University of
St Andrews

CS5001: Object Oriented Modelling Design & Programming

Practical 4: Mandelbrot Set Explorer

Student ID: 180019653

Name: ZHAOHUI XU

The program has achieved all basic functional requirements. The possible enhancements which are described in the "Enhancement" part are implemented successfully as well.

How to run program:

Extract the file firstly and use terminal to the path such as "CS5001-p4-graphics/src",

And input the text "javac *.java" in terminal,

Then, input the text "cd CS5001-p4-graphics/src/main" in terminal,

Finally, input the text "java Main" in terminal.

Functionality:

- **Basic display** – display the Mandelbrot set in black and white for a sensible initial parameter setting.
- **Zoom** - permit user to select a square (or rectangular area) on the image by dragging the mouse over the area to zoom, the square start coordinates collected by using the mouse pressed function and the square end coordinate collected by using the mouse released function. User could also use the mouse wheel to change the zoom value.(The function cannot be used when the user has selected "Pan Permission" button")
- **Pan** – permit user to select a distance and direction on the image by dragging the mouse over the area to change, the line start coordinates collected by using the mouse pressed function and the line end coordinate collected by using the mouse released function. (The function can be used when the user has selected "Pan Permission" button)
- **Estimate of zoom magnification** - permit user to choose to see an estimate of the zoom magnification on the top-left of the screen but below the toolbar.(The function can be used when the user has selected "Show Magnification" button)
- **Change maximum iteration value** - Permit user to alter the value used for max iterations of the Mandelbrot image by clicking the text field on the tool bar to edit the max iteration value and click "Change" button to upload the data in the system.
- **Und/Redo/Reset** – permit the user to undo/redo/reset the last zoom or pan operation by clicking "Undo"/"Redo"/"Reset" buttons on the tool bar
- **Change colour** – permit user to change different colour mappings by clicking "Change Colour" button on the tool bar. (In the program, I have set 4 extra colours that allows user to switch)
- **Save/Load** – parameter settings (and potentially the computed image) to be saved into the ser file and loaded to/from local ser file. To use this function, user should click the "File" button on the menu bar and then select "Save Data" / "Load Local Data".
- **Save Image** – permit the user to save the image in different situations such as different zoom value or different colour styles. To use this function, user should click the "File" button on the menu bar and then select "Save Image".

Design Pattern:

The program use the Model-Delegate(MD) architecture. In order to make the program concise and complete and the user interface(delegate file) is independent form the model.

Model:

The most important class in the model package is the MandelbrotModel class. The class creates an instance of the MandelbrotCalculator class and stores the state index value in a specified array list

used to record user operations for undo/redo functions. If the reset functions is activated, the class will remove the current data and back to the original setting. Additionally, the class contains the methods which are used to save the current Mandelbrot parameter settings as file(ser format) and images (jpeg format) respectively and load the stored Mandelbrot parameter settings file(ser format) from specified paths. When the delegate file require the image details, the MandelbrotModel class will use updateImage() method to handle the image data in specified state and display the image for the user.

The calculateColour() method supports the services that can switch the Mandelbrot images in different colours and handle the brightness according to the Mandelbrot set. Once a user choose to reset the data, removedData() method will be called and clear all state index value in buffer area and thus user cannot use the undo/redo functions back to the last/next state after resetting.

There are four versions to set up the new state and add it up to the data list. The first version do not require any arguments and used to change the colour setting for the image. Therefore, the colour setting sequence is built in the parameter class and user could use undo/redo functions for colour changing. Second version is to handle the action that user make a square in current Mandelbrot Set image and generate new Mandelbrot Set image according to the data passed from the delegate file. The third version is using the max iteration argument passed from the delegate file and thus update the max iteration number of Mandelbrot Set and display the new state/result on screen. The final version is similar to the second version but its arguments includes the coordinates from the start point of the line and the end point of the line and override the old state on screen.

The canvas class is responsible to store initial parameter values fromMandelbrotCalculator class as initial state data which will not be cleared. The class implements Serializable file so it also can handle the new parameter values and stored the new data in the data list.

Delegate:

The delegate file is stored in the “guiDelegate” package in the “src” folder of the upload file. Only one delegate class “GuiDelegate” is produced but it is powerful and central.

This class contains instances of MandelbrotModel class and MandelbrotCalculator class. The main duty of this class is to set up the user interface, build up the addMouseListener() method which can monitor the user input and transport the data transport the relevant parameters to MandelbrotModel class as well as Canvas class.

Menu method

The menu method make the change according to the user input on the menu bar and thus all menu items are need to add ActionListeners and then call the respective methods in the model. The figure below gives the code of “Load Local Data” menu item.

```
// set the menu item which allows user to load the local SER file
load = new JMenuItem("Load Local Data");
load.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        // A file fileChooser is implemented to choose the data to load from local file.
        JFileChooser fileChooser = new JFileChooser();
        FileNameExtensionFilter filter = new FileNameExtensionFilter("SER files", "ser");
        fileChooser.setFileFilter(filter);
        int returnVal = fileChooser.showOpenDialog(null);
        if (returnVal == JFileChooser.APPROVE_OPTION) {
            try {
                // invoke the loading method to update the model.
                mandelbrotModel.loadParamFile(fileChooser.getSelectedFile());
                frame.repaint();
            } catch (FileNotFoundException e1) {
                e1.getMessage();
            } catch (IOException e1) {
                e1.getMessage();
            } catch (ClassNotFoundException e1) {
                e1.getMessage();
            }
        }
    }
});
/**
```

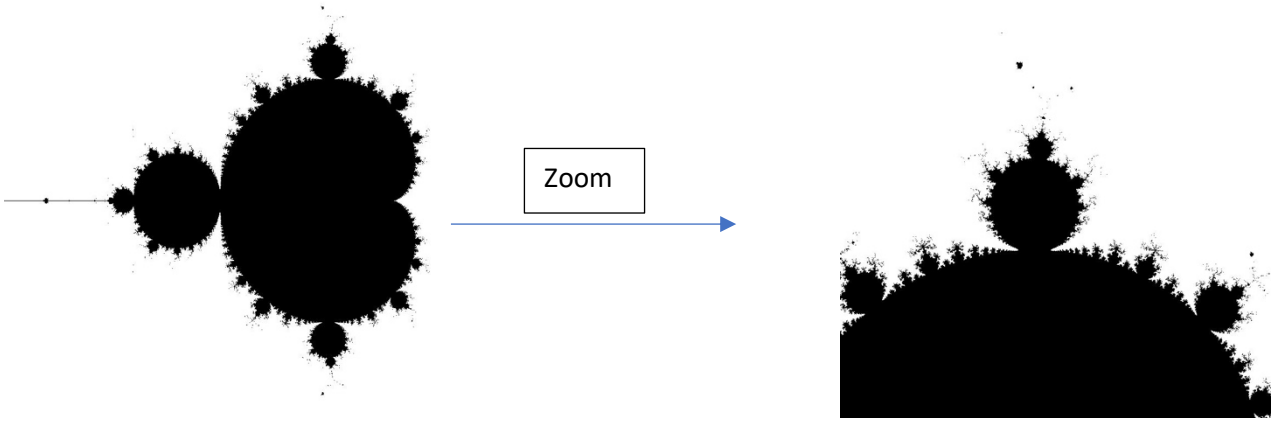
Toolbar method

The toolbar method is similar to the menu method and requires the user input. To achieve the target, all the toolbar button adds ActionsListeners and trigger respective methods.

Zoom method and screen display

It provides operations to permit a rectangular area for zooming and permit a direction and magnitude to change to be selected for panning. The algorithm was stored in the MandelbrotModel class and the figure below will display the details of the respective method and how the effect display on the screen.

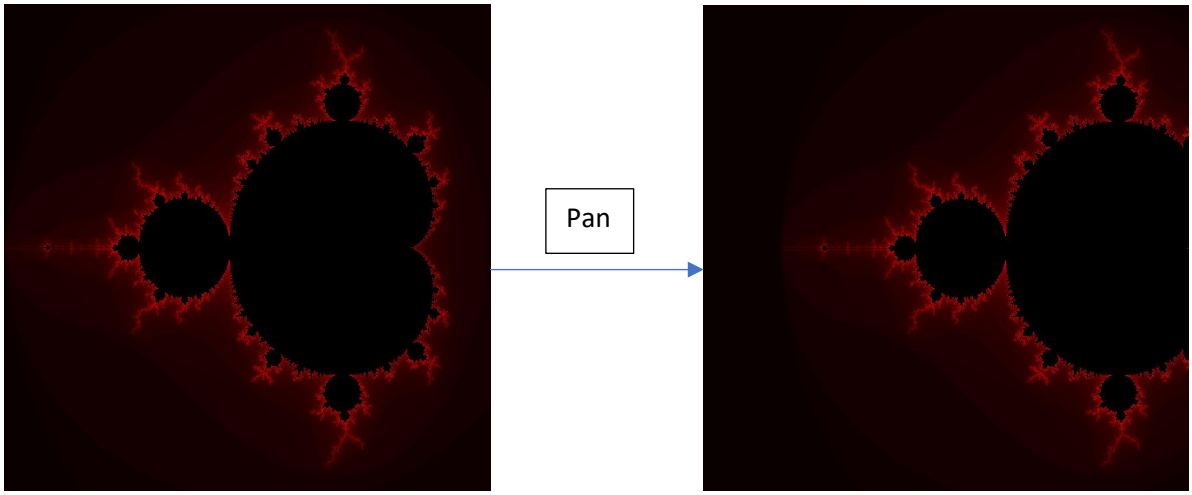
```
public void setNextState(double getxMin, double getxMax, double getyMin, double getyMax) {
    stateIndex++;
    last = canvas.get(stateIndex - 1);
    double minr = getyMin / 850.0 * (last.getMaxReal() - last.getMinReal()) + last.getMinReal();
    double mini = getxMin / 850.0 * (last.getMaxImag() - last.getMinImag()) + last.getMinImag();
    double maxr = getyMax / 850.0 * (last.getMaxReal() - last.getMinReal()) + last.getMinReal();
    double maxi = getxMax / 850.0 * (last.getMaxImag() - last.getMinImag()) + last.getMinImag();
    canvas.add(new Canvas(minr, maxr, mini, maxi, last.getColorStyle(), last.getMaxIteration()));
}
```



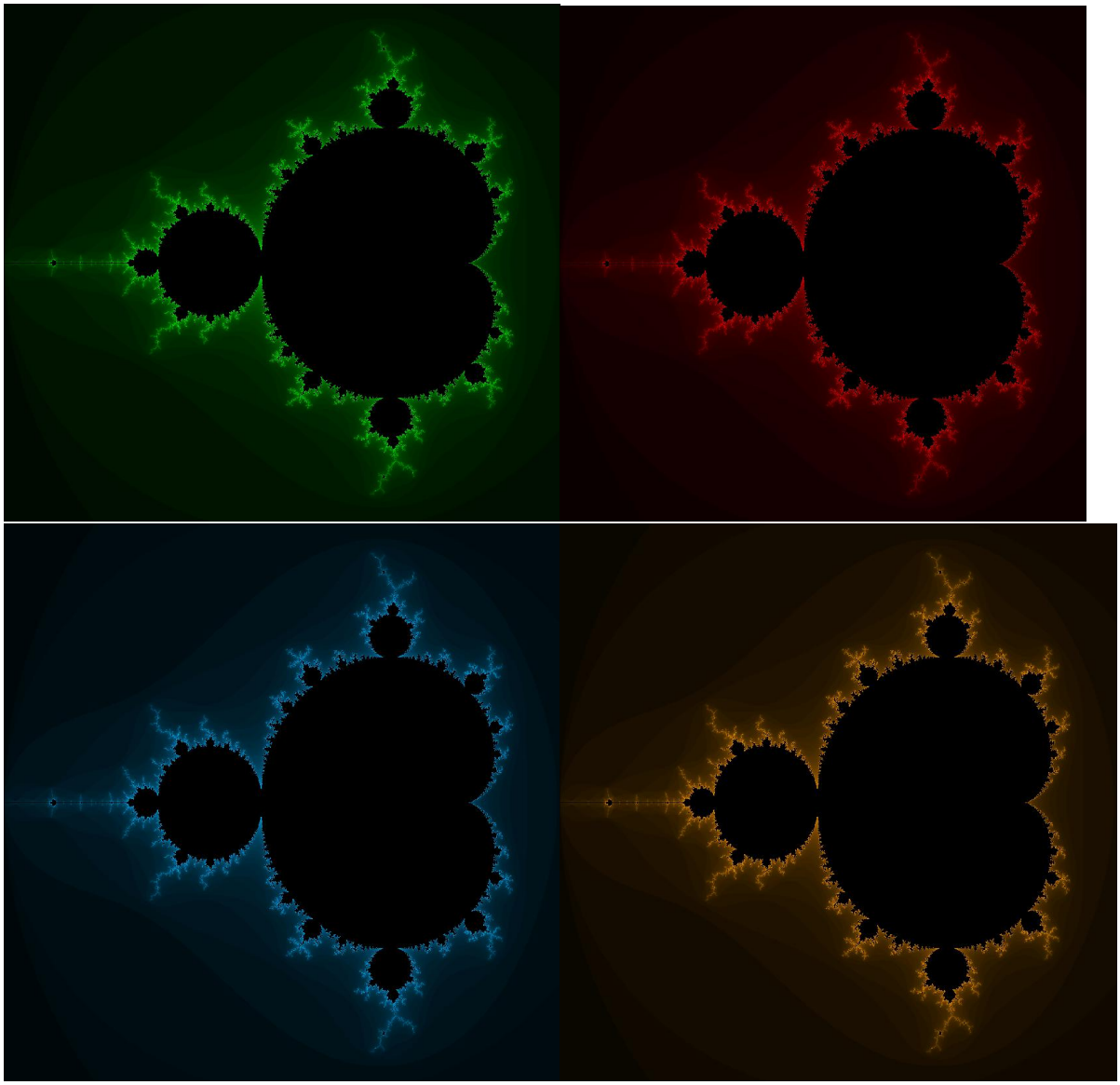
Pan method and screen display

It generates a line that starts at the first point, points to the end point's direction and has the side length of the longer projection.The algorithm was stored in the MandelbrotModel class and the figure below will display the details of the respective method and how the effect display on the screen.

```
public void setNextState2(double x1, double x2, double y1, double y2) {
    stateIndex++;
    last = canvas.get(stateIndex - 1);
    double minr = last.getMinReal()
            - ((y2 - y1) * (last.getMaxReal() - last.getMinReal()) / last.getMaxIteration() * 0.5);
    double mini = last.getMinImag()
            - ((x2 - x1) * (last.getMaxImag() - last.getMinImag()) / last.getMaxIteration() * 0.5);
    double maxr = last.getMaxReal()
            - ((y2 - y1) * (last.getMaxReal() - last.getMinReal()) / last.getMaxIteration() * 0.5);
    double maxi = last.getMaxImag()
            - ((x2 - x1) * (last.getMaxImag() - last.getMinImag()) / last.getMaxIteration() * 0.5);
    canvas.add(new Canvas(minr, maxr, mini, maxi, last.getColorStyle(), last.getMaxIteration()));
}
```

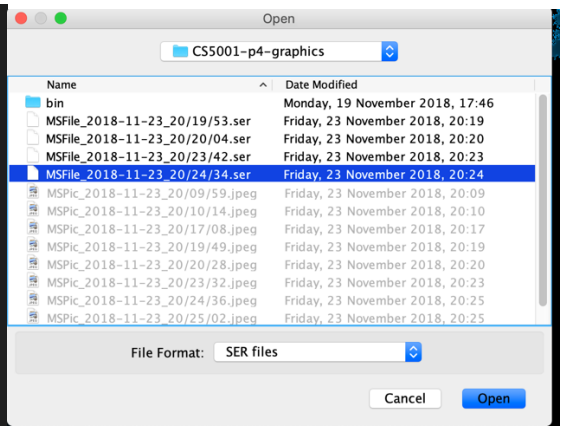


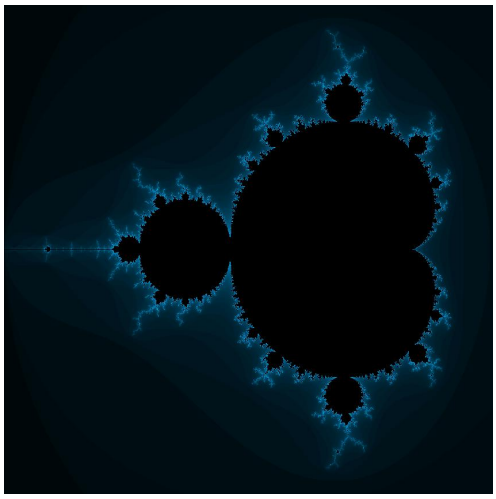
Enhancement: Colour Change (screen display)



Enhancement: Save and load(screen display)

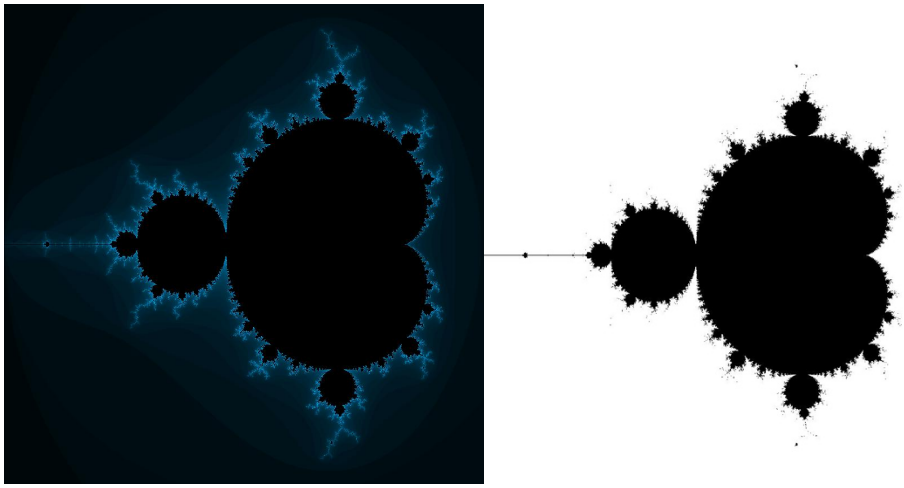
The first graph show already saved file in uploaded file, the second graph show if we open one of the ser file. The final graph show the situation that the data was loaded successfully(shown on the program screen).

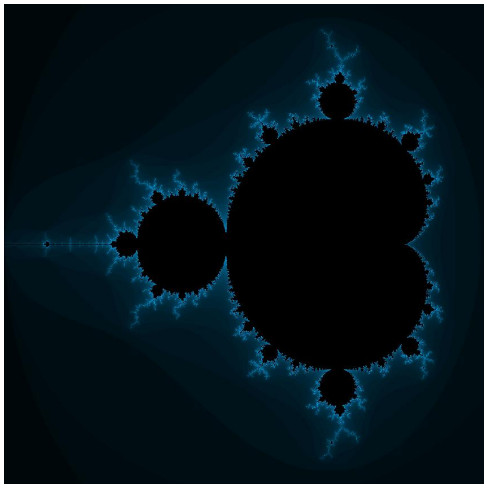




Enhancement: Full Undo, Redo, Reset

The first graph show the second state, the second graph show the situation if user click redo button the screen showing the first state, the third graph show the situation user click undo button showing the second state. If reset button was clicked, user will jump to the first state and cannot undo/redo.





Testing and thinking

The program has been tested in different situation, for example if I click the Pan Permission button and then check whether undo/redo/reset button can work. All the result is as I wish and I have added the dialog to remind user if he/she select the Pan Permission button, user can make the graph move without zoom and if user select the button again, the dialog will remind user again that user can zoom the graph now. If the Show Magnification button is selected, the screen will show some string on the screen as well. The graph below is one of the example.