Main.py

```
from aspose.slides import Presentation
import tkinter as tk
from tkinter import ttk
from tkinter import filedialog
import subprocess

root = tk.Tk()  # create the root window
root.title("Virtual Whiteboard - A Gesture control pen free tool")

# Create a style object for the button
style = ttk.Style()



def choose_directory():
    tk.Tk().withdraw()
    directory = filedialog.askdirectory()
    return directory



def run_ppt():
    ppt_images_directory = choose_directory()
    subprocess.run(
        'python ppt.py --ppt-dir "{}"'.format(ppt_images_directory), shell=False)
```

```
def run_vb():
    subprocess.run("python vb.py", shell=False)



# Create a canvas widget that occupies the full screen
canvas = tk.Canvas(root, width=root.winfo_screenwidth(),
                   height=root.winfo_screenheight())
canvas.pack(fill=tk.BOTH, expand=True)

# Define the start and end colors for the gradient
start_color = "#FFD1DC"  # Baby pink
end_color = "#FFE5B4"  # Peach

# Create a rectangle on the canvas with a gradient fill
for i in range(root.winfo_screenheight()):
    # Calculate the color at this point in the gradient
    r = int((i * int(end_color[1:3], 16) + (root.winfo_screenheight() - i)
            * int(start_color[1:3], 16)) / root.winfo_screenheight())
    g = int((i * int(end_color[3:5], 16) + (root.winfo_screenheight() - i)
            * int(start_color[3:5], 16)) / root.winfo_screenheight())
    b = int((i * int(end_color[5:], 16) + (root.winfo_screenheight() - i)
            * int(start_color[5:], 16)) / root.winfo_screenheight())
    color = "#" + hex(r)[2:].zfill(2) + \
        hex(g)[2:].zfill(2) + hex(b)[2:].zfill(2)

    # Draw a horizontal line of the gradient color
    canvas.create_line(0, i, root.winfo_screenwidth(), i, fill=color)
```

```
# create a label widget on the canvas
title_label = tk.Label(
    canvas, text="Welcome to Virtual Whiteboard- A Gesture Control Pen Free Tool", bg="#FFD1DC", fg='#171002')
title_label.place(relx=0.5, rely=0.5, anchor=tk.CENTER)

# Set the background of the label to the same gradient as the canvas
title_label.config(font=('Helvetica', 40), highlightthickness=0)
title_label.configure(bg="#FFD1DC")
canvas.create_window(root.winfo_screenwidth(
)/2, root.winfo_screenheight()*0.2, anchor=tk.CENTER, window=title_label)


# create a label widget on the canvas
vb_label = tk.Label(
    canvas, text="Tired of traditional drawing tools? \nOur cutting-edge program lets you create stunning artwork using just your hands.\nUnleash your creativity and draw like never before with our state-of-the-art Python app. \nClick the button below to get started!", fg='#171002', bg="#FBD6D2")
vb_label.place(relx=0.5, rely=0.4, anchor=tk.CENTER)
vb_label.config(font=('Helvetica', 20), highlightthickness=0)

# Configure the style for the button
style.configure('RoundedButton.TButton', foreground='#171002', background='#FFACAC',
                font=('Helvetica', 40), relief='groove', borderwidth=1, width=20)

# Configure the style for the button when the mouse is hovering over it
```

```
style.map('RoundedButton.TButton', foreground=[
        ('active', '#171002')], background=[('active', '#F190B7')])

# Create the button on the canvas using the style
vb_button = ttk.Button(canvas, text="Let's Draw",
                       style='RoundedButton.TButton', command=run_vb)
vb_button.place(relx=0.5, rely=0.5, anchor=tk.CENTER)

# create a label widget on the canvas
p_label = tk.Label(
    canvas, text="Revolutionize your presentations with our innovative gesture control feature, allowing you to navigate your slides, \ntake control and draw on your presentation without touching a single button. With intuitive and natural hand gestures, \nyou can deliver a seamless and engaging experience that takes your presentations to the next level.", fg='#171002', bg="#f9dcc5")
p_label.place(relx=0.5, rely=0.7, anchor=tk.CENTER)
p_label.config(font=('Helvetica', 20), highlightthickness=0)

# Configure the style for the button
style.configure('RoundedButton.TButton',                foreground='#171002',
background='#FFACAC',
              font=('Helvetica', 40), relief='groove', borderwidth=1, width=20)

# Configure the style for the button when the mouse is hovering over it
style.map('RoundedButton.TButton', foreground=[
        ('active', '#171002')], background=[('active', '#F190B7')])

# Create the button on the canvas using the style
p_button = ttk.Button(canvas, text="Let's Present",
```

```
                    style='RoundedButton.TButton', command=run_ppt)
p_button.place(relx=0.5, rely=0.8, anchor=tk.CENTER)
root.mainloop()  # start the main event loop
```

handTracker.py

```
import mediapipe as mp
import numpy as np
import cv2



class HandTracker():
    def __init__(self, mode=False, maxHands=2, detectionCon=0.5, trackCon=0.5):
        self.mode = mode
        self.maxHands = maxHands
        self.detectionCon = detectionCon
        self.trackCon = trackCon


        self.mpHands = mp.solutions.hands
        self.hands = self.mpHands.Hands(self.mode, self.maxHands, self.detectionCon, self.trackCon)
        self.mpDraw = mp.solutions.drawing_utils


    def findHands(self, img, draw=True):
        imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        self.results = self.hands.process(imgRGB)


        if self.results.multi_hand_landmarks:
            for handLm in self.results.multi_hand_landmarks:
                if draw:
```

```
                self.mpDraw.draw_landmarks(img, handLm,
self.mpHands.HAND_CONNECTIONS)
        return img

    def getPostion(self, img, handNo = 0, draw=True):
        lmList =[]
        if self.results.multi_hand_landmarks:
            myHand = self.results.multi_hand_landmarks[handNo]
            for lm in myHand.landmark:
                h, w, c = img.shape
                cx, cy = int(lm.x*w), int(lm.y*h)
                lmList.append((cx, cy))

                if draw:
                    cv2.circle(img, (cx, cy), 5, (255,0,255), cv2.FILLED)
        return lmList

    def getUpFingers(self, img):
        pos = self.getPostion(img, draw=False)
        self.upfingers = []
        if pos:
            #thumb
            self.upfingers.append((pos[4][1] < pos[3][1] and (pos[5][0]-pos[4][0]>
10)))
            #index
            self.upfingers.append((pos[8][1] < pos[7][1] and pos[7][1] < pos[6][1]))
            #middle
```

```
        self.upfingers.append((pos[12][1] < pos[11][1] and pos[11][1] < pos[10][1]))
        #ring
        self.upfingers.append((pos[16][1] < pos[15][1] and pos[15][1] < pos[14][1]))
        #pinky
        self.upfingers.append((pos[20][1] < pos[19][1] and pos[19][1] < pos[18][1]))
    return self.upfingers
```

Vb.py

```
from handTracker import *
import cv2
import mediapipe as mp
import numpy as np
import random
import ctypes

user32 = ctypes.windll.user32
screensize = user32.GetSystemMetrics(0), user32.GetSystemMetrics(1)



class ColorRect():
    def __init__(self, x, y, w, h, color, text='', alpha=0.5):
        self.x = x
        self.y = y
        self.w = w
        self.h = h
        self.color = color
        self.text = text
        self.alpha = alpha


    def drawRect(self, img, text_color=(255, 255, 255), fontFace=cv2.FONT_HERSHEY_SIMPLEX, fontScale=0.8, thickness=2):
        # draw the box
        alpha = self.alpha
        bg_rec = img[self.y: self.y + self.h, self.x: self.x + self.w]
```

```
        white_rect = np.ones(bg_rec.shape, dtype=np.uint8)
        white_rect[:] = self.color
        res = cv2.addWeighted(bg_rec, alpha, white_rect, 1-alpha, 1.0)

        # Putting the image back to its position
        img[self.y: self.y + self.h, self.x: self.x + self.w] = res

        # put the letter
        tetx_size = cv2.getTextSize(self.text, fontFace, fontScale, thickness)
        text_pos = (int(self.x + self.w/2 -
                    tetx_size[0][0]/2), int(self.y + self.h/2 + tetx_size[0][1]/2))
        cv2.putText(img, self.text, text_pos, fontFace,
                    fontScale, text_color, thickness)

    def isOver(self, x, y):
        if (self.x + self.w > x > self.x) and (self.y + self.h > y > self.y):
            return True
        return False


# initilize the habe detector
detector = HandTracker(detectionCon=0.8)

# initilize the camera
cap = cv2.VideoCapture(0)
cap.set(3, screensize[1])
cap.set(4, screensize[0])
```

```
# creating canvas to draw on it
canvas = np.zeros((screensize[1], screensize[0], 3), np.uint8)

# define a previous point to be used with drawing a line
px, py = 0, 0
# initial brush color
color = (255, 0, 0)
#####
brushSize = 5
eraserSize = 20
####

########### creating colors ########
# Colors button
colorsBtn = ColorRect(200, 0, 100, 100, (120, 255, 0), 'Colors')

colors = []
# random color
b = int(random.random()*255)-1
g = int(random.random()*255)
r = int(random.random()*255)
print(b, g, r)
colors.append(ColorRect(300, 0, 100, 100, (b, g, r)))
# red
colors.append(ColorRect(400, 0, 100, 100, (0, 0, 255)))
# blue
```

```
colors.append(ColorRect(500, 0, 100, 100, (255, 0, 0)))
# green
colors.append(ColorRect(600, 0, 100, 100, (0, 255, 0)))
# yellow
colors.append(ColorRect(700, 0, 100, 100, (0, 255, 255)))
# erase (black)
colors.append(ColorRect(800, 0, 100, 100, (0, 0, 0), "Eraser"))

# clear
clear = ColorRect(900, 0, 100, 100, (100, 100, 100), "Clear")

########## pen sizes #######
pens = []
for i, penSize in enumerate(range(5, 25, 5)):
    pens.append(ColorRect(1750, 50+100*i, 100,
                100, (50, 50, 50), str(penSize)))

penBtn = ColorRect(1750, 0, 100, 50, color, 'Pen')

# white board button
boardBtn = ColorRect(50, 0, 100, 100, (255, 255, 0), 'Board')

# define a white board to draw on
whiteBoard = ColorRect(
    50, 120, screensize[0] - 260, screensize[1] - 140, (255, 255, 255), alpha=0.6)

coolingCounter = 20
```

```
hideBoard = True
hideColors = True
hidePenSizes = True

while True:

    if coolingCounter:
        coolingCounter -= 1
        # print(coolingCounter)

    ret, frame = cap.read()
    if not ret:
        break
    frame = cv2.resize(frame, (screensize[0], screensize[1]))
    frame = cv2.flip(frame, 1)

    detector.findHands(frame)
    positions = detector.getPostion(frame, draw=False)
    upFingers = detector.getUpFingers(frame)

    if upFingers:
        x, y = positions[8][0], positions[8][1]
        if upFingers[1] and not whiteBoard.isOver(x, y):
            px, py = 0, 0

            ##### pen sizes ######
            if not hidePenSizes:
```

```
    for pen in pens:
        if pen.isOver(x, y):
            brushSize = int(pen.text)
            pen.alpha = 0
        else:
            pen.alpha = 0.5

####### chose a color for drawing #######
if not hideColors:
    for cb in colors:
        if cb.isOver(x, y):
            color = cb.color
            cb.alpha = 0
        else:
            cb.alpha = 0.5

    # Clear
    if clear.isOver(x, y):
        clear.alpha = 0
        canvas = np.zeros(
            (screensize[1], screensize[0], 3), np.uint8)
    else:
        clear.alpha = 0.5

# color button
if colorsBtn.isOver(x, y) and not coolingCounter:
    coolingCounter = 10
```

```
            colorsBtn.alpha = 0
            hideColors = False if hideColors else True
            colorsBtn.text = 'Colors' if hideColors else 'Hide'
        else:
            colorsBtn.alpha = 0.5

        # Pen size button
        if penBtn.isOver(x, y) and not coolingCounter:
            coolingCounter = 10
            penBtn.alpha = 0
            hidePenSizes = False if hidePenSizes else True
            penBtn.text = 'Pen' if hidePenSizes else 'Hide'
        else:
            penBtn.alpha = 0.5

        # white board button
        if boardBtn.isOver(x, y) and not coolingCounter:
            coolingCounter = 10
            boardBtn.alpha = 0
            hideBoard = False if hideBoard else True
            boardBtn.text = 'Board' if hideBoard else 'Hide'

        else:
            boardBtn.alpha = 0.5

    elif upFingers[1] and not upFingers[2]:
        if whiteBoard.isOver(x, y) and not hideBoard:
```

```
            # print('index finger is up')
            cv2.circle(frame, positions[8], brushSize, color, -1)
            # drawing on the canvas
            if px == 0 and py == 0:
                px, py = positions[8]
            if color == (0, 0, 0):
                cv2.line(canvas, (px, py), positions[8], color, eraserSize)
            else:
                cv2.line(canvas, (px, py), positions[8], color, brushSize)
            px, py = positions[8]

        else:
            px, py = 0, 0

    # put colors button
    colorsBtn.drawRect(frame)
    cv2.rectangle(frame, (colorsBtn.x, colorsBtn.y), (colorsBtn.x +
                colorsBtn.w, colorsBtn.y+colorsBtn.h), (255, 255, 255), 2)

    # put white board button
    boardBtn.drawRect(frame)
    cv2.rectangle(frame, (boardBtn.x, boardBtn.y), (boardBtn.x +
                boardBtn.w, boardBtn.y+boardBtn.h), (255, 255, 255), 2)

    # put the white board on the frame
    if not hideBoard:
        whiteBoard.drawRect(frame)
```

```
        ########### moving the draw to the main image #########
        canvasGray = cv2.cvtColor(canvas, cv2.COLOR_BGR2GRAY)
        _, imgInv = cv2.threshold(canvasGray, 20, 255,
cv2.THRESH_BINARY_INV)
        imgInv = cv2.cvtColor(imgInv, cv2.COLOR_GRAY2BGR)
        frame = cv2.bitwise_and(frame, imgInv)
        frame = cv2.bitwise_or(frame, canvas)

    ########## pen colors' boxes #########
    if not hideColors:
        for c in colors:
            c.drawRect(frame)
            cv2.rectangle(frame, (c.x, c.y), (c.x + c.w,
                    c.y+c.h), (255, 255, 255), 2)

        clear.drawRect(frame)
        cv2.rectangle(frame, (clear.x, clear.y), (clear.x +
                clear.w, clear.y+clear.h), (255, 255, 255), 2)

    ########## brush size boxes ######
    penBtn.color = color
    penBtn.drawRect(frame)
    cv2.rectangle(frame, (penBtn.x, penBtn.y), (penBtn.x +
            penBtn.w, penBtn.y+penBtn.h), (255, 255, 255), 2)
    if not hidePenSizes:
        for pen in pens:
            pen.drawRect(frame)
            cv2.rectangle(frame, (pen.x, pen.y), (pen.x + pen.w,
```

```
                    pen.y+pen.h), (255, 255, 255), 2)

    cv2.imshow('video', frame)
    # cv2.imshow('canvas', canvas)
    if cv2.waitKey(1) == 27:
        cv2.destroyAllWindows()
        cap.release()
        break
cap.release()
cv2.destroyAllWindows()
```

ppt.py

```
from cvzone.HandTrackingModule import HandDetector
import cv2
import os
import numpy as np
import argparse

parser = argparse.ArgumentParser(description="Just an example",

formatter_class=argparse.ArgumentDefaultsHelpFormatter)
parser.add_argument("--ppt-dir", help="Source location of PPT images")

args = parser.parse_args()
config = vars(args)

# Parameters
# ppt
widthSlide, heightSlide = 1400, 720
# video
width, height = 640, 480
gestureThreshold = 300

folderPath = "D:\\Virtual Whiteboatrd\\Presentation"
if config.get("ppt_dir", False):
    print(config)
    folderPath = config["ppt_dir"]
```

```
# Camera Setup
cap = cv2.VideoCapture(0)
cap.set(3, width)
cap.set(4, height)


def resizeAndPad(img, size, padColor=0):
    h, w = img.shape[:2]
    sh, sw = size
    # interpolation method
    if h > sh or w > sw:  # shrinking image
        interp = cv2.INTER_AREA
    else:  # stretching image
        interp = cv2.INTER_CUBIC

    # aspect ratio of image
    # if on Python 2, you might need to cast as a float: float(w)/h
    aspect = w/h

    # compute scaling and pad sizing
    if aspect > 1:  # horizontal image
        new_w = sw
        new_h = np.round(new_w/aspect).astype(int)
        pad_vert = (sh-new_h)/2
        pad_top, pad_bot = np.floor(pad_vert).astype(
            int), np.ceil(pad_vert).astype(int)
```

```
        pad_left, pad_right = 0, 0
    elif aspect < 1:  # vertical image
        new_h = sh
        new_w = np.round(new_h*aspect).astype(int)
        pad_horz = (sw-new_w)/2
        pad_left, pad_right = np.floor(pad_horz).astype(
            int), np.ceil(pad_horz).astype(int)
        pad_top, pad_bot = 0, 0
    else:  # square image
        new_h, new_w = sh, sw
        pad_left, pad_right, pad_top, pad_bot = 0, 0, 0, 0

    # set pad color
    # color image but only one color provided
    if len(img.shape) == 3 and not isinstance(padColor, (list, tuple, np.ndarray)):
        padColor = [padColor]*3

    # scale and pad
    scaled_img = cv2.resize(img, (new_w, new_h), interpolation=interp)

    return scaled_img



# Hand Detector
detectorHand = HandDetector(detectionCon=0.8, maxHands=1)

# Variables
```

```
imgList = []
delay = 30
buttonPressed = False
counter = 0
drawMode = False
imgNumber = 0
delayCounter = 0
annotations = [[]]
annotationNumber = -1
annotationStart = False
hs, ws = int(120 * 1), int(213 * 1)  # width and height of small image

# Get list of presentation images
pathImages = sorted(os.listdir(folderPath), key=len)
print(pathImages)

while True:
    # Get image frame
    success, img = cap.read()
    img = cv2.flip(img, 1)
    pathFullImage = os.path.join(folderPath, pathImages[imgNumber])
    imgCurrent = cv2.imread(pathFullImage)

    # Find the hand and its landmarks
    hands, img = detectorHand.findHands(img)  # with draw
    # Draw Gesture Threshold line
    cv2.line(img, (0, gestureThreshold),
```

```
        (width, gestureThreshold), (0, 255, 0), 10)

if hands and buttonPressed is False:  # If hand is detected

    hand = hands[0]
    cx, cy = hand["center"]
    lmList = hand["lmList"]  # List of 21 Landmark points
    fingers = detectorHand.fingersUp(hand)  # List of which fingers are up

    # Constrain values for easier drawing
    xVal = int(np.interp(lmList[8][0], [width // 2, width], [0, width]))
    yVal = int(np.interp(lmList[8][1], [150, height-150], [0, height]))
    indexFinger = xVal, yVal

    if cy <= gestureThreshold:  # If hand is at the height of the face
        if fingers == [1, 0, 0, 0, 0]:
            print("Left")
            buttonPressed = True
            if imgNumber > 0:
                imgNumber -= 1
                annotations = [[]]
                annotationNumber = -1
                annotationStart = False
        if fingers == [0, 0, 0, 0, 1]:
            print("Right")
            buttonPressed = True
            if imgNumber < len(pathImages) - 1:
```

```
            imgNumber += 1
            annotations = [[]]
            annotationNumber = -1
            annotationStart = False

    if fingers == [0, 1, 1, 0, 0]:
        cv2.circle(imgCurrent, indexFinger, 12, (0, 0, 255), cv2.FILLED)

    if fingers == [0, 1, 0, 0, 0]:
        if annotationStart is False:
            annotationStart = True
            annotationNumber += 1
            annotations.append([])
        print(annotationNumber)
        annotations[annotationNumber].append(indexFinger)
        cv2.circle(imgCurrent, indexFinger, 12, (0, 0, 255), cv2.FILLED)

    else:
        annotationStart = False

    if fingers == [0, 1, 1, 1, 0]:
        if annotations:
            annotations.pop(-1)
            annotationNumber -= 1
            buttonPressed = True

else:
```

```
    annotationStart = False

if buttonPressed:
    counter += 1
    if counter > delay:
        counter = 0
        buttonPressed = False

for i, annotation in enumerate(annotations):
    for j in range(len(annotation)):
        if j != 0:
            cv2.line(imgCurrent, annotation[j - 1],
                     annotation[j], (0, 0, 200), 12)

imgSmall = cv2.resize(img, (ws, hs))
h, w, _ = imgCurrent.shape
imgCurrent[0:hs, w - ws: w] = imgSmall

imgCurrent = resizeAndPad(imgCurrent, (heightSlide, widthSlide))

cv2.imshow("Image", img)
cv2.imshow("Slides", imgCurrent)

if cv2.waitKey(1) == 27:
    cv2.destroyAllWindows()
    cap.release()
    break
```