set we use the following algorithm.

1. Let $z_0 = 0$
2. If $|z_0| > 2$, c is outside the set, return 0
3. Compute $z_{n+1} = z_n^2 + c$
   (a) If $|z_{n+1}| > 2$, c is outside the set, return $n + 1$
   (b) If $n + 1 = n_{\mathrm{max}}$, c is in the set, return n_{max}
   (c) Else, set $n \leftarrow n + 1$ and go to 3.

To generate the pictures of the Mandelbrot set, we choose a region of the complex plane and divide it up into discrete values. For each such value, apply the Mandelbrot recursion. We colour each pixel by the number of iterations it took before it was decided as bounded or unbounded. For example, if $|z_{15}| > 2$ then the point is coloured with value 15. The provided utility code takes care of converting this single value into a colour.

A brief aside on complex numbers

Fortran has built in support for complex numbers which we can use. C didn't until C99, so in the C version of the practical, you need to do all the complex arithmetic yourself. Here's a quick, non-exhaustive refresher. Let $z = x + \mathrm{i}y$, where $\mathrm{i} = \sqrt{-1}$. Then

$$|z| = \sqrt{x^2 + y^2} \tag{2}$$

and

$$z^2 = x^2 - y^2 + \mathrm{i}(2xy) \tag{3}$$

Although we use Fortran's built-in complex number support, you may still need to do computations with the real and imaginary parts separately. You may need:

```fortran
rpart = real(c)                   ! The real part of c
ipart = aimag(c)                  ! The imaginary part of c
z = cmplx(rpart, ipart)           ! A new complex number
```

Setup

You will be carrying out these exercises on `ness`, so log in and download the tar file for the exercise (available on WebCT) and unpack it. Change into the `mandelbrot` directory that has just been created. There are two subdirectories `C` and `F` corresponding to the C and Fortran versions of the exercise respectively. These contain a `utils` directory which you should not have to touch, and a `exercise1/template` directory with template code. To build the template `C` code, do

```
cd C/exercise1/template
make
```

This builds the utility code and the template code to produce a `mandelbrot` executable. The Fortran template code can be built in the same way, but in the directory `F/exercise1/template` instead.

If you run the `mandelbrot` executable, it will produce an output image. Until you have implemented the code to calculate the Mandelbrot set, this image will just be black. To view the image, type `display output.ppm`.

The executable accepts a number of command line arguments, which you can see by passing the `-h` flag.

```
$ ./mandelbrot -h
Usage:
mandelbrot [-SixXyYh]
   -S NPIXEL    Set number of pixels in X dimension of image
                Y dimension is scaled to ensure pixels are square
   -i ITS       Set max number of iterations for a point to be inside
   -x XMIN      Set xmin coordinate
   -X XMAX      Set xmax coordinate
   -y YMIN      Set ymin coordinate
   -Y YMAX      Set ymax coordinate
   -h           Show this help
```

The default values show the whole Mandelbrot set.