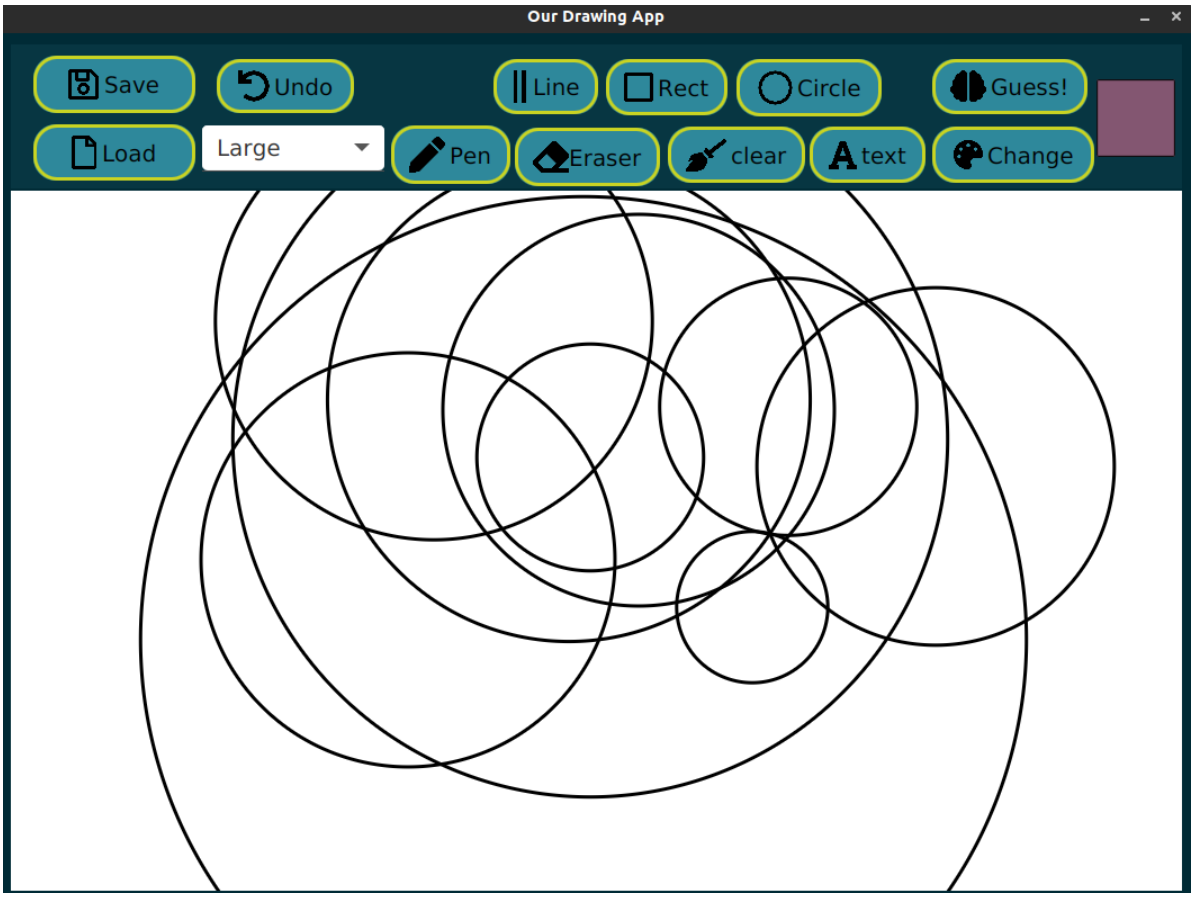
JavaFX_Paint



This project is our recreation of the Paint application written in **Java** and **JavaFX**.
This application hosts a lot of the classic features, as well as new creative spins including an AI assistant that can guess the sketch you are drawing!

Built With

- JavaFX
- SceneBuilder
- Tensorflow for Java

Getting Started

To get a local copy up and running follow these simple steps.

Prerequisites

- All required dependencies can be installed with **Maven** by importing the external packages in `pom.xml` file.

Installation

1. Clone the repo

   ```
   git clone https://github.com/anisdismail/JavaFX_Paint.git
   ```

2. Change to the project repositry:

   ```
   cd JavaFX_Paint
   ```

3. Install required packages with Maven
4. Mark the root folder of the project as **resources root**.
   When Opening the Project on `Intellij`, right-click on the root folder `JavaFX_Paint` and set is as `Set Directory As` -> `Resources Root`

Usage

To compile the project, run the following command:

```
javac Starter.java
```

Then to run the project, run the following command:

```
java Starter
```

Roadmap

See the open issues for a list of proposed features (and known issues).

About The Project

1) Loading and Saving

Saving was done by using `SwingFXUtils.fromFXImage()` which converts a canvas `snapshot` to a `png` image written to a file.
Loading uses the `Graphic Context`'s `drawImage()` function to load an image into the canvas.

2) Undo

For undo, we defined a stack `Stack<Image> undoStack = new Stack<>();` to which we then pushed the canvas snapshots.

With each change, we would `undoStack.push()` the current canvas before the change.

On each undo, we would `undoStack.pop()` an Image and then set it as canvas to restore it.

3) Lines, Rectangles and Circles

Lines, Rectangles and Circles were simply implemented using the `strokeLine()`, `strokeOval()` and `strokeRect()` functions of the `Graphics Context`.

4) Pen, Eraser and Tool Size

The main drawing and erasing functions were implemented by adding a listener to the `Canvas` object of type `MouseEvent.MOUSE_DRAGGED` and `MouseEvent.MOUSE_CLICKED`.
In the body of both listeners, and for the case were the `Pen` was selected, we use the `fillOval()` method of `Graphic Context` to draw a circle of the chosen color
at the current Mouse location. When using the `Eraser`, `clearRect()` is called ,clearing an area of the canvas depending on the selected Tool size.

Moving on to Selecting the Tool size, the `ComboBox` allows the user to select between 3 sizes: `Small`, `Medium` and `Large`. This set size would not only affect the `Pen` and `Eraser`, but also the line thickness of drawn `Circles`, `Rectangles` and `Lines`.

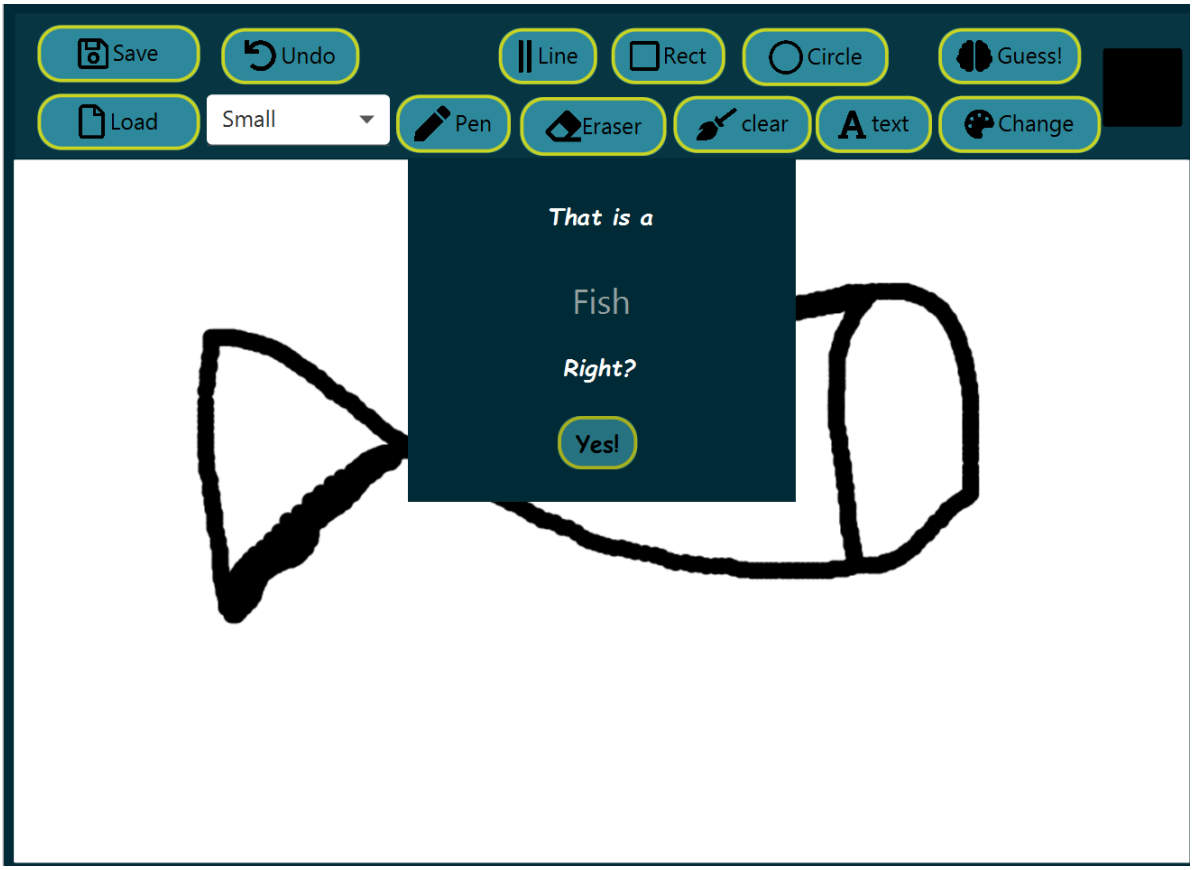
5) Clearing and adding Text

For clearing the drawing, we used the `clearRect( 0 , 0 , canvas.getWidth() , canvas.getHeight() )` method of the `Graphics Context`.

To add text to the canvas, the user would the Press the `Text` Button and then press on the canvas. Because we wanted direct feedback
of what the user was writing, we spawned a `Label` at that location and used `setText()` to set its content that would change after
each keypress.

The user would then press `ESC` to exit Text mode. The method `fillText()` of the `Graphics Context` is called
to write on the canvas. Afterwards, the `Label` would then be deleted. The creation and deletion of the `Label` would be transparent to the user, thinking he was directly writing in the canvas all along.

6) Guess the Sketch using AI



This features allows the user to draw some doodles and let the AI model guess!
The model is based on a Convolutional Neural Network model trained on sketches of the following classes:

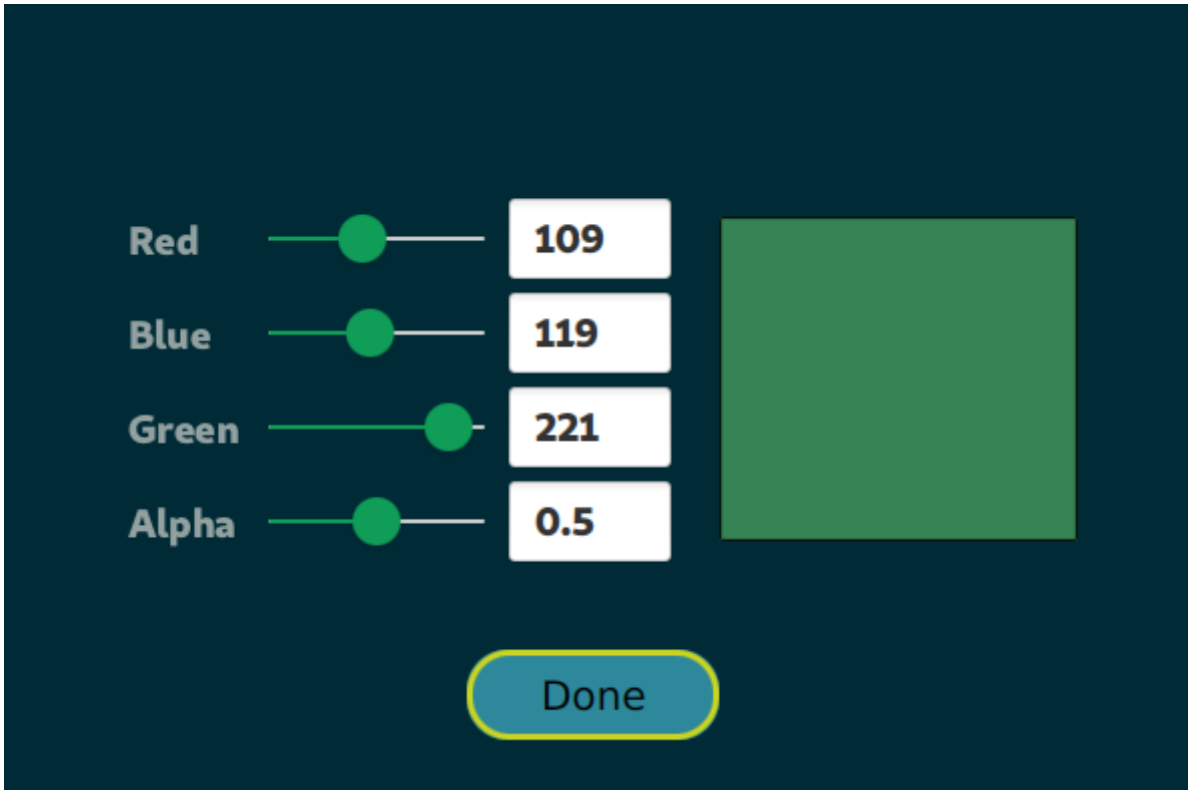
```
{"Apple", "Bowtie", "Candle", "Door", "Envelope", "Fish", "Guitar", "Ice Cream",
            "Lightning", "Moon", "Mountain", "Star", "Tent", "Toothbrush",
"Wristwatch"};
```

Further details on the model can be found here.

When the `Guess` button is pressed, a `snapshot` of the canvas is saved in the temp folder. Before feeding the
image into the CNN model, the image is first preprocessed using a `blurrImage` filter to make it less pixelated. Then
the image is resized to a `28x28` image using the `scaleImage` method. Next, we feed the image into the CNN model, and we select the class with the highest confidence predicted.

7) Color Picker

The color picker is implemented in a separate window, to avoid clutter in the main toolbar. We chose a simple layout as shown below.



Each Color Channel (Red, Green, Blue and Alpha) has a corresponding slider
and text fields. Also a Pane on the right is showing the resulting Color.

The `Done` button closes this window and returns the user to the main drawing screen.

The value of each Color Channel can be changed by moving the Slider or entering the value in the text fields.
The link between Slider and its corresponding text-field was done through properties listeners.
The Slider has a change listener on its `valueProperty()` which would change its value and the value of the textField.
The Text field has a change listener as well on its `textProperty()` which would also change its value and the value of the corresponding slider.