DaVinci Resolve Batch Rendering Script

This Python script automates the process of rendering multiple timelines in a DaVinci Resolve project. It allows users to choose between different render formats, checks the required write permissions and inserts timelines into the render queue for batch processing. Below you will find a detailed summary of the functions and use of the programme.

Prerequisites

- DaVinci Resolve must be open before you run this script.
- Make sure that the specified render directories have write permissions.

Features

1. Render format selection: The script allows the user to choose between the ProRes and H.264 render formats.
2. Write permission check: The script checks whether the target directory has the necessary write permissions for rendering.
3. Load render preset: Loads the appropriate render preset based on the selected format.
4. Batch Timeline Rendering: Adds all timelines in the project to the render queue and starts the rendering process.

Script workflow

1. DaVinci Resolve Initialisation:
 - The script initialises the API connection with DaVinci Resolve.
 - Retrieves the project manager, the current project, the media storage and the media pool.

2. Proxy mode handling:
 - Disables proxy mode to ensure full resolution rendering.

3. Render format selection:
 - Prompts the user to choose between ProRes and H.264 formats.
 - Depending on the selection, specifies the corresponding render preset and the target directory.

4. Check write authorisation:
 - Checks whether the script has write authorisation for the render target directory.
 - Terminates the script if the directory does not have the required authorisations.

5. Load render preset:
 - Loads the selected render preset.
 - Terminates if the preset cannot be loaded.

6. Batch rendering:
 - Retrieves the number of timelines in the project.
 - Adds each timeline with the corresponding settings to the render queue.
 - Starts the rendering process.

Instructions for use

1. Make sure that DaVinci Resolve is open:
 - Start DaVinci Resolve before executing the script.

2. Run the script:
 - Execute the script in a Python environment.

3. Follow the prompts:
 - Select the rendering format (ProRes or H.264).
 - Make sure that the target directory has write permissions.

4. Rendering:
 - The script adds all timelines to the render queue and starts rendering.

Example for the execution

1. Initialise DaVinci Resolve:
 - Make sure that DaVinci Resolve is open and a project is loaded.

2. Run the script:
 ```bash
 python render_timelines.py
 ```

3. Select the render format:
 ```
 Select your render format:
 1. ProRes
 2. H.264
 Enter the number that corresponds to your choice: 1
 ```

4. Result:
 ```
   Render location has write access. Proceeding with rendering...
 There are 3 timelines to be added to the render queue
   Currently adding timeline 1 to the render queue...
   Currently adding timeline 2 to the render queue...
   Currently adding timeline 3 to the render queue...
 Start rendering...
 Rendering started.
 ```

Script details

Functions

- check_write_permission(directory):
 - Checks whether the specified directory has write permission by attempting to create and delete a temporary file.

- add_timeline_to_render(timeline, target_dir, timeline_name):
 - Configures the render settings and adds the specified timeline to the render queue.

Main execution flow

- DaVinci Resolve initialisation:
 - Establishes a connection to DaVinci Resolve and retrieves the required project components.

- Proxy mode handling:
 - Disables proxy mode to ensure high quality rendering.

- User interaction:
 - Prompts for the render format and checks the write permissions for the target directory.

- Load render preset:
 - Loads the appropriate render preset of the user's choice.

- Batch rendering:
 - Adds all timelines to the render queue and starts the rendering process.

Conclusion

This script provides an efficient way to batch render multiple timelines in DaVinci Resolve to save time and ensure consistency. By processing user input and checking the necessary conditions before rendering, it provides a robust and user-friendly solution for automated project rendering.