CS293 Project: Mandelbrot Zoom

Name: Govind Saju
Roll: 200070020

Demo Video:
https://drive.google.com/file/d/17lueXVocgGQyVuKkwt_Eur13XWqpUxTD/view?usp=sharing

Brief Description
In this project, I have implemented the Mandelbrot Zoom in C++ using the **SFML** graphics library. The project has been done on Ubuntu 20.04. The program renders the Mandelbrot set in a coloured manner, and gives the user complete control over the zooming process. The features provided include

- User can zoom in or out with a particular point as focus
- Translate horizontally or vertically without zooming
- Control the rate of zooming
- Choose the algorithm to be used for colouring
- Cyclically shift the colour palette
- Save the current screen as an image

The code has been sufficiently documented with comments and is compatible with Doxygen for generating the documentation in HTML or LaTeX formats.

Setup

1. Installing SFML
   SFML can be installed by running the following command:

   ```
   sudo apt-get install libsfml-dev
   ```

2. Compiling the source code
   A makefile has been included in the submission which takes care of compiling. In the directory of the source code, run the command

   ```
   make mandelbrot
   ```

   This should create an executable file called mandelbrot

3. Running the executable
   In the same directory, run

   ```
   ./mandelbrot
   ```

Design of the application

- The main function is found in the file “main.cpp”. It creates an object of the Renderer class and runs the application frame by frame.

- The Renderer class is in the renderer.h and renderer.cpp files. This is a wrapper class around the entire application. It contains objects of Mandelbrot and EventManager classes. In each frame, the renderer class logs user events using EventManager, updates them in the Mandelbrot object, and finally renders the updated data on the screen.

- The EventManager class is responsible for logging all user events (including mouse clicks and keyboard controls) and changing the corresponding variables. The renderer class reads values from this object every frame.

- The Details class is used to store the text to be shown to the user. It stores values of each parameter and is updated by the renderer class every frame. It is also responsible for printing this data onto the screen in the ‘Instructions and Parameters’ view.

- The Mandelbrot class is the central class that performs the operation of rendering the Mandelbrot set.
    - It is given bounds of the current frame (Complex values mapped to bottom left and top right of the screen).
    - It uses the CoordinateMapper class to find the complex value to which each pixel is mapped.
    - It performs the normalized escape time algorithm for each pixel and uses the palette provided by the ColorMapper class to decide which colour is to be mapped to the current pixel.
    - The maximum number of iterations performed per pixel is defined as a function of the logarithm of the area covered in the current frame, to ensure that the image doesn’t saturate when the total zoom increases.

- The CoordinateMapper class maps each pixel to its corresponding complex number based on the bounds given for the current frame.

- The ColorMapper class is the class responsible for setting up a palette. A palette refers to a continuous cyclic wheel of colours, which programmatically is a circular array. It contains 6 different modes of setting up the palette based on the users choice. Modes 1 and 2 are deterministic, while modes 3-6 are randomized. Each mode is described below:

    - Mode 1: In this mode, we generate the palette by representing the colours in HSV format. In the palette, all colours have V = 1 and S = 1, while H is uniformly distributed between 0 degrees and 360 degrees.

    - Mode 2: This mode is similar to mode 1. It generates a palette uniformly using HSV values. It then takes the photographic negative of each colour in the palette. The S and V values are smaller here which leads to reduced brightness.

    - Mode 3: This mode is a randomised algorithm. It generates H, S, V randomly between (0,360), (0,1) and (0,1) respectively. It then performs a heap sort by sorting the colour array in ascending order by first sorting the H values, followed by the S values and finally the V values in decreasing order of priority.

    - Mode 4: This mode is almost the same algorithmically as Mode 3. The only difference is that the sorting is in descending order here.

    - Mode 5: In this mode, colours are generated randomly using the RGB format. R, G and B are randomly selected from (0,255). We then compute the luminosity of each colour and sort the array in ascending order using heap sort based on the luminosity of the colour.

    - Mode 6: This mode is almost the same algorithmically as Mode 5. The only difference is that we sort in descending order based on luminosity.

- Complex class: A utility class defined for doing complex number arithmetic easily.

Instructions to use the application

- There are two views supported:
    - Mandelbrot: Renders the Mandelbrot set in the current region of the display
    - Instructions and Parameters: This shows the instructions to use the application and also the current parameters related to the zoom

  To switch between these 2 views, use the **D** or **Esc** Keys.

- Clicking on any point inside the window (in the mandelbrot view) with the mouse leads to the application zooming in with that point as the focus. The rate of zoom is called the zoom-factor and has a default value of 1.25. Whenever a point is clicked on, we zoom in by a factor of the zoom factor.

- The zoom factor can be increased or decreased by units of 0.03. Pressing the **=** or **+** (Equals and Plus Key) key on the keyboard increases the zoom-factor, and pressing the **-** or **_** (Minus and Underscore Key) key on the keyboard reduces the zoom-factor.

- Zoom-factor greater than 1 means we are zooming in, and Zoom-factor between 0 and 1 means we are zooming out. Negative values will also lead to an inversion of the screen at every frame and lead to oscillating figures.

- The **arrow keys** can be used for translating the frame being seen. Up arrow renders a region above the current region and similarly translates in other directions for each arrow key.

- To toggle automatic zoom into a hardcoded point (to zoom without a mouse), press key **A**.

- To switch between the 6 modes of colouring, use the **number keys 1,2,3,4,5,6**. Key number i corresponds to the i^{th} mode of colouring

- Pressing the key **I** cyclically shifts the current palette being used for colouring

- Clicking on the key **S** saves the current image in png format. The filename is autogenerated by processing the system date and time and is saved in the same directory.

Code Documentation

The code has been extensively documented and is compatible with Doxygen. To generate the documentation in LaTeX or HTML formats, the following steps can be followed:

1. Installing doxygen

   ```
   sudo apt-get install doxygen
   ```

2. Generating doxyfile
   In the same directory as the code, run

   ```
   doxygen -g
   ```

3. Generating the documentation

   ```
   doxygen Doxyfile
   ```

4. This will generate the documentation in two subfolders, namely HTML and LaTeX.

   To open the HTML documentation, go into the folder and open the file “index.html” in a browser. All the documentation would be accessible from there.

   To generate PDF from LaTeX, go into the LaTeX folder. Use a LaTeX compiler like pdflatex and run the command “pdflatex refman.tex”. This should generate a pdf file refman.pdf which contains the documentation.