

The background of the slide features a dark, slightly blurred image of several wooden baskets filled with fresh produce. In the foreground, a wooden sign is prominently displayed. The sign has a circular logo at the top with a sunburst design. Below the logo, the word "BUY" is written in a smaller font, and the word "LOCAL" is written in large, bold, distressed letters. The sign is mounted on a wooden post.

CIS241

System-Level Programming and Utilities

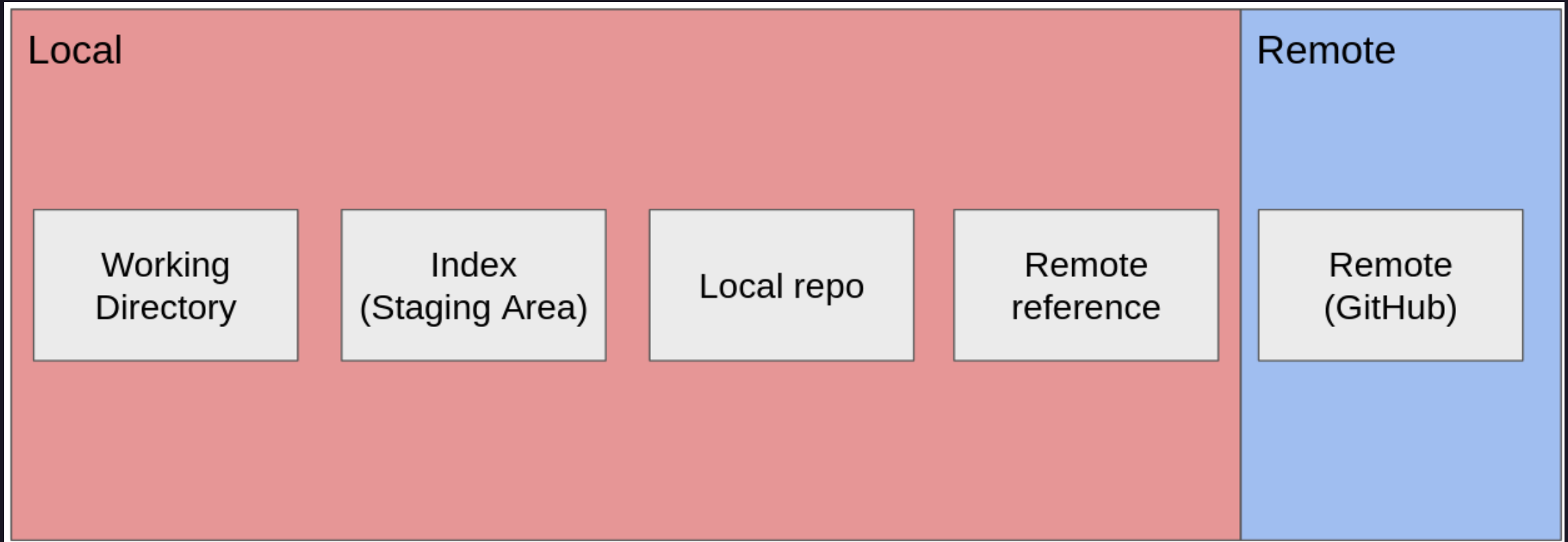
git (local)

Erik Fredericks, frederer@gvsu.edu

Fall 2025

Based on material provided by Erin Carrier, Austin Ferguson, and Katherine Bowers

Local!



git: repositories

Git repositories contain:

- A set of commit objects, and
- A set of references to commit objects (heads)

Git repositories are stored:

- In a .git sub-directory in the same directory as the project
- There is no central repository server

Git is distributed!

git: commit objects

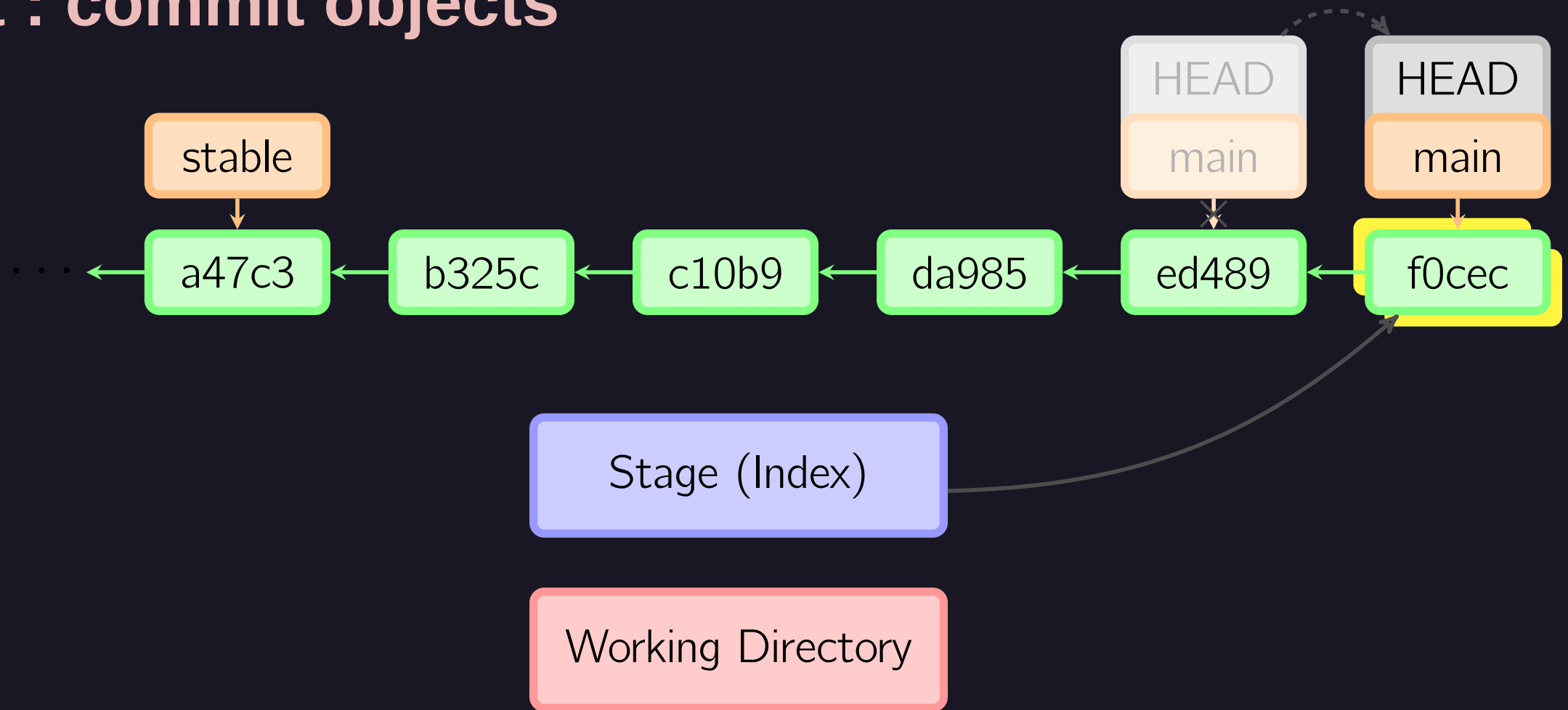
Commit objects contain:

- A set of files (one version each),
- References to parent commit objects, and
- An SHA1 identifier that uniquely* identifies the commit

If two commits are exactly the same, the identifier will be the same.

git commit

git : commit objects



Common references

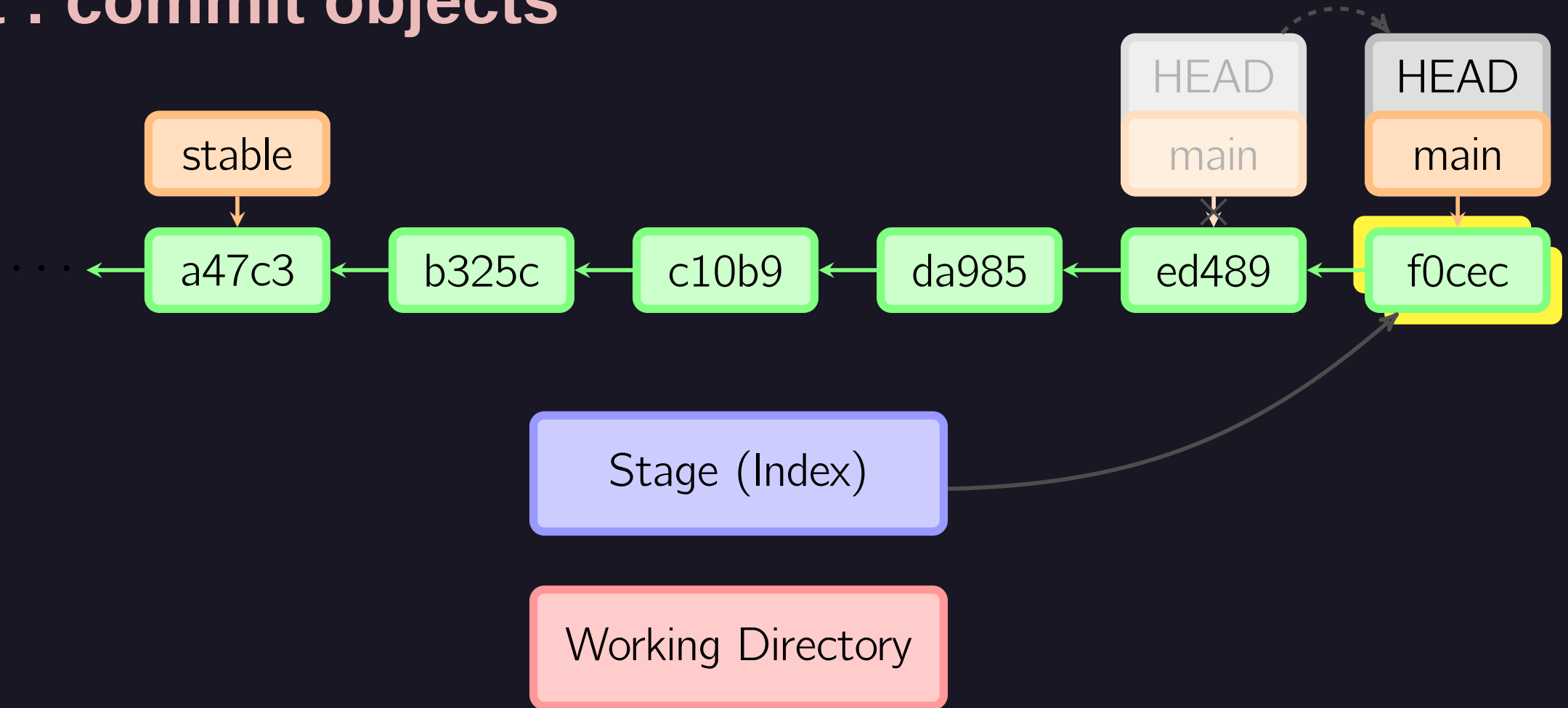
- **HEAD**: reference to tip of current branch
- **index**: staging area
- **SHA-1** hashes: commit references

Common local operations

- `git init` : create new repository
- `git add` : stage changes to file (or add untracked file)
- `git rm` : remove (add `-rf` to recursively remove a directory)
- `git mv` : rename/move
- `git commit` : commit staged changes

git commit

git : commit objects



well then

**How do you access
commit objects if you
don't know their
identifier?**



Common local operations

- `git status` : view ... status
- `git diff` : view changes/differences
- `git log` : view list of commits
- `git show <sha1>` : view info about a particular commit

NOTE

- Running `git checkout <sha1>` will leave you in a `DETACHED HEAD` state!
 - Run `git checkout main` to get back to your normal tree
 - Unless if you were branching...
 - <https://stackoverflow.com/questions/7539130/go-to-a-particular-revision>

Common fixes to mistakes!

- Local mistakes



Common fixes to mistakes

I added (staged) a file but didn't mean to

- Meaning, when you commit/push it will be tracked forever

Options:

- `git restore --staged filename` (the new way)
- `git reset filename` (old way)

Common fixes to mistakes

I've really screwed up a file, I just want to get rid of all my changes

Options:

- `git restore filename` (the new way)
- `git checkout filename` (the old way)
 - `git checkout -- filename` (if branch exists with same name as file)

However, if already staged (meaning you did `git add`):

- `git restore --staged --worktree filename`

Common fixes to mistakes

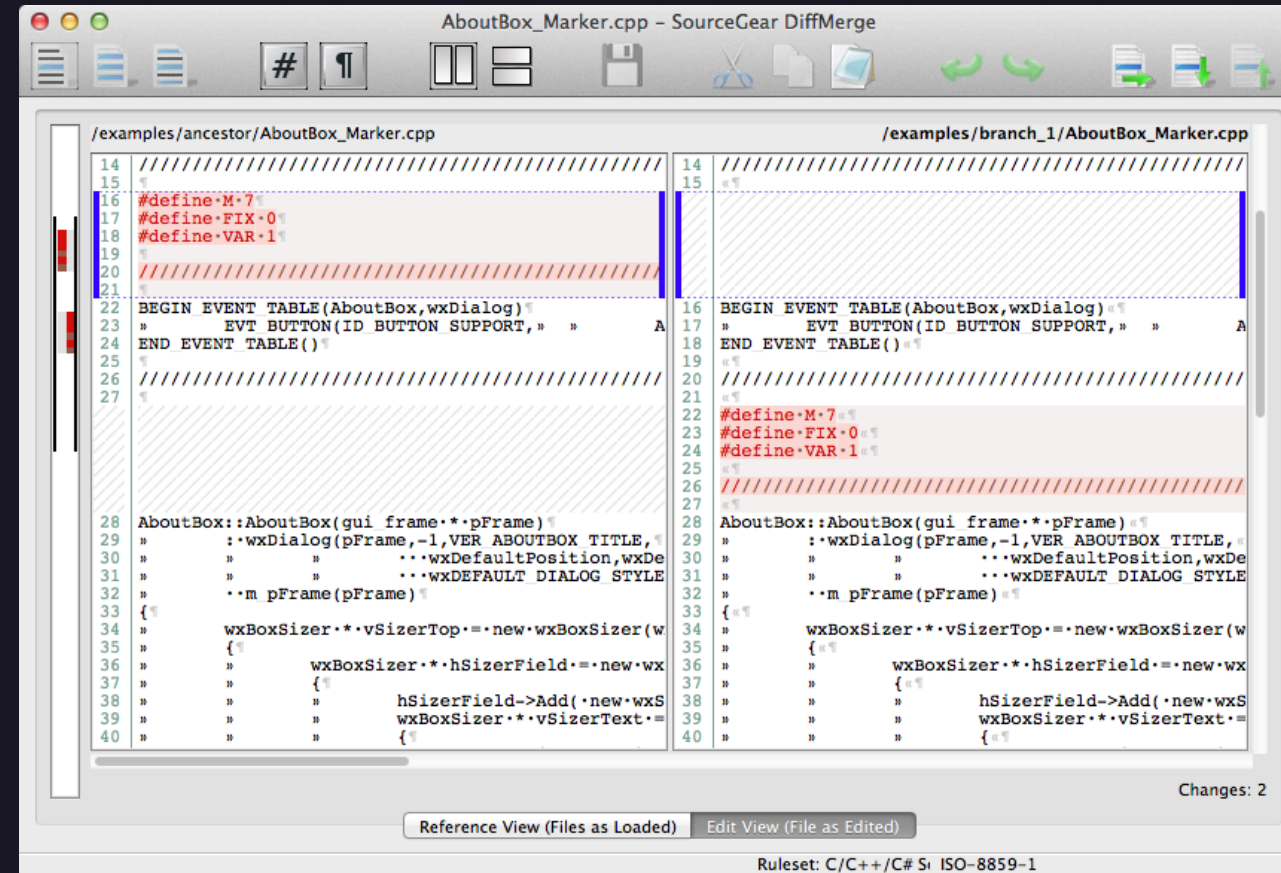
Now you've done it - you messed up your local working directory! How do I *go back*

- `git reset --hard`
 - **CAREFUL -- THIS DELETES THINGS**
- `git restore --staged --worktree .` (from top level directory in repo)

One more helpful thing

Get yourself a visual diff editor

- Muuuuch easier than using `diff`
- DiffMerge, BeyondCompare, Meld, WinMerge...



Command cheatsheet

Main file manipulation git commands:

- `git init` - Initialize git repo in folder
- `git add file` - Add files/changes to staging area (truck)
- `git mv source dest` - Move/rename file and stage
- `git rm file` - Delete file and stage
- `git commit -m "message"` - Commit staged changes to local repo

Status commands

- `git status` - View which files changed since last commit
- `git diff file` - View how file changed since last commit
- `git diff` - View how all files changed
- `git log` - View list of commits