Practical 4 - Mandelbrot Set Explorer

Student ID: 180029539

Problem Description:

- Create a Mandelbrot set Generator with functionalities like Basic display, zoom, pan, see magnification, set maxIterations and Undo/Redo functionalities.
- Implement different colour mappings, implement full undo/redo and reset functionalities.
- Permit user to save, load the current settings into an external file.
- Implement animations for zooming.

Structure of the Code:

- I have used the model-delegate model to achieve this problem.
- I have placed the functionalities of the model in 'com.gui.model' package.
- Then, placed all the functionalities of the delegate in 'com.gui.delegate' package.
- Finally the main function implementation in 'com.gui.main' package.

  Model Structure:
  - The model package includes a single class (MandelbrotCalculator.java) which is used to calculate the array with the output of the Mandelbrot function.
  - It is called by the delegate classes when it needs to calculate the values of the Mandelbrot set.
  - It has two functions which are used to calculate the values of the Mandelbrot equation.

  Delegate Structure:
  - The delegate package consists of two classes (MandelbrotGenerator.java, MandelbrotPanel.java) which contain the properties of the GUI elements of the code.
  - The first class is a combination of properties of the JFrame and the toolbar used to represent the user interface part of the code.
  - The JFrame is just an outer shell that houses a menubar, a toolbar and a panel which displays the graphical output.
  - The menu bar contains the option to save the settings, load the settings, and also save the panel as image.
  - The 'save' and 'save as' adds a valid extension after the name if its not provided.
  - The 'load' restricts users from loading only valid files.
  - The toolbar contains a Reset button which calls the initial configuration of the Mandelbrot set.
  - Next, it contains radio buttons to select between zoom and pan actions
    - For, zoom functionality, we allow the user to make a rectangular selection, which I implemented using Graphics2D set and mouse action implementations.
    - The pixel coordinates are then converted into the complex coordinates and then the updated Mandelbrot set is updated by the model.
    - After, which the panel class repaints the image with new coordinates.

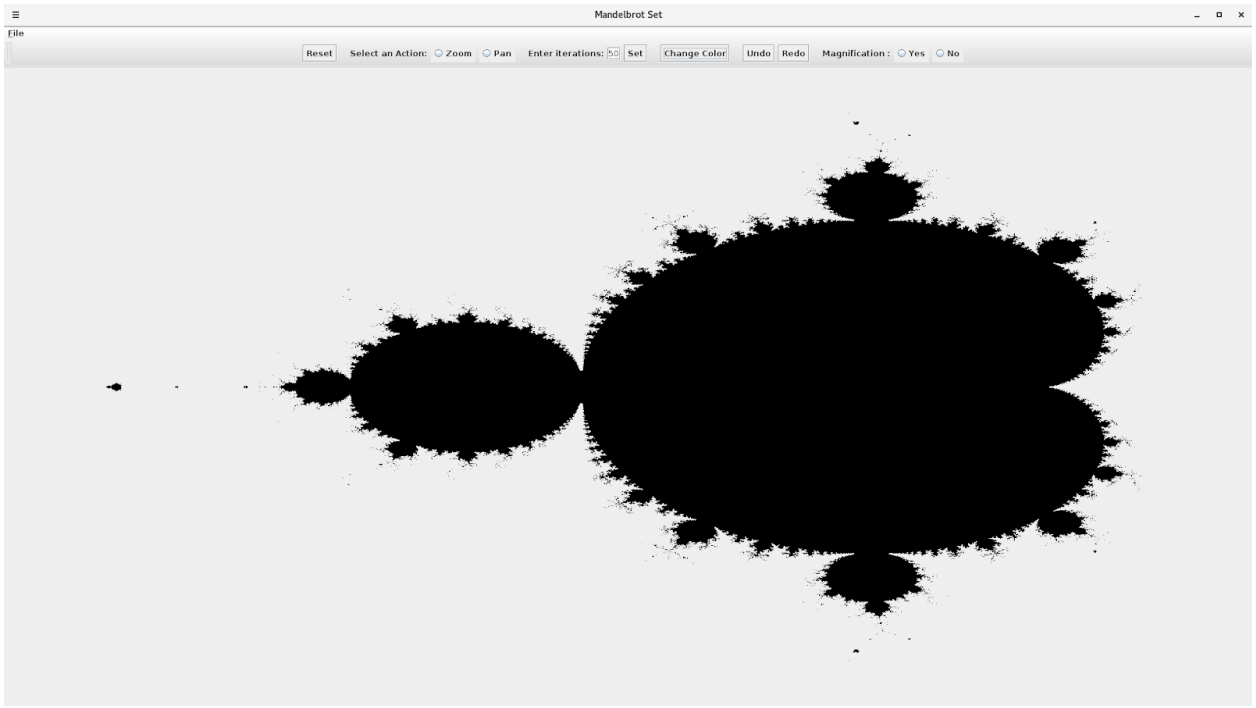
    - For, pan functionality, it works similar to the zoom functionality, but the input is one dimensional.
    - The image is panned in the direction the user drags the image.
    - The model class recomputes the new coordinates and the panel class outputs the new panned image.
- Then, I have an Iteration text field which takes in the new number of max iterations and recomputes the image based on the new settings from model class. This is achieved by actionListener linked to a “Set” button.
- Then, I have a Change color button, which changes between three colors (Red, green and blue).
- Next, I have implemented the Undo/Redo functionality with the help of an ArrayList which holds the history of all configurations from the initial configuration.
- This is implemented with help of two buttons (undo & redo) with attached actionListeners.
- Finally, the user is allowed an option to see how much zoomed in he is into the image with help of radio buttons.
- After, every operation the panel is refreshed to reflect the changes made.
- The next class is the panel class which includes functionalities to call the model class whenever some operation is invoked by the user.
- It also contains the elements of previously discussed features like the ArrayList with the previous configurations and mouse listener activities to accept the input from the user.

Design Decisions:

- I have picked the model-delegate architecture to implement for this code as the user interface and the controller functionalities are tightly coupled.
- I have also merged the toolbar and the frame together as the frame itself doesn’t have independent functionalities.
- But, I separated the panel into a different class as it has its own functionalities like drawing graphics and registering mouse inputs from the user.
- In the zoom functionality, I have used the rectangular selection as the initial configuration of the complex coordinate system was rectangular.
- I used an ArrayList instead of a stack as we had to implement both undo and redo functionalities and I thought an ArrayList made more sense as we had to traverse in both directions on the version scale.
- For, the resolution, I took a dynamic resolution that takes the current screen resolution as input.

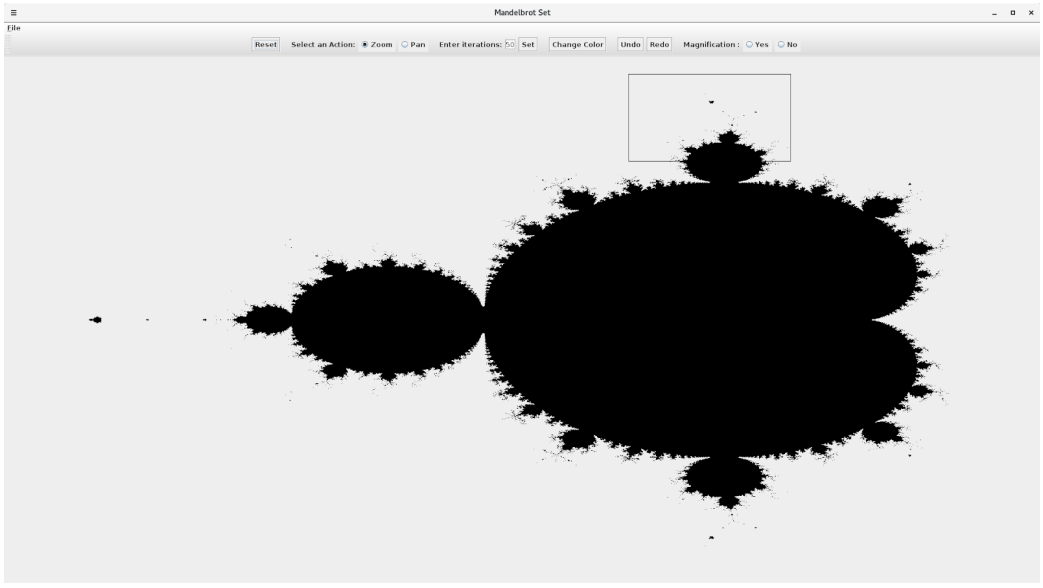
Operations & Outputs:

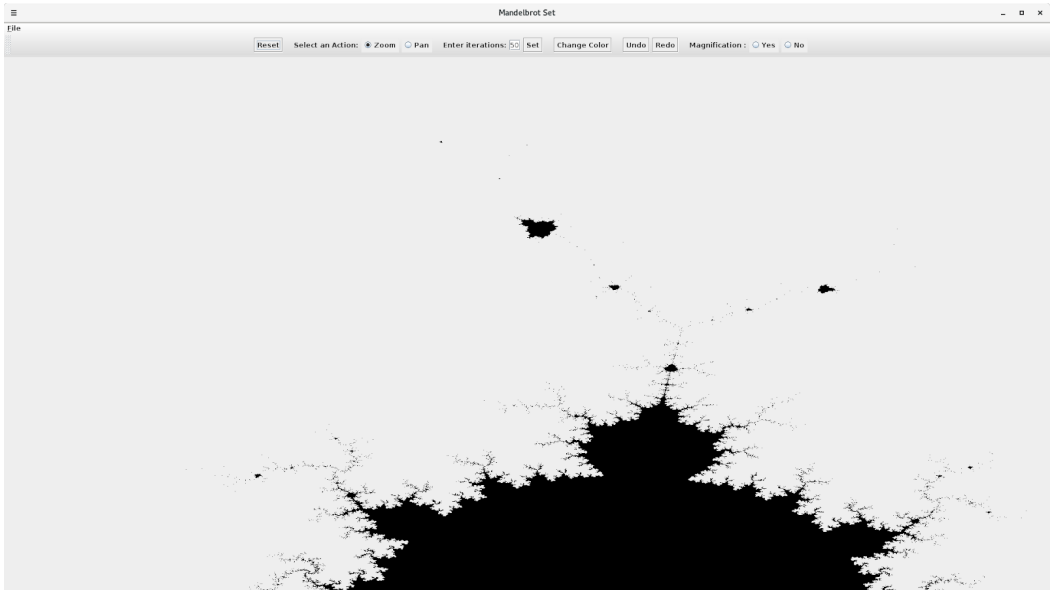
1. **Basic Display:**



   a. **Testing:** I have tested the system in multiple resolutions and it works fine in all the resolutions
   b. **Problems Faced:** When I was using fixed resolution, different systems outputted the image differently. Then, I changed to dynamic resolution.

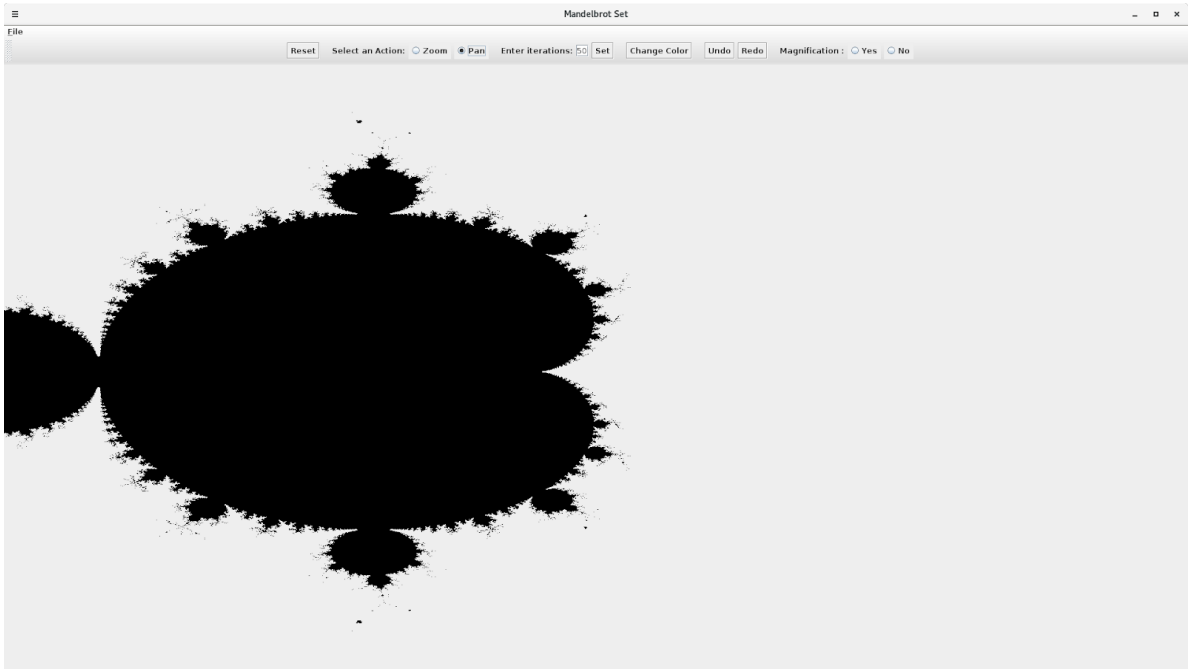
2. **Zoom Functionality:**





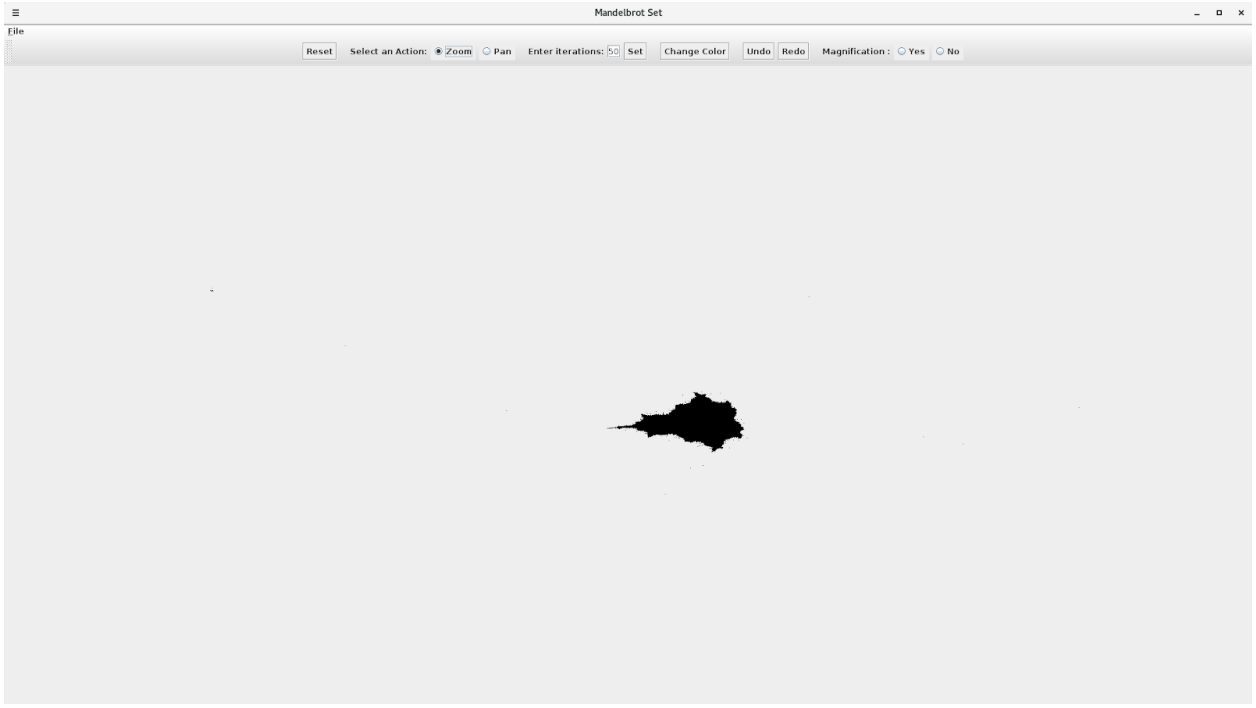
a. **Testing:** I have tested the zoom functionality in both the directions from top to bottom and left to right and in the reverse directions. The image zooms in perfectly.
b. **Problems Faced:** In the initial stages of development, only top-left to bottom-right zoom was working. In other cases, image was getting inverted. Then, I have used the min and max functions to rectify the initial and final points and to enable zoom in all directions

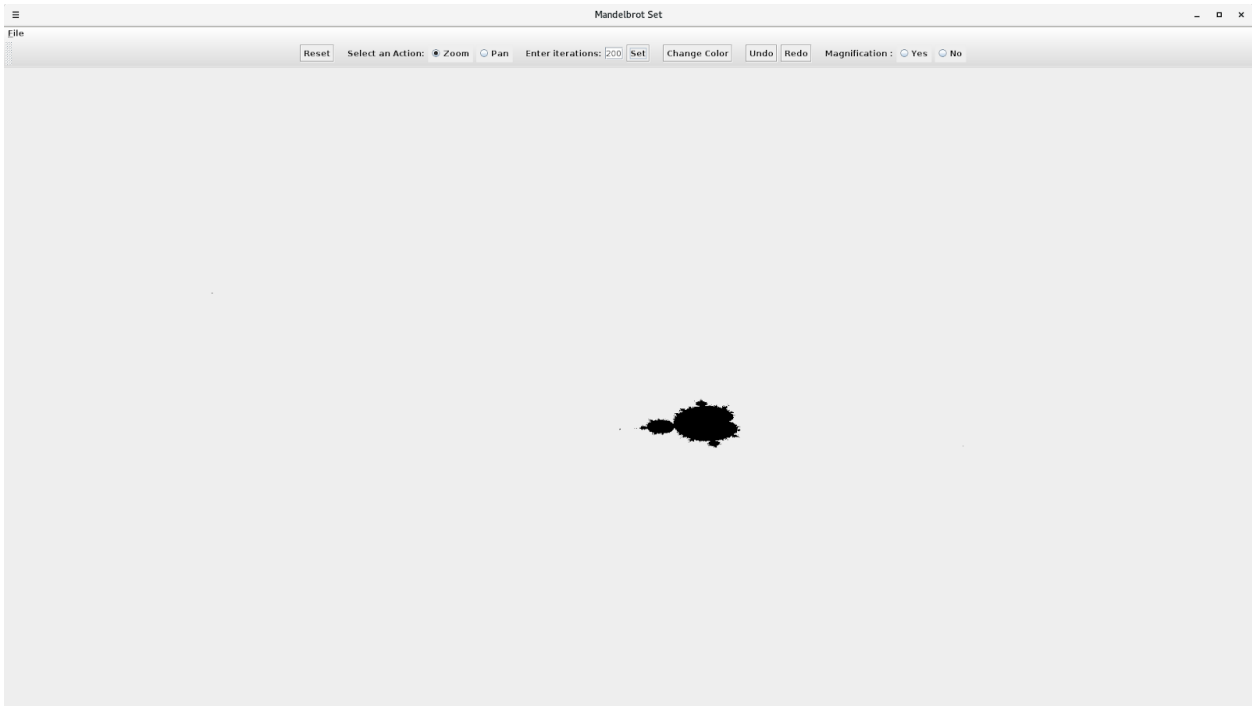
3. **Pan Functionality:**



a. **Testing:** I have tested the pan functionality in all the directions and it works in all directions.

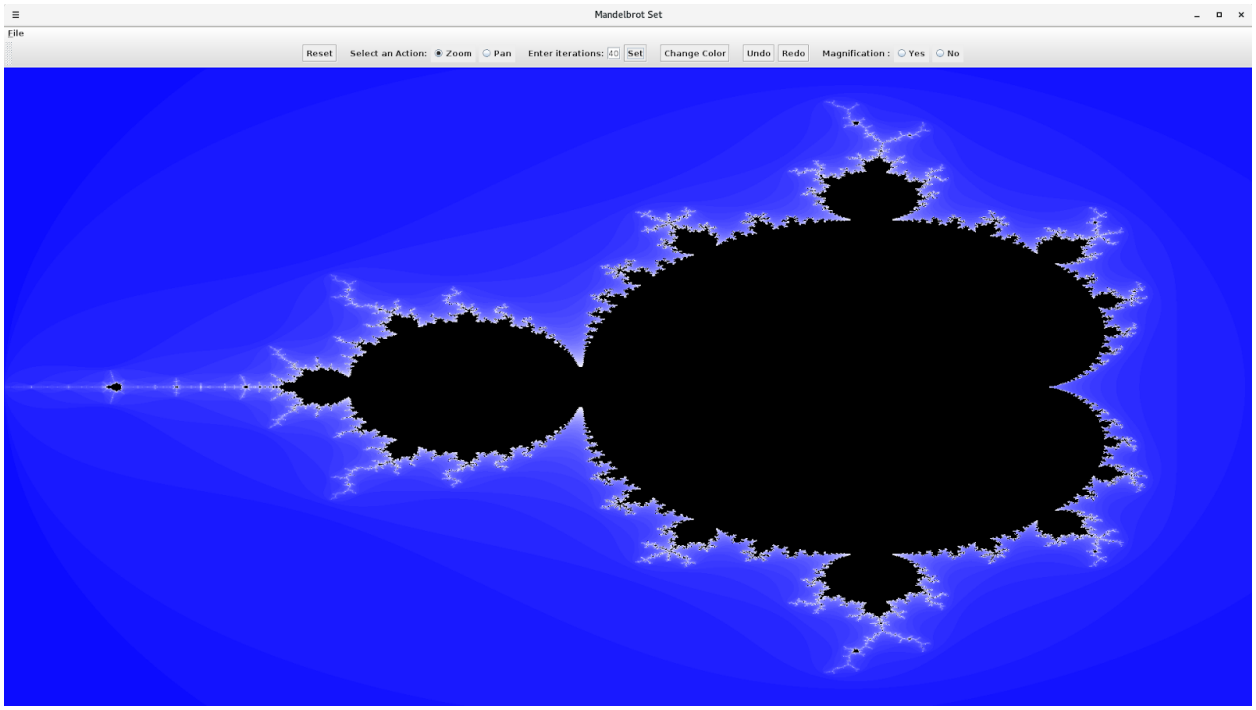
4. **Set Iterations Functionality:**





a. **Testing:** I have tested the set Iterations functionality in different conditions of the program and it works fine in all cases.
b. **Problems Faced:** Initially, after I updated the iterations the panel was refreshing the changes. But, the text field in toolbar didn't. Then, I ran the update text in the re-configure method. So, it gets updated every time the panel is refreshed.

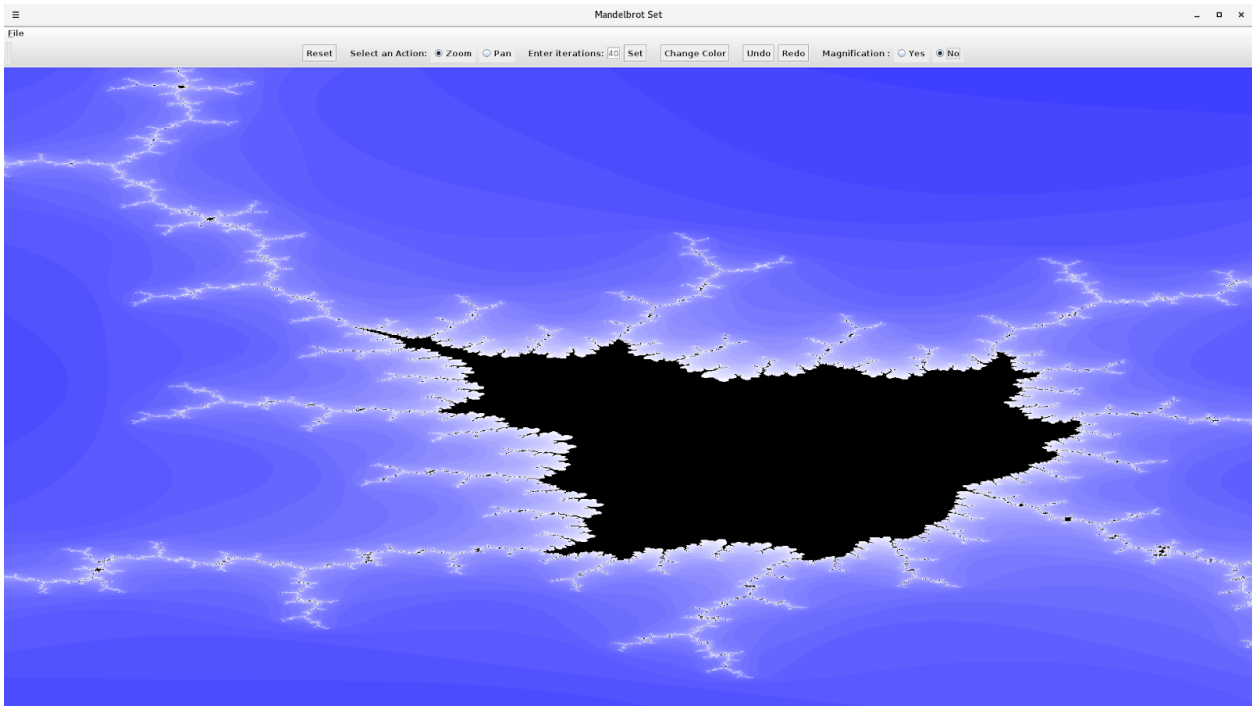
5. **Change Color Functionality:**

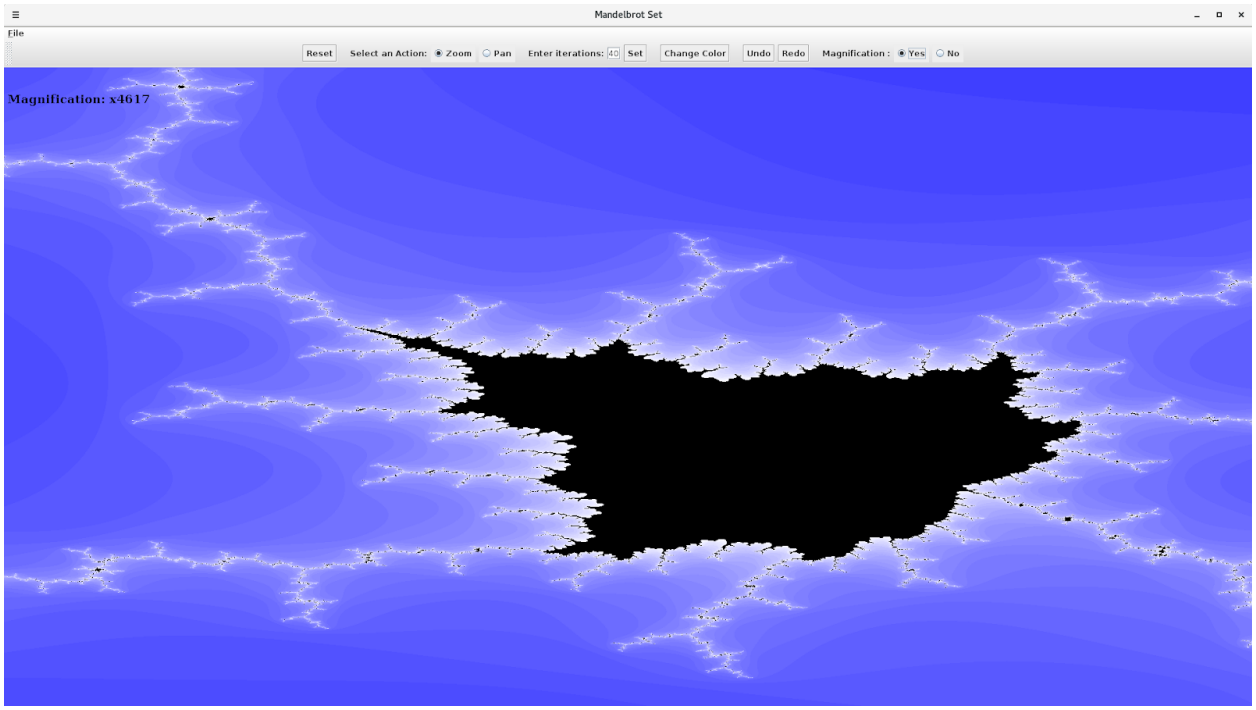


   a. **Testing:** All, the three colors work properly.
   b. **Problems Faced:** If the max iterations is higher, the change in gradient is lower.

6. **Undo/Redo Functionality:**
   a. **Testing:** I did multiple tests to test the functionality of undo and redo. The gui remains in the same screen, if it reaches the maximum redos or undos. Even the color changes are recorded in the undo and redo functionality. The color changes are also recorded in the undo and redo actions.
   b. **Problems Faced:** It was hard to incorporate a new data structure to store the configurations. It was also hard to implement the logic to keep track of older versions.

7. Magnification Functionality:





a. **Testing:** I have tested the magnification show/hide option. It persists even during undos and redos.
b. **Problems Faced:** I have not faced any problems in implementing the functionality.

8. **File Save/Load/Save As Image Functionality:**
    a. **Testing:** I have tested the Save functionality with different extensions, it appends ".list" to invalid names to make it valid. In the load functionality, it only accepts the files with ".list" extension. The "Save as Image" option appends ",png" if extension is not found or invalid.