Git Guide for Rudder\_Code

This guide is intended for new contributors to the Rudder\_Code project who have little or no prior experience with Git. It covers the essential commands and workflows you will use in day-to-day development.

# Basic Git Terminology

• Repository (repo): A collection of files and history tracked by Git.

• Branch: A parallel version of the repository, used to develop features or fix bugs without affecting the main branch.

• Commit: A snapshot of changes in the repo, with a message describing them.

• Remote: A copy of the repository stored on a server like GitHub.

• Push: Uploading your local commits to the remote.

• Pull: Downloading and merging commits from the remote into your local repo.

• Tag: A marker for a specific commit, often used for restore points or releases.

# Setup

1. Install Git (https://git-scm.com/downloads).

2. Configure your name and email:

git config --global user.name "Your Name"

git config --global user.email "you@example.com"

3. Clone the Rudder\_Code repository:

git clone https://github.com/macsinvan/rudder\_3D.git

4. Change into the repository directory:

cd Rudder\_Code

# Day-to-Day Workflow

1. Ensure you are on the correct branch:

git branch

git checkout branch-name

2. Pull the latest changes from the remote:

git pull origin branch-name

3. Make your code changes.

4. Stage changes for commit:

git add file1 file2

# or add all changes

git add -A

5. Commit changes:

git commit -m "Short, descriptive message about the changes"

6. Push your changes to the remote:

git push origin branch-name

# Creating Restore Points

In Rudder\_Code, we use a helper script to create guaranteed restore points.

Run the following command from the repository root:

./scripts/make\_restore\_point.sh -m "Restore point: description here"

This script will:

• Stage all files (including new ones).

• Commit them if needed.

• Create a timestamped tag.

• Optionally push to GitHub with -p.

# Branching

To create a new branch for a feature or fix:

git checkout -b new-branch-name

Push the branch to GitHub so others can see it:

git push origin new-branch-name

# Merging

Once your feature/fix branch is ready, create a Pull Request on GitHub to merge it into main or the appropriate base branch.

Make sure to resolve any conflicts before merging.

# Good Practices

• Commit often with clear messages.

• Pull before starting work to avoid conflicts.

• Use branches for all changes; keep main clean.

• Create restore points before risky changes.

• Push your work frequently so it’s backed up.