Git, GitHub and ROBOTC

A Winning Alliance for FIRST Teams

Presented by FTC #6055 Gearticks

Lincoln, MA

Outline of presentation

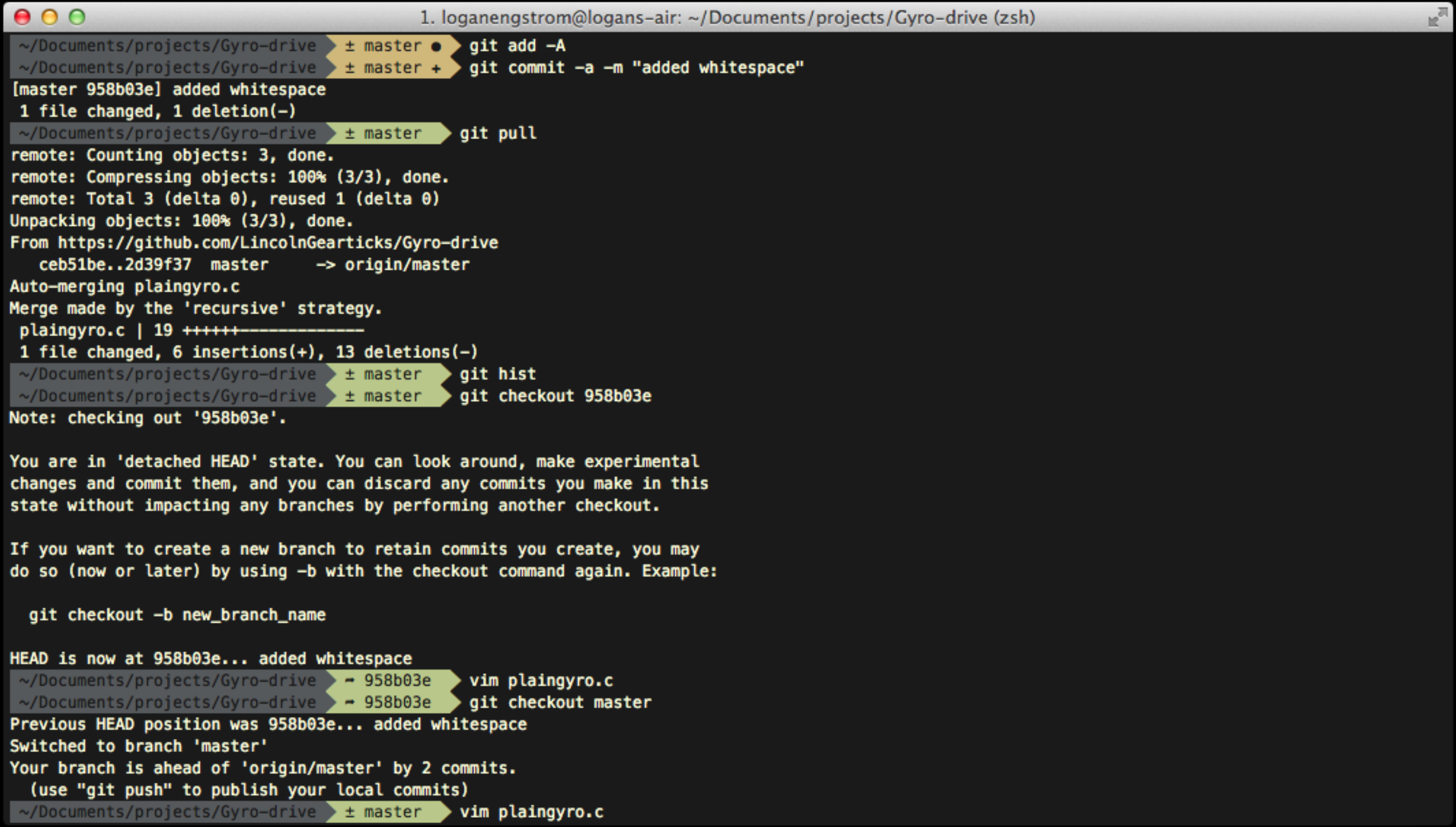
- What is Git and GitHub?
- How can it benefit FTC and FRC teams?
- Our experience without and with Git and GitHub
- How to obtain it
- Helpful sources for learning
- Basic concepts of Git and GitHub
- Involving beginner programmers
- Summary of advantages and potential pitfalls

Terminology and follow up

- File paths and commands, as well as any code, are designated with `monospace`
- This presentation is located at:
  - `www.gearticks.org`
  - `email us at FTC6055gearticks@gmail.com`

Terminology

- RCS: Revision control system
- SVN: Apache Subversion, another RCS
- SCM: Source control management
- GUI: Graphical User Interface
- CLI: Command Line Interface


1. loganengstrom@logans-air: ~/Documents/projects/Gyro-drive (zsh)

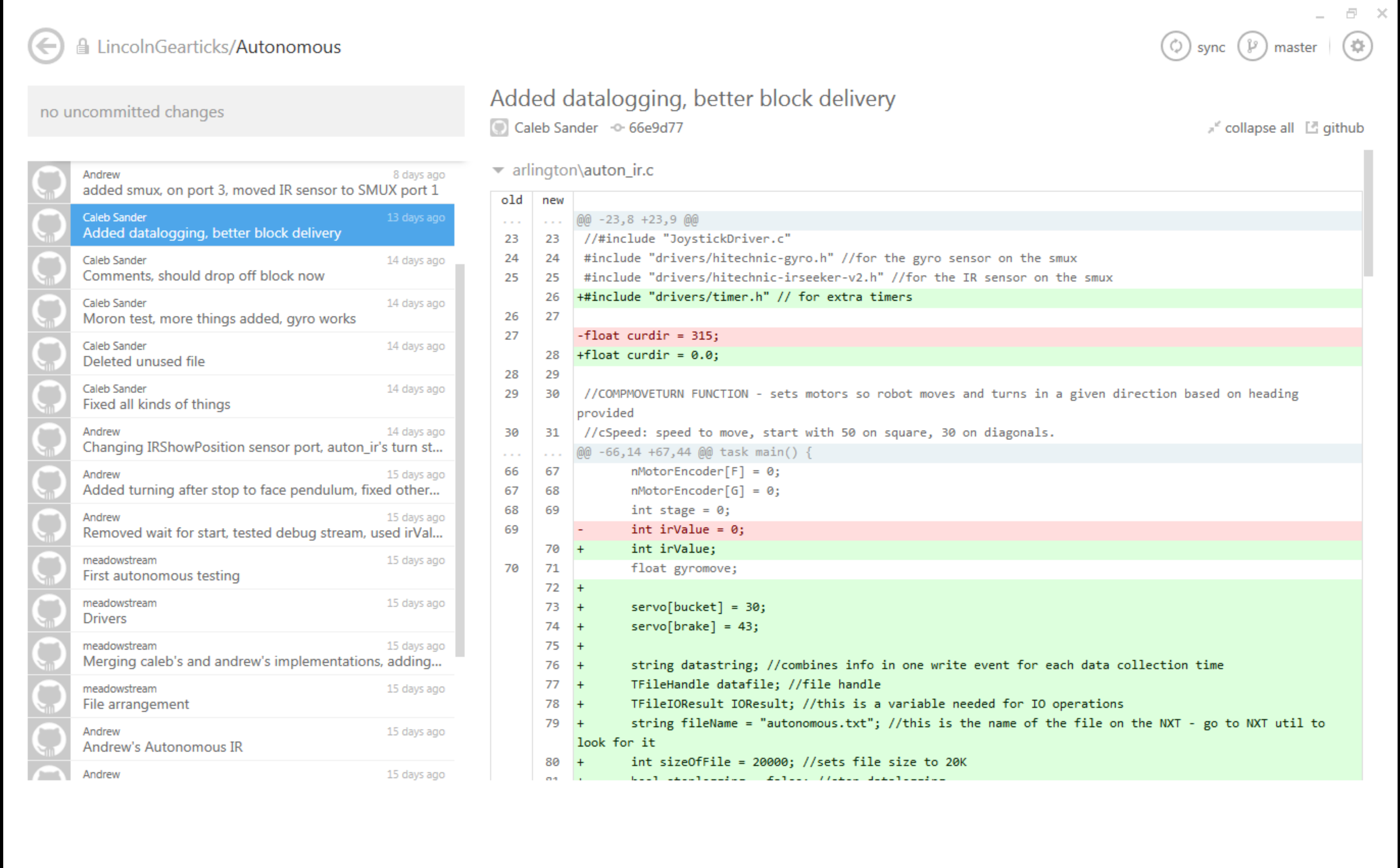
```
 ~/Documents/projects/Gyro-drive  ± master ●  git add -A
 ~/Documents/projects/Gyro-drive  ± master +  git commit -a -m "added whitespace"
[master 958b03e] added whitespace
 1 file changed, 1 deletion(-)
 ~/Documents/projects/Gyro-drive  ± master  git pull
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 1 (delta 0)
Unpacking objects: 100% (3/3), done.
From https://github.com/LincolnGearticks/Gyro-drive
   ceb51be..2d39f37  master     -> origin/master
Auto-merging plaingyro.c
Merge made by the 'recursive' strategy.
 plaingyro.c | 19 ++++++-------------
 1 file changed, 6 insertions(+), 13 deletions(-)
 ~/Documents/projects/Gyro-drive  ± master  git hist
 ~/Documents/projects/Gyro-drive  ± master  git checkout 958b03e
Note: checking out '958b03e'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b new_branch_name

HEAD is now at 958b03e... added whitespace
 ~/Documents/projects/Gyro-drive  ➦ 958b03e  vim plaingyro.c
 ~/Documents/projects/Gyro-drive  ➦ 958b03e  git checkout master
Previous HEAD position was 958b03e... added whitespace
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 2 commits.
  (use "git push" to publish your local commits)
 ~/Documents/projects/Gyro-drive  ± master  vim plaingyro.c
```



Andrew — 8 days ago
added smux, on port 3, moved IR sensor to SMUX port 1

Caleb Sander — 13 days ago
Added datalogging, better block delivery

Caleb Sander — 14 days ago
Comments, should drop off block now

Caleb Sander — 14 days ago
Moron test, more things added, gyro works

Caleb Sander — 14 days ago
Deleted unused file

Caleb Sander — 14 days ago
Fixed all kinds of things

Andrew — 14 days ago
Changing IRShowPosition sensor port, auton_ir's turn st...

Andrew — 15 days ago
Added turning after stop to face pendulum, fixed other...

Andrew — 15 days ago
Removed wait for start, tested debug stream, used irVal...

meadowstream — 15 days ago
First autonomous testing

meadowstream — 15 days ago
Drivers

meadowstream — 15 days ago
Merging caleb's and andrew's implementations, adding...

meadowstream — 15 days ago
File arrangement

Andrew — 15 days ago
Andrew's Autonomous IR

Andrew — 15 days ago

Added datalogging, better block delivery

Caleb Sander · 66e9d77

arlington\auton_ir.c

```
@@ -23,8 +23,9 @@
 //#include "JoystickDriver.c"
 #include "drivers/hitechnic-gyro.h" //for the gyro sensor on the smux
 #include "drivers/hitechnic-irseeker-v2.h" //for the IR sensor on the smux
+#include "drivers/timer.h" // for extra timers

-float curdir = 315;
+float curdir = 0.0;

 //COMPMOVETURN FUNCTION - sets motors so robot moves and turns in a given direction based on heading provided
 //cSpeed: speed to move, start with 50 on square, 30 on diagonals.
@@ -66,14 +67,44 @@ task main() {
        nMotorEncoder[F] = 0;
        nMotorEncoder[G] = 0;
        int stage = 0;
-       int irValue = 0;
+       int irValue;
        float gyromove;
+
+       servo[bucket] = 30;
+       servo[brake] = 43;
+
+       string datastring; //combines info in one write event for each data collection time
+       TFileHandle datafile; //file handle
+       TFileIOResult IOResult; //this is a variable needed for IO operations
+       string fileName = "autonomous.txt"; //this is the name of the file on the NXT - go to NXT util to look for it
+       int sizeOfFile = 20000; //sets file size to 20K
```

What is Git?

- Git is a fast, distributed version control and SCM system
- Git is a protocol for communicating with a Git server (like GitHub)
- Git is free and open source

What is GitHub?

- GitHub is a service that stores Git repositories, or version-controlled directories, remotely on a server
- Only select people can access the repository (your team, e.g.)
  - Lets multiple people on multiple computers work on the same codebase locally at the same time

How can Git benefit FTC and FRC Teams?

- Git will:
  - Manage your code across multiple computers
  - Keep a copy of your code online
  - Keep track of the changes to your code
    - let you revert to different versions of your code
    - let you maintain different versions of the same code at the same time
  - Allow you to keep local copies of the code

Git Uses and Restrictions

- Can be used with any text based format (if you want to use it to its full extent)
  - ROBOTC, Java, C++, .txt, HTML
  - Limitations of graphical programming languages
    - diff-ing, merging can be hard to do
    - Apparently people are working on this though!
- Can be used with any sized project
  - FTC, FRC, team websites, and more

Our Experience Without and With Git

Without Git

- Used Google Drive
- Had to upload and download each version and merge manually
- Could not track changes
- A pain for multiple programmers to modify the same file at the same time
- So it was hard for new programmers to start programming because others would already be modifying the files

With Git

- Can track changes
- Can see update history
- Programming is a collaborative process
- New team members can participate and learn
- Much easier merging

How to Get Git and GitHub

Getting Git Client

- http://git-scm.com/downloads
  - Download per platform
- Not comfortable with a CLI?
  - Get Github’s GUI
  - http://git-scm.com/downloads/guis
  - We’ve found that GitHub’s Windows and Mac GUIs are the easiest to use for people not experienced with the command line

Getting Started on GitHub

- http://www.github.com
- Make an account
- Unlimited public repos (viewable by anyone, editable by contributors)
  - Private (viewable only by contributors) usually cost money
  - GitHub offers free private repos to many FTC teams

Get Git

- http://git-scm.com/book/en/Getting-Started-Installing-Git
- Or, search “install git”, and follow the instructions

Overview

- Git is a decentralized version control system
  - each repository contains every change, every revision of work in the directory
  - computers do not have to be online, or connected to a central server all the time
- This means that each repository can use all the functions of the git scm system without being online

GUIs and the Command Line Interface

- Command line interface
  - The GUIs for Git (most notably offered by GitHub) are very simplistic, their functions are easy to use
  - GUIs and the command line interface use the same terms, in general, so understanding how the command line interface works, even at a superficial level, will let one use the GUI effectively
    - Command line is far more powerful, though
    - Interesting article: http://pauillac.inria.

Git Basics

- Git repositories are stored locally in a directory
- Transform a directory into a git repository by initializing the repository
- Edit/add files in folder as you would normally
    - Changes are automatically tracked in the `.git` directory

Staging and committing changes

- When you make changes to the directory and want to have them tracked, “commit” the changes and then push them to the repository
- This format is central to how git works

Staging changes

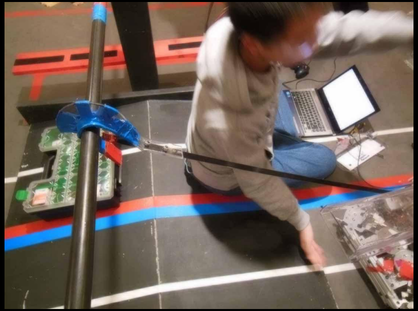
- “Staging” - selecting files to track for each commit
- Note that these are changes being added, not files
  - When you change files after you’ve added staged in them, you need to add the file again to retain the extra changes

Committing changes

- When you've finished adding all the changes you want, you commit them
  - `git commit -a -m "Commit message"` to commit all tracked files (`-a` is short for "all")
    - Tracked: files present in the last commit - if you add new files, you still have to add them!
    - The CLI/GUI might need a special command to add newly added files to the commit (`git add filename` or `git add --all`)

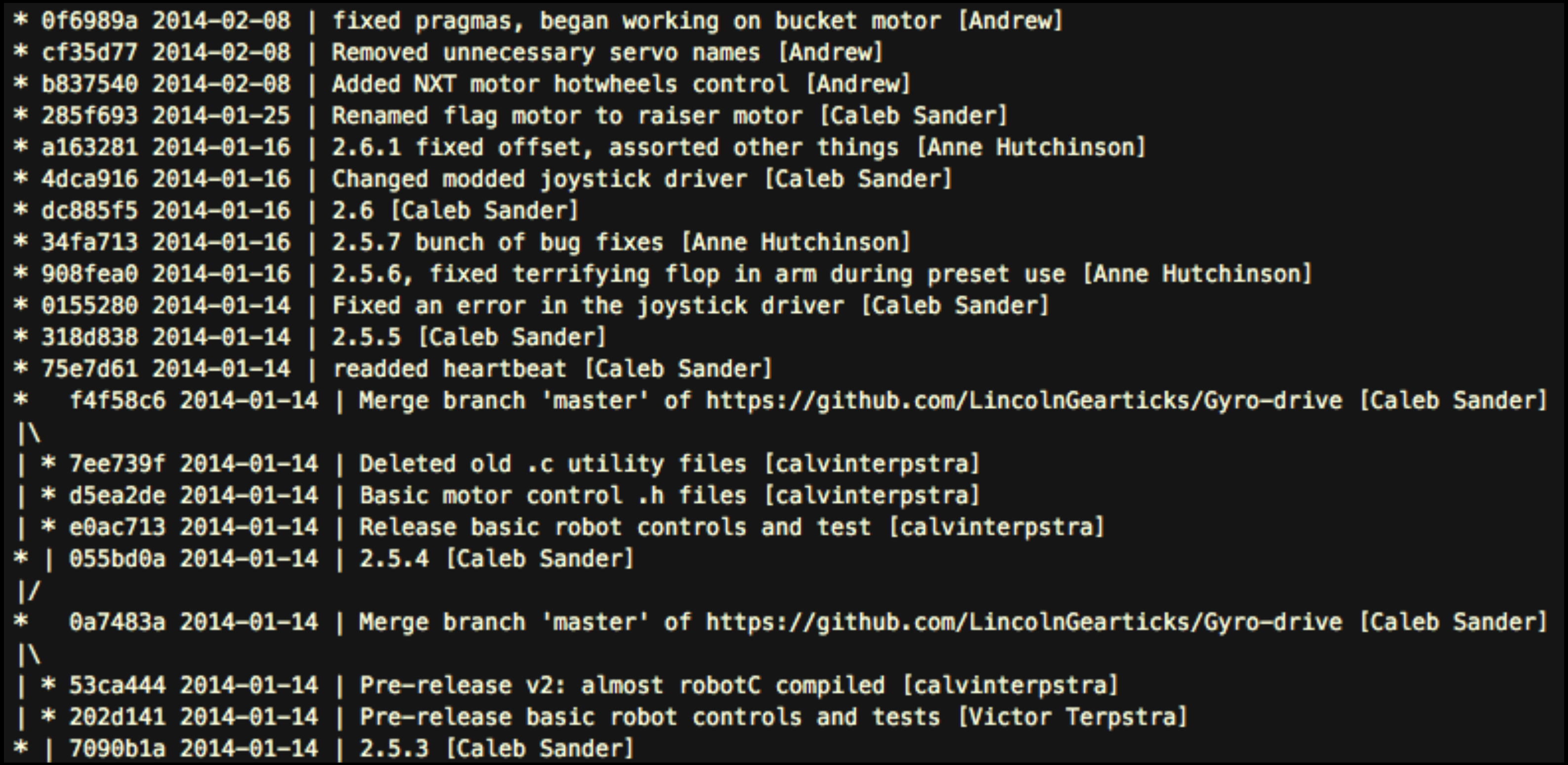
Looking at the commit history

- Easily look at each change
  - Listed: every commit, with description and hash
    - Hashes let you jump to each commit
  - Shows commits of branches, merges
  - GitHub has great interfaces online, (go to github.com/username/repo to see it)



Example

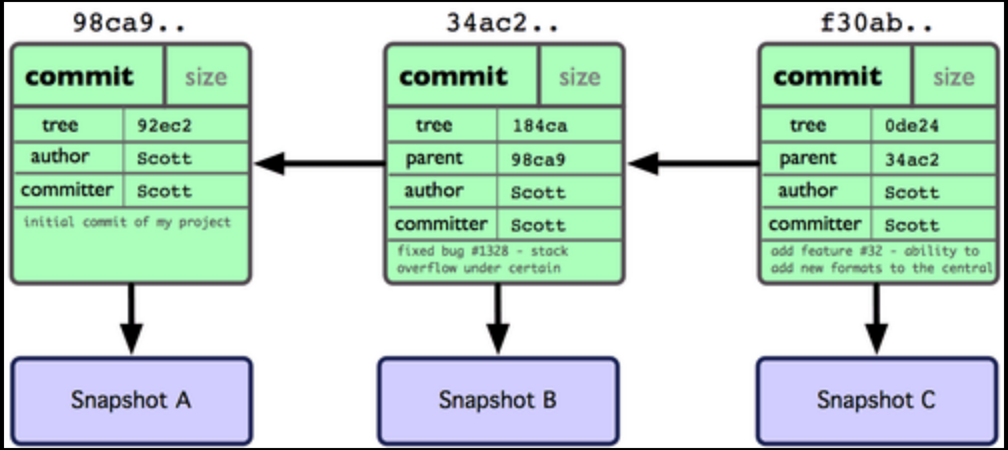
The “bumps” are when people have a different head than the master, and merge their version with the master’s version (in the remote repository on Github.com)



```
* 0f6989a 2014-02-08 | fixed pragmas, began working on bucket motor [Andrew]
* cf35d77 2014-02-08 | Removed unnecessary servo names [Andrew]
* b837540 2014-02-08 | Added NXT motor hotwheels control [Andrew]
* 285f693 2014-01-25 | Renamed flag motor to raiser motor [Caleb Sander]
* a163281 2014-01-16 | 2.6.1 fixed offset, assorted other things [Anne Hutchinson]
* 4dca916 2014-01-16 | Changed modded joystick driver [Caleb Sander]
* dc885f5 2014-01-16 | 2.6 [Caleb Sander]
* 34fa713 2014-01-16 | 2.5.7 bunch of bug fixes [Anne Hutchinson]
* 908fea0 2014-01-16 | 2.5.6, fixed terrifying flop in arm during preset use [Anne Hutchinson]
* 0155280 2014-01-14 | Fixed an error in the joystick driver [Caleb Sander]
* 318d838 2014-01-14 | 2.5.5 [Caleb Sander]
* 75e7d61 2014-01-14 | readded heartbeat [Caleb Sander]
*   f4f58c6 2014-01-14 | Merge branch 'master' of https://github.com/LincolnGearticks/Gyro-drive [Caleb Sander]
|\
| * 7ee739f 2014-01-14 | Deleted old .c utility files [calvinterpstra]
| * d5ea2de 2014-01-14 | Basic motor control .h files [calvinterpstra]
| * e0ac713 2014-01-14 | Release basic robot controls and test [calvinterpstra]
* | 055bd0a 2014-01-14 | 2.5.4 [Caleb Sander]
|/
*   0a7483a 2014-01-14 | Merge branch 'master' of https://github.com/LincolnGearticks/Gyro-drive [Caleb Sander]
|\
| * 53ca444 2014-01-14 | Pre-release v2: almost robotC compiled [calvinterpstra]
| * 202d141 2014-01-14 | Pre-release basic robot controls and tests [Victor Terpstra]
* | 7090b1a 2014-01-14 | 2.5.3 [Caleb Sander]
```

Branches

- A branch is a path of commits that allow a project to move in different directions, and then be combined again if necessary
    - People can make their own versions of a program, but still keep the same codebase, and can easily merge in their changes
- This is especially useful for experimenting or for modifying code without getting in someone else’s way


98ca9..
commit
size
tree
92ec2
author
Scott
committer
Scott
initial commit of my project
34ac2..
commit
size
tree
184ca
parent
98ca9
author
Scott
committer
Scott
fixed bug #1328 - stack overflow under certain
f30ab..
commit
size
tree
0de24
parent
34ac2
author
Scott
committer
Scott
add feature #32 - ability to add new formats to the central
Snapshot A
Snapshot B
Snapshot C

Working with remote repositories: Getting changes

- Remote repositories' files are not forcefully synced with the ones locally, so changes must be pushed and pulled
- Fetching the remote repository puts it in your `.git` (which stores all the version control info)
- Pulling the remote repository fetches the remote repository's changes, and then merges them into your own program (alerting you if it runs into any conflicts between your changes and the repository's)

Pushing changes

- Push changes to move them onto the remote repository from your computer
- If your local repo is not up to date, it will require you to perform a pull first

Conflicts

- Conflicts between versions
  - Case: one person pushed changes to the remote repository, and then another person tried to merge the remote repository into his/her own code, which had changes made on the same line(s))
- If there are conflicts running a merge...
  - Git marks each conflict to be resolved manually, showing each version and a notification in the file(s)

Example:

Original:

This is a line

First editor

This is a line, and it’s awesome

Second editor

This is a line. It is short.

Resolution:

This is an awesome line. It is short.

<<<<<<<: Indicates the start of the lines that had a merge conflict. The first set of lines are the lines from the file that you were trying to merge the changes into.

=======: Indicates the breakpoint used for comparison. Breaks up changes that user has committed (above) to changes coming from merge (below) to visually see the differences.

>>>>>>>: Indicates the end of the lines that had a merge conflict.

How does git work?

- `.git` (hidden) directory contains every change
  - Changes are compressed and stored as hashes
  - Branches are saved as well
  - HEAD: Contains reference to current branch and commit
    - Generally reference to master

Potential Pitfalls

- “Forcing” things (with the -f flag): When merging is too difficult, you can overwrite everything (difficult to reverse--can lose work, loss of time)
- Requires a certain level of training/education to use it effectively.
    - Understand Git before using it, don’t just use it without understanding
- Deleting `.git`: destroys the entire revision control system
- Merge conflicts
    - Make sure that you resolve them! Don’t just push again!
- Internet syncing
    - Make sure that you sync, otherwise other people won’t have access to your code
- Git isn’t google documents: changes aren’t pushed immediately!

Involving Beginner Programmers

- Not using source control discourages new programmers
  - Only one person can realistically program at once: in the stress of the season, that’s generally the most experienced person
- Git changes that: everyone can program at once

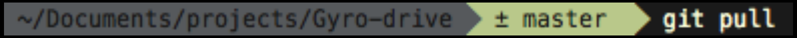
Summary

- Git is a fantastic RCS that lets many people with many computers work on different parts of the same project at the same time
- Decentralized revision control systems reduces dependence on a server, such as with SVN
- Track changes very efficiently to debug and mark different versions - also see which users edited where

Beginning to Learn Git

- Start when you have a bit of time to learn Git
  - Though Git is pretty simple, it still takes time to become familiar with
  - We started at the end of the season last year, and learned / taught our team throughout the summer
- Look at the resources on the next page

Learn more

- Everything you need to know about Git: http://git-scm.com/book
- Learn the CLI interactively: http://gitimmersion.com/
  - Get zsh: http://www.zsh.org/
    - Easier to work with than bash for the command line
    - Looks nicer, has more information
      - 

```
~/Documents/projects/Gyro-drive  ± master  git pull
```
- Questions? Email ftc6055Gearticks@gmail.com

Summary: Advantages

- Git is a fantastic RCS that lets many people with many computers work on the same project at the same time with trees, pulling, pushing, and merging
- Decentralized revision control systems reduces dependence on a server, such as with SVN
- Track changes very efficiently to debug and mark different versions - also see which users edited where