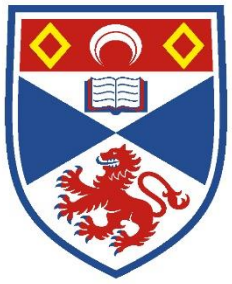
CS5001 Object Oriented Modelling Design & Programming

Practical 4 – Mandelbrot Set Explorer



University of
St Andrews

Contents

Mandelbrot Set Explorer

1. Finishing Part

In this assignment, the parts I have done as follows:

- All tasks of Basic Requirements
- First two tasks of Enhancements(i.e. Different Color Mappings, Save and Load)

2. Introduction

This practical utilize GUI in Java to offer a user an interface which make them able to view and explore the Mandelbrot set graphically. In this report, how I design my program is described in detail. Firstly, design part will be introduced, including the choice of design pattern, how I classified classes and methods. Secondly, I demonstrate the result after testing and give an example. Finally, I summarizes this practical and the functions of this program I implemented.

3. Design

3.1 Design pattern (MD or MVC)

The Model-Delegate design pattern is adopted in my program. Model is used to calculate Mandelbrot set and set parameter and realize undo, redo and reset functions. Delegate is use to implement user interface, including showing graphics, allowing users to zoom in, input value, click buttons. Basically, Model is to operate background

data. The function of using delegate is to show information on UI and give users a better visual experience. To some extent, Delegate serves Model.

Under MD design pattern, I created 3 packages (guiDelegate, main and model) which contains 6 classes in total.

- guiDelegate pakage includes Delegate, DrawMandelbrot
- main package class only has Main class.
- Model package includes MandelbrotCalculator, Model and Record classes.

More detail is introduced in next sections.

3.2 Delegate class in guiDelegate package

The purpose of Delegate class is to implement UI. Therefore, there contains setupToolbar method, setupMouse method and setupComponents method.

- setupToolbar method: 7 buttons, one label and one text field are add into tool bar to realize functions including changing the maxIteration, undo, redo, reset, save, load and random color. Those button is listened. When a button is triggered, corresponding manipulation is executed, Mandelbrot set is updated and the Mandelbrot image is updated.
- setupMouse method: This method record the initial position when a user clicks in the UI and the final position when a user releases mouse in the UI. According to current coordinates value, new coordinates value can be calculated by calling calPosition method in model class. Then the current model object is updated. The new graphic is updated. In addition, the purpose of this method is to zoom in, and the updated model should be add in to the first stack which will be used when call undo method.

- setupComponents method: This method invoke setupToolbar method and setupMouse method because tool bar and mouse are components. It is more reasonable and tidy when these two method are invoked here. Besides, Jpanel is added in this frame. Plus, JFrame sets up here. Therefore, in fact this method is basically build an interface that shows to users.

3.3 DrawMandelbrot class in guiDelegate package

The main purpose of DrawMandelbrot class is to give color to every pixel. Color in Java is expressed by integer. There are one constructor and two method in this class, including paint and setColorsRandom.

- paint method: When the iteration of every element in Mandelbrot set equal to the maximum iteration, the color is black. Or the color is random color. By default, random color includes red, green and blue.
- setColorsRandom method: When the user click “random color” button, this button is triggered and setColorsRandom is called. Hence color will be changed. Then the random color will be updated.

3.4 Main class in main package

Main class is dependent outside this MD design patter. In the main method, it create a model object and transmit this object to delegate class to implement operations of these two part.

3.5 MandelbrotCalculator class in model package

This class is provided in advance. This class is mainly used to calculate iterations which is putted into array eventually.

3.6 Record class in model package

Record class includes constructors, addModel, undo and redo, save and load methods. There are two stacks in this class. To realize undo and redo functions. The points here are:

- These two stacks is to store model because every model object because every model object includes all information needed in calculate Mandelbrot set.
- When the initial Mandelbrot set is calculated, maxIterations is changed by users and the Mandelbrot is zoomed in, the addModel method is called, namely the current model object is pushed into the first stack
- When the undo method is invoked, the head element of the first stack is move to another stack by using pop() because it is the current model object. When undo method is invoked, the last model is needed. After this operation, the head element of the stack is returned to delegate class where the undo method is invoked. By using the information the element offers, the minReal, maxReal, minImaginary, maxImaginary and maxIterations of the old model object can be settled, and then new Mandelbrot set is calculated.
- When redo method is invoked, the head element is took out to add push into the first stack by using peek(). Meanwhile, this element is also used to update the old model object in Delegate class which is similar to the processes is mentioned just now.

To realize save and load functions. The points here are:

- In save method, the model in the first stack should be peeked. Those information or attributes from this model object is saved in “save_record.txt” through using BufferedWriter.

- In load method, the information stored in “save_record.txt” is read by BuffereReader. Then attributes of model object are settled by information from the txt. File. Finally when the model call calMandelbrot in Delegate class, Mandelbrot set is calculated and the Mandelbrot image is updated. Note that the model calling calMandelbrot can only be called in Delegate class because the Delegate class implements Observer.

3.7 The mechanism of running the system

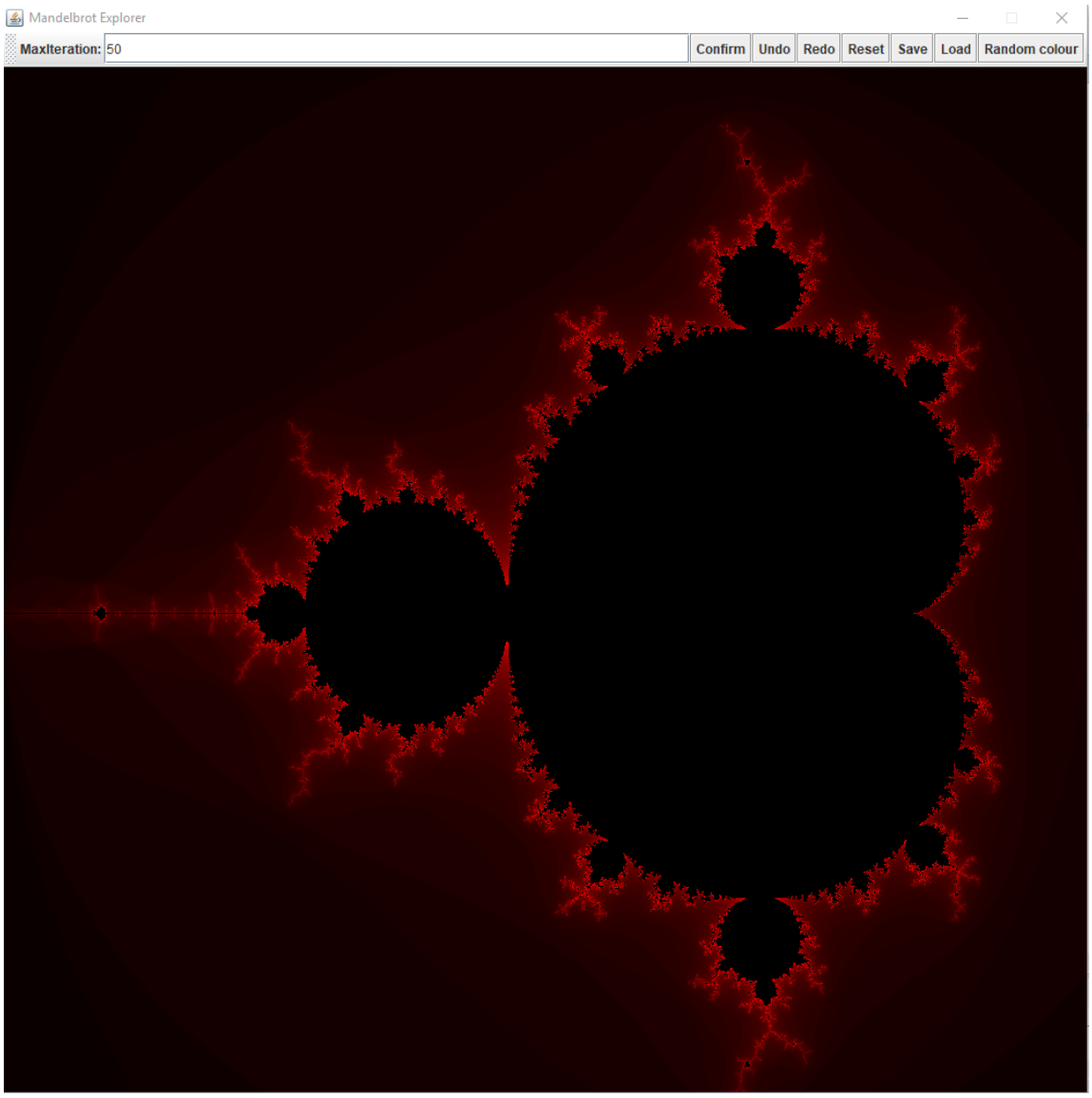
Because of using MD design pattern, the Delegate and model are separated. They need to be connected by using of Observer. Every time when model changes, it should notifies its observers. And delegate will update the image.

To implement this. In Delegate should implements Observer and override update method. When the observed object (i.e. model) changes, Mandelbrot is drawn again. Model is also an observable object and that is why it can be observed by Observers.

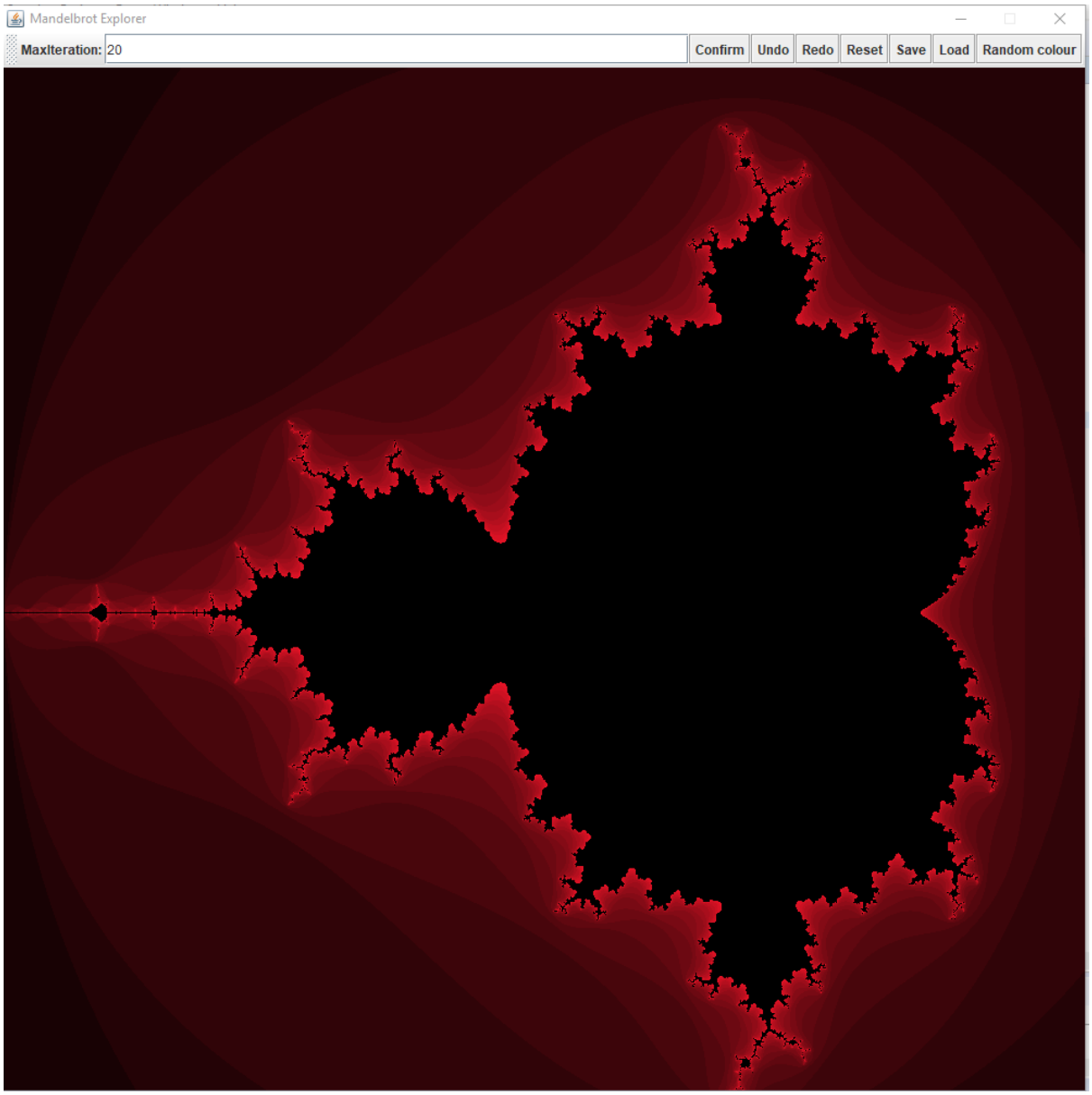
4. Examples

Only some of functions are displayed as follows:

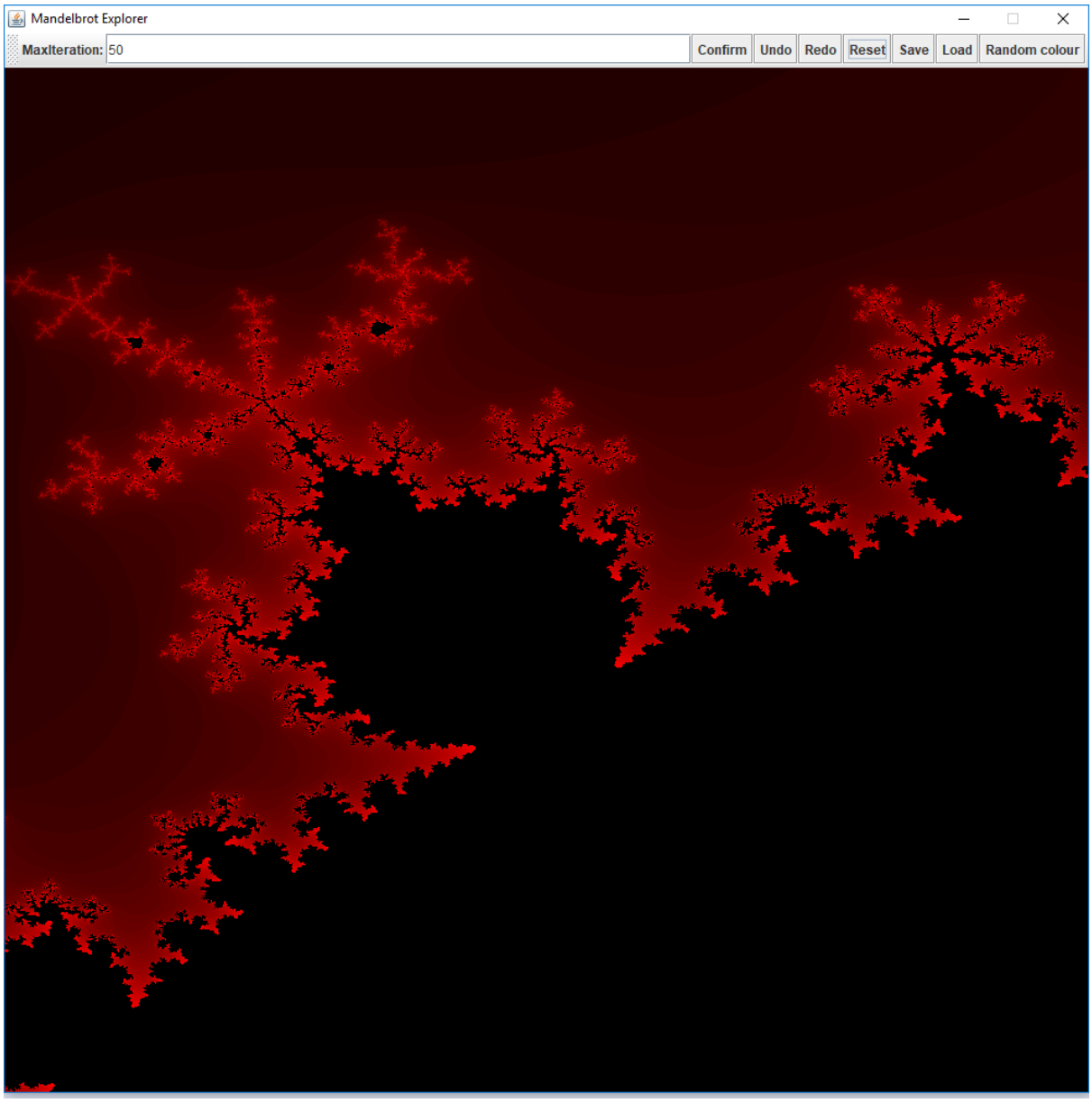
4.1 Display default Mandelbrot image



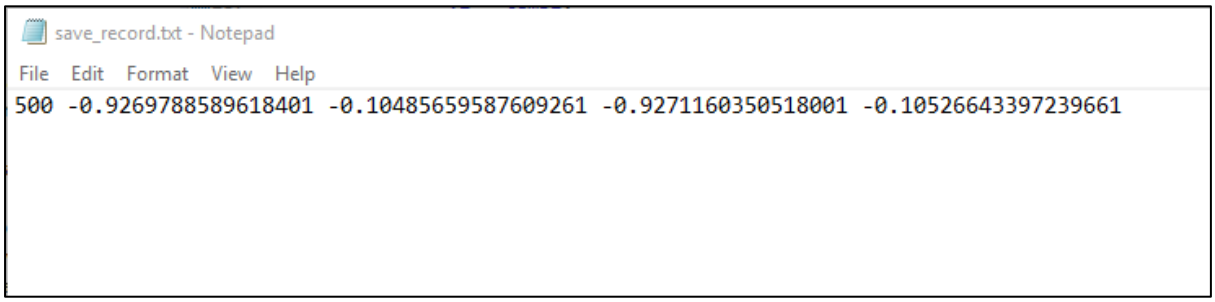
4.2 Change MaxIteration



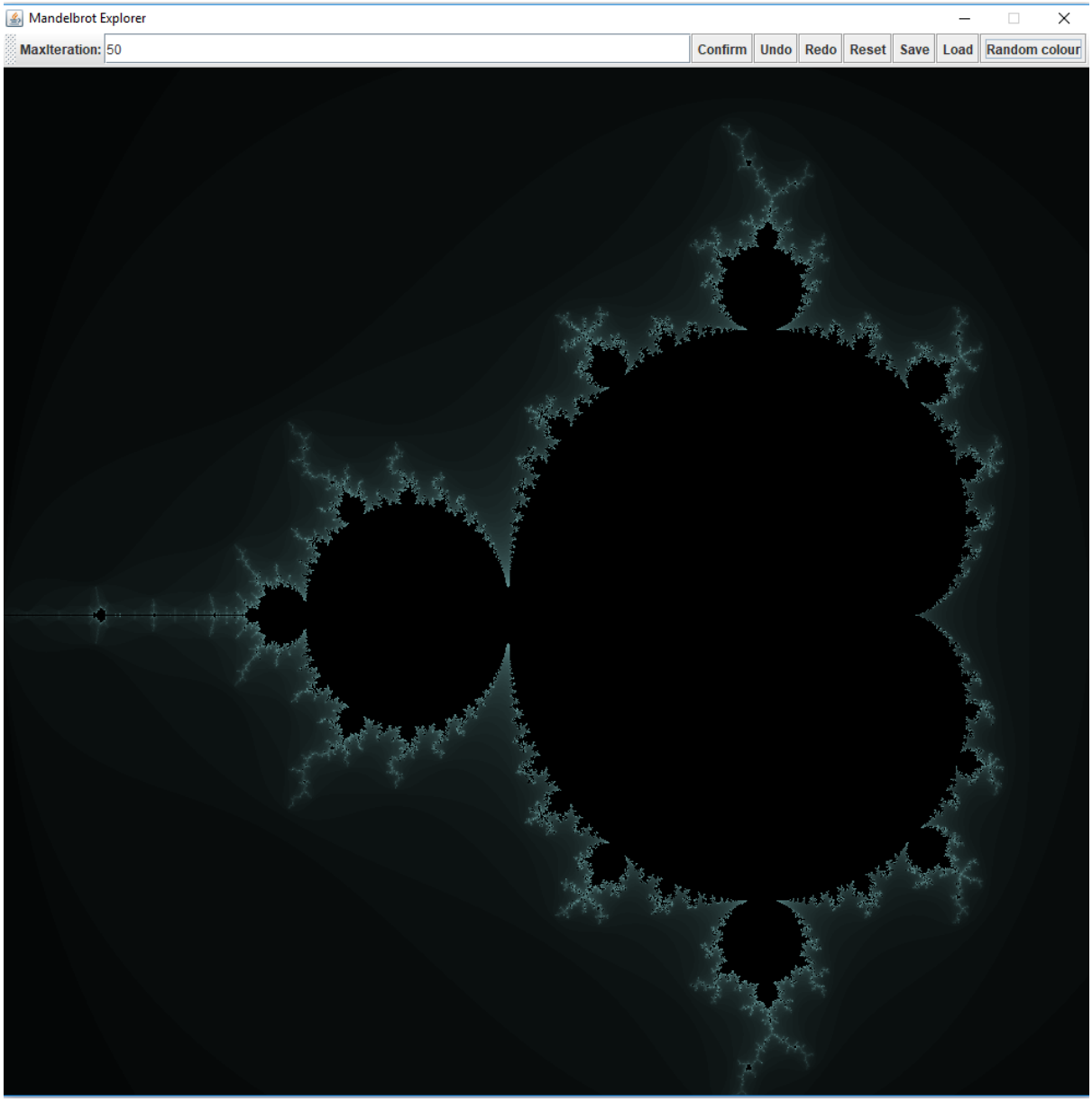
4.3 Zoom In



4.4 Save



4.5 Random color



5. Conclusion

This practical is to practice making CUI and using MD or MVC design pattern. A beautiful and concise user interface can provide convenience and bring good experiences to users. The rationale of design pattern helps to add other components or functions and improve security to the system.

In this practical, the program is implemented by applying MD design pattern. Some functions have been implemented as follows:

- Display Mandelbrot set, and the graphic is shown combing with different colors divided by the boundary.
- Users can zoom in the graphic by select a rectangular area.
- Users can change the maximum iterations, and the Mandelbrot image can change corresponding to the new value.
- It offers undo, redo and reset button to make users be able to undo and redo operations.
- Users can save information like position of graphics by clicking save button and according to those information stored by computers, the graphic can be reload to let users explore Mandelbrot sets from the onward position.