

[<< Back](#)

# Pipeline

## Root

### Drive structure

Real drives

| Drive letter | Function                                        |
|--------------|-------------------------------------------------|
| C:           | Windows drive                                   |
| D:           | Sync work drive here + other work related stuff |
| E:           | Sync all media here (foto, music, video)        |

Mapped drives

| Drive letter | Function                             |
|--------------|--------------------------------------|
| I:           | Active projects:                     |
|              | Audio                                |
|              | Photo                                |
|              | Visual                               |
|              | VJ                                   |
|              | Web                                  |
| M:           | Music                                |
| P:           | Files for software (settings, adons) |
| F:           | Personal files                       |

From the command line:

```
subst A: /d
```

Replace A with drive you want deleted

## Startup

### Create Drives

Using VisualSubst

### Open programs on correct desktop

Put this .ps1 file in any local location on your pc, navigate to your Windows startup folder

(shell:startup), right-click → new → shortcut, and insert this into the textbox:  
powershell.exe -NoProfile -ExecutionPolicy Bypass -File  
"C:\Users\flori\4\_StartupScript\_MoveToVirtualDesktop1.ps1"

## Audio

### Folder structure

#### Artwork finder

- MusicProduction
  - Practice
  - Ideas
  - InProgress
  - FinalTouches
  - Finished
  - Publish
- DJSets
- Library
  - Samples
    - AmbiguousRoyaltySounds
    - FreeUse

### Configure MusicBee and Traktor for EZ playlist management

**MusicBee:** Ensure that MusicBee is consistently exporting the "M:\iTunes Music Library.xml" file on close. You've already set this up, but double-check under MusicBee's Preferences > Library > "export the library as an iTunes formatted XML file" is enabled and pointing to "M:\iTunes Music Library.xml".

**Traktor:** In Traktor Preferences → File Management:

Set the "iTunes/Music Library" path to "M:\iTunes Music Library.xml". This ensures Traktor always looks at your MusicBee-exported XML.

Enable "Import Music-Folders at Startup" and add the folder(s) where your actual music files reside (e.g., M:\Music\ or wherever your tracks are stored). This keeps the Track Collection updated with new files, though it won't handle playlists directly.

Fire up the Python script:

It enables synchronisation of playlists from any iTunes.xml compatible music library to any DJ software of your choice.

Here's what it does:

1. It reads an iTunes XML library file to extract tracks from user playlists
2. It copies these tracks to a designated "DJ Library" folder
3. If enabled, it converts audio files to WAV format using FFmpeg

4. It creates a new XML file with updated file paths that point to the DJ Library folder

Key features:

- Filters tracks to only include supported audio formats (.mp3, .flac, .wav, etc.)
- Skips system and auto-generated playlists
- Can skip files that already exist in the destination folder
- Provides detailed analysis of missing tracks for debugging
- Updates file locations in the XML for compatibility with DJ software

The script runs through three main steps:

1. Analyzing the iTunes library to extract playlist tracks
2. Copying and converting tracks to the DJ Library folder
3. Creating an updated XML file with the new file paths

This allows a you to maintain a playlists in iTunes/Musicbee while having those same tracks available in any DJ software in the preferred format.

## Photo

### Folder structure

- Year
  - 01\_Incoming
  - 02\_Outgoing

### File structure

Template: YYYY-MM-DD\_Location\_Subject(abbreviated, PubLib (Public Library))\_FileVersion.Extention

Example: 2020-12-19\_Zele\_PubLib\_001.jpg

## Visual

### Folder structure

- YYYY-MM-DD\_NameClient\_NameProject
  - \_Library
  - 00\_Incoming
    - 01\_Preproduction
    - 02\_Production
  - 01\_Preproduction
    - 01\_Moodboard
    - 02\_Storyboard
  - 02\_Production
    - \_Blender

- \_DaVinci
- \_Houdini
- Scene\_01
- 03\_Outgoing
  - YYYY-MM-DD

## File structure

### Naming Convention

[CamelCaps](#), geen spaties, underscores/streepjes enkel gebruiken bij opdeling filenaam.  
Maak genoeg iteraties van je bestand, zodat je steeds een versie terug kan gaan, moest er iets mislopen.

### Models

Export files (FBX, OBJ) hebben steeds zelfde versienummer als work file (Blender, Maya).  
Afkortingen:

- Character: CH
- Prop: P
- Foilage: F

Template: [Project\\_SceneNumber\\_Modeltype-ModelName\\_FileVersion.Extension](#)

Example: [Silver\\_000\\_CH-Nora\\_v001.blend](#)

### Texture Maps

Export files (PNG, JPG) hebben steeds zelfde versienummer als work files (Substance, Photoshop)  
Afkortingen:

- Diffuse: DIFF
- Ambient Occlusion: AO
- Glossiness: G
- Normal: N
- Roughness: R
- Subsurface Scattering: SSS
- Metallic: M

Template: [Project\\_SceneNumber\\_TextureName\\_FileVersion\\_TextureType.Extension](#)

Example: [Silver\\_010\\_NoraHair\\_v001\\_DIFF.PNG](#)

### Scenes

The count of scenes starts from 010, models with the scene number 000 are not restricted to one

scene.

Template: Project\_SceneNumber-Description\_FileVersion.Extension

Example: Silver\_010-MudWorld\_v001.blend

## In 3D Software

Max 5 letters voor afkortingen

Afkortingen:

- Group: GRP
- Mesh: MESH
- Curve: CRVE
- Joint: JNT
- Light: LGHT
- Camera: CAM
- Locator: LOC
- Controller: CTRL
- Low Poly: LP
- High Poly: HP
- Forces: FRCE

Template: Type\_Name\_PolyType

Example: GRP\_Nora\_LP

## Reference Library

For my collection of reference images, I only use broad categories as folders, the other information I put in the name and metadata.

I'm using a combination of tools, including Advanced Renamer, digiKam, Python and a tool from GitHub called [taggui](#).

### Workflow

First, I will change the name with Advanced Renamer.

Using TagGUI, I will then let the program generate captions into separate text files, which are named after the image they are describing.

Now we need the Python script to take the information from the text file, and add it to the metadata of the images.

For clean organisation, I opted to convert all files to .jpg. This to make sure the metadata is displayed in a clean way.

### The name

CATEGORY\_SUBJECT\_inc Nr

## The metadata

A description/caption in the images "Title" and "Subject" fields.

## The script

```
1. import os
2. from PIL import Image as PilImage, PngImagePlugin
3. from pyexiv2 import Image as Pyexiv2Image
4.
5. def add_text_to_image_metadata(image_folder):
6.     try:
7.         # Iterate through all files in the specified folder
8.         for filename in os.listdir(image_folder):
9.             image_path = os.path.join(image_folder, filename)
10.            text_file_path = os.path.join(image_folder,
11. f"{os.path.splitext(filename)[0]}.txt")
12.
13.            print(f"Processing {image_path}") # Debugging line
14.
15.            if os.path.exists(text_file_path):
16.                # Read the content from the text file
17.                with open(text_file_path, 'r') as file:
18.                    caption_text = file.read()
19.
20.                # Convert filename to lowercase for extension checking
21.                filename_lower = filename.lower()
22.
23.                if filename_lower.endswith(('.jpg', '.jpeg')):
24.                    # Load the image's metadata
25.                    image = Pyexiv2Image(image_path)
26.
27.                    # Add the caption to the image's EXIF UserComment
28.                    and ImageDescription fields
29.                    image.modify_exif({'Exif.Photo.UserComment':
30. caption_text, 'Exif.Image.ImageDescription': caption_text})
31.                elif filename_lower.endswith('.png'):
32.                    # Convert the PNG image to JPEG
33.                    image = PilImage.open(image_path)
34.                    rgb_image = image.convert('RGB')
35.                    rgb_image.save(os.path.splitext(image_path)[0] +
36. '.jpg', 'JPEG')
37.
38.                    # Load the image's metadata
39.                    image =
40. Pyexiv2Image(os.path.splitext(image_path)[0] + '.jpg')
41.
42.                    # Add the caption to the image's EXIF UserComment
```

```
    and ImageDescription fields
38.         image.modify_exif({'Exif.Photo.UserComment':
caption_text, 'Exif.Image.ImageDescription': caption_text})
39.
40.         # Delete the original PNG image
41.         os.remove(image_path)
42.
43.         # Delete the text file
44.         os.remove(text_file_path)
45.
46.         print("Processing completed for all images.")
47.     except Exception as e:
48.         print(f"Error: {e}")
49.
50. # Example usage: Specify the folder containing your images
51. image_folder_path = '/path/to/your/source/folder'
52. add_text_to_image_metadata(image_folder_path)
53. }();
```

## Sources

[https://www.youtube.com/watch?v=6KCtPnam6Sk&ab\\_channel=TimvanHeldsdingen](https://www.youtube.com/watch?v=6KCtPnam6Sk&ab_channel=TimvanHeldsdingen)

<https://github.com/alexanderrichtertd/plex/wiki/Project-Structure>

<https://www.youtube.com/watch?v=o2LqKH6ahDA>

Pipeline @ Blender

PSVirtualDesktop - commandlets to manage virtual desktops

From:

<http://floriandheer.com/wiki/> - **Brain II**

Permanent link:

<http://floriandheer.com/wiki/doku.php?id=start:knowledge:vfx:pipeline&rev=1741617094>

Last update: **2025/03/10 15:31**

