Camera Recording Program with Simple Window (Tkinter)

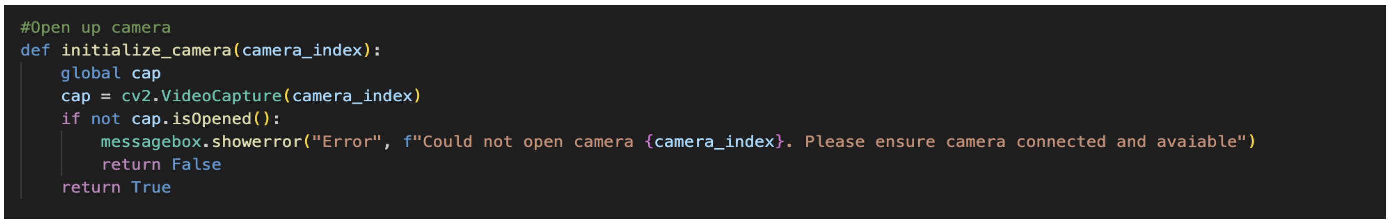
This program allows you to record video from your camera using a simple window with buttons. You can select a camera, preview what it looks like for a few seconds, and record video with a timestamp. The video is saved in a `.mov` format.

Important Things to Know

- **cap**: This stores the connection to your camera, so the program knows which camera to use.
- **out**: This saves the video to a file on your computer.
- **is_recording**: A true/false setting that tracks whether you are currently recording.
- **selected_camera_index**: The number (index) of the camera you choose to record from.
- **camera_output_file**: The name of the video file where your recording will be saved.

What Each Part of the Code Does

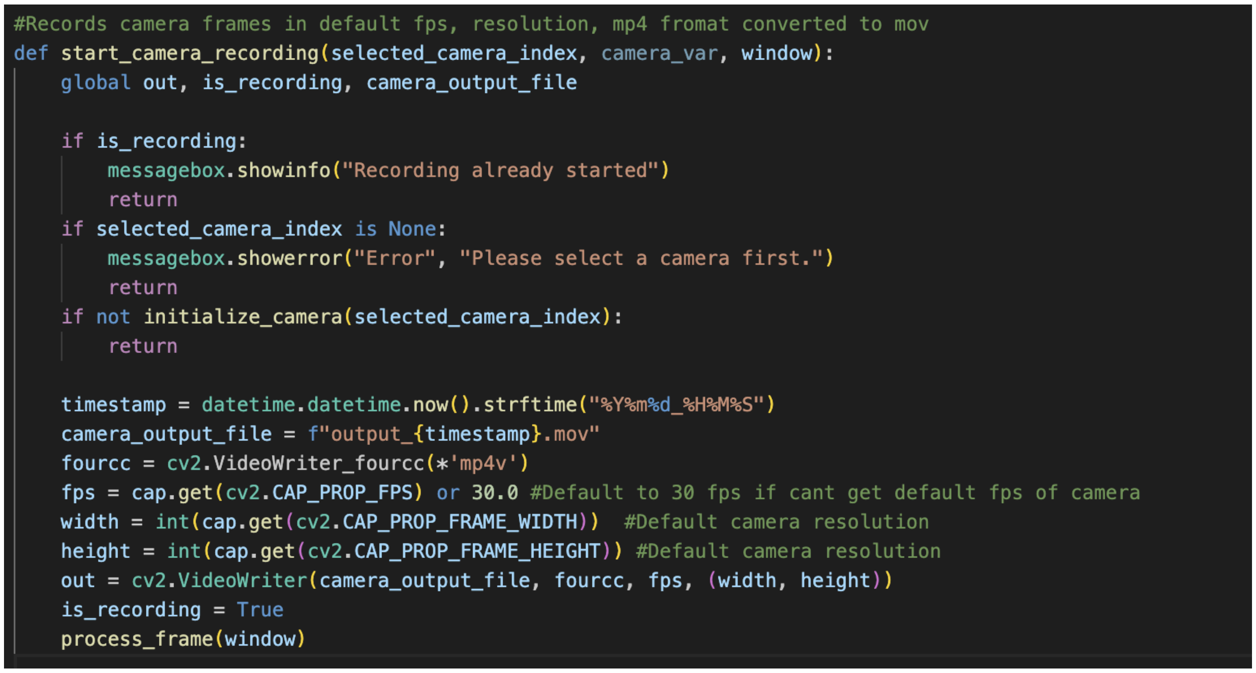
1. `initialize_camera(camera_index)`



```python
#Open up camera
def initialize_camera(camera_index):
    global cap
    cap = cv2.VideoCapture(camera_index)
    if not cap.isOpened():
        messagebox.showerror("Error", f"Could not open camera {camera_index}. Please ensure camera connected and avaiable")
        return False
    return True
```

- **What it does**: This part connects to the camera you choose (like camera 1 or camera 2). If the camera isn't working, it shows an error message.
- **Why it's important**: The program needs to make sure the camera works before it can start recording.

2. `start_camera_recording(selected_camera_index, camera_var, window)`



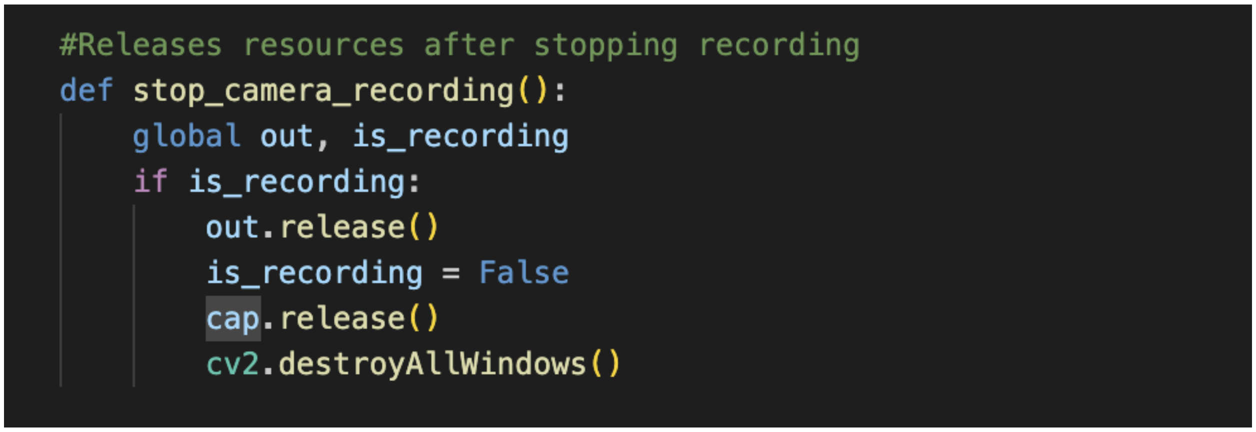
```python
#Records camera frames in default fps, resolution, mp4 fromat converted to mov
def start_camera_recording(selected_camera_index, camera_var, window):
    global out, is_recording, camera_output_file

    if is_recording:
        messagebox.showinfo("Recording already started")
        return
    if selected_camera_index is None:
        messagebox.showerror("Error", "Please select a camera first.")
        return
    if not initialize_camera(selected_camera_index):
        return

    timestamp = datetime.datetime.now().strftime("%Y%m%d_%H%M%S")
    camera_output_file = f"output_{timestamp}.mov"
    fourcc = cv2.VideoWriter_fourcc(*'mp4v')
    fps = cap.get(cv2.CAP_PROP_FPS) or 30.0 #Default to 30 fps if cant get default fps of camera
    width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))  #Default camera resolution
    height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT)) #Default camera resolution
    out = cv2.VideoWriter(camera_output_file, fourcc, fps, (width, height))
    is_recording = True
    process_frame(window)
```

- **What it does**:
    - This part starts recording video from the camera you selected.
    - It saves the video to a file with a name that includes the current date and time (e.g., `output_20240101_120000.mov`).
    - The video is recorded in `.mp4` format, but saved as `.mov`.
    - Video recorded using default fps and resolution of camera (if can't retrieve default fps defaults to 30)
- **Why it's important**: It handles everything needed to start recording, including picking the correct file type and setting up the camera properly.

3. `stop_camera_recording()`

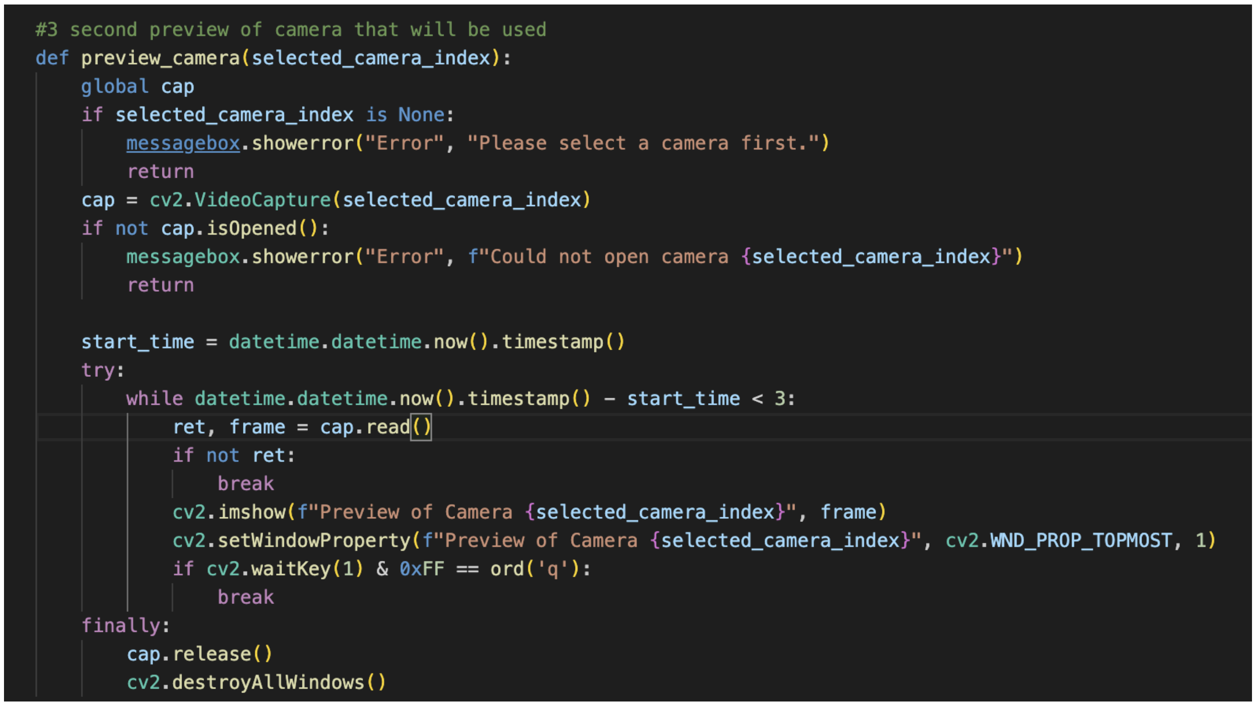


```python
#Releases resources after stopping recording
def stop_camera_recording():
    global out, is_recording
    if is_recording:
        out.release()
        is_recording = False
        cap.release()
        cv2.destroyAllWindows()
```

- **What it does**:
    - This part stops the recording and closes the camera when you're done.

    ○ It also closes the window where you see the live video.
- **Why it's important**: It releases the camera and saves the recording properly.

4. `preview_camera(selected_camera_index)`

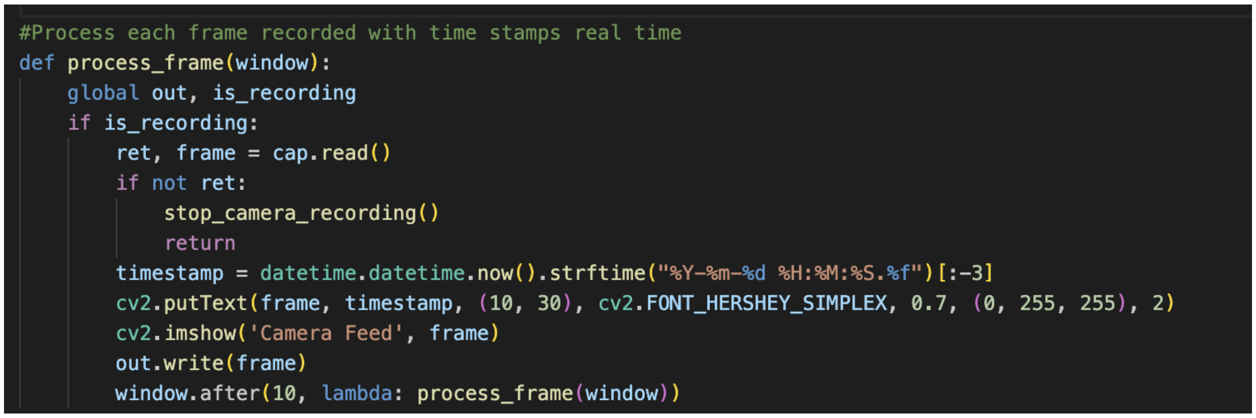


```python
#3 second preview of camera that will be used
def preview_camera(selected_camera_index):
    global cap
    if selected_camera_index is None:
        messagebox.showerror("Error", "Please select a camera first.")
        return
    cap = cv2.VideoCapture(selected_camera_index)
    if not cap.isOpened():
        messagebox.showerror("Error", f"Could not open camera {selected_camera_index}")
        return

    start_time = datetime.datetime.now().timestamp()
    try:
        while datetime.datetime.now().timestamp() - start_time < 3:
            ret, frame = cap.read()
            if not ret:
                break
            cv2.imshow(f"Preview of Camera {selected_camera_index}", frame)
            cv2.setWindowProperty(f"Preview of Camera {selected_camera_index}", cv2.WND_PROP_TOPMOST, 1)
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break
    finally:
        cap.release()
        cv2.destroyAllWindows()
```

- **What it does**:
    - This shows a quick 3-second preview of the camera you selected and can end early by pressing q.
    - You can see what the camera will record before you start.
- **Why it's important**: It helps you check that the camera is working and showing what you want to record.

5. `process_frame(window)`

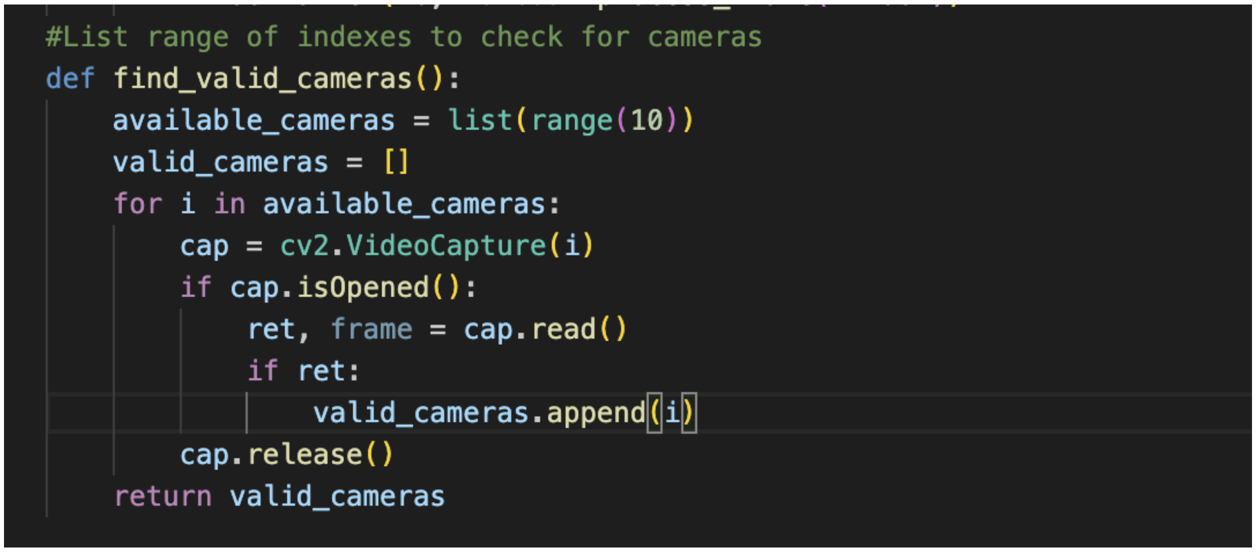


```python
#Process each frame recorded with time stamps real time
def process_frame(window):
    global out, is_recording
    if is_recording:
        ret, frame = cap.read()
        if not ret:
            stop_camera_recording()
            return
        timestamp = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S.%f")[:-3]
        cv2.putText(frame, timestamp, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 255), 2)
        cv2.imshow('Camera Feed', frame)
        out.write(frame)
        window.after(10, lambda: process_frame(window))
```

- **What it does**:
    - While recording, this part processes (captures) each frame from the camera and adds the current time and date to the video (as text in the corner of the screen).
    - It saves each frame to the video file.
- **Why it's important**: This ensures that each frame is saved with the correct timestamp while you're recording.

6. `find_valid_cameras()`

- **What it does**:
    - This checks all possible camera options (like camera 0, camera 1, camera 2) to see which ones are available.
    - It then creates a list of all the cameras that are working and ready to use.
- **Why it's important**: It helps the program figure out which cameras are plugged in and working so you can select one.



```python
#List range of indexes to check for cameras
def find_valid_cameras():
    available_cameras = list(range(10))
    valid_cameras = []
    for i in available_cameras:
        cap = cv2.VideoCapture(i)
        if cap.isOpened():
            ret, frame = cap.read()
            if ret:
                valid_cameras.append(i)
        cap.release()
    return valid_cameras
```

Some Important Things You Can Change

- **Changing the Video File Format**:
    - Right now, the video is saved as `.mov`. If you want to save it as `.mp4` or another format, you can change the part where it says `camera_output_file = f"output_{timestamp}.mov"` to `camera_output_file = f"output_{timestamp}.mp4"`.
- **Recording Time**:

    - You can change how long the preview lasts by adjusting the `3` in this part: `while datetime.datetime.now().timestamp() - start_time < 3:`. For example, change the `3` to `5` if you want the preview to last for 5 seconds.

- **Video Resolution**:
    - The program automatically uses your camera's default resolution. If you want to set a specific resolution, you can change the `width` and `height` values in the `start_camera_recording` function. For example, to record at 1280x720, change this line:
      width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
      height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
      To
      Width = 1280
      Height = 720

- **Camera Index**:
    - The program currently checks over 10 index's to open cameras that may be connected to your machine but if you had more then 10 cameras connected or you are sure your connected camera definitely works but is not showing up as a valid camera in the drop down list you can adjust this line:
      Available_cameras = list(range(10)) to a higher index say 100 instead of 10 to try locate the camera connected