Utsav Maloo

AIM: - MINI PROJECT: - [Virtual Paint Box]

❖ OBJECTIVE: -

In this project, we are going to create a virtual painter using AI. We will first track our hand and get its landmarks and then use the points to draw on the screen. We will use two fingers for selection and one finger for drawing. And the best part is that all of this will be done in real-time. An application that enables one to virtually paint in the air using their fingers. It is developed in python using OpenCV and MediaPipe

❖ Tech Stacks:

I built it with Python and using OpenCV Module.

1. OpenCV (for image processing and drawing)
2. Mediapipe (for Hand Tracking)

❖ Features:

a) Can draw on your System screen based on your Index finger movement

b) Can track your hand in real-time

❖ Working:

Virtual Painter is a painting app but not a normal painting app it will work on your camera, as soon as you open it a camera will pop then you need to show your hand with all five fingers open use one finger for painting and two fingers for selection of brush and eraser. It is a fun app that children love a lot.

This project is a use case of Hand Tracking technology. MAs soon as the user shows up his hand in the camera the application detects it & draws a bounding box around the hand.

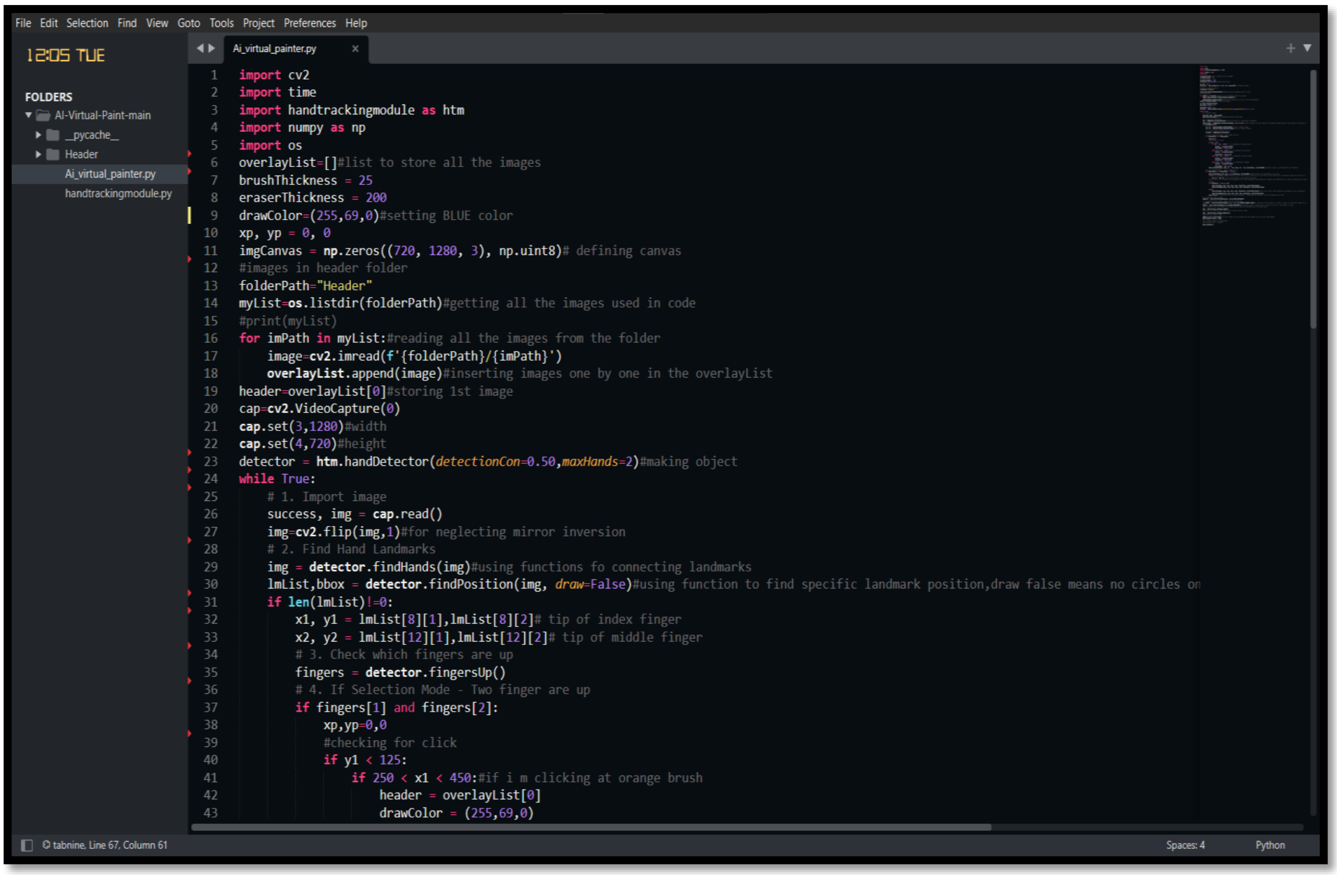
If User shows only Index Finger than he/she is in drawing mode.
To Select different color or eraser from the top of Canvas, User must select it by taking his both Index and Middle finger together at the top of icon.

❖ FUTURE GOAL: -

I did an app which children love and it is a fun app. Draw your imagination by just waiving your finger in air.
It is easy to use and easy to draw.
It is easy explain children about shapes drawing and much more.

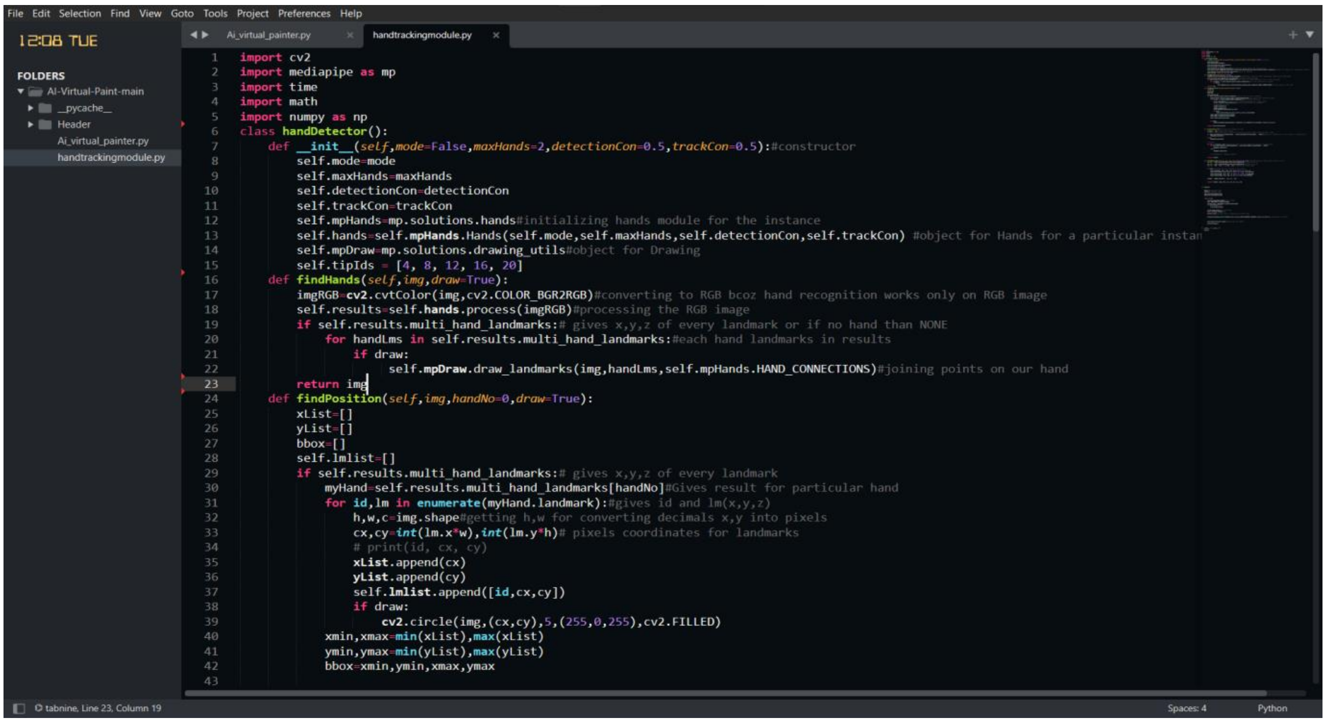
CODE: -

AI_Virtual_Paint.py



```python
import cv2
import time
import handtrackingmodule as htm
import numpy as np
import os
overlayList=[]#list to store all the images
brushThickness = 25
eraserThickness = 200
drawColor=(255,69,0)#setting BLUE color
xp, yp = 0, 0
imgCanvas = np.zeros((720, 1280, 3), np.uint8)# defining canvas
#images in header folder
folderPath="Header"
myList=os.listdir(folderPath)#getting all the images used in code
#print(myList)
for imPath in myList:#reading all the images from the folder
    image=cv2.imread(f'{folderPath}/{imPath}')
    overlayList.append(image)#inserting images one by one in the overlayList
header=overlayList[0]#storing 1st image
cap=cv2.VideoCapture(0)
cap.set(3,1280)#width
cap.set(4,720)#height
detector = htm.handDetector(detectionCon=0.50,maxHands=2)#making object
while True:
    # 1. Import image
    success, img = cap.read()
    img=cv2.flip(img,1)#for neglecting mirror inversion
    # 2. Find Hand Landmarks
    img = detector.findHands(img)#using functions fo connecting landmarks
    lmList,bbox = detector.findPosition(img, draw=False)#using function to find specific landmark position,draw false means no circles on
    if len(lmList)!=0:
        x1, y1 = lmList[8][1],lmList[8][2]# tip of index finger
        x2, y2 = lmList[12][1],lmList[12][2]# tip of middle finger
        # 3. Check which fingers are up
        fingers = detector.fingersUp()
        # 4. If Selection Mode - Two finger are up
        if fingers[1] and fingers[2]:
            xp,yp=0,0
            #checking for click
            if y1 < 125:
                if 250 < x1 < 450:#if i m clicking at orange brush
                    header = overlayList[0]
                    drawColor = (255,69,0)
```

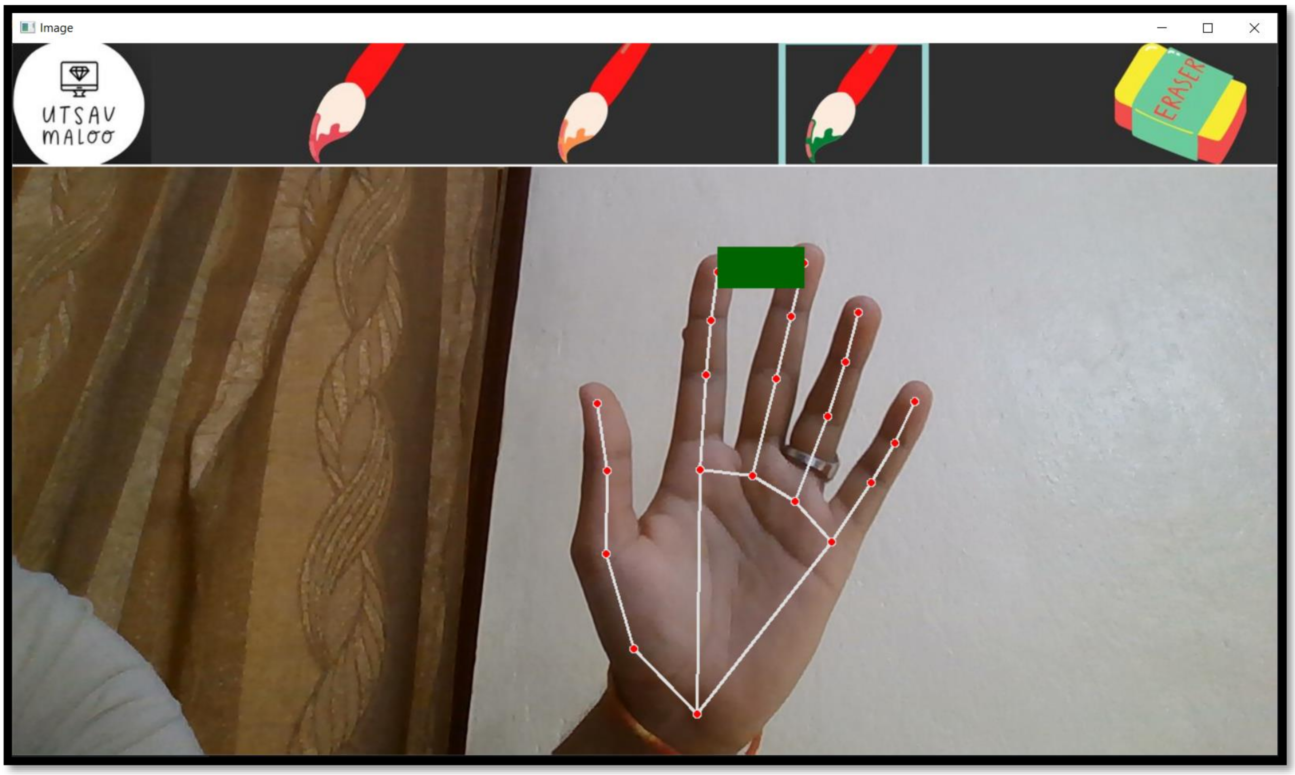
handtrackingmodules.py



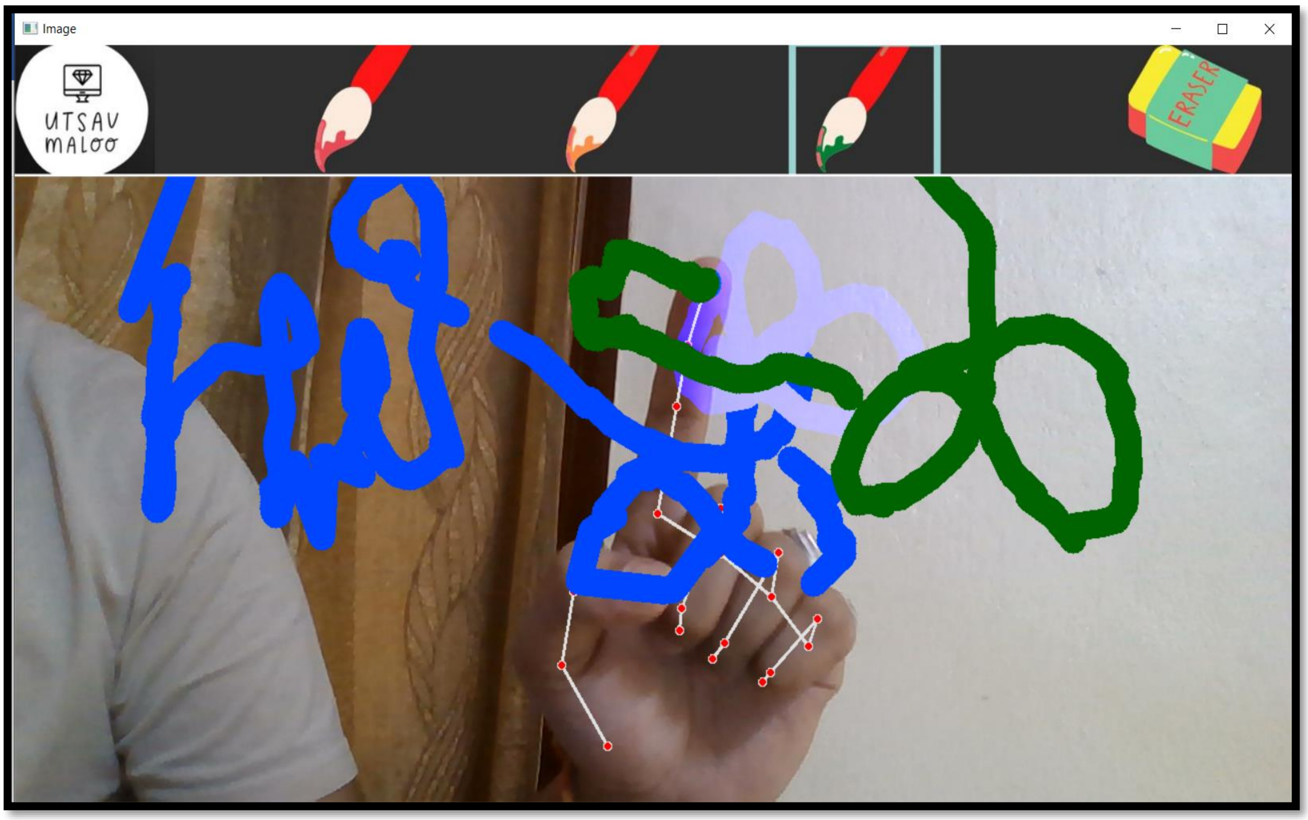
```python
import cv2
import mediapipe as mp
import time
import math
import numpy as np
class handDetector():
    def __init__(self,mode=False,maxHands=2,detectionCon=0.5,trackCon=0.5):#constructor
        self.mode=mode
        self.maxHands=maxHands
        self.detectionCon=detectionCon
        self.trackCon=trackCon
        self.mpHands=mp.solutions.hands#initializing hands module for the instance
        self.hands=self.mpHands.Hands(self.mode,self.maxHands,self.detectionCon,self.trackCon) #object for Hands for a particular instan
        self.mpDraw=mp.solutions.drawing_utils#object for Drawing
        self.tipIds = [4, 8, 12, 16, 20]
    def findHands(self,img,draw=True):
        imgRGB=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)#converting to RGB bcoz hand recognition works only on RGB image
        self.results=self.hands.process(imgRGB)#processing the RGB image
        if self.results.multi_hand_landmarks:# gives x,y,z of every landmark or if no hand than NONE
            for handLms in self.results.multi_hand_landmarks:#each hand landmarks in results
                if draw:
                    self.mpDraw.draw_landmarks(img,handLms,self.mpHands.HAND_CONNECTIONS)#joining points on our hand
        return img
    def findPosition(self,img,handNo=0,draw=True):
        xList=[]
        yList=[]
        bbox=[]
        self.lmlist=[]
        if self.results.multi_hand_landmarks:# gives x,y,z of every landmark
            myHand=self.results.multi_hand_landmarks[handNo]#Gives result for particular hand
            for id,lm in enumerate(myHand.landmark):#gives id and lm(x,y,z)
                h,w,c=img.shape#getting h,w for converting decimals x,y into pixels
                cx,cy=int(lm.x*w),int(lm.y*h)# pixels coordinates for landmarks
                # print(id, cx, cy)
                xList.append(cx)
                yList.append(cy)
                self.lmlist.append([id,cx,cy])
                if draw:
                    cv2.circle(img,(cx,cy),5,(255,0,255),cv2.FILLED)
            xmin,xmax=min(xList),max(xList)
            ymin,ymax=min(yList),max(yList)
            bbox=xmin,ymin,xmax,ymax

```

OUTPUT: -

1. **Headtracking and Output of program.**



2. **With help of handtrackingmoduls we select Brush and Draw.**



3. **With the help of handtrackingmoduls we select Eraser to clear the screen.**

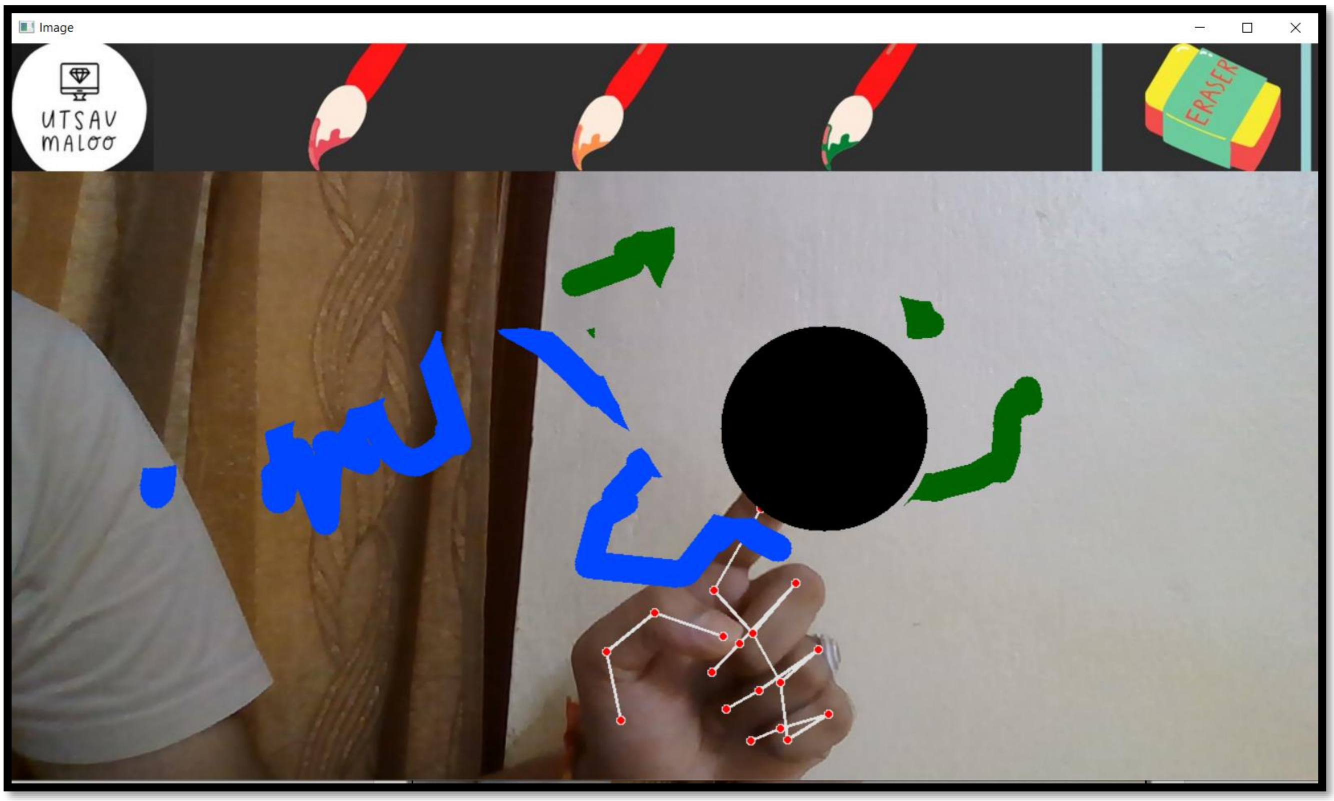


Image
UTSAV
MALOO
ERASER