Camera Recording Script with Tkinter GUI

This Python script uses OpenCV to interact with a camera and Tkinter for a simple graphical user interface (GUI). It allows the user to select a camera, preview it for 3 seconds, and start/stop recording video with real-time timestamps. The recorded video is saved in `.mov` format.

Global Variables

- `cap`: Stores the camera capture object created by OpenCV.
- `out`: Stores the video writer object used to save the recorded frames.
- `is_recording`: A boolean that tracks whether the camera is currently recording.
- `selected_camera_index`: Stores the index of the camera selected by the user.
- `camera_output_file`: Stores the name of the output file for the recording.

Functions

`initialize_camera(camera_index)`

- **Parameters**:
  - `camera_index` (int): The index of the camera to initialize.
- **Description**:
  - This function initializes the camera by creating a `VideoCapture` object using the provided index. If the camera cannot be opened, it shows an error message and returns `False`. On success, it returns `True`.

`start_camera_recording(selected_camera_index, camera_var, window)`

- **Parameters**:
  - `selected_camera_index` (int): The index of the selected camera.
  - `camera_var`: Camera variable (used for GUI control).
  - `window`: Tkinter window object for GUI frame updates.
- **Description**:
  - Starts recording video from the camera. It first checks if a camera is already recording, then verifies if a camera is selected. The function uses the current timestamp to create a unique filename for the output file (`output_<timestamp>.mov`). It records frames in `.mp4` format but saves it as `.mov`. The `process_frame` function is called to handle frame recording.

`stop_camera_recording()`

- **Parameters**: None.
- **Description**:
    - Stops the recording process by releasing both the camera and video writer objects. Also closes any OpenCV windows used during recording.

`preview_camera(selected_camera_index)`

- **Parameters**:
    - `selected_camera_index` (int): The index of the selected camera.
- **Description**:
    - Provides a 3-second preview of the selected camera. It displays the camera feed in a new window for the user to verify the camera's functionality. The window automatically closes after 3 seconds or can be closed manually by pressing 'q'.

`process_frame(window)`

- **Parameters**:
    - `window`: Tkinter window object for updating frames.
- **Description**:
    - This function captures each frame from the camera and adds a real-time timestamp to it. The timestamp is displayed on the frame and the frame is saved to the output file. The function uses Tkinter's `after()` method to call itself recursively at a 10-millisecond interval to continuously process and save frames.

`find_valid_cameras()`

- **Parameters**: None.
- **Description**:
    - Scans through camera indices (from 0 to 9) to check which cameras are available. If a camera is available and can capture frames, its index is added to a list of valid cameras, which is returned by the function.

Additional Notes

- **Dependencies**:
    - The script relies on the following Python libraries:
        - `cv2` (OpenCV): For interacting with the camera and processing video frames.
        - `tkinter`: For creating the graphical user interface (GUI).
        - `datetime`: For generating timestamps and naming the output file.
- **Error Handling**:

    - The script checks whether a camera is connected before starting a recording. If a camera cannot be opened, it shows an error message to the user.
- **Video Format**:
    - The script records video using the `mp4v` codec and saves it as `.mov` files.
- **Real-time Timestamp**:
    - The recorded frames have real-time timestamps displayed on them, formatted as `YYYY-MM-DD HH:MM:SS.mmm`.