Screen Recording Application Code Documentation

This document provides a detailed explanation of the Python screen recording application, built using tkinter for the GUI, Pillow for screen capture, OpenCV for video encoding, PyAudio for microphone input, and FFmpeg for merging video and audio streams.

1. ScreenRecorderApp Class

This is the main application class that manages the GUI and coordinates screen and audio recording.

__init__(self, master)

- **Purpose:** Initializes the main application window and its components.
- **Parameters:**
  - master: The root tkinter window instance.
- **Functionality:**
  - Sets up the main window's title, size, and prevents resizing.
  - Configures ttk.Style for a modern UI appearance.
  - Initializes various state variables (is_recording, is_paused, recording_thread, audio_thread, file paths, etc.).
  - Sets default fps and highlight_mouse_var (mouse highlight option).
  - Initializes PyAudio parameters (FORMAT, CHANNELS, RATE, CHUNK).
  - Creates and packs all UI elements:
    - status_label: Displays current application status.
    - start_button: Initiates the recording process.
    - pause_button: Toggles recording pause/resume.
    - stop_button: Stops the recording.
    - options_frame: A LabelFrame to group recording settings.
    - fps_combobox: Allows selecting the video frame rate.
    - highlight_mouse_checkbox: Toggles mouse cursor highlighting.
    - audio_options_frame: A LabelFrame to group audio settings.
    - no_audio_radio, mic_audio_radio: Radio buttons for audio source selection.
    - mic_device_combobox: Dropdown to select an available microphone device.

        - open_folder_button: Opens the output directory of the saved video.
    - Calls _initialize_pyaudio() to set up audio devices.
    - Sets up a protocol to handle window closing gracefully.

_initialize_pyaudio(self)

- **Purpose:** Detects available microphone input devices and populates the mic_device_combobox.
- **Functionality:**
    - Initializes pyaudio.PyAudio().
    - Iterates through detected audio devices to find those with input channels.
    - Stores microphone device names and their indices in self.mic_devices_map.
    - Updates the mic_device_combobox with detected microphone names.
    - Sets the first detected microphone as default, or displays "No Microphones Found".
    - Disables microphone-related UI elements if no microphones are found or if PyAudio initialization fails.

_update_audio_controls(self)

- **Purpose:** Enables or disables the microphone device selection dropdown based on whether "Microphone" audio source is selected.

start_recording(self)

- **Purpose:** Begins the screen recording setup process.
- **Functionality:**
    - Checks if a recording is already in progress.
    - Retrieves and sets the selected FPS value from the UI.
    - Hides the main application window (self.master.withdraw()).
    - Creates and displays an AreaSelectionWindow instance to allow the user to select the recording region.

_on_area_selected(self, area)

- **Purpose:** Callback method invoked by AreaSelectionWindow once a recording area is selected or the selection is cancelled.
- **Parameters:**

    - area: A tuple (x, y, width, height) representing the selected screen region, or None if cancelled.
- **Functionality:**
    - Shows the main application window again (self.master.deiconify()).
    - If an area is provided, stores it in self.recording_area and calls _start_countdown_and_record().
    - If area is None, displays a cancellation message.

_start_countdown_and_record(self)

- **Purpose:** Initiates a 3-second countdown before the actual recording begins.
- **Functionality:**
    - Sets self.countdown_active to True.
    - Disables all relevant UI buttons and options to prevent changes during countdown/recording.
    - Calls _update_countdown() to start the visual countdown.

_update_countdown(self, count)

- **Purpose:** Recursively updates the countdown status label.
- **Parameters:**
    - count: The current countdown value.
- **Functionality:**
    - Updates status_label with the current count.
    - Uses self.master.after() to schedule itself to run again after 1 second.
    - When count reaches 0, sets self.countdown_active to False and calls _start_recording_process().

_start_recording_process(self)

- **Purpose:** Handles the actual initiation of video and audio recording threads after the countdown.
- **Functionality:**
    - Generates temporary file paths for raw video (.avi) and audio (.wav).
    - Prompts the user to select the final output .mp4 filename and location.
    - If a file is selected, sets self.is_recording to True, updates button states, and changes status_label.
    - Starts the _record_screen method in a new threading.Thread.
    - If "Microphone" audio is selected, starts the _record_audio method in another threading.Thread.

toggle_pause(self)

- **Purpose:** Toggles the recording state between paused and resumed.
- **Functionality:**
  - Updates self.is_paused boolean.
  - Changes the pause_button text ("Pause" / "Resume").
  - Updates the status_label.
  - Stops or starts the PyAudio stream if audio recording is active.

_record_screen(self)

- **Purpose:** Captures screen frames and writes them to a raw video file (.avi). This runs in a separate thread.
- **Functionality:**
  - Defines the MJPG codec for cv2.VideoWriter.
  - Enters a loop that runs while self.is_recording is True.
  - If not self.is_paused:
    - Captures a screenshot of the self.recording_area using PIL.ImageGrab.grab().
    - Converts the image to an OpenCV-compatible NumPy array.
    - If highlight_mouse_var is True, gets the current mouse position using pyautogui.position() and draws a red circle on the frame.
    - Writes the processed frame to the self.out (VideoWriter object).
    - Updates the FPS status in the status_label.
  - Includes time.sleep() to control the frame rate and prevent busy-waiting when paused.
  - Handles exceptions during recording and calls stop_recording() on error.

_record_audio(self)

- **Purpose:** Captures audio from the selected microphone device and stores it in self.audio_frames. This runs in a separate thread.
- **Functionality:**
  - Opens a PyAudio input stream using the selected microphone device.
  - Enters a loop that runs while self.is_recording is True or self.is_paused is True.
  - If not self.is_paused, reads audio data chunks from the stream and appends them to self.audio_frames.
  - Includes a small time.sleep() during pause to prevent busy-waiting.
  - Closes the audio stream when recording stops or an error occurs.

stop_recording(self)

- **Purpose:** Stops both video and audio recording, finalizes the video file, and cleans up temporary files.
- **Functionality:**
  - Sets self.is_recording and self.is_paused to False.
  - Waits for both recording_thread and audio_thread to finish using join().
  - Releases the cv2.VideoWriter object.
  - If microphone audio was recorded, it saves the self.audio_frames to a temporary .wav file.
  - Calls _merge_video_audio() to combine the raw video and audio into a final .mp4 file. If no audio was recorded or merging fails, it renames the raw video to the final output path.
  - Deletes temporary raw video and audio files.
  - Resets UI button states and re-enables options.

_merge_video_audio(self)

- **Purpose:** Uses the FFmpeg command-line tool to merge the raw AVI video and WAV audio into a final MP4 file.
- **Functionality:**
  - Constructs the FFmpeg command with input video, input audio, video copy, and AAC audio encoding.
  - Executes the FFmpeg command using subprocess.run().
  - Handles success by showing an info message.
  - Handles FileNotFoundError if FFmpeg is not in PATH, and other exceptions during merging, providing informative error messages and keeping the raw video file in case of failure.
  - On Windows, uses subprocess.STARTUPINFO to prevent a console window from popping up.

open_output_folder(self)

- **Purpose:** Opens the directory where the final recorded video is saved.
- **Functionality:**
  - Determines the folder path of self.final_output_filename.
  - Uses os.startfile (Windows), subprocess.Popen(["open", ...]) (macOS), or subprocess.Popen(["xdg-open", ...]) (Linux) to open the folder in the system's file explorer.

on_closing(self)

- **Purpose:** Handles the main window's close event, prompting the user if recording is active.
- **Functionality:**
  - If recording or countdown is active, asks for confirmation to stop and quit.
  - If confirmed, calls stop_recording() and self.master.destroy().
  - Ensures self.p.terminate() is called to properly close PyAudio resources.

2. AreaSelectionWindow Class

This class provides a transparent overlay window for the user to select a rectangular screen region to record.

__init__(self, parent, callback)

- **Purpose:** Initializes the transparent selection window.
- **Parameters:**
  - parent: The parent tkinter window (usually the ScreenRecorderApp's master).
  - callback: A function in the parent class to call with the selected area or None on cancellation.
- **Functionality:**
  - Creates a tk.Toplevel window.
  - Removes window decorations (overrideredirect(True)), sets transparency (attributes('-alpha', 0.3)), and keeps it on top (attributes('-topmost', True)).
  - Attempts to set the window geometry to cover the entire virtual screen (all monitors combined) using ImageGrab.grab(). Includes basic error handling for screen detection, falling back to primary monitor dimensions if needed.
  - Creates a tk.Canvas to draw the selection rectangle.
  - Initializes variables for tracking mouse drag coordinates.
  - Binds mouse events (<ButtonPress-1>, <B1-Motion>, <ButtonRelease-1>) for drawing the rectangle.
  - Binds the <Escape> key to cancel the selection.
  - Adds "Confirm" and "Cancel" buttons at the bottom right of the overlay.

on_button_press(self, event)

- **Purpose:** Records the starting coordinates of the mouse drag and initializes a

red dashed rectangle on the canvas.

on_mouse_drag(self, event)

- **Purpose:** Continuously updates the dimensions of the dashed rectangle as the mouse is dragged.

on_button_release(self, event)

- **Purpose:** Finalizes the selected rectangle when the mouse button is released.
- **Functionality:**
  - Normalizes the coordinates (ensures x1 < x2, y1 < y2).
  - Calculates current_x, current_y, current_width, current_height.
  - Performs a basic check to ensure the selected area is not zero-sized.
  - Deletes the dashed rectangle and draws a solid blue rectangle to highlight the final selection.
  - Makes the window less transparent temporarily for better visibility of control buttons.

confirm_selection(self)

- **Purpose:** Calls the callback function with the selected recording area tuple and destroys the selection window. Includes a check to ensure a valid area has been selected.

on_cancel(self, event=None)

- **Purpose:** Calls the callback function with None (indicating cancellation) and destroys the selection window. Can be triggered by the "Cancel" button or the Escape key.

3. Main Execution Block

if __name__ == "__main__":

- **Purpose:** Ensures the application runs only when the script is executed directly.
- **Functionality:**
  - Creates the root tkinter window (tk.Tk()).

- Instantiates ScreenRecorderApp.
- Starts the tkinter event loop (root.mainloop()), which keeps the GUI responsive and handles events.