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)

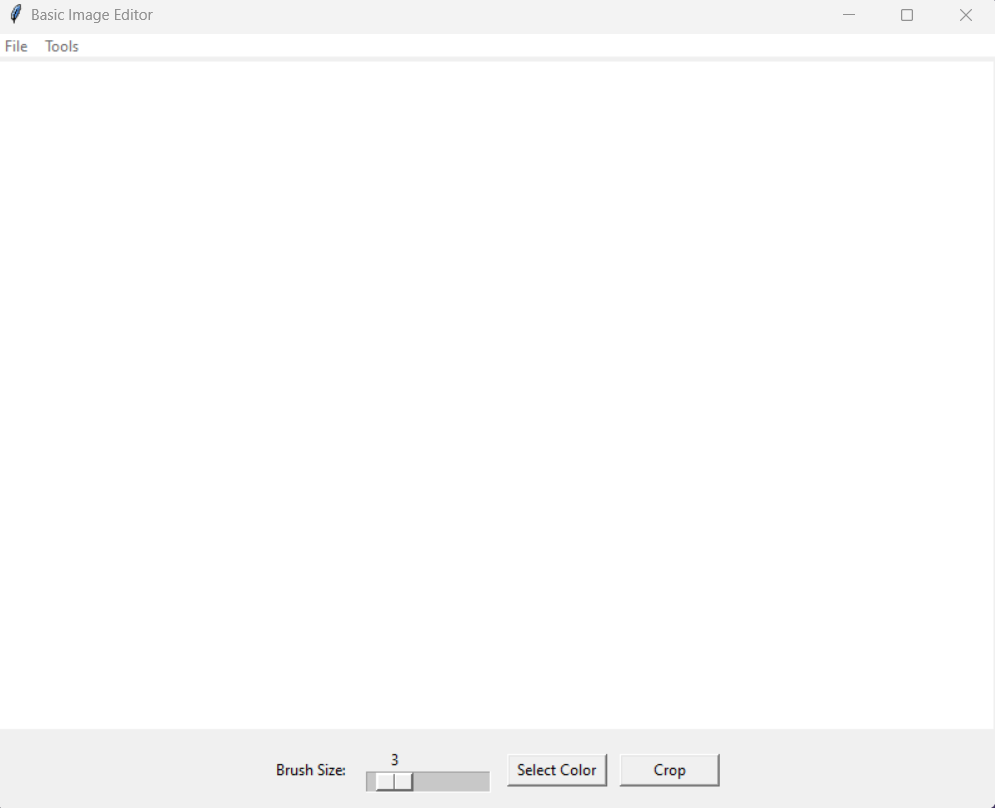
if \_\_name\_\_ == "\_\_main\_\_":

root = Tk()

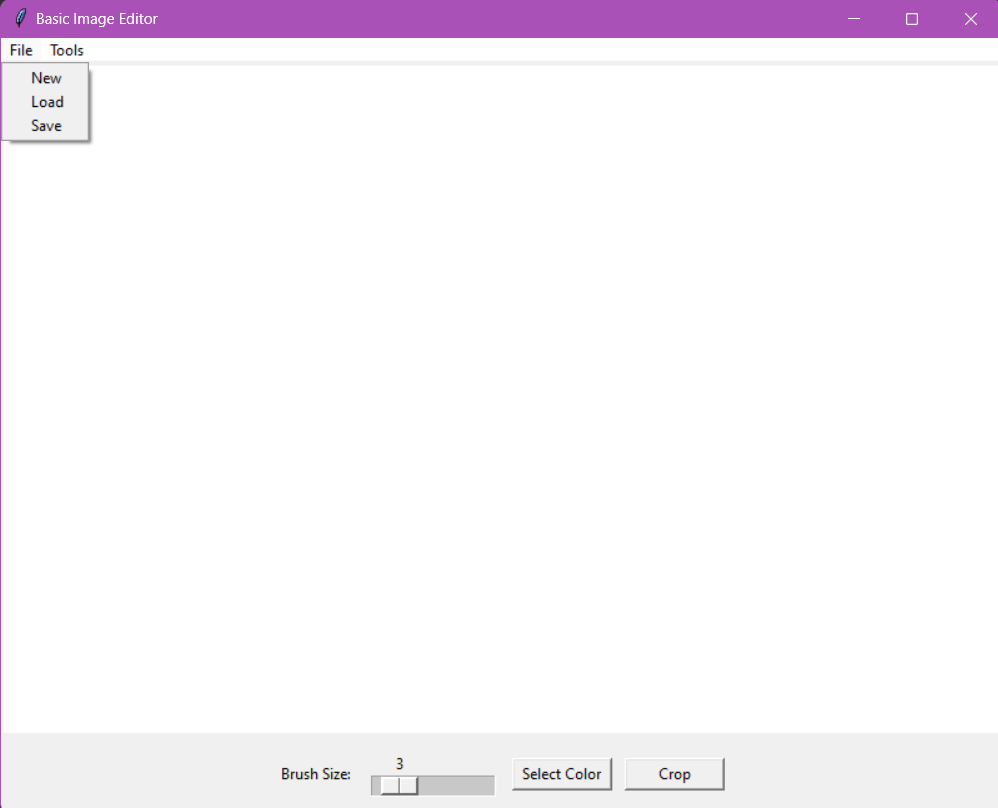
app = ImageEditor(root)

root.mainloop()

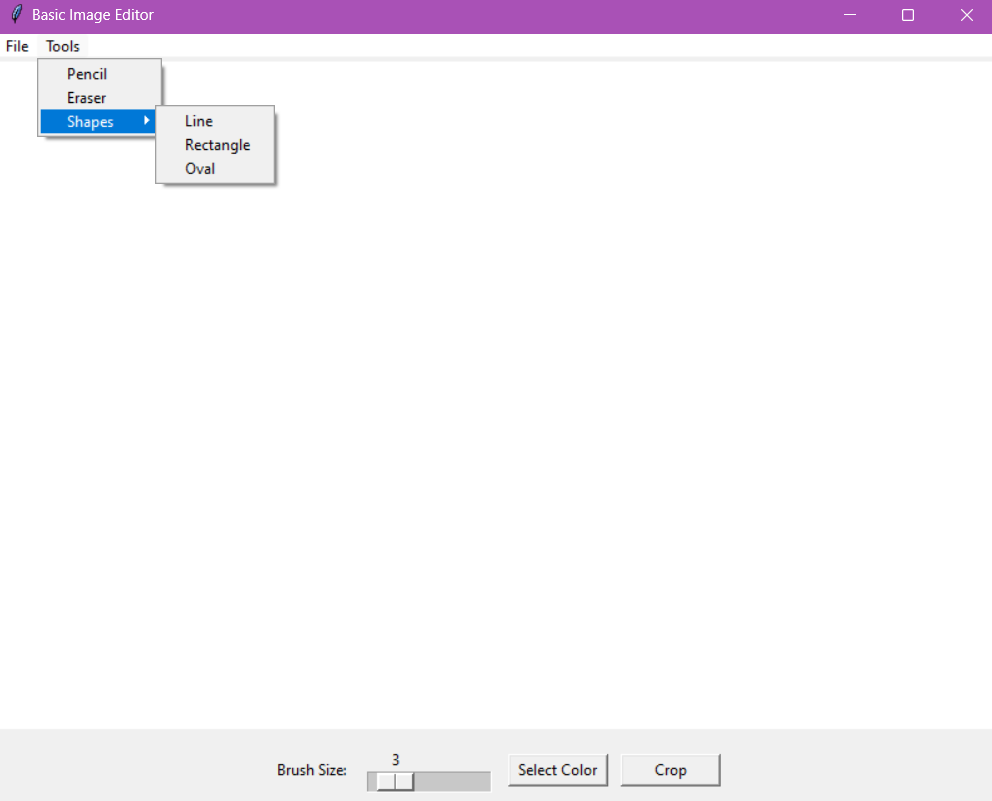
OUTPUT:



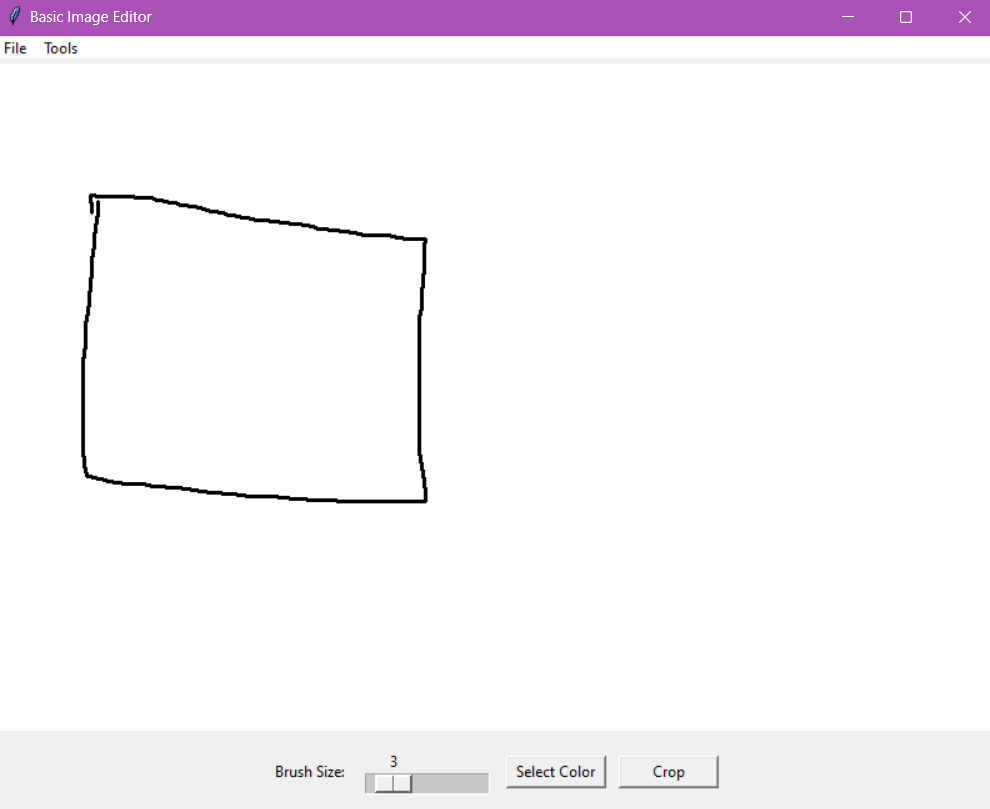
What you see when you run the program

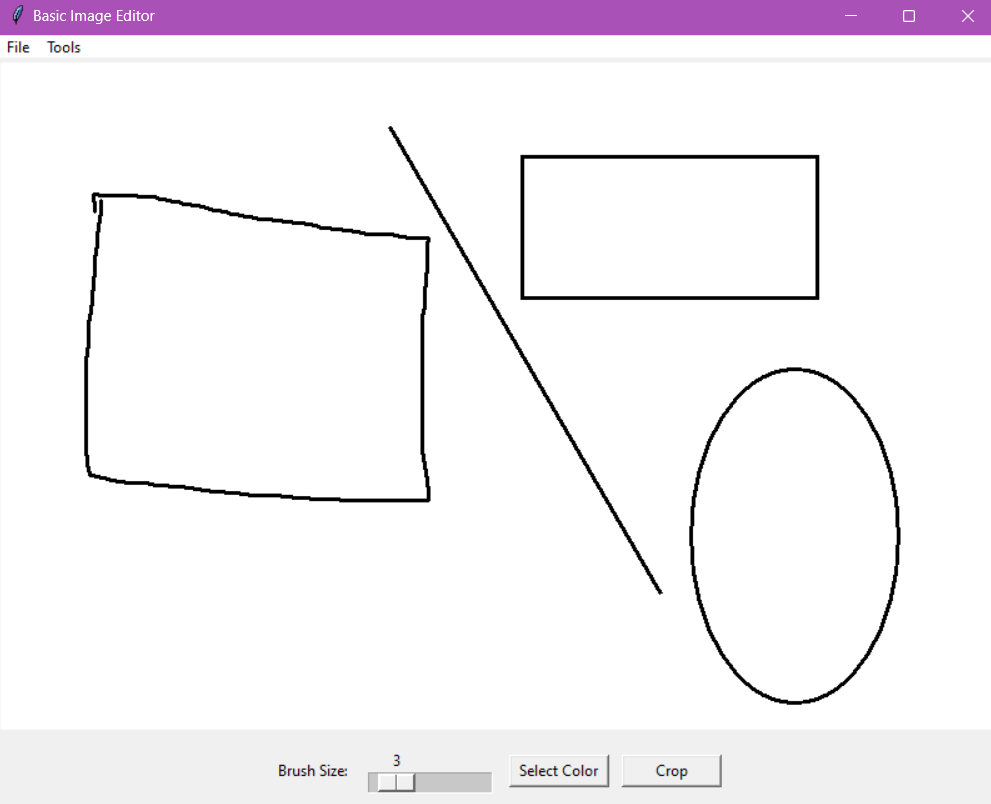


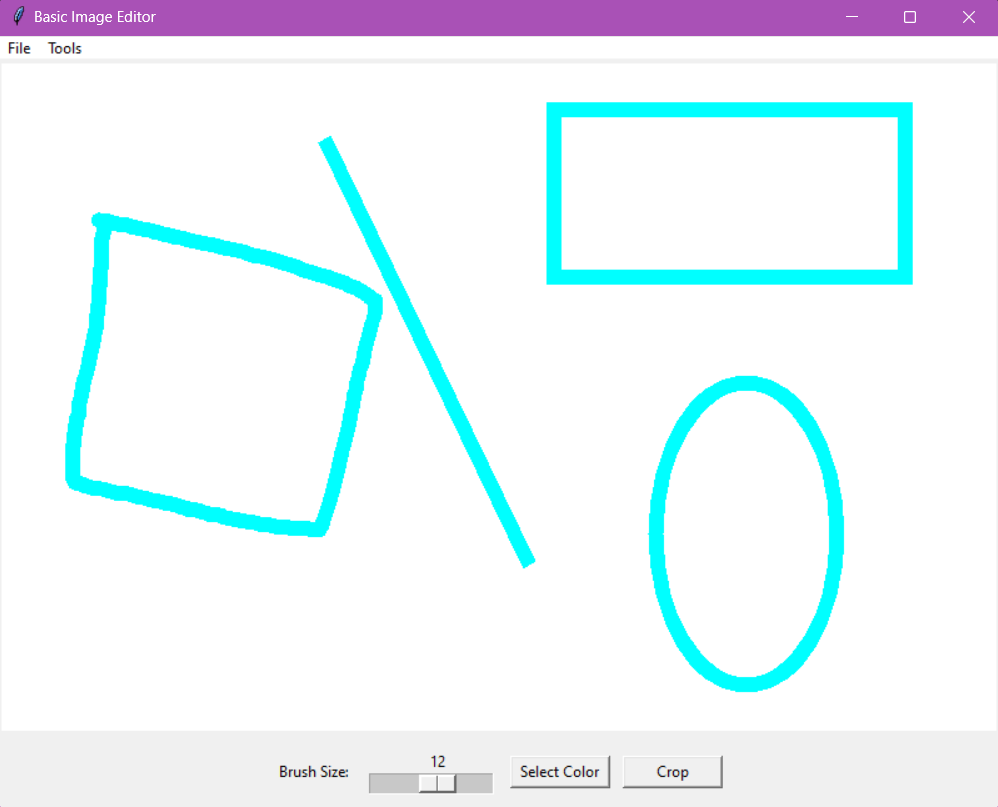
When you click File

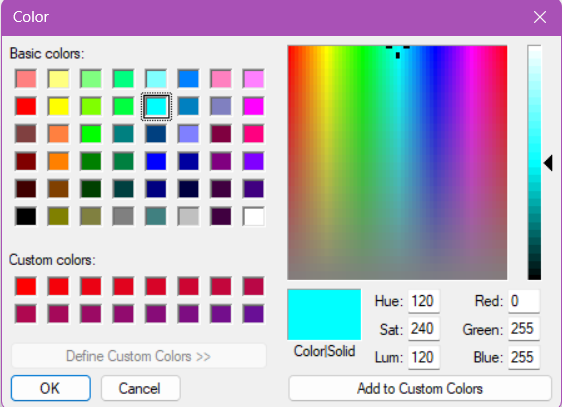


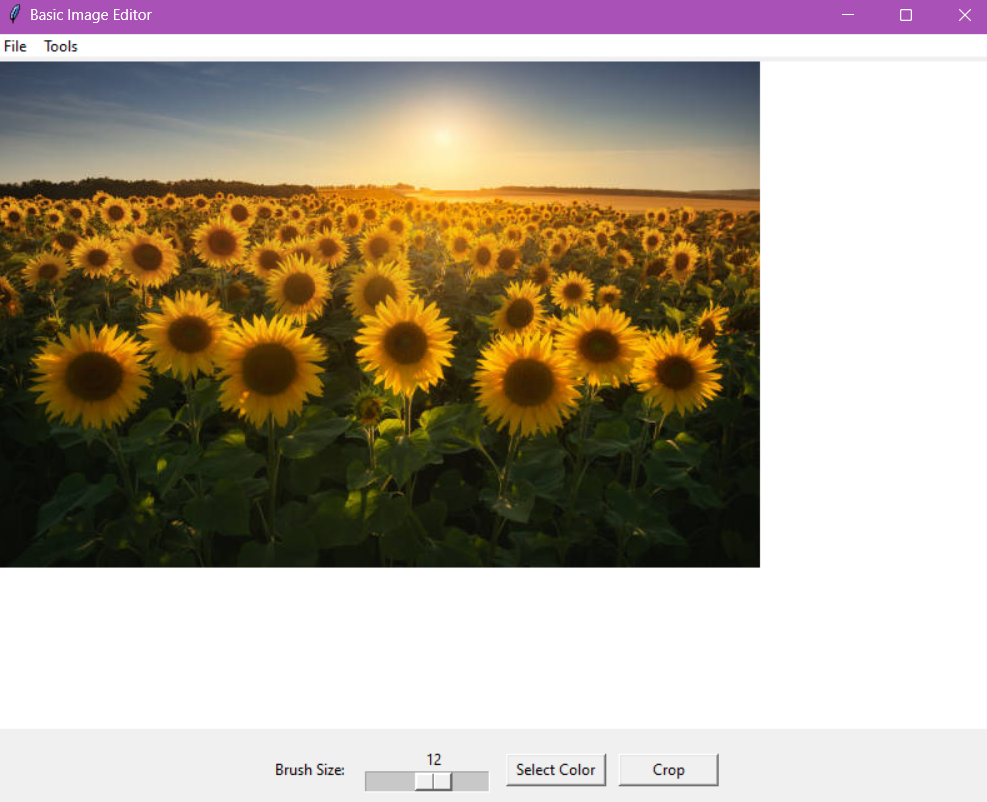
When you click Tools and hover over Shapes

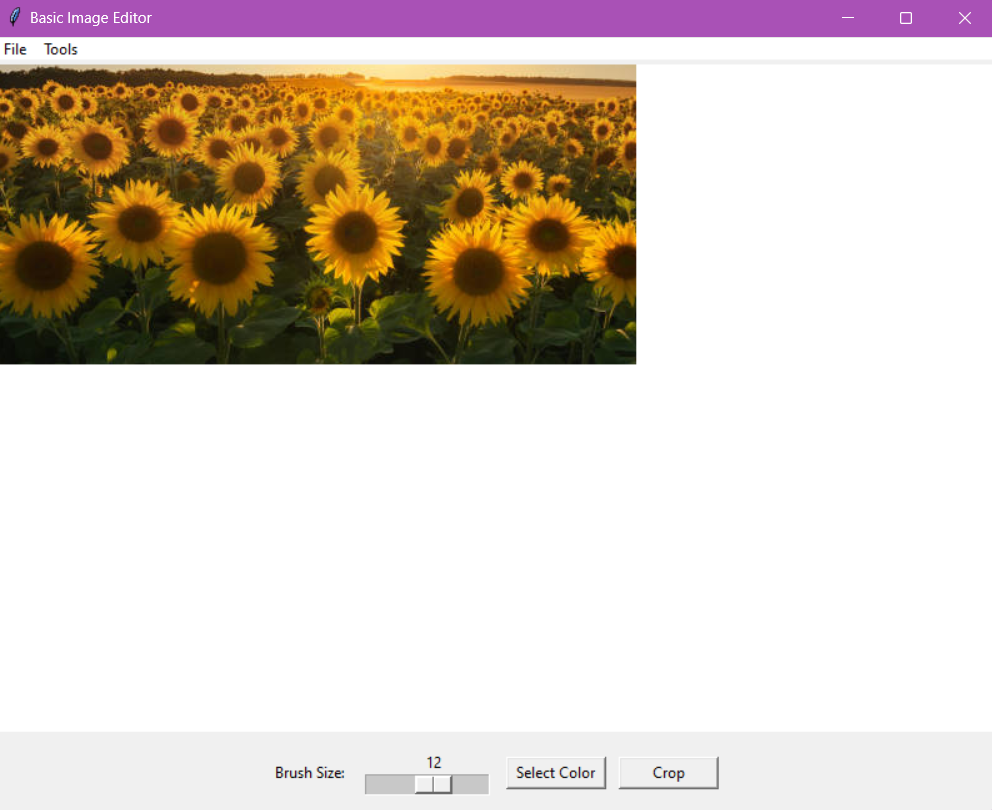
  
When you draw with pencil

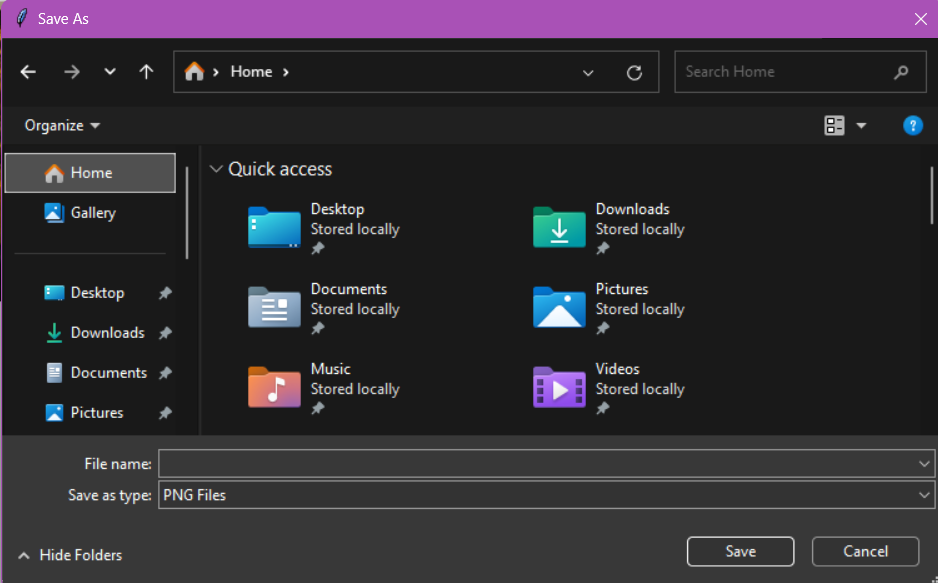
  
When you draw a line, a rectangle and an oval

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

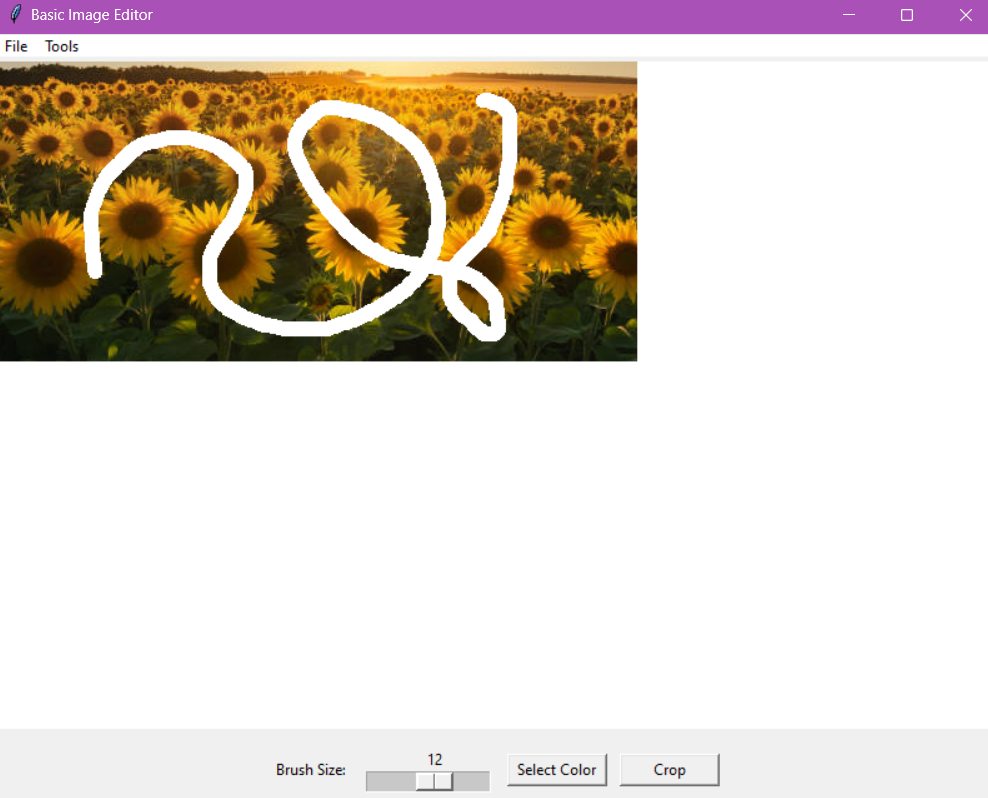
  
The color menu

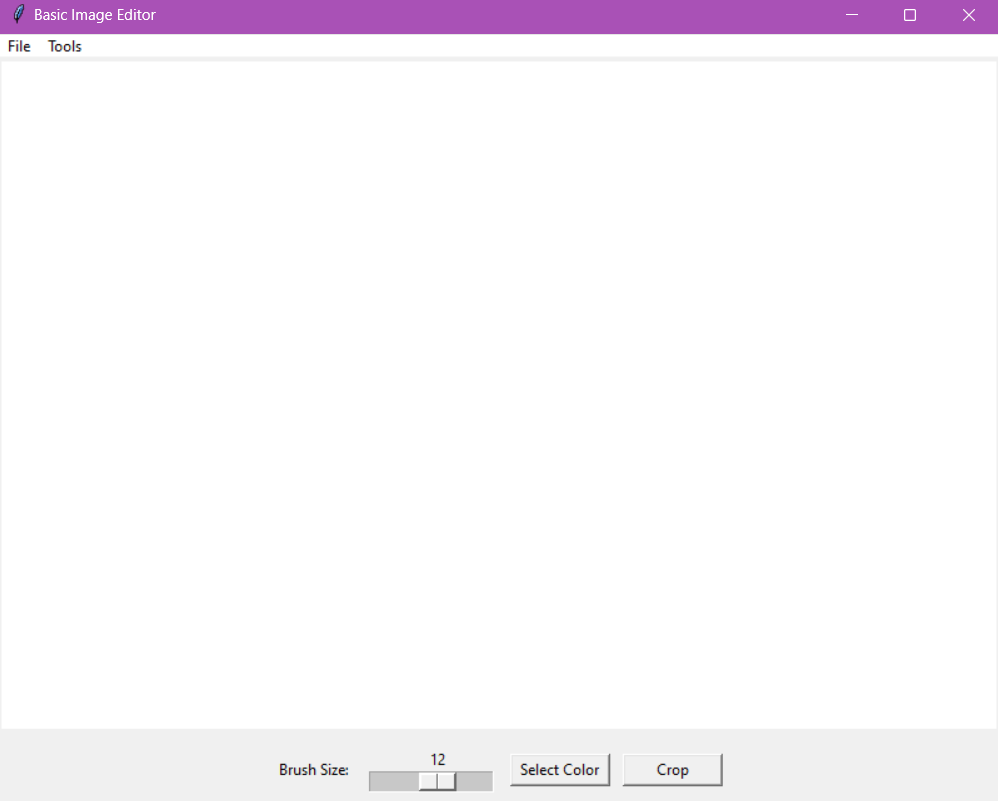
  
When you load an image

  
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.