FRACTAL GENERATOR

June 3, 2022

Contents

1 Introduction:

This implementation of a fractal generation algorithm, while not unique, does represent an attempt to keep it simple and efficient. The focus is on computation using complex data types and parallelization, along with an embedded macro language, to provide a simple but flexible capability to produce fractal images. Output is provided in a simple, standard portable bitmap format that can be piped to appropriate translation utilities to produce images files, avoiding the need to implement support for any specific output format.

The algorithm used in this implementation is focused on the Mandelbrot set and related Julia sets arising from the iterative solution of the simple equation $z = z^2 + c$ where z and c are complex numbers ($c = [a + bi]$ where $i^2 = -1$). The Mandelbrot set is defined as the set of complex numbers c for which the function $z = z^2 + c$ does not diverge to infinity with an initial value $z_0 = 0$. The rate at which values of z diverge can be visualized by assigning a color, based on the iteration count n that occurs when the absolute value of z exceeds the escape value 2 (i.e., $|z_n| > 2$), to an image pixel placed at a coordinate derived from the real and imaginary parts of c. An iteration count that exceeds a threshold, without $|z|$ exceeding the escape value, implies that the assigned value of c is a "likely" candidate for inclusion in the Mandelbrot set. The resulting "fractal" visualization is in fact a graphical depiction of the rate of divergence of z for all values of c in the defined view, that are by definition, not part of the Mandelbrot set.

In contrast, a Julia set is defined as the set of initial values of z_0 for which the function $z = z^2 + c$ does not diverge to infinity for a constant static value of c. For initial values of z that diverge, the resulting iteration count is visualized using a image color value assigned to the pixel at the coordinate derived from the real and imaginary parts of z_0.

If the specified number of threads defined is greater than 1 then image computation will be allocated to each thread in column sequence for the most equitable and efficient division of computational resources. Hopefully, it is obvious that the number of threads specified should be equal to or less than the available hardware cores or hyper threads.

2 Usage:

From command shell type:

-> fractal [rexx script file]

2.1 Embedded REXX Macro Language

This version of Fractal registers itself as an embedded Rexx interpreter which will process commands not directly understood by the Rexx macro processor. This effectively adds all the capabilities of the Rexx macro language to Fractal program macros.

3 Command Reference:

3.1 reset(XRES, YRES, MAXITER, NTHREAD)

This command resets the image resolution, maximum number of iterations, and number of allocated threads. The resolution (XRES, YRES) defines the screen resolution of the generated image.

An example of this command is:

```
reset(1600, 1200, 4000, 4)
```

3.2 palette(DMAX, SHIFT, COLORS...)

This command sets the palette of color spectrum to the defined set of colors, with the gradient between color values defined by the DMAX parameter (number of color gradient values between defined color values). The SHIFT value defines the starting color value with zero (0) specifying the first color value. If the iteration value exceeds the maximum color value (number of colors multiplied by DMAX) then the color selected will cycle through the color spectrum starting with the first color value.

An example of this command is:

```
palette(64, 2, 070f19, 7B98F1, 010015, 2D62F3, 22010b, 4FDCE5)
```

3.3 view(XC, YC, SIZE)

This command defines the fractal view center coordinates and size.

An example of this command is:

```
view(0.0, 0.0, 2.6)
```

3.4 mandel(CTYPE)

An example of this command is:

```
mandel(0)
```

3.5 julia(CX, CY, CTYPE)

An example of this command is:

```
julia(0.355534, -0.337292, 1)
```

3.6 spectrum

An example of this command is:

```
spectrum
```

3.7 save(FILENAME)

This command will save a generated image to a file.

The FILENAME argument specifies a file to save the resulting image to. The file output will be in portable bit map (PBM) format. If the file name begins with a pipe ('|') then the output will be provided as a standard input stream to the specified process.

An example of this command is:

```
save(|pnmtojpeg -quality 90 > output.jpg)
```

4 Example

```
* macro for Fractal */
reset 3200 2400 4000 4
view 0.0 0.0 2.6
julia 0.355534 '-0.337292' 1
'save(|pnmtojpeg -quality 90 > julia.jpg)'
exit
```