Asset Guidelines

1 About this documentation

1.1 Adherence to Documentation

Please thoroughly read the provided documentation and strive to follow it as closely as possible. The file naming system outlined here is designed to work effectively in most scenarios. If you come across any issues, conflicts, or logical inconsistencies, don't hesitate to notify the Engine Team. Together, we will address and resolve any concerns.

1.2 Proficiency in Git and Git Branching

It's crucial to have a solid understanding of Git and Git Branching because this is the exclusive method for supplying your files to the engine. Regardless of your role, access to GitHub is essential as everyone will contribute to the engine at some point. So pls also fully read the supplied Useful Links.

1.3 Git as Primary File Storage

Git will serve as your primary file storage system. You are encouraged, and in some cases, expected to include all your work-in-progress files, such as .psd or. blend files, in the repository. This practice ensures a seamless transition for team members who may need to continue your work and allows everyone to access essential assets at any time.

1.4 Regular Committing:

Commit your work regularly, especially after making significant progress or at the end of your work session. Frequent commits, approximately every hour, maintain a detailed and up-to-date history of your work, making it easier for the team to collaborate and track changes effectively.

1.5 Device Storage Requirement:

Minimum 100 GB of Free Storage: Each team member is required to maintain at least 100 GB of available storage on their device. This capacity is necessary to accommodate project files, work-in-progress assets, and other essential data. Please be aware that no exceptions can be made during the project, so please ensure your device has the required storage. Use this software if you have trouble cleaning up files: TreeSize Free (Software for seeing what takes up how much storage)

2 Submission Guidelines

2.1 Naming Convention

- **Avoid Special Characters**: Do not use special characters such as ~, !, @, #, $, %, ^, &, *, `, ;, :, <, >, ?, ., ,, ', ", |, ö, ä, ü.
- **No Spaces**: Do not include spaces in file names; use alphanumeric characters, dashes, or underscores.
- **Keep It Short**: Keep names as concise as possible while conveying the asset's function in the game.
- **CamelCase**: Utilize CamelCase naming conventions for clarity.
- **English Only**: Only use English names (no mix between German and English)
- **Functional Naming**: Name assets after their intended function in the game. Avoid vague or redundant descriptions.
- **Example**:
    - **Wrong**: hairDark, hairBlonde (Dark and Blonde serve different purposes)
    - **Correct**: hairBlack, hairBlonde

2.2 Consistency:

- Maintain consistency across all file names and metadata. Apply the same information to all relevant files.

2.3 Brackets for Numbering Systems

- Use brackets only when implementing numbering systems for folders. Each folder should have a number in brackets, such as [1]FolderOne, [2]FolderTwo.

2.4 Branch Creation for Asset Submission:

- To submit assets, create a new branch from the current main branch.
- Name the branch after the change you made (e.g., added couch.fbx or added furniture).
- Notify the Engine Team about the change you've made.

2.5 Engine Team Review and Merging:

- The Engine Team will review the asset for correct formatting and usability for the game.
- If approved, the Engine Team will merge your branch into the main branch.

2.6 Avoid Redundant Information:

- When naming files, avoid repeating information already present in the file structure. For instance, if an asset is located at "[1]Assets/[4]Gamescreens/[6]UI/[4]Scorebar," there is no need to include all these details in the file name. Instead, name the file "mask.png," as the filepath already provides the necessary context.

2.7 Handling Complex Issues:

- In cases where issues become complex, you may need to schedule a call with someone from the engine team for precise specifications.

2.8 Rebasing for Updates

- Be prepared to rebase your branch when the main project undergoes significant changes. This involves importing the latest files from the main branch into your branch.
- In the event of folder name changes, follow these steps:
  - Download the latest version of your file.
  - Discard your current changes.
  - Paste the file in the updated folder.

2.9 Provide Detailed Asset Usage Specifications

- When submitting an asset, it's crucial to provide precise specifications on how the asset is intended to be used and its expected behavior within the game.
- Include details such as when the asset should appear, its appearance on mouse hover, exact position, and any other relevant information.
- Be as specific and detailed as possible in your communication with the Engine Team to ensure accurate asset integration.
- You always have the possibility to use a ReadMe files in the assets folder to specify usage (only use .txt files for that)

2.10 Special Rules for 2D Assets

- **Nine-Slicing Requirement**: Every 2D asset must be nine-slicable. Exceptions require discussion with the engine team.
- **Size for Nine-Slicables**: Use the smallest dimensions possible for them.
- **Size Guidelines for Non-Slicables**: For non-slicable assets, their dimensions must be a power of 2.
- **Anti-Aliasing Restrictions**: Avoid using anti-aliasing on assets unless specified by the engine team.
- **Self-Nine-Slicing**: When importing nine-sliced assets, please perform the nine-slicing in the Unity Sprite Editor yourself.
- **File Format**: .png unless specified differently

2.11 Special Rules for Soundfiles

- Hand all of them in wav format.
- Use a CSV Table to specify where which sound goes and how it is used
- Trim the audio files before importing them.

2.12 Useful Links

- **Using Git**
  - Git Tutorial
  - What is Git? - Git Book
  - Learn Git Branching (Chapters 1.1-1.3)
  - GitHub Desktop (Most of us use this program as it has a simple UI)
- **Camelcase**
  - Camel Case Notation