```
import cv2

import mediapipe as mp

import time

import numpy as np

import math


# MediaPipe setup

mp_hands = mp.solutions.hands

mp_draw = mp.solutions.drawing_utils

hands = mp_hands.Hands(False, 1, 1, 0.3)


tips_pts = np.array([[]], np.int32)

Draw_pts = np.array([[]], np.int32)

colour = (255, 0, 0) # Default to blue

prev_frame_time = 0

curr_frame_time = 0


is_Draw_curr_Frame = False

is_Draw_prev_Frame = False

is_drawing_enabled = True


# Define color circles

Color_Circle = {

"Blue": {

"Center": (40, 40),
"Radius": 40,

"Color": (255, 0, 0),

"is Active": False,

"Drawing": [np.array([[]], np.int32)],

"Distance": 300

},

"Green": {

"Center": (40, 140),

"Radius": 40,

"Color": (0, 255, 0),

"is Active": False,

"Drawing": [np.array([[]], np.int32)],

"Distance": 300

},

"Red": {

"Center": (40, 240),

"Radius": 40,

"Color": (0, 0, 255),

"is Active": False,

"Drawing": [np.array([[]], np.int32)],

"Distance": 300

},

"Black": {

"Center": (40, 340),

"Radius": 40,
"Color": (0, 0, 0),
"is Active": False,
"Drawing": [np.array([[]], np.int32)],
"Distance": 300
},
"Purple": {
"Center": (40, 440),
"Radius": 40,
"Color": (200, 0, 200),
"is Active": False,
"Drawing": [np.array([[]], np.int32)],
"Distance": 300
},
"Yellow": {
"Center": (40, 540),
"Radius": 40,
"Color": (0, 255, 255),
"is Active": False,
"Drawing": [np.array([[]], np.int32)],
"Distance": 300
}
}

def Bounding_box_coords(lms):
b_y1 = int(min([lm.y for lm in lms]) * h)
```

```
b_y2 = int(max([lm.y for lm in lms]) * h)

b_x1 = int(min([lm.x for lm in lms]) * w)

b_x2 = int(max([lm.x for lm in lms]) * w)

return (b_x1, b_y1), (b_x2, b_y2)


def distance(a, b):

return int(math.sqrt(pow(a[0] - b[0], 2) + pow(a[1] - b[1], 2)))


def Is_in_Draw_Position(handlms, w, h):

thumb_tip = (handlms[4].x * w, handlms[4].y * h)

index_tip = (handlms[8].x * w, handlms[8].y * h)

thumb_dip = (handlms[3].x * w, handlms[3].y * h)

ref_d = distance(thumb_tip, thumb_dip)

if ref_d == 0:

return False

d = distance(thumb_tip, index_tip)

return int(d / ref_d) < 1


# Start capturing

cap = cv2.VideoCapture(0)

while cap.isOpened():

ret, img = cap.read()

if not ret:

print("Camera not working!")

break
```

```
h, w, _ = img.shape

empty_img = 255 * np.ones((h, w, 3), np.uint8)

img = cv2.flip(img, 1)


for color in Color_Circle:

cv2.circle(img, Color_Circle[color]["Center"],

Color_Circle[color]["Radius"],

Color_Circle[color]["Color"], -1)

cv2.circle(empty_img, Color_Circle[color]["Center"],

Color_Circle[color]["Radius"],

Color_Circle[color]["Color"], -1)


RGB_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

results = hands.process(RGB_img)


if not is_drawing_enabled:

cv2.putText(img, "DRAWING HALTED", (w//2 - 200, 50),
cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0, 0, 255), 4)

else:

if results.multi_hand_landmarks:

for handlm in results.multi_hand_landmarks:

mp_draw.draw_landmarks(img, handlm, mp_hands.HAND_CONNECTIONS)

for id, lm in enumerate(handlm.landmark):

lm_pos = (int(lm.x * w), int(lm.y * h))
```

```
if id == 8:

cv2.circle(img, lm_pos, 18, (255, 255, 255), -1)

for color in Color_Circle:

Color_Circle[color]["Distance"] = distance(lm_pos, Color_Circle[color]["Center"])

if Color_Circle[color]["Distance"] < 35:

for c in Color_Circle:

Color_Circle[c]["is Active"] = False

Color_Circle[color]["is Active"] = True


if Color_Circle[color]["is Active"]:

cv2.circle(empty_img, lm_pos, 18, Color_Circle[color]["Color"], -1)

if Is_in_Draw_Position(handlm.landmark, w, h):

is_Draw_curr_Frame = True

if not is_Draw_prev_Frame and is_Draw_curr_Frame:

Color_Circle[color]["Drawing"].append(np.array([[]], np.int32))


Color_Circle[color]["Drawing"][-1] = np.append(

Color_Circle[color]["Drawing"][-1], lm_pos).reshape(-1, 1, 2)

else:

is_Draw_curr_Frame = False

is_Draw_prev_Frame = is_Draw_curr_Frame


# Draw bounding box and fingertip trace

Box_corner1, Box_corner2 = Bounding_box_coords(handlm.landmark)

cv2.rectangle(img, Box_corner1, Box_corner2, (0, 0, 255), 2)
```

```
cv2.polylines(img, [tips_pts], False, (255, 0, 255), 2)


for color in Color_Circle:

for pts in Color_Circle[color]["Drawing"]:

if pts.shape[0] >= 2:

cv2.polylines(empty_img, [pts], False, Color_Circle[color]["Color"], 18)


curr_frame_time = time.time()

fps = int(1 / (curr_frame_time - prev_frame_time)) if curr_frame_time != prev_frame_time else 0

prev_frame_time = curr_frame_time

cv2.putText(img, "FPS : " + str(fps), (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 0), 3)


cv2.imshow("Virtual Painter", img)

cv2.imshow("Canvas", empty_img)


key = cv2.waitKey(5) & 0xFF

if key == ord('q'):

break

elif key == ord('c'):

for color in Color_Circle:

Color_Circle[color]["Drawing"].clear()

elif key == ord('h'):

is_drawing_enabled = not is_drawing_enabled

elif key == ord('s'):
```

```
filename = f"drawing_{int(time.time())}.png"

cv2.imwrite(filename, empty_img)

print(f"Drawing saved as {filename}")


cap.release()

cv2.destroyAllWindows()
```