Group 13: Finger painting in air using a webcam

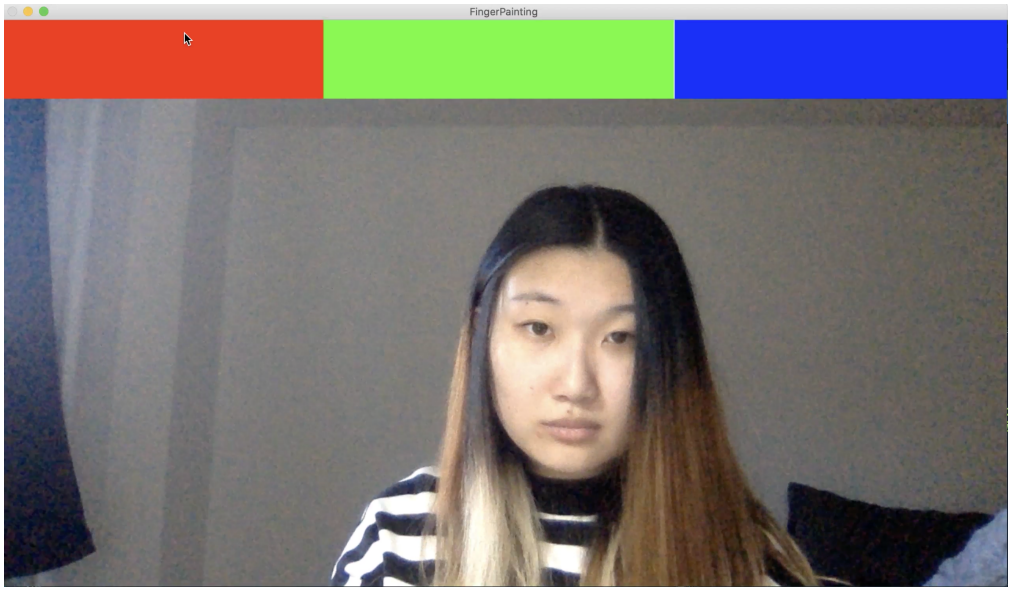
Yueyi Li (yl7544)

1. Introduction

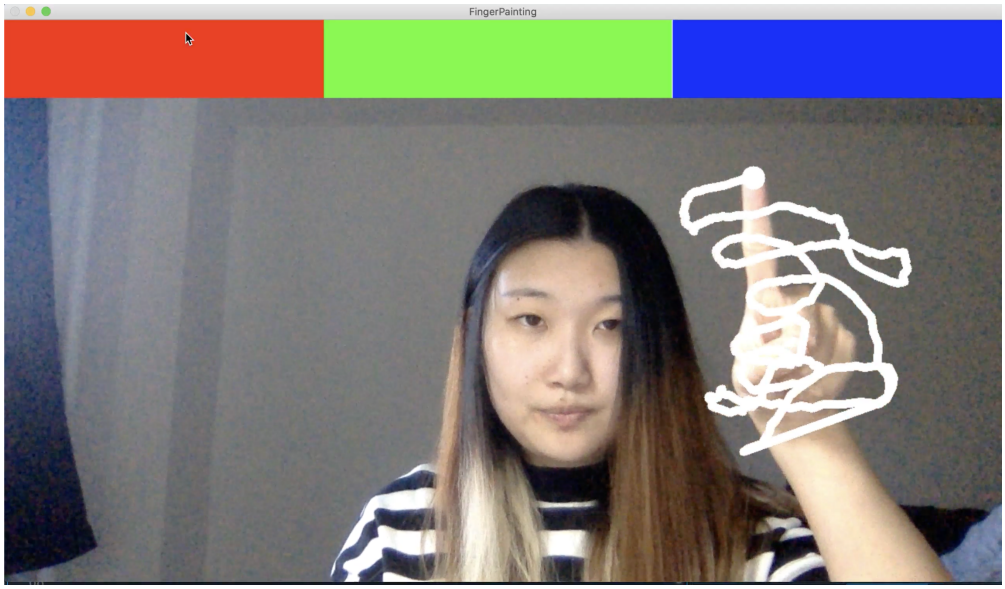
In this project, we can draw in air using a webcam with one finger. The brush will follow our finger and paint on the canvas. The default color of brush is white, besides, there are three different color brushes for us to choose: red, green, and blue. We can change the color by putting our finger on the corresponding area.

2. Demonstration

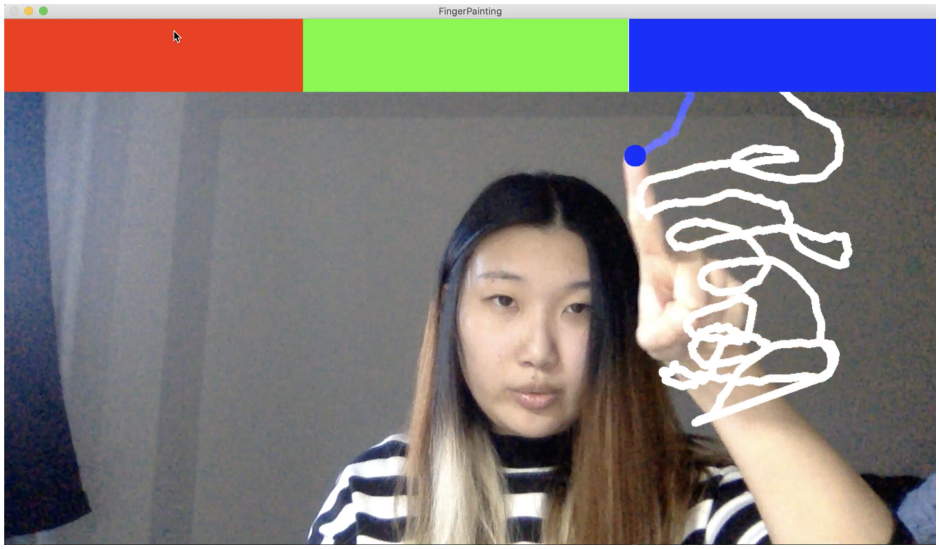
2.1 Run our program, and we can see that the webcam turns on and there is a three-color header picture at the top of screen where we can choose the brush color.



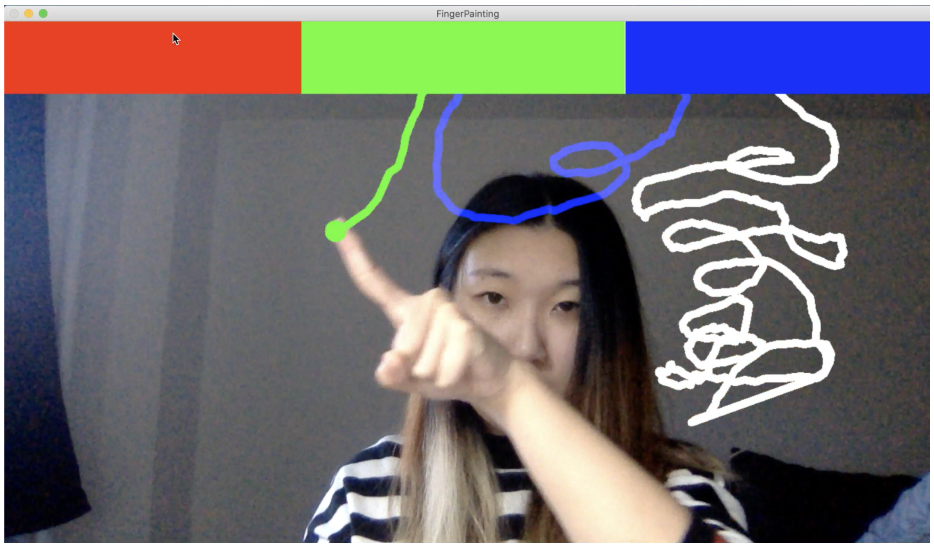
2.2 When we put our finger up, the program will recognize it and start to paint on our canvas. The default color of brush is white, and the lines represent our painting, the circle represent our finger position.

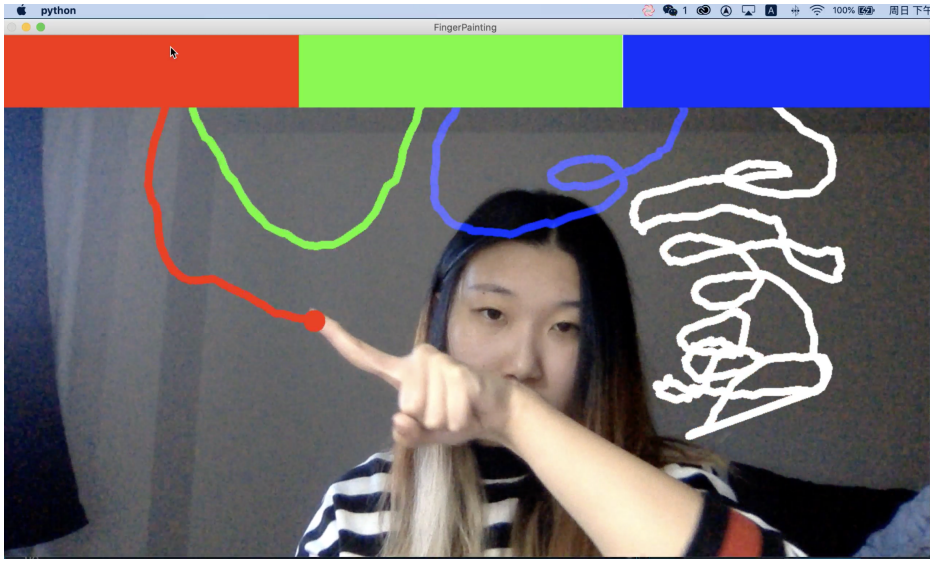


2.3 There are three colors at the top of screen, if we want to change the brush color, we can put our finger on the corresponding color area. As the screenshot, if we put our finger on the blue, the brush will turn to blue and continue our painting.



2.4The same goes for green and red.





3. Implementation

3.1 We need to initialize some variables. Read in our header picture, set the default color to white, connect our video device and set our canvas to 720*1280. Then initialize the width of brush and the default position value.

```
header = cv2.imread('1.png')
drawColor = (255, 255, 255) # default color
cap = cv2.VideoCapture(0)  # video device linux -1, win 0/1
cap.set(3, 1280)
cap.set(4, 720)
imgCanvas = np.zeros((720, 1280, 3), np.uint8)
```

```
brushWidth=10
x0, y0 = 0, 0
```

3.2 The detector is a class to detect our hand and find out the position of finger, I use a python library called mediapipe to help.

```
detector = Detector()
```

```
import mediapipe as mp

class Detector():
    def __init__(self):
        self.hands = mp.solutions.hands.Hands(False, 1, 0.85, 0.5)
    def getPosition(self, img):
        imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        self.results = self.hands.process(imgRGB)
        self.lmList = []
        if self.results.multi_hand_landmarks:
            for id, lm in enumerate(self.results.multi_hand_landmarks[0].landmark):
                h, w, c = img.shape
                cx, cy = int(lm.x * w), int(lm.y * h)
                self.lmList.append([id, cx, cy])
            return self.lmList
```

3.3 In the while loop, the current frame image will be passed into the getPosition method, and the coordinate of finger will be returned.

```
while True:
    success, img = cap.read()
    img = cv2.flip(img, 1)
    lmList = detector.getPosition(img)
```

y1 is less than 80 means we are trying to change color, so the drawColor will be corresponding color value.

```
    if lmList!=None and len(lmList) != 0:
        x1, y1 = lmList[8][1:]
        if x0 == 0 and y0 == 0:
            x0, y0 = x1, y1
        if y1 < 80:
            if 100 < x1 < 450:
                drawColor = (0, 0, 255)
            elif 500 < x1 < 850:
                drawColor = (0, 255, 0)
            elif 900 < x1 < 1250:
                drawColor = (255, 0, 0)
```

Then we can draw the circle and lines which is the painting by our finger. Here we use the cv2 to help which is a library for processing images. In addition, we need to update x0 and y0.

```
        cv2.circle(img, (x1, y1), 15, drawColor, cv2.FILLED)
        cv2.line(img, (x0, y0), (x1, y1), drawColor, brushWidth)
        cv2.line(imgCanvas, (x0, y0), (x1, y1),
                    drawColor, brushWidth)
        x0, y0 = x1, y1
```

In the end, put the new image on our canvas and set the header picture.

```
    img[0:100, 0:1280] = header
    cv2.imshow("FingerPainting", img)
    cv2.waitKey(1)
```

In this way, we read in each frame, calculate the finger position, and paint on the canvas, and then this project is finished.

4. Further

4.1 We have to put our finger on the specific area to change our brush color which leads to unnecessary painting leaving on our canvas. Therefore, it would be better to use another way to change the color which will not leave the painting, such as a specific gesture.

4.2 Now that we have captured the position of one finger, we can record the positions of the remaining four fingers at the same time to realize other functions. For example, different fingers represent different brush effects: pen, pencil, highlighter, etc.