Git Basics

Daniel Barry

November, 2016

University of Hertfordshire

Introduction

Why use Git?

- Replace the old "copy and rename" workflow
- History of how the project was built
- Ability to go back to any point during the project
- Always having a working version
- Work in teams easily
- Access to large open source projects
- Employers prefer you know how repositories work

This presentation was built in a Git repository!

Quick Git History

- *Patches and Archives* for Linux Kernel (1991 - 2002)
- *BitKeeper* for Linux Kernel (2002 - 2005)
- In 2005, BitKeeper no longer free (Larry McVoy)
- Linus Torvalds designed *Git* to be fast, simple, non-linear, distributed and scalable
- *Git* for Linux Kernel (2005 - now)

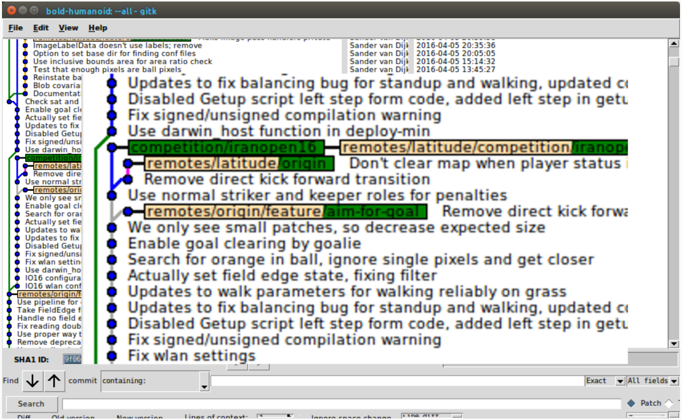
More: `https://en.wikipedia.org/wiki/Git`

What is Git?

- Inspired by BitKeeper
- Version control system
- Fully distributed
- Merges, even three-way merges
- Robust against corruption

Git was named after Linus...

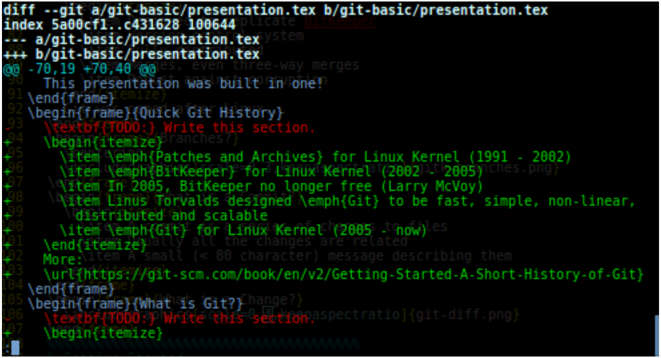
Branches?


bold-humanoid: --all - gitk
File
Edit
View
Help
ImageLabelData doesn't use labels; remove
Option to set base dir for finding conf files
Use inclusive bounds area for area ratio check
Test that enough pixels are ball pixels
Sander van Dijk 2016-04-05 20:35:36
Sander van Dijk 2016-04-05 20:05:05
Sander van Dijk 2016-04-05 15:14:32
Sander van Dijk 2016-04-05 13:45:27
Updates to fix balancing bug for standup and walking, updated c
Disabled Getup script left step form code, added left step in getu
Fix signed/unsigned compilation warning
Use darwin_host function in deploy-min
competition/iranopen16
remotes/latitude/competition/iranop
remotes/latitude/origin
Don't clear map when player status
Remove direct kick forward transition
Use normal striker and keeper roles for penalties
remotes/origin/feature/aim-for-goal
Remove direct kick forwa
We only see small patches, so decrease expected size
Enable goal clearing by goalie
Search for orange in ball, ignore single pixels and get closer
Actually set field edge state, fixing filter
Updates to walk parameters for walking reliably on grass
Updates to fix balancing bug for standup and walking, updated c
Disabled Getup script left step form code, added left step in getu
Fix signed/unsigned compilation warning
Fix wlan settings
SHA1 ID:
Find
commit
containing:
Exact
All fields
Search
Patch

What is a Commit?

- A commit is a series of changes to files
- Contains meta information (who, when, etc)
- Usually all the changes are related
- A small (typically $\leqslant$ 80 character) message describing them

What is a Change?



Getting Started

Requirements

- `git` - The repository program
- `git-gui` - Graphical representation

Setup an Environment

Create the working directory and navigate to it

```
$ mkdir wrk_dir
$ cd wrk_dir

```

Initialise the repository in the working directory

```
$ git init
Initialised empty Git repository in wrk_dir/.git/
```

Workflow

Fetching

Check for changes from remote repositories

```
$ git fetch
fatal: No remote repository specified.  Please,
    specify either a URL or a
remote name from which new revisions should be fetched
    .
```

This means we must setup a remote

```
$ git remote add origin git@github.com:danielbarry/
    test.git
```

Status

Find out the state of the working tree

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean
```

- We are on our master branch
- Our latest commit
- The fact there is nothing to commit
- How to commit if we need to

Adding Files

Create simple read me file

```
$ echo -e '#ReadMe\n\nHello World.' > readme.md
```

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)

  readme.md

nothing added to commit but untracked files present (use "git add" to track)
```

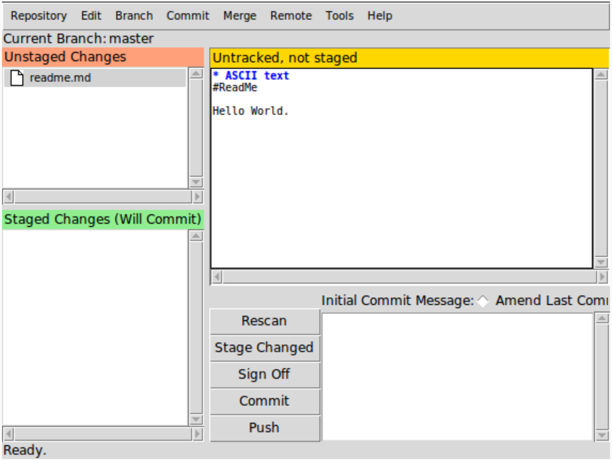
Add the file to the commit

```
$ git add readme.md
```

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

  new file:   readme.md
```

Visually Adding Files



Commiting & Pushing

Next we commit our changes

```
$ git commit -m 'First commit'
[master f51ed2a] First commit
 1 file changed, 3 insertions(+)
 create mode 100644 readme.md
```

We fetch to check nothing happened in origin

```
$ git fetch
```

And then we push

```
$ git push origin master
To git@github.com:danielbarry/test.git
   fb7db94..f51ed2a  master -> master
```

Now we check our working tree

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean
```

Stashing

We can create changes and check them

```
$ echo -e '\n\nNew words' > readme.md; git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes not staged for commit:
  (use "git add <file>..." to update what will be
    committed)
  (use "git checkout -- <file>..." to discard changes
    in working directory)

	modified:   readme.md

no changes added to commit (use "git add" and/or "git
    commit -a")
```

Stash our changes

```
$ git stash
Saved working directory and index state WIP on master:
    f51ed2a First commit
HEAD is now at f51ed2a First commit
```

Check the status of the repository

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean
```

Pop our changes back

```
$ git stash pop
On branch master
Your branch is up-to-date with 'origin/master'.
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

  modified:   readme.md

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (154c631be9fd03b1be0347aad9f9d9fc204f3afd)
```

We are told the repository status

Pulling

Somebody on our team has pushed a change we want

```
$ git fetch
From github.com:danielbarry/test
   f51ed2a..be5e774  master     -> origin/master
```

Stash our unstaged changes

```
$ git stash
Saved working directory and index state WIP on master:
    f51ed2a First commit
HEAD is now at f51ed2a First commit
```

Pull in the changes (auto-rebase)

```
$ git pull
First, rewinding head to replay your work on top of it
    ...
Fast-forwarded master to
    be5e774af0029431e7479271cc2bf42455816c14.
```

And pop our stashed changes back

```
$ git stash pop
On branch master
Your branch is up-to-date with 'origin/master'.
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

  modified:   readme.md

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (54735f7330c6907178b770792702a1b44c15983c)
```

Feature Branch

New features developed on a separate branch

```
$ git checkout -b feature/new-branch
Switched to a new branch 'feature/new-branch'
M readme.md
```

Check which branch we are on

```
$ git branch
* feature/new-branch
  master
```

Create change on branch and commit it

```
$ echo -e 'extra data' > feature.txt; git add feature.txt; git commit -m 'New commit'
[feature/new-branch 888dfd2] New commit
 1 file changed, 1 insertion(+)
 create mode 100644 feature.txt
```

Push change

```
$ git push origin feature/new-branch
To git@github.com:danielbarry/test.git
 * [new branch]      feature/new-branch -> feature/new-branch
```

Merge Branch

Checkout the master branch

```
$ git checkout master
Switched to branch 'master'
M readme.md
Your branch is up-to-date with 'origin/master'.
```

Merge the branch with master

```
$ git merge feature/new-branch
Updating be5e774..888dfd2
Fast-forward
 feature.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 feature.txt
```

Now check the status

```
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
Changes not staged for commit:
  (use "git add <file>..." to update what will be
    committed)
  (use "git checkout -- <file>..." to discard changes
    in working directory)

  modified:   readme.md

no changes added to commit (use "git add" and/or "git
    commit -a")
```

Commits from the branch are waiting to be pushed

Something Went Wrong

Undo-ing Changes

Check our bad change

```
$ git status -s
 M readme.md
```

Undo all changes in file

```
$ git checkout readme.md
```

Undo-ing Commits

Show the previous commit

```
$ git log -n 1
commit 888dfd2da79a98b1437dfea6a8417efec251a916
Author: Dan <danbarry16@googlemail.com>
Date:   Wed Nov 16 11:37:47 2016 +0000

    New commit
```

Undo the previous commit

```
$ git reset --mixed HEAD^
```

Show the previous commit (again)

```
$ git log -n 1
commit be5e774af0029431e7479271cc2bf42455816c14
Author: Dan <danbarry16@googlemail.com>
Date:   Wed Nov 16 11:37:41 2016 +0000

    New data
```

"Bang, and the dirt is gone!"

Actually, it is still staged

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)

  feature.txt

nothing added to commit but untracked files present (use "git add" to track)
```

So we can do something with this

Advanced

Re-Writing History

WARNING: This is dangerous (but powerful ;))

Reasons not to do this:

- Could make data hard to recover
- Will could problems for team mates
- More difficult to undo these changes (but not impossible)

Reasons to do this:

- Something bad happened
- "You're a Git Wizard Harry"

Re-Write What?

- Amend previous commit `git commit --amend`
- Amend many commits `git rebase -i HEAD N`
- Squashing commits (Use interactive rebase)
- Split commits (Use interactive rebase)
- Filter-Branch (Scripted re-writing)

Further reading: `https://git-scm.com/book/en/v2/Git-Tools-Rewriting-History`

Conclusion

About Presentation

- Content - `https://github.com/danielbarry/presentations`
- Presentation - `http://www.latex-project.org`
- Theme - `https://github.com/matze/mtheme`