PD LAB
ASSIGNMENT - 6

Name: Raunak Thanawala
Registration Number: 231070051
Branch: Computer Engineering
Batch: 3

Aim:-

Create image editor using tkinter and OpenCV in Python

Theory:-

- OpenCV stands for Open Source Computer Vision Library.
- It is a powerful library focused on real-time computer vision and image processing.
- It was initially developed by Intel, now maintained by the OpenCV community.

- Applications:
  - Image analysis: Basic image processing techniques.
  - Object detection: Identifying and recognizing objects in images.
  - Motion tracking: Analyzing the movement of objects.

- Main Concepts:
  - Image representation: Uses multi-dimensional arrays for images.
  - Color spaces: Supports various representations (e.g., RGB, HSV).
  - Image filtering: Convolution operations for enhancing or extracting features.

- Main Modules:
  - Core module: Consists of basic data structures and functions.
  - Imgproc module: Filtering and transformations.
  - Objdetect module: Object detection algorithms.
  - HighGUI module: Tools for GUIs and image display.

- Algorithms and Techniques:
  - Filtering: Gaussian blur, median filtering.
  - Edge detection: Canny and Sobel operators.
  - Contour detection: Identifying shape boundaries.

- Performance Optimization:
  - Multithreading: Utilizes multiple CPU cores.
  - Hardware acceleration: Supports CUDA and OpenCL for GPU processing.

-It is compatible with Windows, macOS, Linux, Android, and iOS and is versatile for developers and researchers in computer vision.

Code and Output:

CODE:

```
import cv2
import numpy as np
from tkinter import *
from tkinter import filedialog, messagebox, colorchooser, Menu
from PIL import Image, ImageTk, ImageDraw

class ImageEditor:
    def __init__(self, master):
        self.master = master
        self.master.title("Basic Image Editor")
```

```
        self.master.geometry("800x600")

        self.image = None
        self.drawing_canvas = Image.new("RGB", (800, 600), "white")
        self.start_x = self.start_y = None
        self.drawing_shape = "pencil"
        self.brush_color = "black"
        self.brush_size = 3
        self.is_drawing = False
        self.shape_obj = None
        self.saved_filename = None

        self.canvas = Canvas(master, bg='white')
        self.canvas.pack(fill=BOTH, expand=True)

        self.setup_menu()
        self.setup_buttons()

    def setup_menu(self):
        menubar = Menu(self.master)

        file_menu = Menu(menubar, tearoff=0)
        file_menu.add_command(label="New", command=self.new_image)
        file_menu.add_command(label="Load", command=self.load_image)
        file_menu.add_command(label="Save", command=self.save_image)
        menubar.add_cascade(label="File", menu=file_menu)

        tools_menu = Menu(menubar, tearoff=0)
        tools_menu.add_command(label="Pencil", command=lambda:
self.set_shape("pencil"))
        tools_menu.add_command(label="Eraser", command=lambda:
self.set_shape("eraser"))

        shapes_menu = Menu(tools_menu, tearoff=0)
        shapes_menu.add_command(label="Line", command=lambda:
self.set_shape("line"))
        shapes_menu.add_command(label="Rectangle", command=lambda:
self.set_shape("rectangle"))
        shapes_menu.add_command(label="Oval", command=lambda:
self.set_shape("oval"))
```

```
        tools_menu.add_cascade(label="Shapes", menu=shapes_menu)
        menubar.add_cascade(label="Tools", menu=tools_menu)

        self.master.config(menu=menubar)

    def setup_buttons(self):
        button_frame = Frame(self.master)
        button_frame.pack(pady=10)

        Label(button_frame, text="Brush Size:").pack(side=LEFT, padx=5)
        self.brush_size_scale = Scale(button_frame, from_=1, to=20,
orient=HORIZONTAL, command=self.set_brush_size)
        self.brush_size_scale.set(3)
        self.brush_size_scale.pack(side=LEFT, padx=5)

        color_button = Button(button_frame, text="Select Color",
command=self.choose_color, width=10)
        color_button.pack(side=LEFT, padx=5)

        crop_button = Button(button_frame, text="Crop",
command=self.start_crop, width=10)
        crop_button.pack(side=LEFT, padx=5)

    def new_image(self):
        self.image = None
        self.drawing_canvas = Image.new("RGB", (800, 600), "white")
        self.canvas.delete("all")
        self.saved_filename = None

    def load_image(self):
        filename = filedialog.askopenfilename()
        if filename:
            self.image = cv2.imread(filename)
            self.display_image(self.image)
            self.saved_filename = filename
    def display_image(self, img):
        img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        img_tk = ImageTk.PhotoImage(Image.fromarray(img_rgb))
        self.canvas.create_image(0, 0, anchor=NW, image=img_tk)
```

```
        self.canvas.image = img_tk

    def choose_color(self):
        if (color := colorchooser.askcolor()[1]):
            self.brush_color = color

    def set_brush_size(self, size):
        self.brush_size = int(size)

    def set_shape(self, shape):
        self.drawing_shape = shape

        self.canvas.unbind("<B1-Motion>")
        self.canvas.unbind("<ButtonPress-1>")
        self.canvas.unbind("<ButtonRelease-1>")

        if shape in ["pencil", "eraser"]:
            self.canvas.bind("<B1-Motion>", self.draw)
            self.canvas.bind("<ButtonPress-1>", self.start_drawing)
            self.canvas.bind("<ButtonRelease-1>", self.stop_drawing)
        else:
            self.canvas.bind("<ButtonPress-1>", self.start_shape)
            self.canvas.bind("<B1-Motion>", self.draw_temp_shape)
            self.canvas.bind("<ButtonRelease-1>", self.finish_shape)

    def start_drawing(self, event):
        self.is_drawing = True
        self.last_x, self.last_y = event.x, event.y

    def stop_drawing(self, event):
        self.is_drawing = False

    def draw(self, event):
        if self.is_drawing:
            if self.drawing_shape == "pencil":
                self.canvas.create_line(self.last_x, self.last_y, event.x,
event.y,
                                        fill=self.brush_color,
width=self.brush_size, capstyle=ROUND, smooth=True)
```

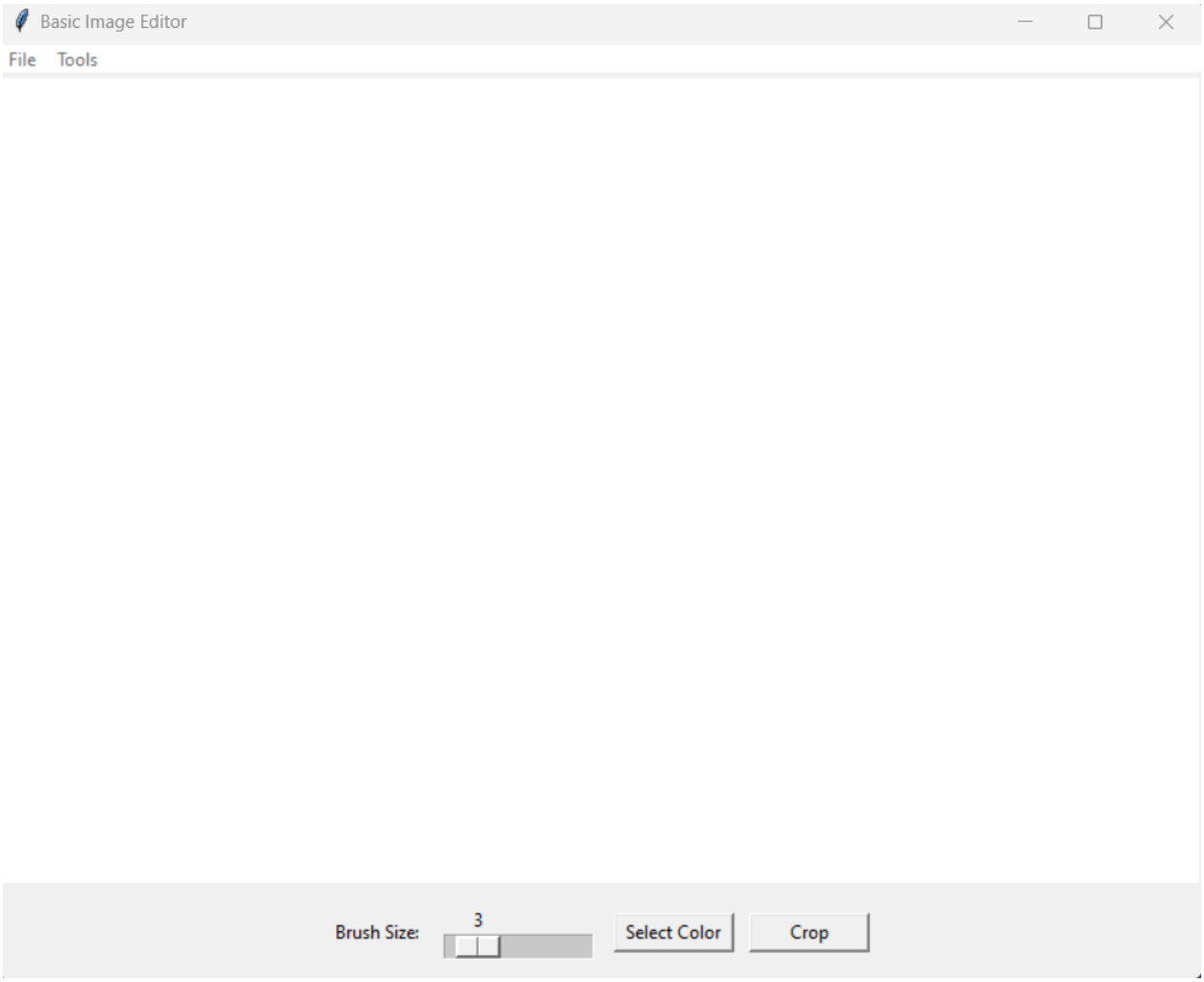
```
                self.draw_on_image(self.last_x, self.last_y, event.x,
event.y)
            elif self.drawing_shape == "eraser":
                self.canvas.create_line(self.last_x, self.last_y, event.x,
event.y,
                                        fill='white',
width=self.brush_size, capstyle=ROUND, smooth=True)
                self.draw_on_image(self.last_x, self.last_y, event.x,
event.y, erase=True)
            self.last_x, self.last_y = event.x, event.y

    def start_shape(self, event):
        self.start_x, self.start_y = event.x, event.y
        self.shape_obj = None

    def draw_temp_shape(self, event):
        self.canvas.delete("temp")
        if self.drawing_shape == "line":
            self.shape_obj = self.canvas.create_line(self.start_x,
self.start_y, event.x, event.y,

fill=self.brush_color, width=self.brush_size, tags="temp")
        elif self.drawing_shape == "rectangle":
            self.shape_obj = self.canvas.create_rectangle(self.start_x,
self.start_y, event.x, event.y,

outline=self.brush_color, width=self.brush_size, tags="temp")
        elif self.drawing_shape == "oval":
            self.shape_obj = self.canvas.create_oval(self.start_x,
self.start_y, event.x, event.y,

outline=self.brush_color, width=self.brush_size, tags="temp")

    def finish_shape(self, event):
        self.canvas.delete("temp")
        if self.drawing_shape == "line":
            self.shape_obj = self.canvas.create_line(self.start_x,
self.start_y, event.x, event.y,

fill=self.brush_color, width=self.brush_size)
```

```
        elif self.drawing_shape == "rectangle":
            self.shape_obj = self.canvas.create_rectangle(self.start_x,
self.start_y, event.x, event.y,

outline=self.brush_color, width=self.brush_size)
        elif self.drawing_shape == "oval":
            self.shape_obj = self.canvas.create_oval(self.start_x,
self.start_y, event.x, event.y,

outline=self.brush_color, width=self.brush_size)

        self.draw_on_image(self.start_x, self.start_y, event.x, event.y)

    def draw_on_image(self, x1, y1, x2=None, y2=None, erase=False):
        draw = ImageDraw.Draw(self.drawing_canvas)
        if x2 is None and y2 is None:
            r = self.brush_size // 2
            if erase:
                draw.ellipse([x1 - r, y1 - r, x1 + r, y1 + r],
fill="white")
            else:
                draw.ellipse([x1 - r, y1 - r, x1 + r, y1 + r],
fill=self.brush_color, outline=self.brush_color)
        else:
            if self.drawing_shape == "line":
                if erase:
                    draw.line([x1, y1, x2, y2], fill="white",
width=self.brush_size)
                else:
                    draw.line([x1, y1, x2, y2], fill=self.brush_color,
width=self.brush_size)
            elif self.drawing_shape == "rectangle":
                if erase:
                    draw.rectangle([x1, y1, x2, y2], fill="white")
                else:
                    draw.rectangle([x1, y1, x2, y2],
outline=self.brush_color, width=self.brush_size)
            elif self.drawing_shape == "oval":
                if erase:
                    draw.ellipse([x1, y1, x2, y2], fill="white")
```

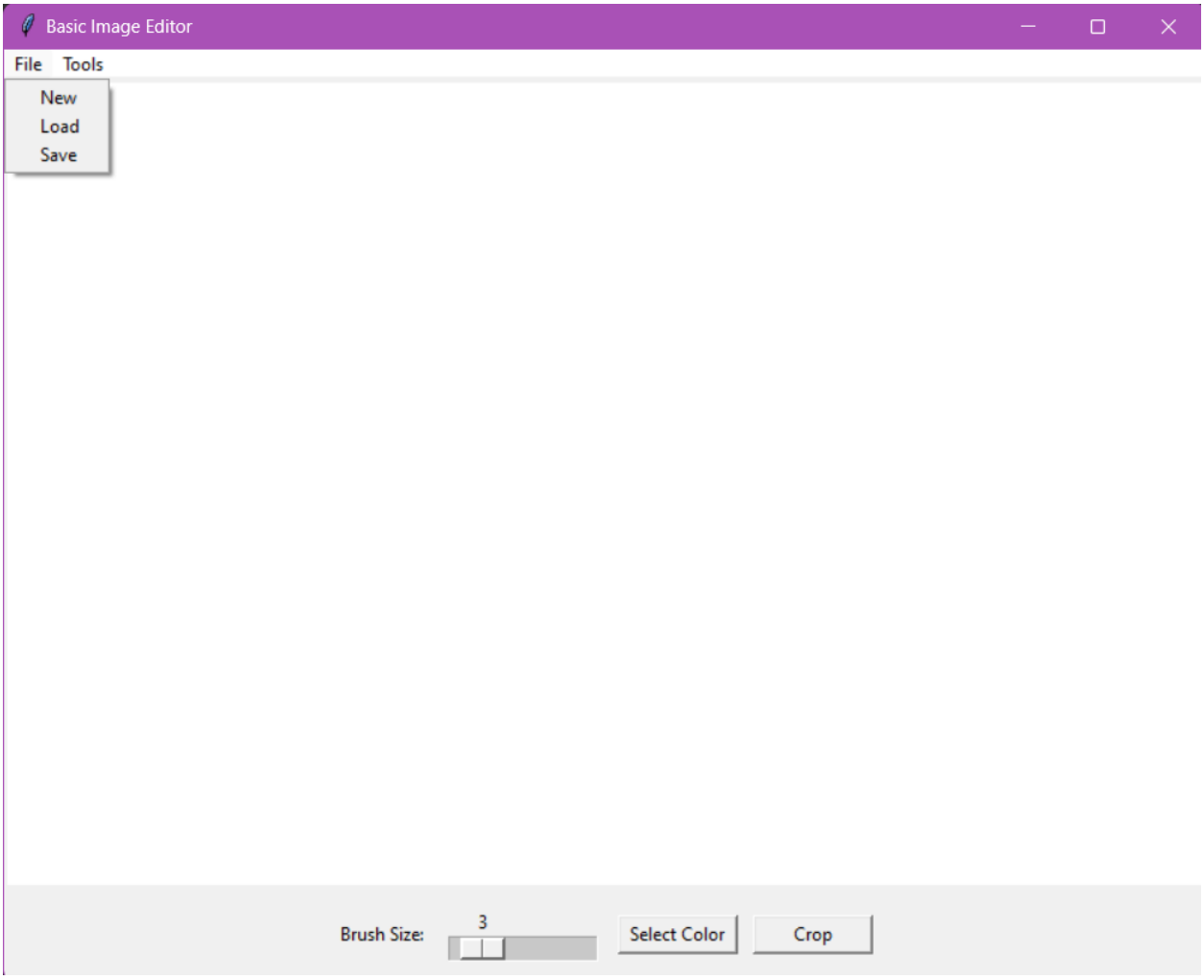
```
                else:
                    draw.ellipse([x1, y1, x2, y2],
outline=self.brush_color, width=self.brush_size)

    def start_crop(self):
        self.canvas.bind("<ButtonPress-1>", self.start_crop_rectangle)
        self.canvas.bind("<B1-Motion>", self.draw_crop_rectangle)
        self.canvas.bind("<ButtonRelease-1>", self.finish_crop)

    def start_crop_rectangle(self, event):
        self.start_x, self.start_y = event.x, event.y
        self.crop_rectangle = self.canvas.create_rectangle(self.start_x,
self.start_y, self.start_x, self.start_y,
                                                           outline='red',
width=2)

    def draw_crop_rectangle(self, event):
        self.canvas.coords(self.crop_rectangle, self.start_x,
self.start_y, event.x, event.y)

    def finish_crop(self, event):
        self.canvas.unbind("<ButtonPress-1>")
        self.canvas.unbind("<B1-Motion>")
        self.canvas.unbind("<ButtonRelease-1>")

        end_x, end_y = event.x, event.y
        crop_box = (min(self.start_x, end_x), min(self.start_y, end_y),
                    max(self.start_x, end_x), max(self.start_y, end_y))

        self.crop_image(crop_box)
        self.canvas.delete(self.crop_rectangle)

    def crop_image(self, crop_box):
        if self.image is not None:
            img_pil = Image.fromarray(self.image)

            # Crop the image using the bounding box
            cropped_img = img_pil.crop(crop_box)

            # Display the cropped image back in the canvas
```

```
            self.drawing_canvas = Image.new("RGB", (800, 600), "white")  # Reset drawing canvas
            self.display_image(np.array(cropped_img))  # Show cropped image
            self.image = np.array(cropped_img)  # Update the main image to the cropped one
            self.saved_filename = None

    def save_image(self):
        if self.image is not None:
            combined_image = Image.new("RGB", (800, 600))
            combined_image.paste(Image.fromarray(self.image), (0, 0))

            # Create a new image with an alpha channel to combine the drawings
            drawing_with_alpha = self.drawing_canvas.convert("RGBA")

            # Create a mask for the drawing
            mask = Image.new("L", drawing_with_alpha.size, 255)  # White mask
            combined_image.paste(drawing_with_alpha, (0, 0), mask)

            file_path = filedialog.asksaveasfilename(defaultextension=".png",
                                                     filetypes=[("PNG Files", "*.png"), ("JPEG Files", "*.jpg")])
            if file_path:
                combined_image.save(file_path)

if __name__ == "__main__":
    root = Tk()
    app = ImageEditor(root)
    root.mainloop()
```

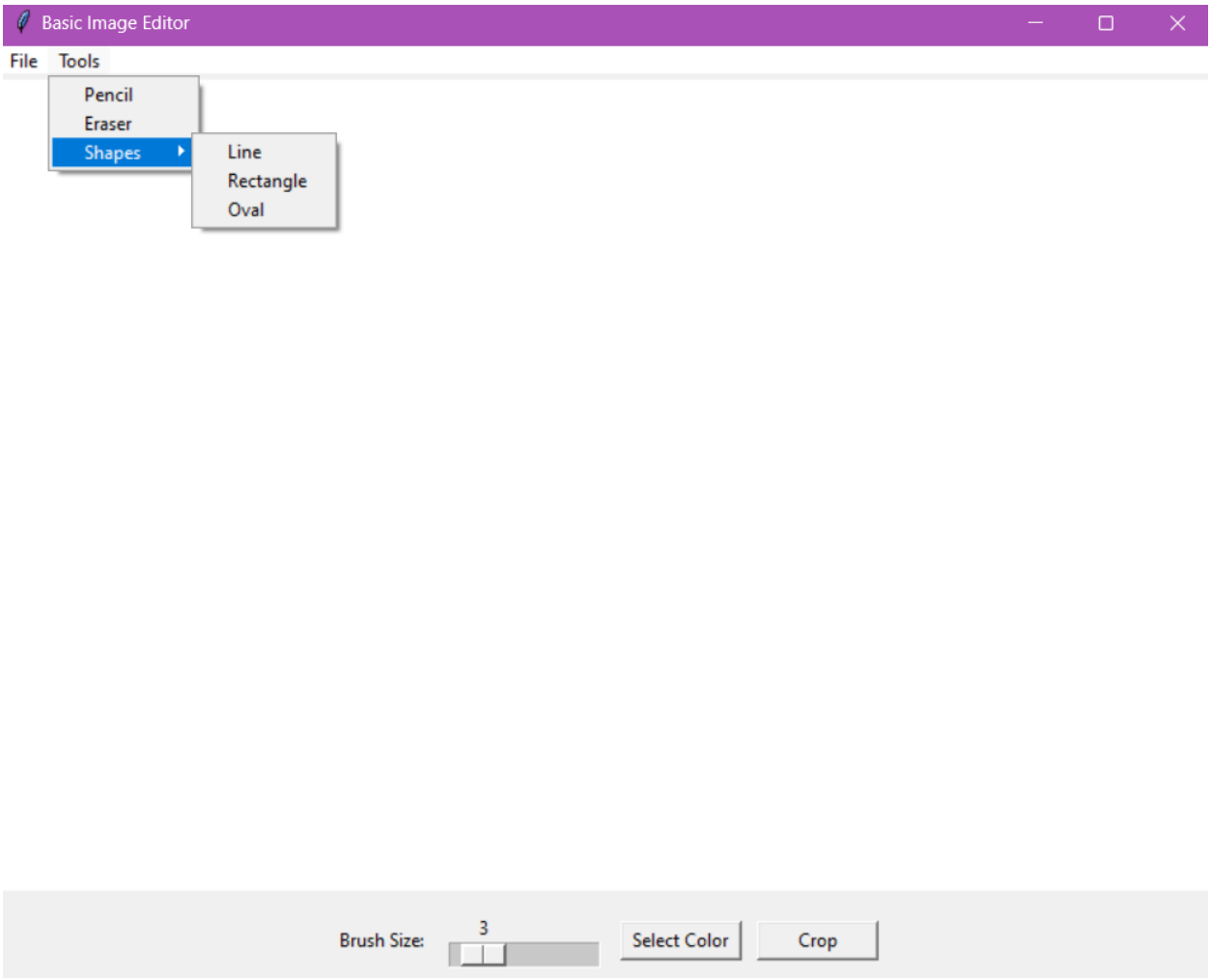
OUTPUT:



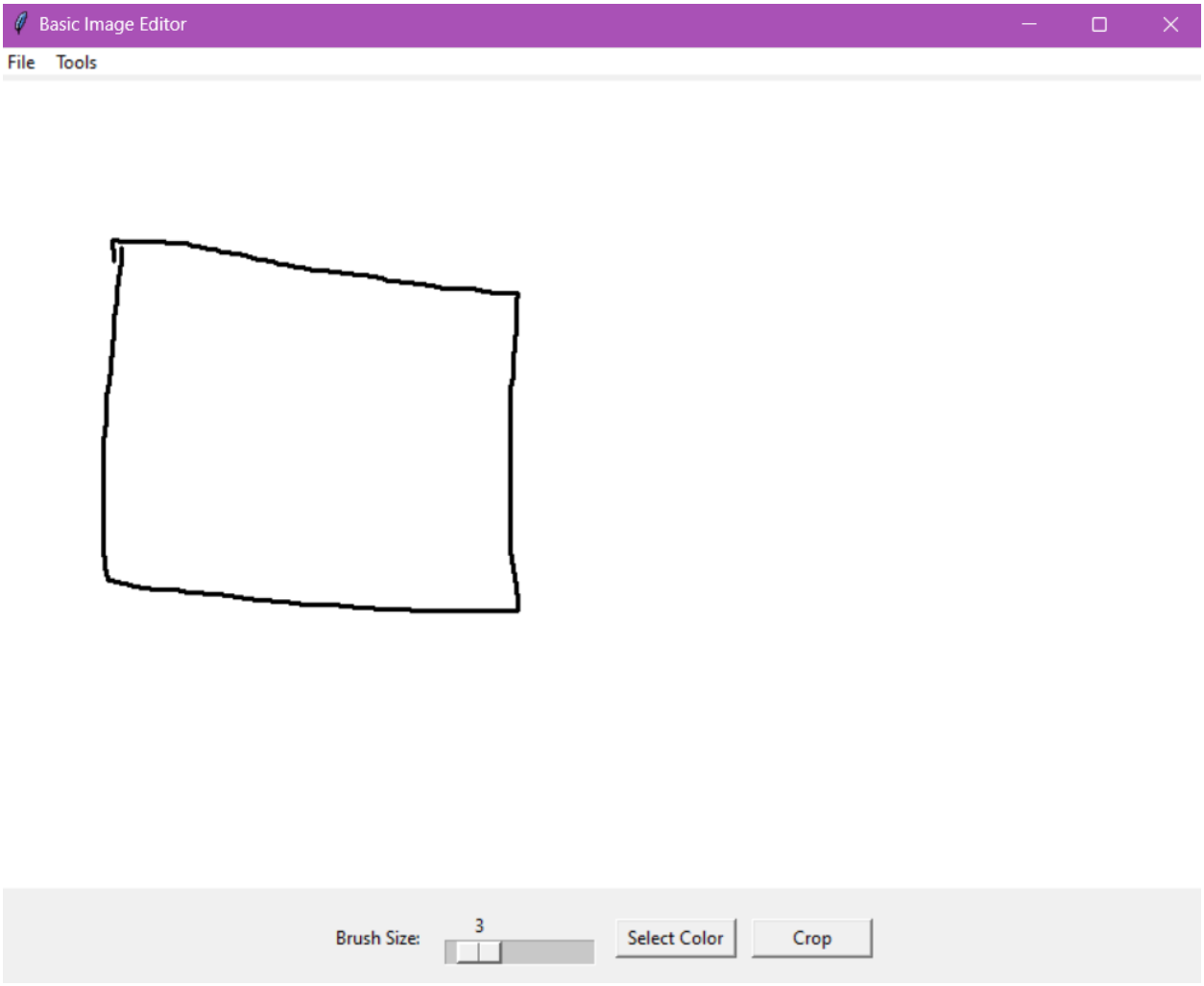
What you see when you run the program



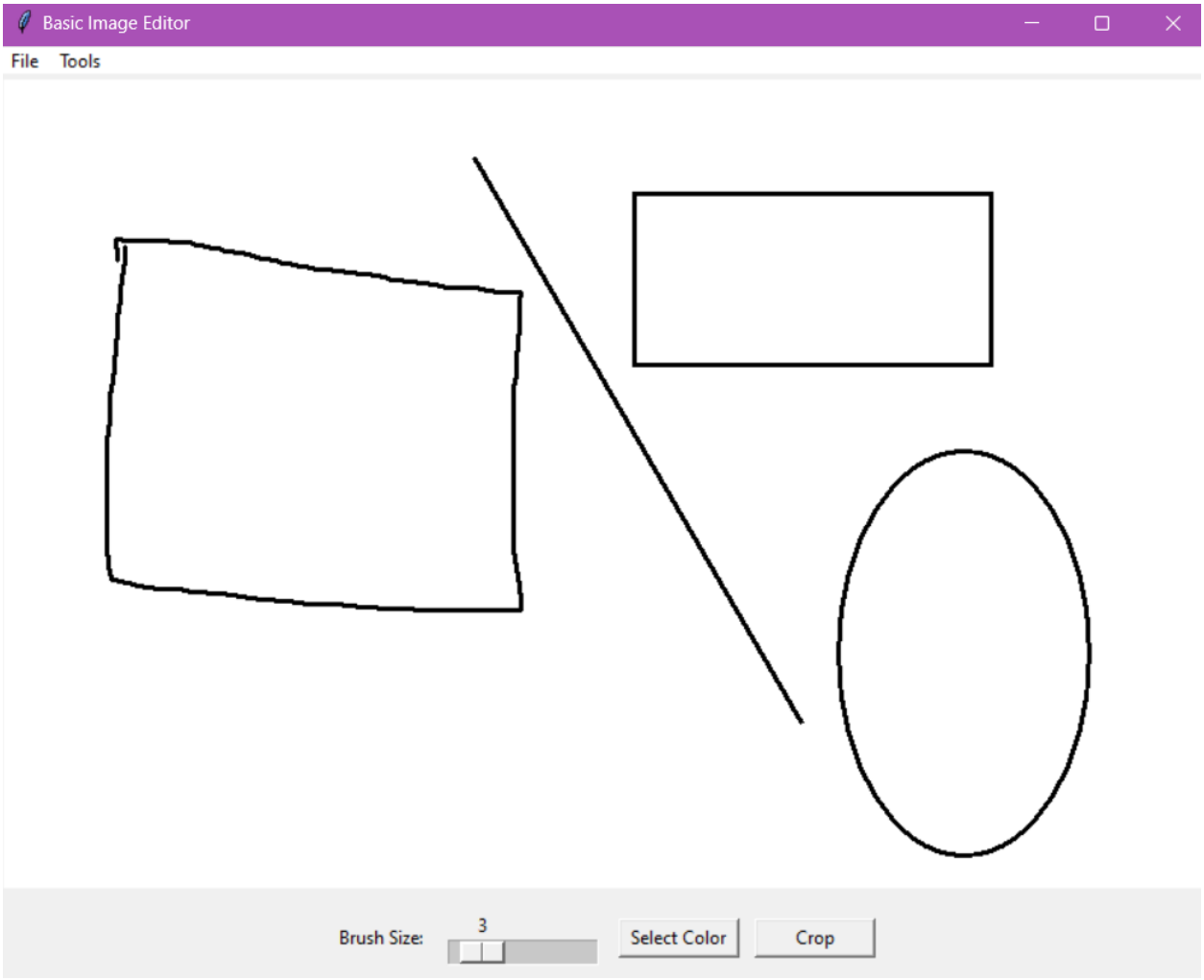
When you click File



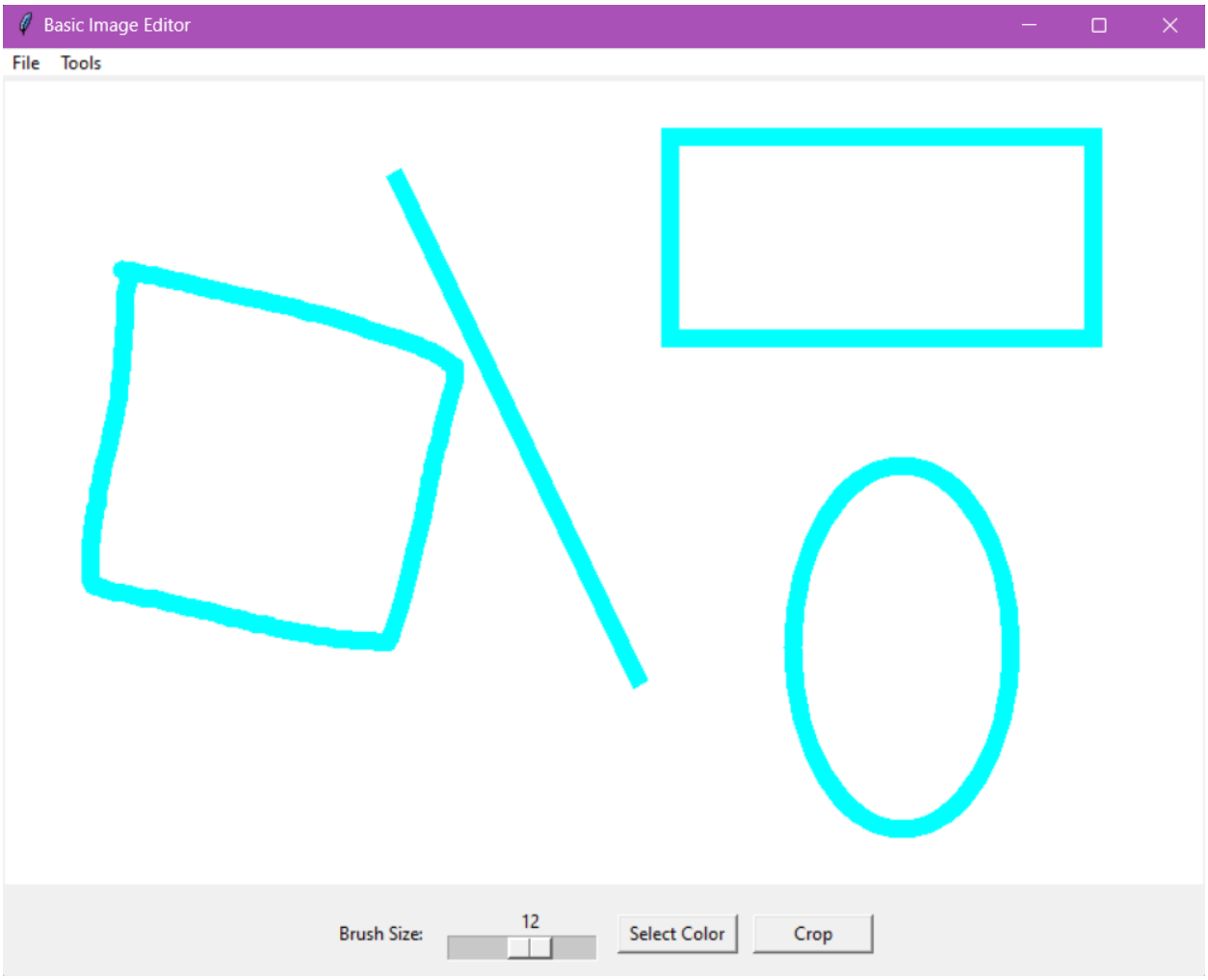
When you click Tools and hover over Shapes



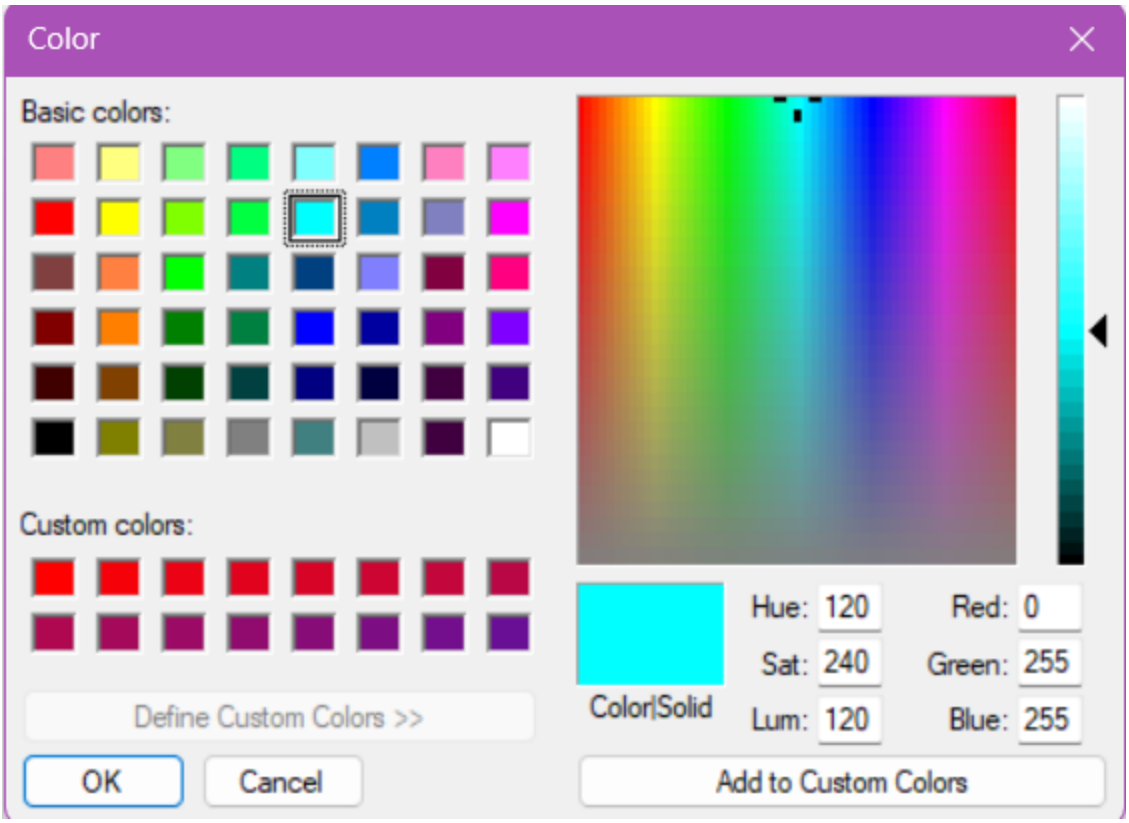
When you draw with pencil



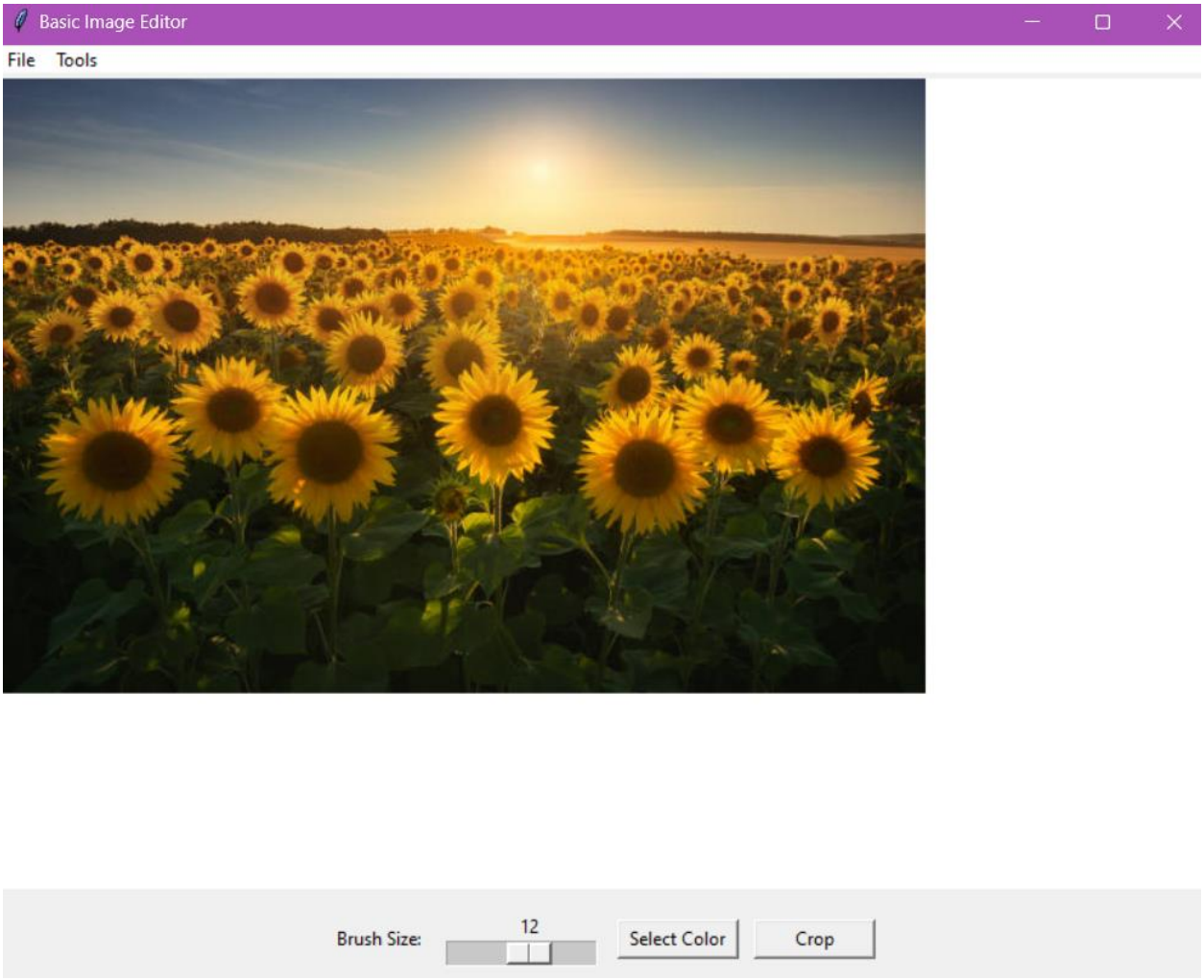
When you draw a line, a rectangle and an oval



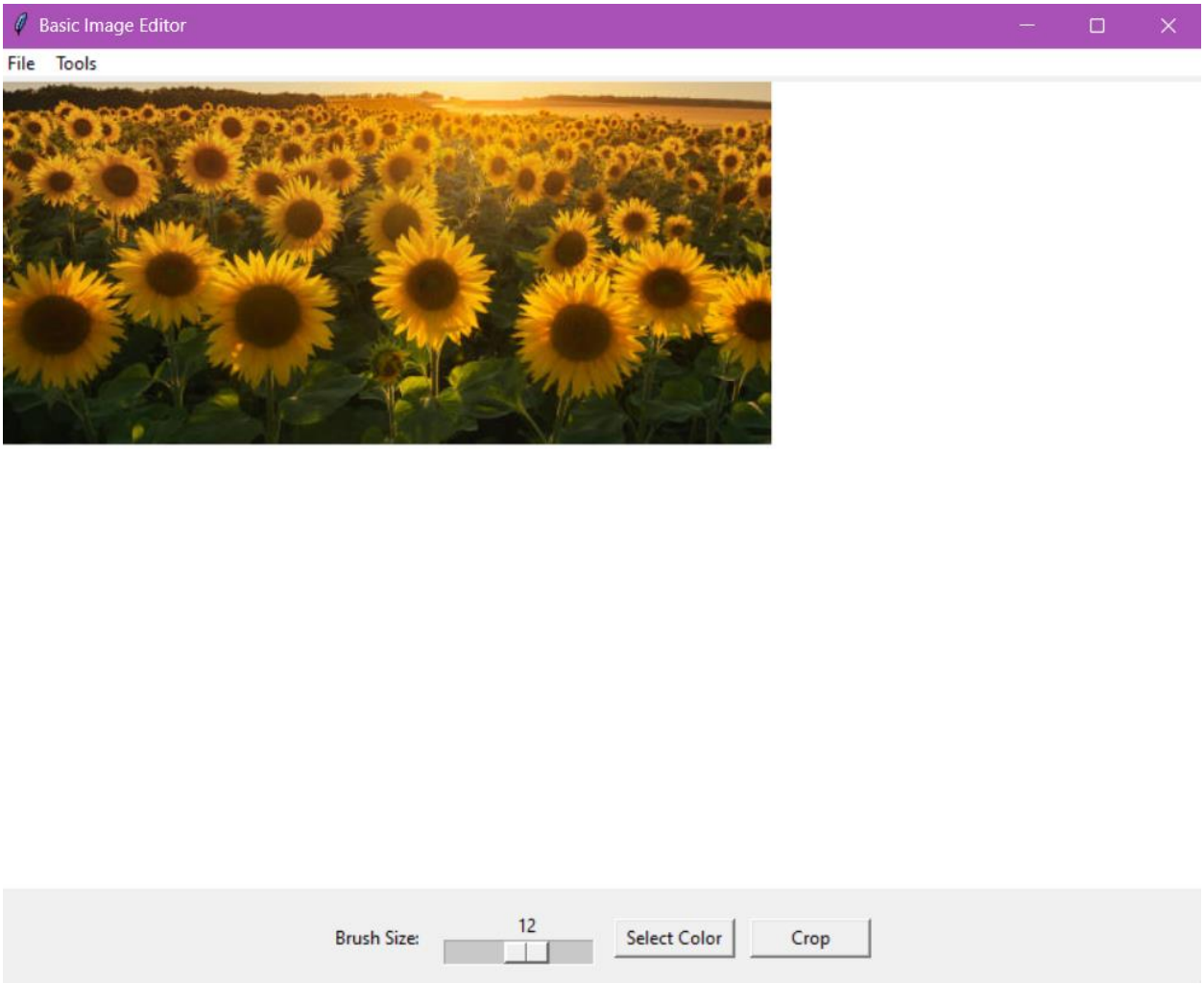
When you increase brush size and change color and redraw with all the brushes and the shapes



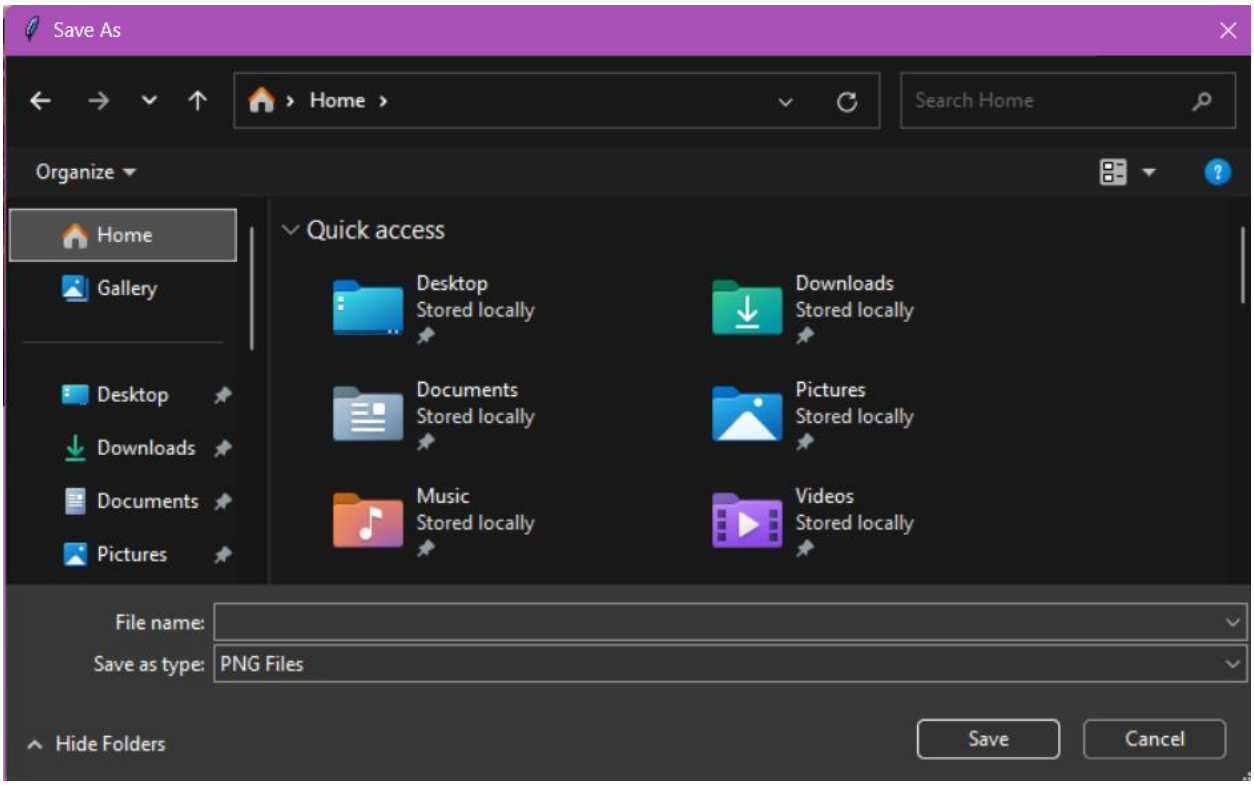
The color menu



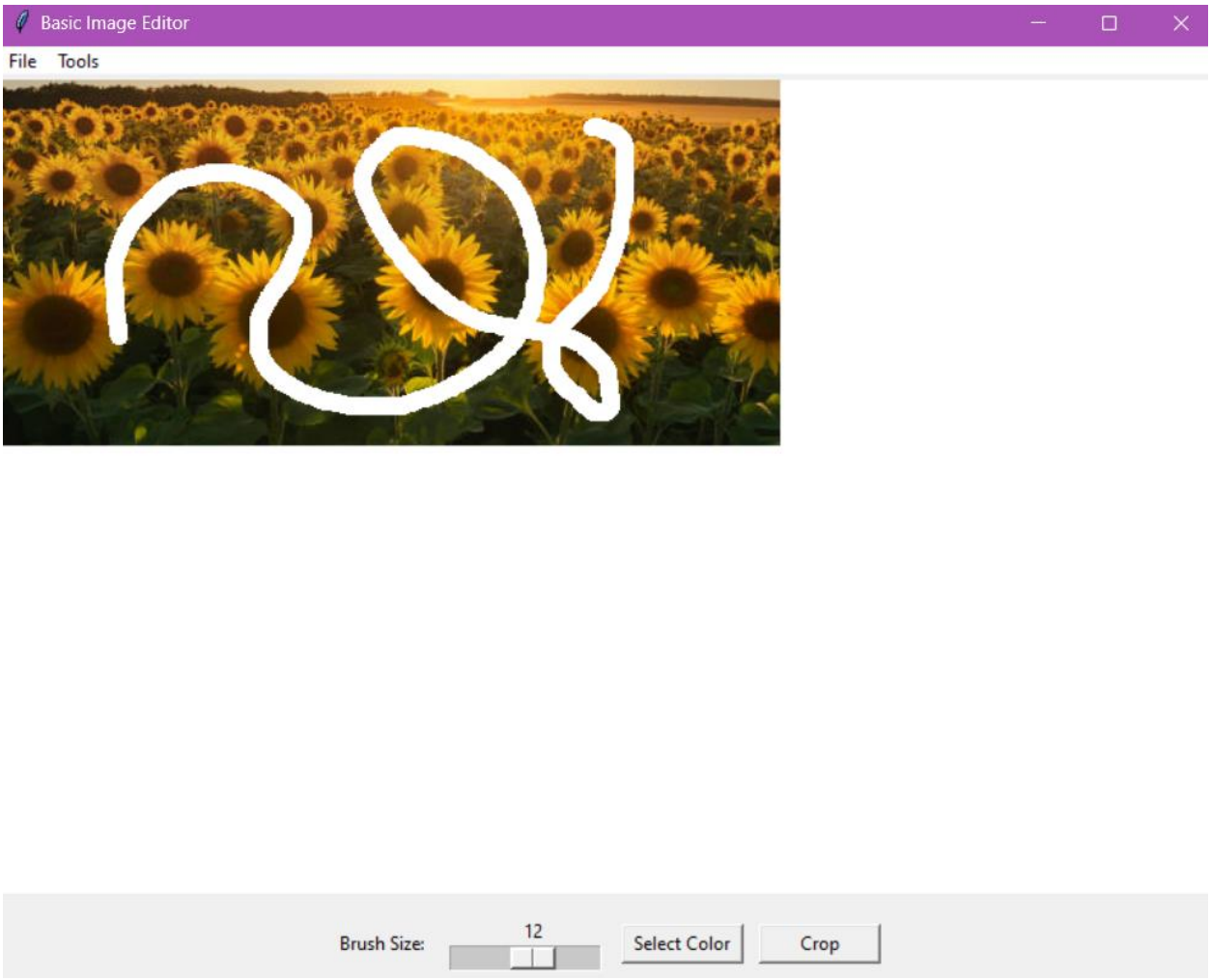
When you load an image



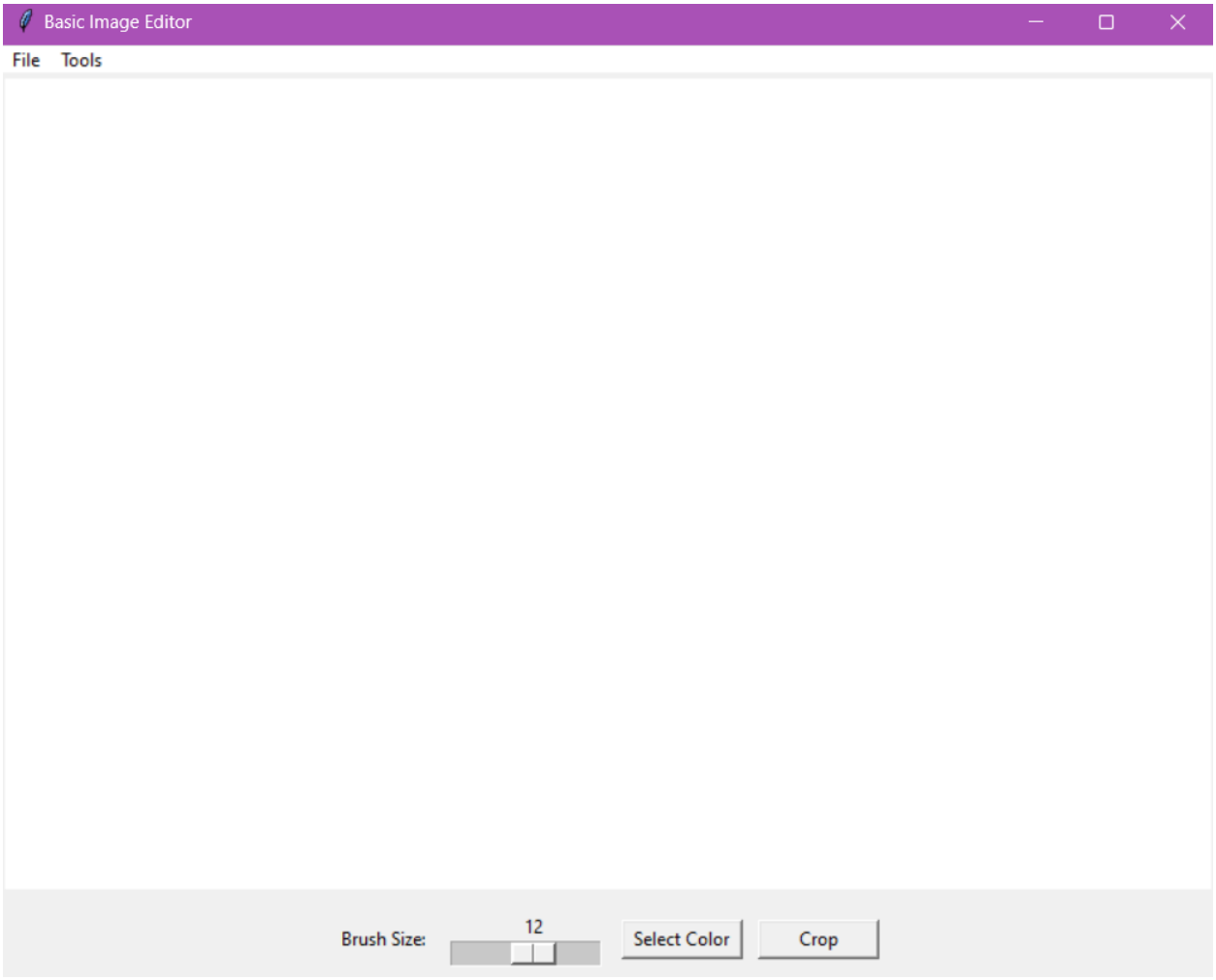
After cropping the image



When you click Save



When you use the Eraser



When you click New

Conclusion:

Thus we have written a program to make an image editor using opencv in python with the help of tkinter.

The Image Editor has the basic functions of Making a new blank canvas, Loading in any image from your pc, Saving the new image, Drawing with the help of a pencil or shapes such as Lines, Rectangles and Ovals, We also have an Eraser Feature.

We can also increase the size of the brush which changes the thickness of the pencil, and the multiple different shapes.
We can also crop loaded images how we see fit.