Google Summer of Code

GSoC 2021

SCoRe Lab organization

Project: Image Lab

Project Idea: Improve the Image Lab user experience

Pola Hano Shehat

Faculty of Engineering – Fayoum University

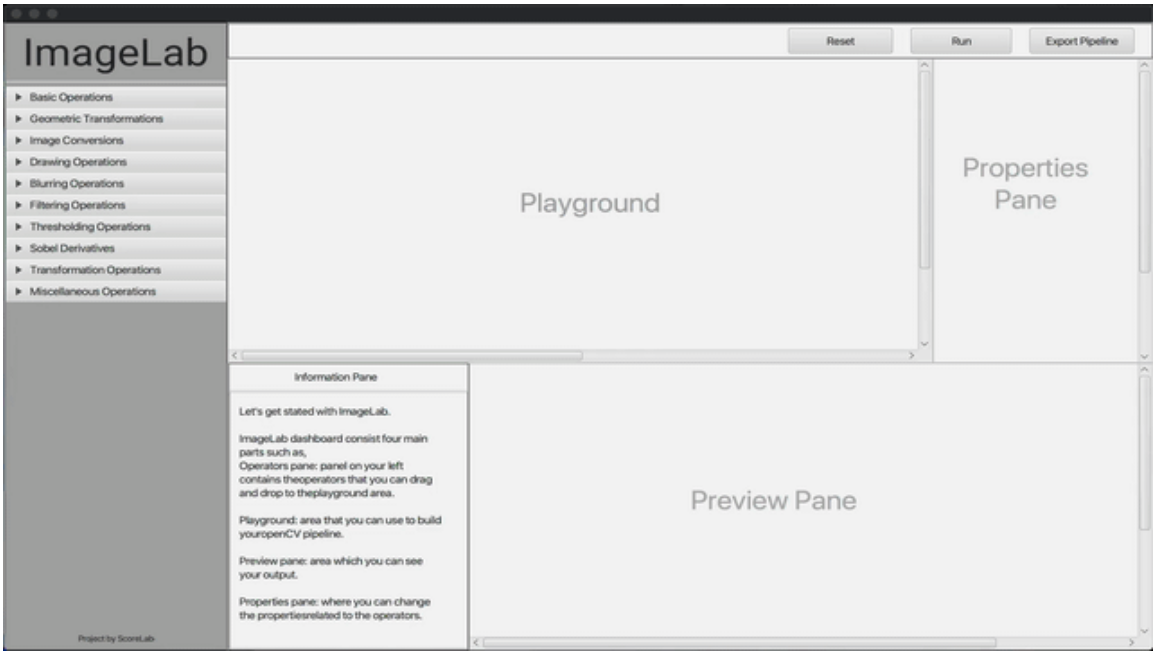
Fayoum, Egypt

Computer Engineering department

Table of Contents

1. About the project

ImageLab is an application that enables everyone to learn image processing principles and techniques in an easier, less logical manner. As a result, by interacting with ImageLab, this platform allows and motivates new users to learn how image processing concepts work. Furthermore, ImageLab provides a test environment for users who are already familiar with image processing activities before moving forward with actual implementation or creation.



Briefly, ImageLab is a project aims to facilitate learning image processing for those who don't have experience with image processing and improve the experience for people already have some experience. You can, by using the app, understand how complicated image filters form by small blocks of operations.

The idea of the project is great and beneficial but I faced some problems when I used it as a user like: the project needs many improvements in user interface to meet user's satisfaction and also it has few features so it would be better if it is supplied with more to ease usage.

So, in my GSoC period I am motivated to work on that useful project and enhance the user experience by designing a modern and interactive interface and will also make blocks of operators used in the processing of image be more attractive and interactive to be suitable for all ages. And I will illustrate all about my ideas on next pages.

2. Project Goals

In this section I will illustrate the ideas I see are so important for the project to apply during GSoC period and will illustrate each point in detail in implementation section:

- Designing a logo for ImageLab project
- Designing a splash screen (loading screen)
- Making a quick guide on the first use
- Improving the project User Interface
- Adding night mode
- Changing block shapes
- Adding zooming in the playground and also in preview pane.
- Add save or don't save pipeline option when closing the app.
- Enhance the view of the project documentation.
- Facilitate names of operator
- Improve error messages UI
- Replacing Information pane with button tooltips
- Adding help section
- Adding Info (About us) section
- Providing availability for all image sizes
- Adding history pane
- Drag and drop in basket to remove a specific filter
- Add (new pipeline) button
- Package the application into an installable exe file in windows
- Run with keyboard shortcut.

In case I have finished all above tasks, I could work on these:

⇒ Design infograph to illustrate all operators
⇒ Add playing music feature
⇒ Adding the ability to search for a block.
⇒ Adding undoing feature
⇒ Asking about personal name and photo at the first use
⇒ Add Arabic Translation
⇒ Adding feedback screen before closing application.
⇒ Adding drawing operations feature by adding ruler like MS word ruler in preview pane

3. Implementation

3.1. Designing a logo for ImageLab project

Here I've designed 2 logos for Image Lab project with different colors and styles and used the one who attracted me for dedicating the UI colors. But of course, we can change colors of UI depending on the choice of logo of SCoRe organization and my mentor.

3.2. Designing a splash screen (loading screen)

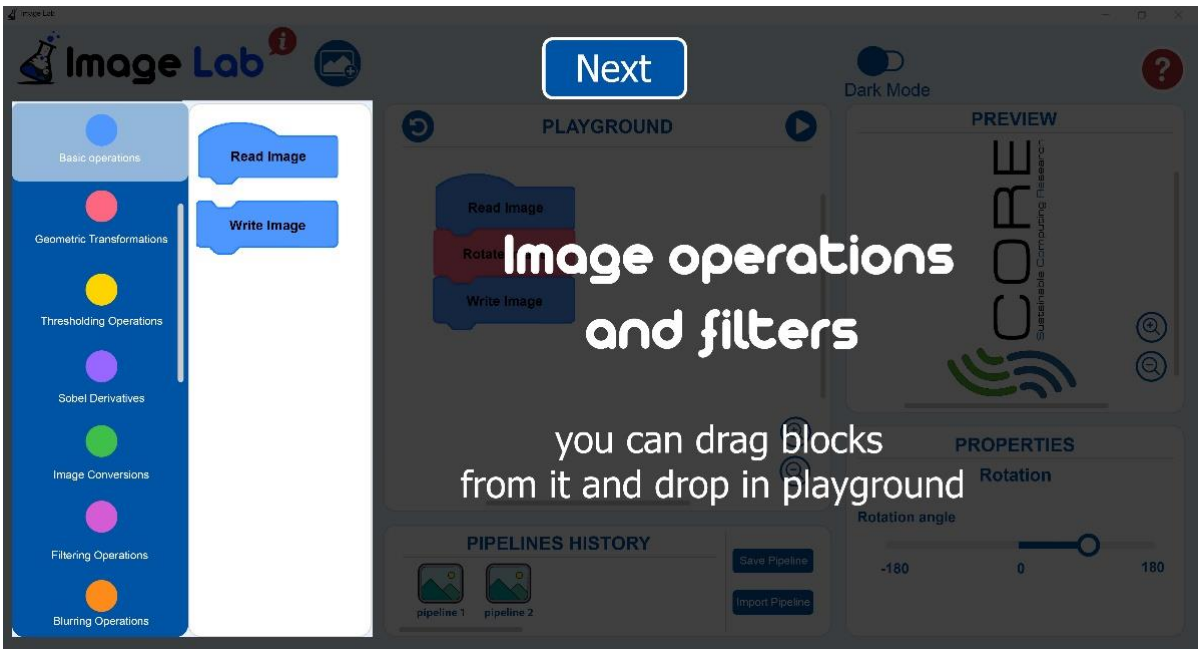
This is a design for the splash screen of the project that will load the bar on the bottom for about 3-4 seconds then the program will open.

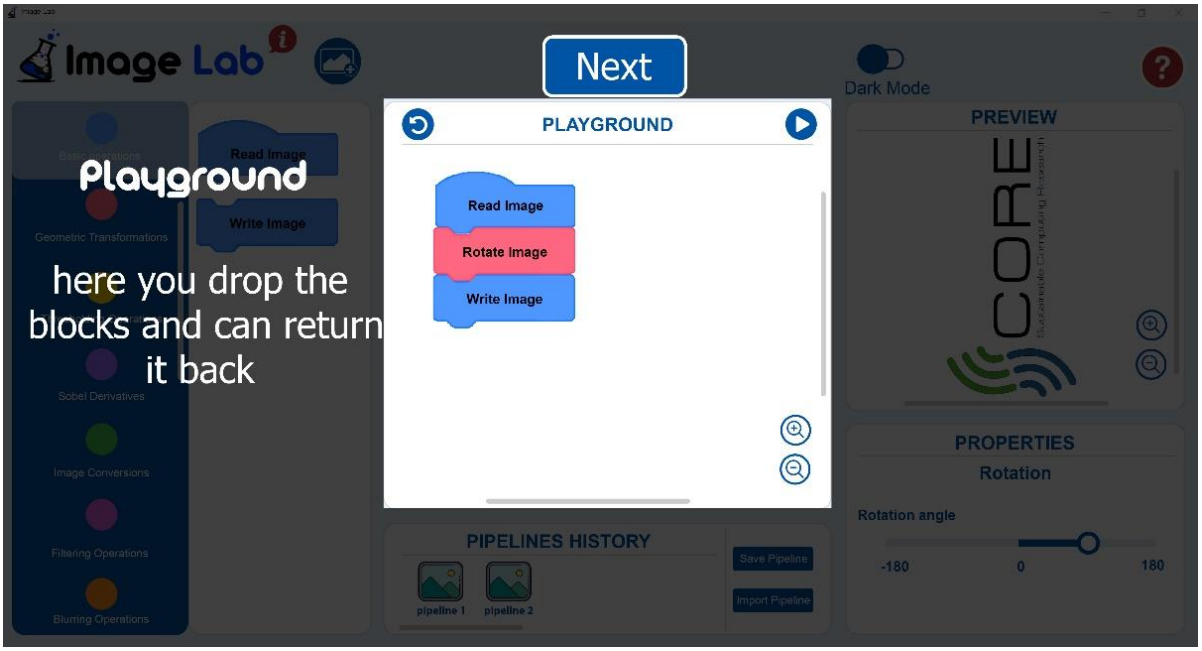
I think it will be attractive and give a good introduction of the app to the user.

3.3. Introducing a quick guide on the first use

When the user first uses the app, he will need a quick tour in the application to tell him about each component and its role and choose path to save exported pipelines.

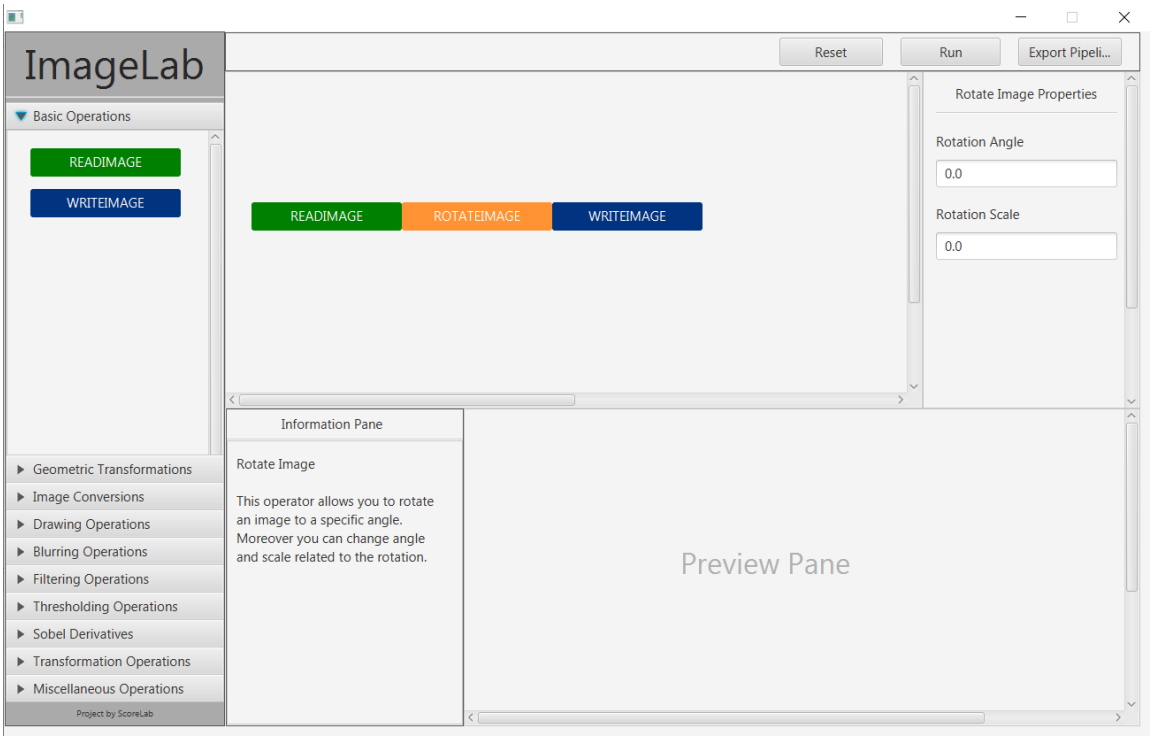
Note: This is a demo of the idea of this guide not showing the exact sentences will be written:



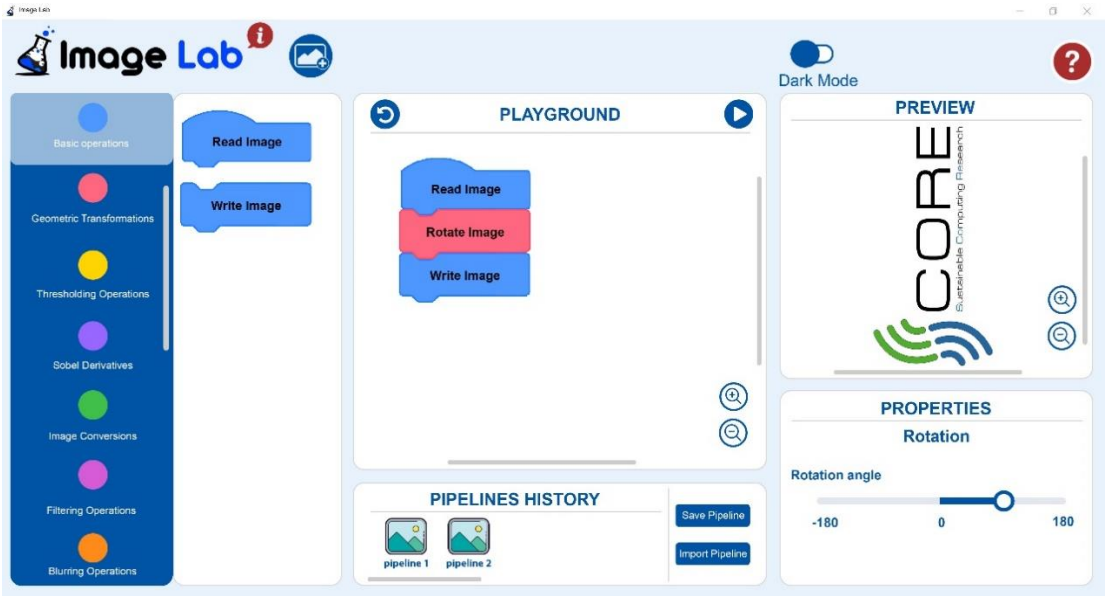


3.4. Improving the project User Interface

I have designed an initial new design for UI of the project. Here you can see the old design:



And here is the new design:

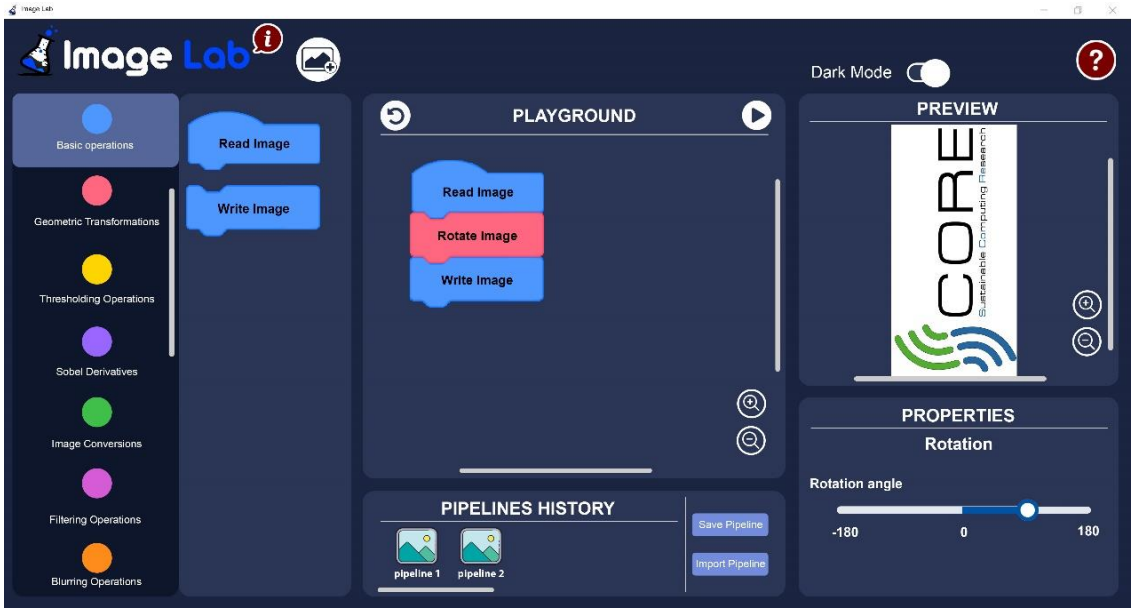


The step after that will be implementing that design using Javafx and Scene Builder so we will have a new **dashboard.xml** file, **DashboardController.java** file and a new **style.css** file then I will introduce code of the old design to the new one in order to be able to work properly on it taking in mind the new sizes, colors, styles etc.

I may use some Javafx libraries with modern components like : MigLayout, Jfoenix, BootstrapFX, FontAwesomeFX.

3.5. Adding night mode

I've also designed a prototype of the dark mode for the project



Here, I will create a new file **style-dark-mode.css** and add the new colors of the components in order to change on turning dark mode on.

And this is pseudo code of how **style-dark-mode.css** could be like:

```
.root {
    -fx-base: #000000;
    -fx-accent: #000000 ;
    -fx-focused-text-base-color : ladder(
            -fx-selection-bar,
            -fx-light-text-color 45%,
            -fx-dark-text-color 46%,
            -fx-mid-text-color 60%
        );
}
```

And we will add a listener to the toggle button and here is pseudo code of it:

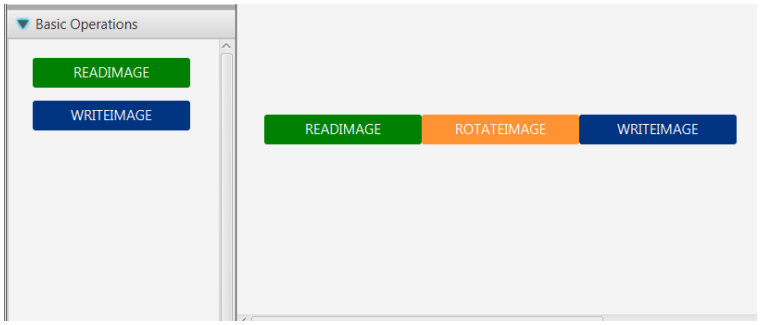
```
checkToggleButton.addListener(parameters) -> {
    if (isSelected)
        show("dark-theme.css")
     else
        hide("dark-theme.css")
}
```

3.6. Changing block shapes

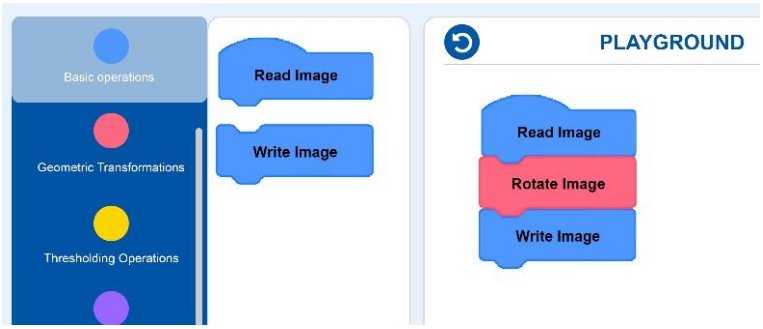
Here, I will Introduce block shapes depending on the operation types which can be merged with each other at the workspace area (playground) by changing shapes of rectangular blocks to be like Scratch blocks to inform user nicely about operators that can be merged with each other at the workspace area without using error messages.

For example, in the new design below we can't put Write Image before Read Image because they can't be composed together.

For more illustration, that is the style of blocks now:



And will be changed to be:



And this change will make interacting with blocks more easily, smoother and looks more beautiful and comfortable to eyes.

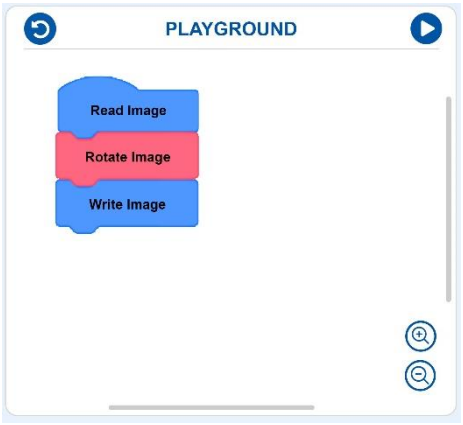
To do that step we will need to see all operations and for every operation what other operations can be merged with it in order to make block designs than can match each other. And for those operations that don't match each other will need to be different in shape in order not to make them merge together and in the same time without showing error message as it exists in old design.

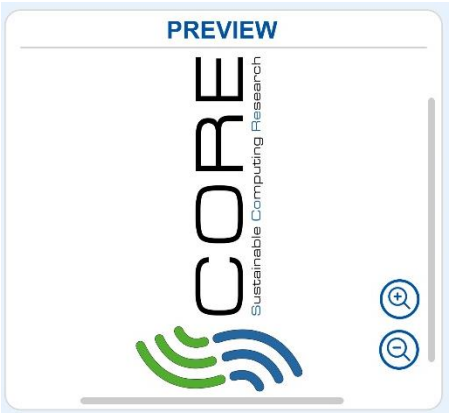
So, we will make a table like the following and then analyze operations that match together and that don't match:

operation	Other operations that are allowed to be merged with it
ReadImage	All operators
ApplyBorder	ReadImage – RotateImage – WriteImage – ColoredImageToBinary – ConvertToGrayscale – ApplyBlurEffect – ApplyGaussianBlurEffect – ApplyMedianBlurEffect – ApplyBoxFilter – ApplySimpleThreshold - ApplyBorder
RotateImage	ReadImage.class - RotateImage.class - WriteImage.class - ConvertToGrayscale.class
And so on for all operators …	

For doing such a task I will use the help of Scratch blocks generator to get very high-quality blocks and may also use editing apps like photoshop to make some modifications on it.

3.7. Adding zooming in the playground and also in preview pane





For the ease of usage of playground and preview pane we will add zooming to them with 2 ways:

1) By using mouse wheel

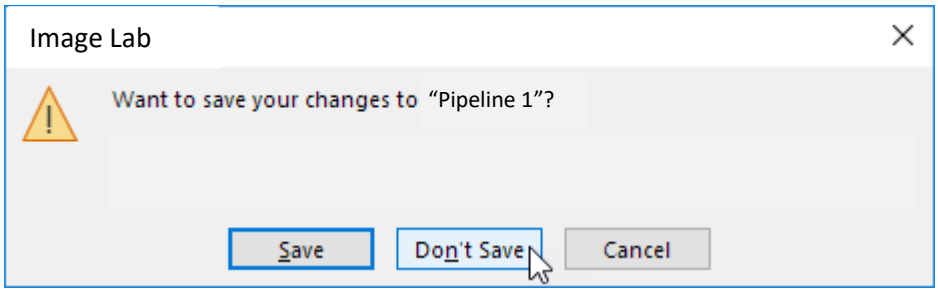
JavaFX provides option to listen for mouse scroll event. We will make use of that. ScrollEvent.SCROLL allows to track the scrolling.

We will do next steps for adding zooming feature:

a. Attach a scroll listener
b. Get how much scroll was done in Y axis.
c. Add it to the Z-axis location.

2) By using zoom in & zoom out buttons

3.1. Add save or don’t save pipeline alert when closing the app.



This part would be implemented by overriding the close button function by adding event handler on pressing the exit button and form the new window style and buttons and follow that logic of pseudo code:

If (save pressed)

save pipeline then application closes

Else if (Don’t save pressed)

Close application immediately without saving pipeline

Else if (close or X sign pressed)

Close save window and continue working on application

3.2. Enhance the view of the project documentation.

By designing a new design for that page and fix errors in links exists in that documentation and add more description for the application about setup guidelines, application GUI, operators …

3.3. Facilitate names of operator

We can change blocks names to be easier to user for example:

Read Image => Open Image

Write Image => Save Image

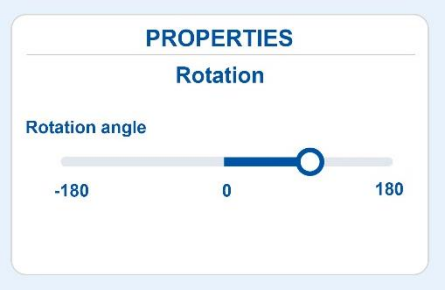
And like this for most of blocks to facilitate understating what they do.

3.4. Improve error messages UI

By designing a new UI for error messages to improve user experience.

3.5. Replacing text input fields with Scroll bars

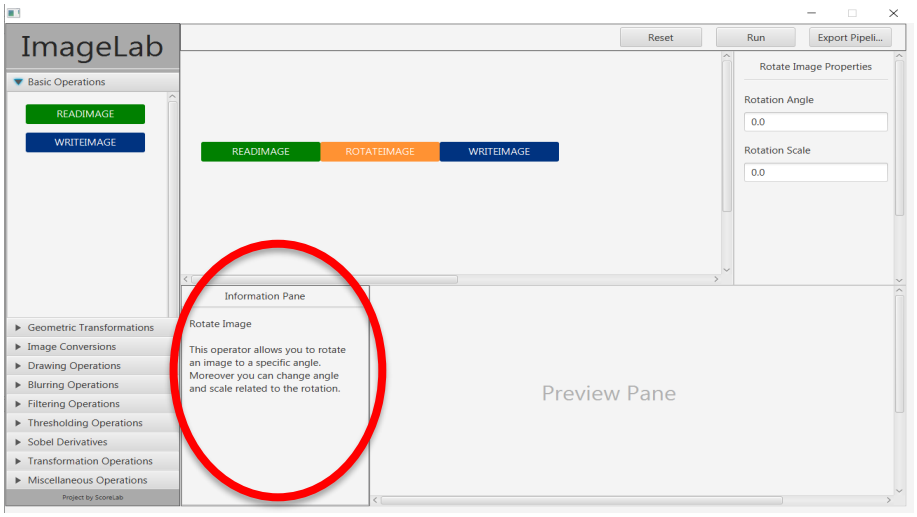
In some operators it asks the user to write a value and if he entered a wrong format it gives us **NumberFormatException** so, to avoid that exception we can replace text input components with scroll bars. It will provide smoother interacting and better user experience.



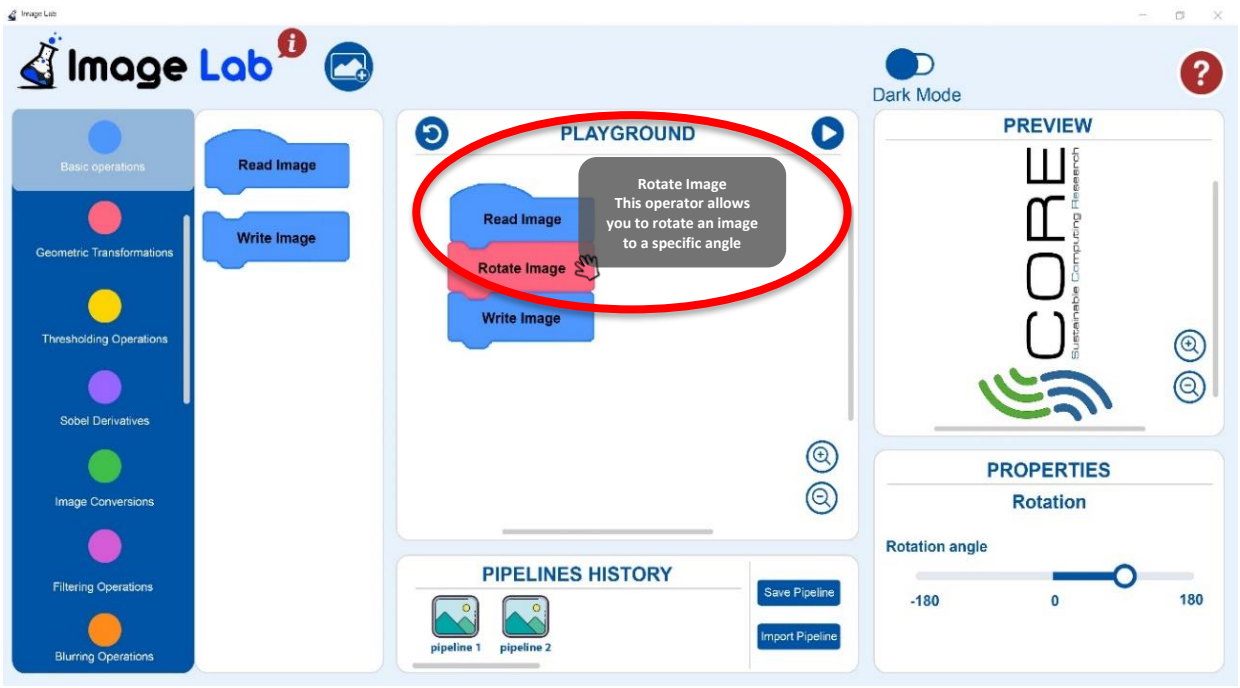
3.6. Replacing Information pane with button tooltips

It is a small change to improve user experience. That change is that we will replace the information pane in our old design with tooltips that gives a brief about the operator you select and if the user wants to know more, he can navigate help section. And by that change we will use the area of information pane to increase size of playground and add a new section that shows saved pipelines before.

The old design Information pane:



The new buttons tooltips:



3.7. Adding help section

I have more an idea about the help section:

- Making button file navigate user to a PDF file online that includes all details about the project (how to use the GUI? - how each operator acts? – info about the project idea – how to contribute – how to setup in windows/Linux/macOS …)
- Opening another window above the app window offline that has all information about project like the PDF but it will be written in a scrollable pane and will be divided into tabs and sections like help of most of programs.
- Improving the page of Image Lab documentation by adding not only the setup guidelines but also all other information about the project and organize the page into sections.
- Providing links on some small tutorials about how to use the program.

These are some ideas but I will discuss my mentor about that and choose which is more suitable and preferable.

3.8. Adding Info (About us) section

That is a small window that appears when you click info button. It provides project github link, chatting groups, information about SCoRe Lab organization and could contain a link (for example: SurveyMonkey) that allows user to provide his feedback about the application if he faces any issues, has a question or provides an idea to enhance the app.

3.9. Providing availability for all image sizes

In the project there is a problem of not allowing all image sizes to be imported. In other words, when importing a large image size, it shows only a small part of it in the preview pane and will solve this problem by fitting imported image in the preview pane by using `fitWidthProperty()` method in `ImageView` class.

3.10. Adding history pane

It shows the previous saved or exported pipelines by the user. When user save a pipeline, it will be saved in the output folder path the user chooses.

We will implement a method that take the path of pipelines saved and its name, the name will appear in pipeline history section and the path will be used to get the blocks used to build it and can redraw them in playground when the user presses any of them.

And here is how the exported pipeline looks like:

```
                    ImageLab v1 | Operation Pipeline
                      Sun Apr 11 01:22:35 EET 2021


READ_IMAGE
|
?
ROTATE_IMAGE
|
?
WRITE_IMAGE
|
?
```

So, we can return these lines into blocks by extracting blocks names from these lines and we will do that by implementing a new algorithm.

3.11. Removing a specific operator

It can be done by two ways:

1- By dragging a specific block from the playground and dropping it back into the operations pane.

2- By dragging a specific block from the playground and dropping it into a bin that appears when dragging a block or bin image can be fixed like a button in the playground and the user drop into it.

3.12. Add (new pipeline) button

When the user presses that button, a message appears asks him if he wants the new pipeline in a new window or in the same window.

3.13. Package the application into an installable exe file in windows

First, we will convert our application to a .jar file by using `maven-assembly-plugin` or `maven-shade-plugin` dependencies because maven uses convention over configuration, so you only need to tell Maven the things that are different from the defaults.

Then we can use one of the famous softwares like **Launch4J** that could convert (.jar) files to executable files (.exe).

3.14. Run with keyboard shortcut.

We can use for example (ctrl+R) shortcut to run the pipeline and we will implement that by using `keyCodeCombination` class and pass the shortcuts to it and by adding an event on pressing the keyboard (ctrl+R).
Implementation will go through these steps:

```
[1] Adding event filter to our button
[2] Initializing Key Combination.
[3] Handling that key event
```