COMP140 WORKSHEET B: MANDELBROT SET

Version 1.0
BSc Computing for Games
COMPXXX

Brian McDonald & Ed Powley

In this worksheet, you will use the SDL2 library (`https://www.libsdl.org/index.php`) to write a program to generate and display the *Mandelbrot set* fractal; see Figure 1. This fractal colours each pixel of the image according to an iterated mathematical formula, as described below. The GitHub repository contains a project named `Mandelbrot` for you to build upon. This contains code to create and display a blank image; you will implement the calculations to generate the Mandelbrot set fractal.

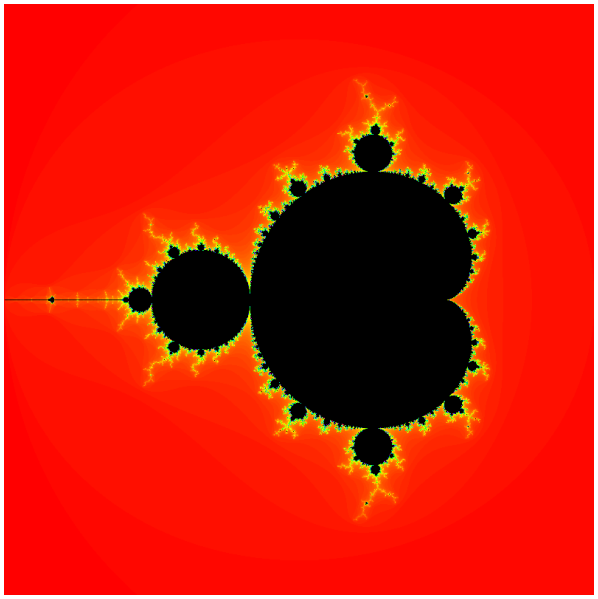

Figure 1: The Mandelbrot set fractal.

1

To generate an interesting fractal, the on-screen x and y coordinates must first be rescaled. In the skeleton project the pixel coordinates range from 0 to 800, whereas the Mandelbrot set fractal is most interesting in the region $-2 \leq x \leq 1$ and $-1.5 \leq y \leq 1.5$.

Let p_x be the x coordinate of the pixel. This can be remapped into the range x_{min} to x_{max} using the following formula:

$$x_0 = \frac{p_x}{\text{image.width}} \times (x_{\text{max}} - x_{\text{min}}) + x_{\text{min}}$$

The y coordinate can be remapped using a similar formula.

Implement the above calculations for the x and y coordinates, at the indicated parts of `Mandelbrot.cpp`.

2

The Mandelbrot set is based on the following sequence of numbers. Let x_0 and y_0 be the coordinates of a point in the image. Then the sequence $x_1, y_1, x_2, y_2, x_3, y_3, \ldots$ is defined[1] recursively for $i = 0, 1, 2, 3, \ldots$ by:

$$x_{i+1} = (x_i)^2 - (y_i)^2 + x_0$$
$$y_{i+1} = (2 \times x_i \times y_i) + y_0$$

The points are coloured according to the *smallest* value of i for which $(x_i)^2 + (y_i)^2 \geq 4$. If such a value of i is not found after a large number of iterations (for example $i = 200$), the pixel is coloured black.

Implement an algorithm which performs the above computation, determining the smallest value of i for which $(x_i)^2 + (y_i)^2 \geq 4$ and selecting the appropriate pixel colour. Implement the algorithm in `Mandelbrot.cpp` so that the program generates the Mandelbrot set fractal (Figure 1) when it is run.

3 Stretch goal

Mathematicians have discovered many other fractals besides the Mandelbrot set.

Research a different type of fractal, using online resources and/or academic literature. Briefly summarise your findings (with URLs and/or citations) in your `readme.md` file.

Implement a generator for the fractal you have researched. Allow the user to choose (by command line entry) which fractal they want to generate when the program starts up: the Mandelbrot set or the new fractal. You may wish to restructure the skeleton program to accommodate the new fractal generation algorithm.

Submission instructions

Begin by **forking** the GitHub repository at the following URL:

`https://github.com/Falmouth-Games-Academy/comp140-worksheetA`

You should complete a pull request before the hand-in on Friday by 5pm on Week 1. Feedback will be given in the pull request and in class.

Marking criteria

Remember that **it is better to submit incomplete work than to submit nothing at all**.

To demonstrate **basic competency**, complete the following:

- **Timely Submission:** Obtain the marks for timely submission, you must submit (as a GitHub pull request). As with other worksheets, you may resubmit after these deadlines in order to collect extra correctness or quality marks. This is awarded as long as you submit *something* for each part by the deadline, even if your submission has bugs or other issues.

[1] If you are familiar with complex numbers, you may notice that this is equivalent to $z_{j+1} = z_j^2 + z_0$, where $z_j = x_j + y_j i$.

To demonstrate **basic proficiency**, complete the following:

- **Achieve basic competency** affect the overall functioning of your programs
- Appropriate use of GitHub, with descriptive commit messages
- Comments are used where appropriate, and are well written.

To demonstrate **novice competency**, complete the following:

- Achieve **basic proficiency** affect the overall functioning of your programs
- Your code is well formatted. Variable and function names are clear and descriptive.

To demonstrate **novice proficiency**, complete the following:

- Achieve **novice competency**
- **Design** an improved word choosing algorithm

To demonstrate **professional competency**, complete the following:

- Achieve **novice proficiency**
- **Implement** an improved word choosing algorithm