3D Modeling Project Organization and Versioning System

This document describes the folder structure, file naming conventions, and versioning system used for organizing 3D modeling projects. The goal of this system is to maintain a consistent and logical organization that makes it easy to manage and navigate projects.

Folder Structure

The base folder structure for each project is as follows:

```
/BASE_FOLDER
├─ 00_base_folder_version.txt
├─ 00_Version_Logs
│   ├─ parts_versions_log.CSV
│   └─ branches_info.CSV
├─ 01_Models
│   ├─ 00_Blender
│   ├─ 01_Rhino
│   └─ 02_Export
│       ├─ 00_STL
│       ├─ 01_OBJ
│       ├─ 02_STEP
│       └─ 03_Vector
├─ 02_Renders
│   ├─ 00_Base_Render_Scene
│   │   └─ Base_Render_Scene.blend
│   └─ 01_Renders
├─ 03_Video
├─ 04_Photos
├─ 05_Assets
├─ 06_Documents
├─ 07_Notes
└─ 08_References
```

The "base_folder_version.txt" file, located in the root of the base folder, must be named as the current base folder version, inside it must have a description of the last changes made. For example:

```
Version: 1.01.001A
Date: YYYY-MM-DD
Changes: Initial version of the base folder.
```

Folder Descriptions

- **00_Version_Logs**: Contains version logs and branching information for each project. Use a spreadsheet for logging part versions and a text file for documenting branches.
- **01_Models**: Contains subfolders for Blender, Rhino, and exported files. Each subfolder is organized by file format (STL, OBJ, STEP, Vector, etc.).
- **02_Renders**: Organized into subfolders for different views, materials, and lighting setups. The base Blender scene file (scenario.blend) used for rendering is located in the 00_Scenario subfolder.
- Other folders (**03_Video**, **04_Photos**, **05_Assets**, **06_Documents**, **07_Notes**, **08_References**): Store additional project files, assets, and references as needed.

File Naming Conventions

Assemblies and Parts

In this documentation, the term "assembly" refers to the main file that contains all the parts of your project. The term "parts" refers to the individual components within the assembly. The naming conventions for assemblies and parts are based on semantic versioning. If you're not familiar with semantic versioning, you can learn more about it by visiting the official SemVer website.

Assemblies

Use semantic versioning for assembly files. Name each file as follows:

assembly_X.Y.Z🐔.filetype

- **X.Y.Z**: Version number of the main file, following the Major.Minor.Patch format.
- 🐔: Branch identifier (A, B, C, etc.) for when multiple branches of a project are created.
- **filetype**: File format extension (e.g., **.blend**, **.3dm**).

For example, a file name with a branch could look like this: **assembly_1.01.001A.blend**

Parts

For individual parts, use the following naming convention:

part_name_X.Y.Z🐔.filetype

- **X.Y.Z**: Version number of the part, following the Major.Minor.Patch format.
- 🐔: Branch identifier (A, B, C, etc.) for when multiple branches of a project are created.
- **filetype**: File format extension (e.g., **.stl**, **.obj**).

For example, a file name with a branch could look like this: **partA_1.01.001A.stl**

Renders

Name render files (images and videos) using the following convention:

model_name_x.y.z🐔_S.x.y.c🐔_Vx.y🐔Rx.y🐔.filetype

- **model_name_x.y.z🐔**: Model name, followed by its version (either the assembly or the part) and branch identifier used for the render.
- **Sx.y.z🐔**: Base render scene version and branch identifier used for the render.
- **Vx.y🐔**: Project render version number followed by a branch identifier (e.g., "R1.0A", "R1.1B").
- **Rx.y🐔**: Render version number followed by a branch identifier (e.g., "R1.0A", "R1.1B").
- **filetype**: File format extension (e.g., **.png**, **.jpg**).

For example, a file name with a branch could look like this:
model_1.01.001A_S1.01.001B_V1.01A_R1.01A.png

Document branching information in a text file within the **00_Version_Logs** folder, including the branch identifier, purpose, and any relevant notes.

Version Logging

The version logging system is implemented to keep track of the versions and branches of your project files. It helps you manage your workflow and ensures that you have a clear record of changes and developments throughout the project.

Maintaining version logs is crucial for tracking changes and updates to individual parts and files throughout the project. Use the following strategies to simplify the logging process:

1. **Parts Versions Log (parts_versions_log.csv)**: This CSV file, located in the **00_Version_Logs** folder, is used to log the version numbers of each part in the project. Whenever you make changes to a part, update the corresponding row in the CSV file. Include information such as:
   - Part name
   - Version number (in the format MAJOR.MINOR.PATCH)
   - Date of modification
   - Description of changes made
   - Name of the software used for modification (Blender or Rhino)
2. Integrate logging into your workflow by updating the log as soon as you make changes to a part or file.
3. Use templates or snippets with the basic structure of a log entry to make it easier to fill in the required information for each new entry.

Example of a parts versions log entry:

Part Name	Version	Date	Description	Software	Branch
part1	1.0.001	2023-04-10	Initial design	Blender	A
part2	1.0.001	2023-04-11	Initial design	Rhino	A
part1	1.1.002	2023-04-12	Updated dimensions	Blender	A

Part Name	Version	Date	Description	Software	Branch
part2	1.0.002	2023-04-13	Minor tweaks in geometry	Rhino	A

2. **Branches Log (branches_info.csv)**: This CSV file, located in the **00_Version_Logs** folder, is used to log branching information for your project. Include the following information:
    - Branch name
    - Parent branch (if applicable)
    - Date of creation
    - Description of the purpose of the branch
    - Name of the software used for modification (Blender or Rhino)

Example of a branches log entry:

Branch Name	Parent Branch	Date	Description	Software
A	Main	2023-04-10	Initial project branch	Blender
B	A	2023-04-15	Alternative design exploration	Rhino
C	A	2023-04-20	Testing new materials	Blender

By maintaining these logs consistently, you will have a clear record of the development and evolution of your project. This makes it easier to track changes, identify issues, and collaborate with others.

Archiving Versions:

When a significant file within the base folder is changed or replaced (for example, the base render scene), a snapshot of the folder or the file should be created and stored in an "Archive" subfolder within the relevant folder. The archived version should be named appropriately to reflect its version number. This will allow for easy reference or rollback if necessary.

Branching

Branching refers to the practice of creating different versions or paths in a project to explore alternative designs, features, or solutions while preserving the original version. This allows for experimentation and comparison between various iterations of a project.

When creating multiple branches of a project, use a letter identifier (A, B, C, etc.) at the end of the version number to differentiate between branches. Document branching information in a text file (branches_info.CSV) within the 00_Version_Logs folder, including the branch identifier, purpose, and any relevant notes. Describe the objective of the branch and any significant differences from the main project or other branches. Include the following information:

- Branch name
- Parent branch (if applicable)
- Date of creation
- Description of the purpose of the branch
- Name of the software used for modification (Blender or Rhino)

Handling Multiple Parts in a Single File

When working with multiple parts in a single file, it's essential to manage the changes made to both the main file (the assembly) and the individual parts. To make this process easier to understand, we'll use an example:

Imagine you have a 3D modeling project with two parts, Part A and Part B, both within the same file. To keep track of the changes, you can use a versioning system for each part and the main file.

Main File Versioning

The main file's version number should be updated to represent the overall progress of the entire assembly. For example:

- Version 1.0.0: Initial design of the assembly with Part A and Part B.
- Version 1.1.0: Made significant changes to the assembly, such as adding new connections or modifying the layout.

Parts Versioning

Each part should have its own version number that tracks its specific updates:

Part A

- Version 1.01.001: Initial design of Part A.
- Version 1.02.001: Updated dimensions of Part A.

Part B

- Version 1.01.001: Initial design of Part B.
- Version 1.01.002: Minor tweaks in geometry of Part B.

In this example, it's normal for some parts to have a higher version number than the main file, as they may have undergone more frequent updates. Just remember that the main file's version number reflects the assembly's overall progress, while the parts' version numbers focus on their unique changes.

When exporting the parts, use their individual version numbers in their filenames, e.g., "partA_1.02.001.stl" and "partB_1.01.001.stl".

By maintaining separate versioning systems for the main file and individual parts, you can easily understand how the assembly and its components have evolved over time. This will help you manage your project more effectively and collaborate with others.

Customizing the Folder Structure

The folder structure and naming conventions provided in this documentation serve as a general guideline for organizing 3D modeling projects. However, you may need to modify the structure or add new folders to fit the specific requirements of your project. If you need to add new folders, follow these guidelines:

1. Add the new folder in the appropriate location within the existing folder hierarchy.
2. Name the new folder using the existing numbering system (e.g., 09_New_Folder) to maintain consistency.

Remember that the primary goal of this organizational system is to maintain a clear and logical structure that is easy to navigate and manage. Make sure that any modifications you make still align with this goal.

Base Folder Logs

It's essential to keep track of the changes made to your base project folder structure, especially when these changes affect the way you organize and manage your individual projects. To be able to keep track of the changes made to the base folder an "Archive" folder must exist in the same directory as the base folder. logs and snapshots of the folder structure will be stored on this archive. Here's how to handle versioning and logs for the base folder:

Versioning:

The version of the base folder is recorded in the "base_folder_version.txt" file that is stored in the root of the base folder. Whenever a change is made to the base folder structure, or to files within it, the version number should be updated following semantic versioning rules.

Archiving Versions:

When a significant change is made to the base folder structure, or one of it's files, a snapshot of the folder should be created and stored in the "Archive" folder. The archived version should be named appropriately to reflect its version number. This will allow for easy reference or rollback if necessary. The "base_versions_log.csv" file must also be updated to log the changes made.

Logging Changes:

For minor changes to the base folder (like adding, moving, or deleting a folder), a log entry should be made in "base_versions_log.csv" stored in "00_Version_Logs". This CSV file will include the date of the change, the new version number, and a description of the changes made.

Folder Structure Snapshots:

With every update to the base folder structure, include a text-based snapshot of the updated folder structure in your log. This will give a clear and immediate picture of what the folder looked like at each version, further facilitating tracking of changes and understanding of the folder's evolution over time.

the text-based snapshot should look like this:

```
/base folder v1.01.001A
├─ 00_base_folder_version.txt
├─ 00_Version_Logs
│   ├─ parts_versions_log.csv
│   └─ branches_info.txt
├─ 01_Models
├─ 02_Assemblies
.
.
```

Special Cases

Should a project need special modifications for the structure or something else, you can change the branch letter to indicate that this is a special case and not part of the main branch of base folders.

Project Status Notes

When putting a project on hold or wrapping it up, it's good practice to leave notes about the current state, pending tasks, or any other relevant information. This can help you or other team members to quickly understand the project's status when resuming work on it later. To do this, create a text file named **project_status_notes.txt** (or a similar name) in the **07_Notes** folder. Update this file whenever necessary to keep it current and useful.

Sample Project Folder Structure

To help you visualize how the proposed organization and versioning system can be implemented in a real project, we've provided a sample folder structure below. This example includes various files in different folders to give you an idea of what each folder might contain. Keep in mind that this is just an example, and your actual project may have more or fewer files and folders depending on your specific needs and preferences.

```
/Project_Name
├─ 00_base_folder_version.txt
├─ 00_Version_Logs
│   ├─ parts_versions_log.csv
│   └─ branches_info.csv
├─ 01_Models
│   ├─ 00_BLENDER
│   │   ├─ part1_1.01.001A.blend
│   │   └─ assembly_1.01.001A.blend
│   ├─ 01_RHINO
│   │   └─ part2_1.01.001A.3dm
│   └─ 02_EXPORTS
│       ├─ 00_STL
│       │   └─ part1_1.01.001A.stl
│       ├─ 01_OBJ
│       ├─ 02_VECTOR
│       └─ 03_STEP
├─ 02_Renders
│   ├─ 00_Base_Render_Scene
│   │   └─ Base_Render_Scene_1.01.001A_V1.01A.blend
│   ├─ 01_Renders
│   │   └─ assembly_1.01.001A_V1.01A_R1.01A.png
│   └─ 01_VIDEO
│       └─ assembly_1.01.001A_V1.01A_R1.01A.MP4
├─ 03_Video
│   └─ assembly_animation.mp4
├─ 04_Photos
│   └─ product_photoshoot.jpg
├─ 05_Assets
│   └─ texture_library
│       └─ wood_texture.jpg
├─ 06_Documents
│   └─ project_specifications.pdf
├─ 07_Notes
│   └─ design_notes.txt
└─ 08_References
    └─ inspiration_images
        └─ reference_image.jpg
```

Remember that you can adapt and modify the folder structure as needed to better suit your project requirements. If you need to add new folders or change the existing ones, just make sure to follow the same naming conventions and maintain a clear organization.