Scientific Visualization: How to Git

SciVis TAs

September 2022

1 Introduction

In this course, Scientific Visualization (SV), students are divided into groups of two. In pairs, students are expected to pull assignment-relevant code (download code to their local computers) from a remote repository made by the course TAs (https://github.com/HamidGadirov/ScientificVisualization) as well as to push (upload their code) to their own repository. To do this, students will be using GitHub. Git shouldn't be confused with GitHub. Git is a piece of software that tracks computer programs and files and the changes that are made to them over time. It also allows for collaborative work between people on the same program, code, or file. GitHub and similar services (GitLab and BitBucket) are websites that host a Git server program to hold code.

2 To install Git

Check https://github.com/git-guides/install-git to install Git in different Operative Systems or https://www.atlassian.com/git/tutorials/install-git, for example.

3 Creating your repository

Students should create their own repository where they work on their assignments by creating an account with GitHub, installing Git on their local machines (or using lab computers) and pulling from the course repository, and pushing to their own repository. In the end, the students will submit their zipped work via Brightspace. A repository is like a folder on the cloud (i.e. on GitHub's servers). To create your repository, first, select the top left plus button and click "new repository" - make it private and **don't** initialize it with a README file. Go to your repository, click "code" and copy the link. Open a terminal and "`git clone https://github.com/YourPrivateGitRepo.git`".

3.1 To add the public repository to your private repository

1. `git clone --bare https://github.com/HamidGadirov/ScientificVisualization` (do this only once)
2. `cd ScientificVisualization.git`
3. `git push --mirror https://github.com/YourPrivateGitRepo.git`
4. `cd ..` (move one directory back)
5. `rm -rf ScientificVisualization.git`

3.2 To push changes to your private repository

1. `cd YourPrivateGitRepo` (go to your private repository)
2. add your code for assignment X
3. `git add .`
4. `git commit -m "files for assignment X"`
5. `git push origin main` (push to you branch)

3.3 To update your private repository with the changes from the public repository

1. `cd YourPrivateGitRepo` (go to your private repository)
2. `git remote add public https://github.com/HamidGadirov/ScientificVisualization` (do this only once)
3. `git pull public main` (pull from our public ScientificVisualization repo's main branch)
4. `git push origin main` (push to your branch)

And now you have all changes on your private repository (your code + updates from public repository) :)

4 Git flow: how to use Git and its commands

For this course, the assignment files are given in a repository but the assignment work per group must be submitted in their own repositories or mirrors, so first create your course repository, then work on them under your own branch and push them to your own branch while pulling from main any new frameworks or files that the TAs release for the assignments.

Assuming that you mirror cloned the course repository, the normal workflow or usage of Git using commands involves opening a console in the home folder of the repository and doing the following:

1. ***Work on the code***
2. `git add .` (This adds the current changes (all of them) in your local (in the computer) repo to the list of changes to possibly be committed or made in the remote repository or main branch)
3. `git status` (This shows you the list of indexed files and their status)
4. `git commit -m "This is a commit explaining what changes are being committed"` (This commits the changes)
5. `git push` (This pushes the changes to the remote and effectively makes said changes to the remote or main branch. Now anyone that pulls will see these instead of the previous version)

The first time Git is used this information needs to be given:
git config –global user.name "My Name"
git config –global user.email "myemail@example.com"

A GUI can obviously be used instead of the console interface and is even recommended (check https://www.hostinger.com/tutorials/best-git-gui-clients/ for examples).

4.1 Mistakes happen

If using the console interface when using Git:

4.1.1 Undo a commit and redo

1. `git commit -m "This code is wrong but you noticed too late"`
2. **Fixing it from now on...**
3. `git reset HEAD`
4. ***Edit files to fix the mistake***
5. `git add .`
6. `git commit -c ORIG_HEAD`

4.1.2 Destroy a commit and drop uncommitted changes

`git reset --hard HEAD~1` (there are two hyphens behind the "hard").

4.1.3 Undo a commit but keep changes

`git reset HEAD~1`