Scientific Computing

Assignment 10 – last assignment

Due: Friday, Nov. 30, 2018 at 17:00

Refer to week 1 lecture notes for instructions for submitting your assignment using the `hand-in` web-form. In your submission include

- all required source code and
- any other files asked for in the question.

Put the files in a directory called `a10`. Do not include object or executable files.

For full marks document your work using *meaningful* comments.

Make sure to include

- your name,
- your student number, and
- the assignment number.

in each file you submit.

Also add comments where necessary to clearly label each solution. **Hand in only the source files and makefile, not the objects and executables.**

---

1 Problems

There is only one problem this week, and it is to make a small application that can zoom in on the Mandelbrot set. The Mandelbrot set is a nice beginning programming exercize, because it allows us to calculate something simple that gives a surprisingly nice picture.

To display the Mandelbrot set, we colour a pixel based on whether the magnitude of the sequence of numbers z_n generated by the Mandelbrot equation ever go above 2. The color of the pixel is then set based on the number of iterations it took to get above 2. If the equation does not go above 2 after say 20 iterations, then set the color of the pixel to black. The sequence of numbers is generated for some pixel at (x, y) by setting $z_0 = x + iy$ for the zeroth entry in the sequence. The $n + 1$ th entry in the sequence is then given by:

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

Or, if you prefer in real numbers the values of (x_{n+1}, y_{n+1}) are:

$$x_{n+1} = x_n^2 - y_n^2, \text{ and} \tag{2}$$

$$y_{n+1} = 2x_n y_n. \tag{3}$$

You can find examples of the completed program in Fig.1 just after startup, and in Fig.2 after having set the zoom and center location and pressing the next button.

Instructions for making the application are as follows.

1. Copy `lineDrawing.cpp` from the week11 examples folder into a file `drawMandlebrot.cpp` along with the associated graphics library files as a starting point.
2. Update the code to rename `Lines_Window` to `Mandelbrot_window`, and update it to have three `In_box` objects: one to set a zoom-in scaling factor, one to set the x-coordinate to zoom in on, and finally one to set a y-coordinate to zoom in on.
3. Add a new class `Pixel` that is in the `Graph_lib` namespace that inherits from `Shape` and represents a single pixel. You will need to implement the `draw_lines` method of the class, that uses the `void fl_point( int x, int y)` function from the FLTK library to draw a single pixel at location (x, y) on the screen.

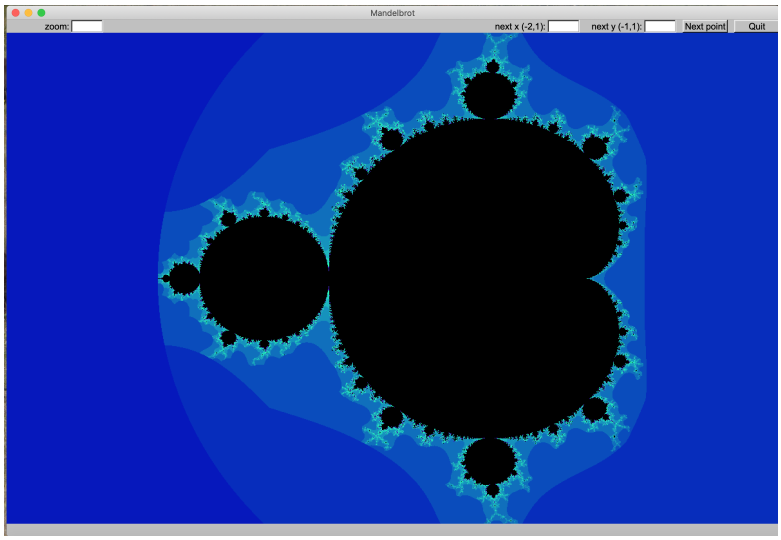

Figure 1: drawMandelbrot.exe application window at startup

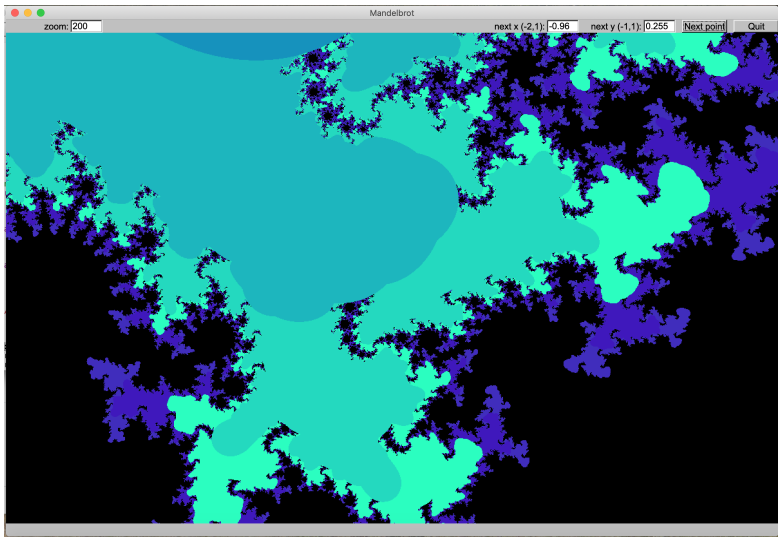

Figure 2: drawMandelbrot.exe applicatoin window after entering zoom factor, x, y values, and pressing the next button.

4. Update the code that gets executed when the `next()` function is called when the user pushes the next button. This is where you will put all of the code to update the colours of a `Vector_ref<Pixel>` that will hold all of the pixels. Remember to only populate the Vector_ref once, and to only attach the `Pixel`s to the window once. After the first time the function is called, you just need to update the colour of each pixel. Note that the Mandelbrot set is most interesting over the range $-2 < x < 1$, and $-1 < y < 1$, so scale the locations on the screen to start with that range for a zoom factor of 1.

5. Prepare a makefile to hand in with the assignment that compiles the fltk wrapper code, and your code. It should also link the code to make an executable.