import tkinter as tk

import pygame

import pygame.camera

from PIL import Image, ImageTk

import matplotlib.pyplot as plt

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

from collections import deque

class VisitorCounterApp:

"""

A GUI application to count visitors, display a video feed, and show analytics.

This application integrates tkinter for the GUI, pygame for video streaming,

and matplotlib for analytics plotting.

"""

def \_\_init\_\_(self, root):

"""

Initialize the VisitorCounterApp instance.

Parameters:

- root: The tkinter root window.

"""

# Initialize tkinter root window

self.root = root

self.root.title("Visitor Counter with Video Feed and Analytics")

# Initialize visitor count and history

self.visitor\_count = 0

self.visitor\_history = deque(maxlen=100)

# Create label to display current visitor count

self.visitor\_label = tk.Label(self.root, text=f"Current Visitors: {self.visitor\_count}", font=("Helvetica", 24))

self.visitor\_label.pack(pady=20)

# Create canvas for video feed

self.video\_canvas = tk.Canvas(self.root, width=640, height=480)

self.video\_canvas.pack()

# Create button to increment visitor count

self.increment\_button = tk.Button(self.root, text="Increment Visitor", command=self.increment\_visitor)

self.increment\_button.pack(pady=10)

# Create button to reset visitor count

self.reset\_button = tk.Button(self.root, text="Reset Count", command=self.reset\_visitor)

self.reset\_button.pack(pady=10)

# Initialize pygame and pygame camera

pygame.init()

pygame.camera.init()

# Initialize camera and start capturing

self.camera = pygame.camera.Camera(pygame.camera.list\_cameras()[0], (640, 480))

self.camera.start()

# Start streaming video feed

self.stream\_video()

# Initialize matplotlib figure and axes for analytics plot

self.fig, self.ax = plt.subplots(figsize=(8, 4))

self.ax.set\_title('Visitor Count Over Time')

self.ax.set\_xlabel('Time')

self.ax.set\_ylabel('Visitor Count')

self.plot\_line, = self.ax.plot([], [], label='Visitors')

self.ax.legend()

# Embed matplotlib plot into tkinter GUI

self.canvas = FigureCanvasTkAgg(self.fig, master=self.root)

self.canvas.get\_tk\_widget().pack()

# Start updating analytics plot periodically

self.update\_analytics\_plot()

def stream\_video(self):

"""

Continuously updates the video feed from the camera.

Uses pygame to capture frames from the camera, converts them to

a tkinter-compatible format, and updates the video canvas.

"""

frame = self.camera.get\_image()

# Convert pygame surface to PIL image

image\_data = pygame.image.tostring(frame, "RGB")

pil\_image = Image.frombytes("RGB", (640, 480), image\_data)

# Update video canvas with the new frame

self.video\_feed = ImageTk.PhotoImage(image=pil\_image)

self.video\_canvas.create\_image(0, 0, anchor=tk.NW, image=self.video\_feed)

# Schedule next video frame update

self.root.after(10, self.stream\_video)

def increment\_visitor(self):

"""

Increment the visitor count by 1 and update the visitor history.

Has to done automatically once integrated with face detection

"""

self.visitor\_count += 1

self.visitor\_history.append(self.visitor\_count)

self.update\_visitor\_label()

def reset\_visitor(self):

"""

Reset the visitor count to 0 and clear the visitor history.

"""

self.visitor\_count = 0

self.visitor\_history.clear()

self.update\_visitor\_label()

def update\_visitor\_label(self):

"""

Update the visitor count label with the current visitor count.

"""

self.visitor\_label.config(text=f"Current Visitors: {self.visitor\_count}")

def update\_analytics\_plot(self):

"""

Update the analytics plot with the latest visitor history data.

"""

self.plot\_line.set\_data(range(len(self.visitor\_history)), self.visitor\_history)

self.ax.relim()

self.ax.autoscale\_view()

self.fig.canvas.draw()

# Schedule next update of analytics plot

self.root.after(1000, self.update\_analytics\_plot)

def \_\_del\_\_(self):

"""

Destructor to ensure the camera is stopped when the object is deleted.

"""

if hasattr(self, 'camera'):

self.camera.stop()

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

# Create the tkinter root window and start the application

root = tk.Tk()

app = VisitorCounterApp(root)

root.mainloop()