Gettin' the hang of git

Version control and why git matters

by Sinthujan Thanabalasingam

What we'll talk about

- Collaborative software development
- Struggles and problems
- git – how does git work?
- Basic workflow (to *git gud* (͡° ͜ʖ ͡°))
- Hints regarding RoboLab 2018

Feel free to raise your hand and ask questions.

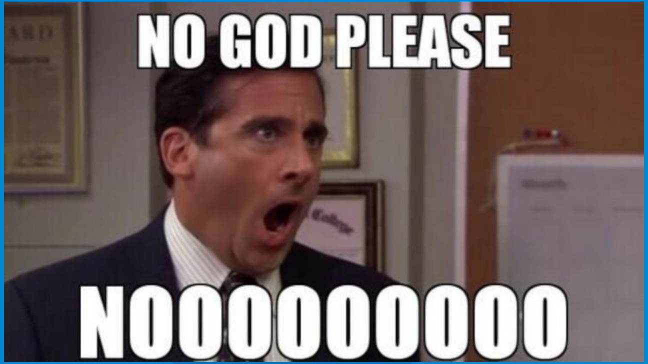
Software Engineering in a team

- Complex projects require teams of developers
- Modern SE is collaborative and distributed
- Division of labor as central aspect
- Integration of sub parts to one whole

→ How do you manage the code base?

Easy!

USB drives! Just share source code between colleagues via flash drives!



Problems with the "easy" solution

- Division of labor is difficult
- File corruptions and other catastrophes?
- Undo changes?
- Versioning?
- What is the current common ground?
- Recover from chaos, hate and despair?

Version Control System (VCS)

- Tracking of distributed documents and changes to them
- Source code management
- Recover previous states
- Automatic integration (merging) of changes and revisions
- Essential for organization of multi-developer-projects

git
SUBVERSION
Final 2
Final Final
FINAL FOR REAL THIS TIME
OK This is definitely final
I prefer the real version control
I said the *real* version control
Perfection

git to the rescue

- Distributed VCS
- Cooperative working on a project
- Managing projects in form of repositories
- Support by web-based hosting services like GitHub or Bitbucket
- Available for Linux, Windows and macOS

The guts of git

- Every developer works on a local copy of the project (local repository)
- git manages changes in the form of commits
- Commits are published on remote repositories
- Developers can integrate changes of other devs by using the remote repository

How to get git

Debian/Ubuntu/WSL

```
sudo apt-get install git-all
```

Windows

- git bash: https://git-scm.com/download/windows
- WSL (Windows Subsystem for Linux)

macOS

https://git-scm.com/download/mac

git – basic workflow

Every directory can be managed by git.

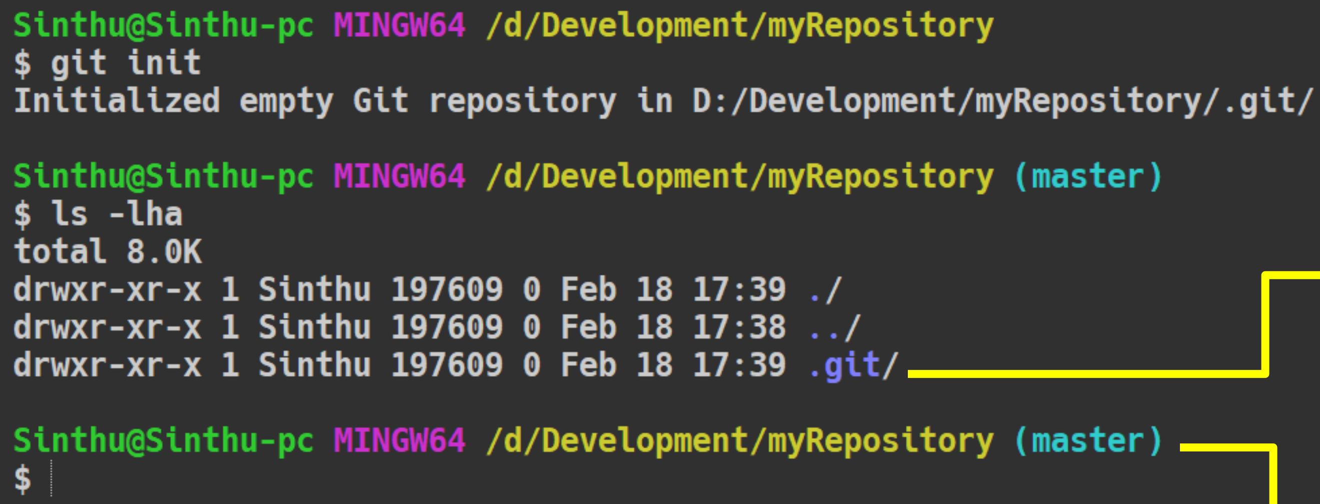
```
git init
```

or

```
git clone <url>
```

- Initializes/copies a repository in the current working directory
- Now files can be added to the index
- git manages all files that were added to the index

git init - result



```
Sinthu@Sinthu-pc MINGW64 /d/Development/myRepository
$ git init
Initialized empty Git repository in D:/Development/myRepository/.git/

Sinthu@Sinthu-pc MINGW64 /d/Development/myRepository (master)
$ ls -lha
total 8.0K
drwxr-xr-x 1 Sinthu 197609 0 Feb 18 17:39 ./
drwxr-xr-x 1 Sinthu 197609 0 Feb 18 17:38 ../
drwxr-xr-x 1 Sinthu 197609 0 Feb 18 17:39 .git/

Sinthu@Sinthu-pc MINGW64 /d/Development/myRepository (master)
$
```

hidden folder for git

master branch (explained later)

git – basic workflow

```
git clone <url>
```

- Copies an existing repository in the current working directory
- URL specifies the location of the remote

git – basic workflow

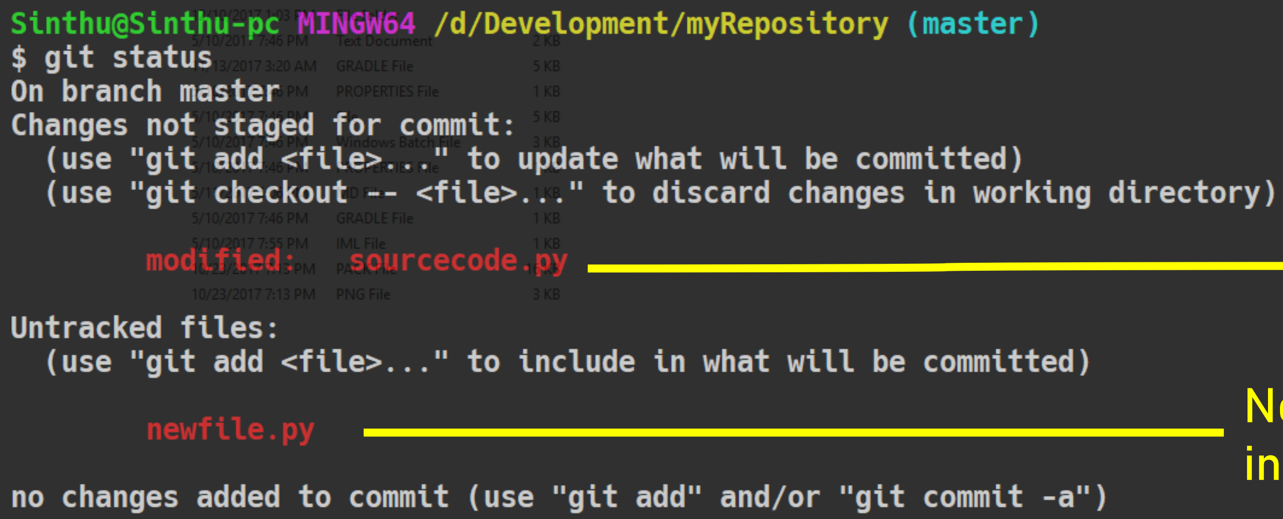
- After initialization or cloning, work can be done on files
- Creating/updating/deleting files
- git registers changes in the current working directory
- Changes can then be published
- add, commit, branch, push , pull and merge are elementary commands

status of the repository

```
git status
```

- Summary of changes since the last commit
- Lists new, deleted and changed files
- Tool to decide which changes should be included in the next commit

git status - result



```
Sinthu@Sinthu-pc MINGW64 /d/Development/myRepository (master)
$ git status
On branch 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:   sourcecode.py

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        newfile.py

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

Tracked, but modified

Not in the index yet

git – basic workflow

Adding files to the git index is called staging.

```
git add <filename>
```

adds file <filename> to the index

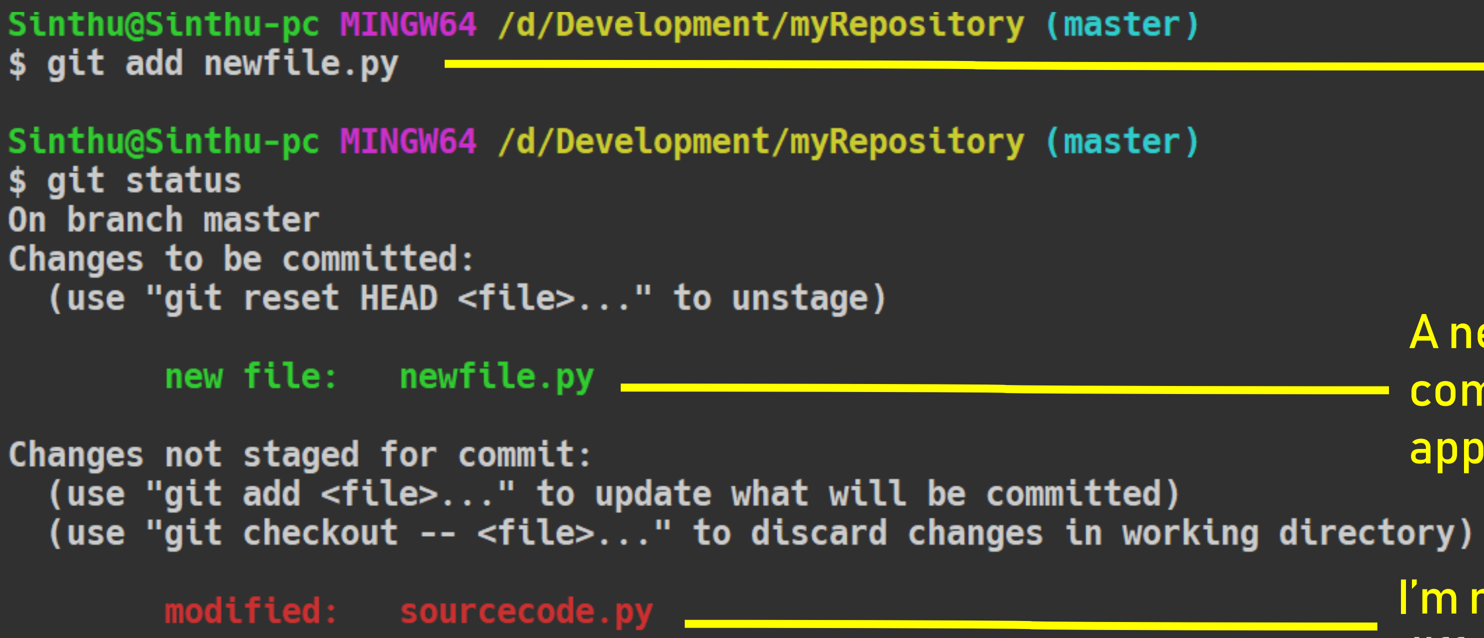
```
git add *.<exstension>
```

adds all files of extension <extension> to the index

```
git add .
```

adds all files to the index

Adding changes and status check



```
Sinthu@Sinthu-pc MINGW64 /d/Development/myRepository (master)
$ git add newfile.py

Sinthu@Sinthu-pc MINGW64 /d/Development/myRepository (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   newfile.py

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:   sourcecode.py
```

adding newfile.py

A new file in the commit appeared!

I'm not new, but different

git commit

„A commit describes and adds a set of changes on resources to a repository.“

git commit

```
git commit -m "commit message"
```

- Bundles changes to a commit
- Changes are committed to the VCS and saved locally
- commit message
  - Should be a meaningful description of changes made by the dev
  - Is visbible information for other developers

The emphasis lays on meaningful…

Bad commit messages I encountered @robolab and/or @work:

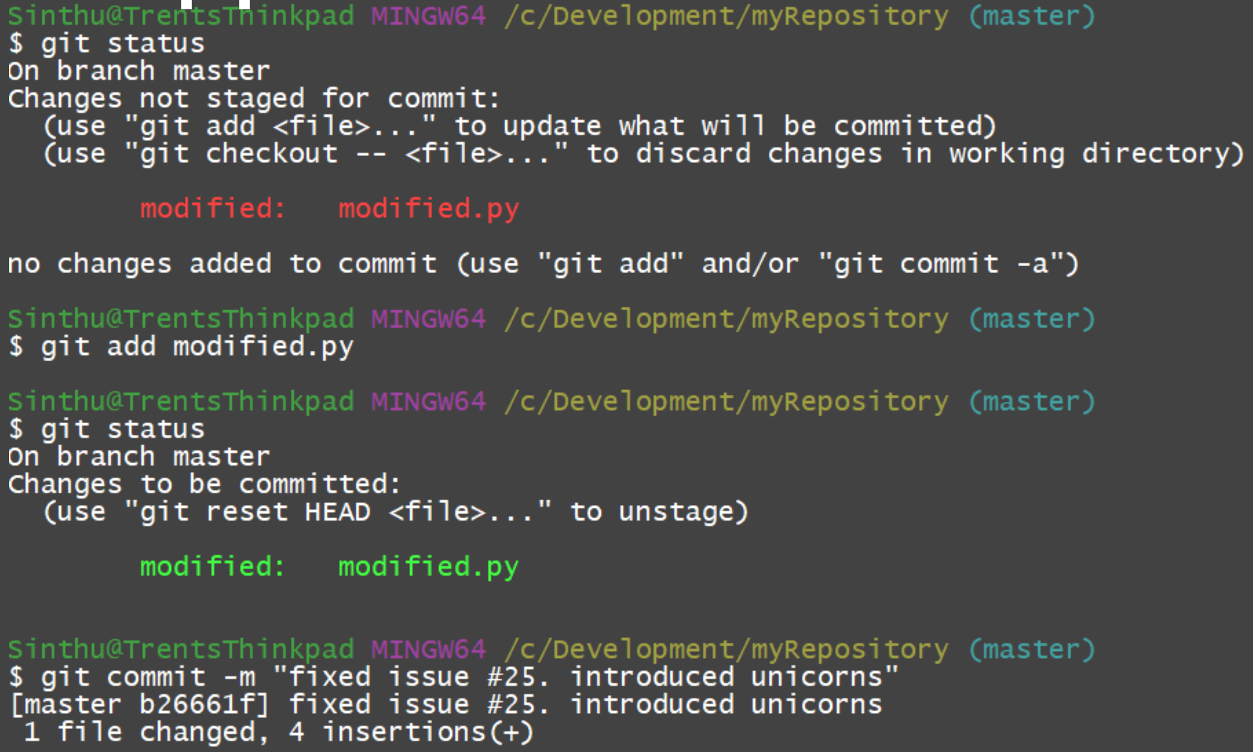
```
git commit -m “still crashes -.- pls end me!”
```

```
git commit -m “fixed some merge conflict sh*t”
```

```
git commit -m “kill it! just kill it with fire.”
```

```
git commit -m “what am even I doing with my life?”
```

Preppin' a little commit…



```
Sinthu@TrentsThinkpad MINGW64 /c/Development/myRepository (master)
$ git status
On branch 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:   modified.py

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

Sinthu@TrentsThinkpad MINGW64 /c/Development/myRepository (master)
$ git add modified.py

Sinthu@TrentsThinkpad MINGW64 /c/Development/myRepository (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   modified.py


Sinthu@TrentsThinkpad MINGW64 /c/Development/myRepository (master)
$ git commit -m "fixed issue #25. introduced unicorns"
[master b26661f] fixed issue #25. introduced unicorns
 1 file changed, 4 insertions(+)
```

git commit

- Every commit can be identified by a hash value associated to it
- Repositories can be reset to a previous state with the hash values
- Commits are initially local and only influence your own working directory

Now, you might ask yourself...

How is this process collaborative?

How do you publish your changes?

And most important: where do you publish your changes to?

git remote got you covered

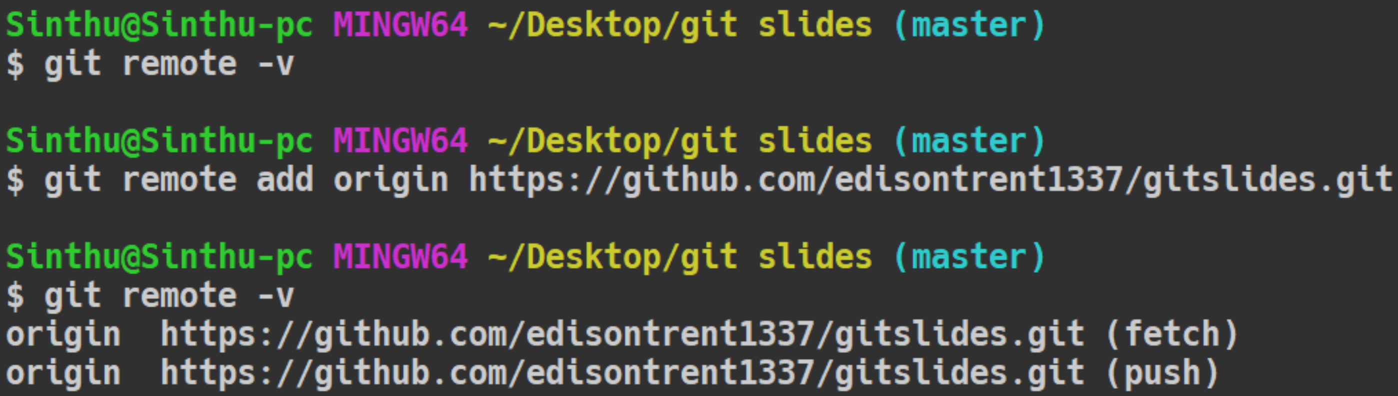
```
git remote add <alias> <url>
```

- Connects local repository with a remote repository
- <alias>: alias, under which the remote is known locally
- <url>: location of the remote repository

```
git remote -v
```

- Lists all available remote repositories

Establishing some connections



```
Sinthu@Sinthu-pc MINGW64 ~/Desktop/git slides (master)
$ git remote -v

Sinthu@Sinthu-pc MINGW64 ~/Desktop/git slides (master)
$ git remote add origin https://github.com/edisontrent1337/gitslides.git

Sinthu@Sinthu-pc MINGW64 ~/Desktop/git slides (master)
$ git remote -v
origin  https://github.com/edisontrent1337/gitslides.git (fetch)
origin  https://github.com/edisontrent1337/gitslides.git (push)
```

Mapping of aliases and
remote URLs

git remote – update URL

```
git remote set-url <alias> <url>
```

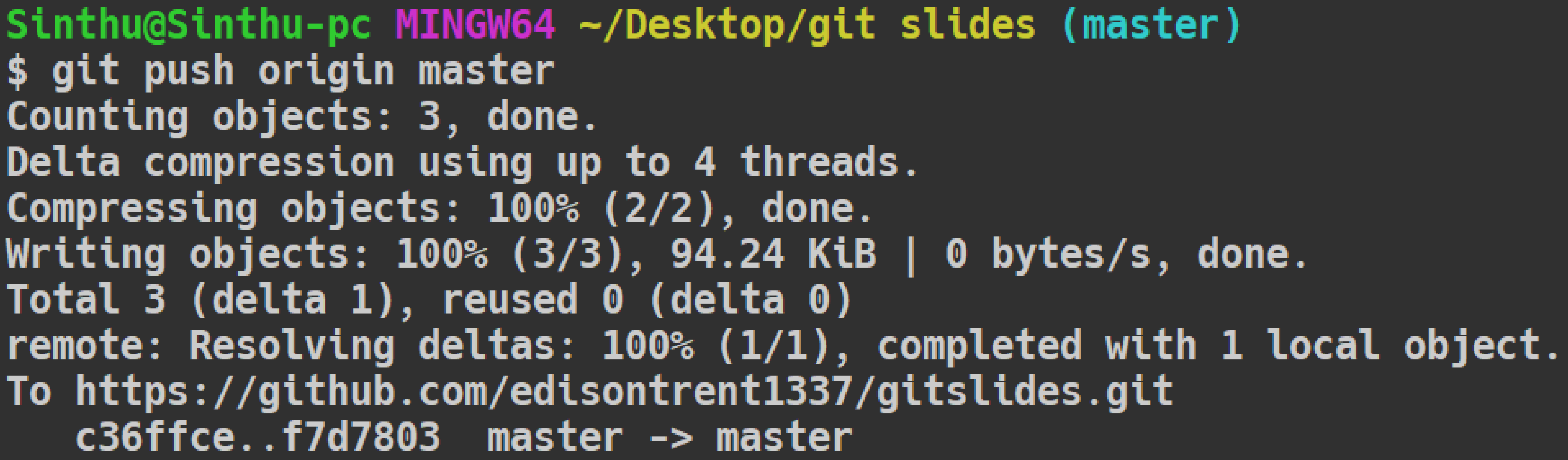
- Updates the remote URL of a given alias
- You need this instruction later when starting to work with the template

Share your bugs with the world!™

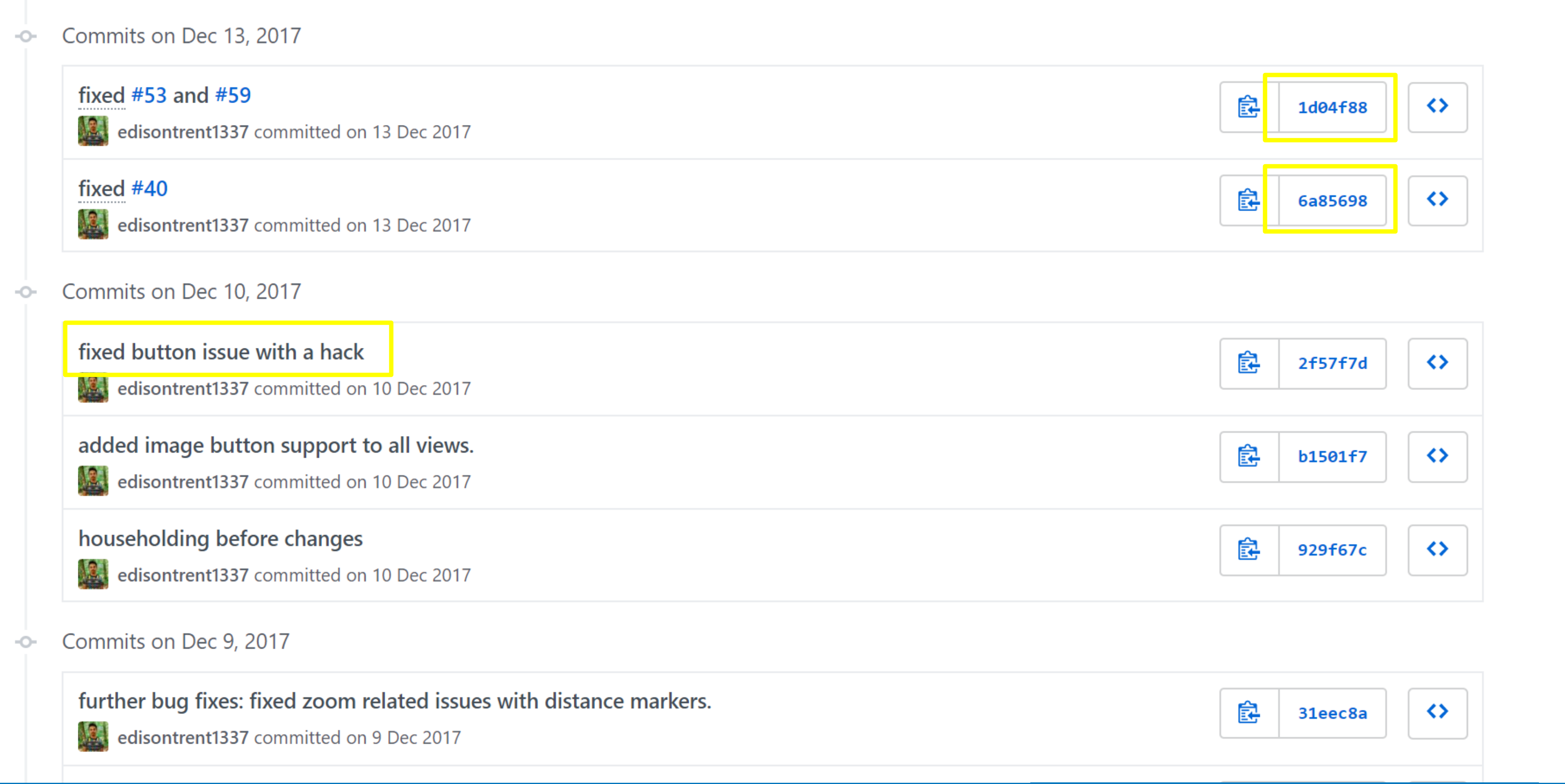
```
git push <alias> <branch>
```

- Pushes all unpublished commits on the branch <branch> of the remote repository called <alias>
- origin is default local alias for remote repository
- Alias simplifies pushes, fully qualifying the URL is not required
- Authentication is required every time when using https

A successful push



```
Sinthu@Sinthu-pc MINGW64 ~/Desktop/git slides (master)
$ git push origin master
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 94.24 KiB | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/edisontrent1337/gitslides.git
   c36ffce..f7d7803  master -> master
```



Commits on Dec 13, 2017

fixed #53 and #59
edisontrent1337 committed on 13 Dec 2017
1d04f88

fixed #40
edisontrent1337 committed on 13 Dec 2017
6a85698

Commits on Dec 10, 2017

fixed button issue with a hack
edisontrent1337 committed on 10 Dec 2017
2f57f7d

added image button support to all views.
edisontrent1337 committed on 10 Dec 2017
b1501f7

householding before changes
edisontrent1337 committed on 10 Dec 2017
929f67c

Commits on Dec 9, 2017

further bug fixes: fixed zoom related issues with distance markers.
edisontrent1337 committed on 9 Dec 2017
31eec8a

Branches – let's get crazy

- Represent separate history of a state of the repository
  - Separate views on the same repository
- Used to develop features independently from other branches and developers
- master is the default branch

git branch

```
git branch <name>
```

- Creates a new branch called <name>
- Branch is an exact copy of the current branch
  → both branches are even

```
git checkout <branch>
```

- Changes the current branch to <branch>
- Future changes now only apply to new branch

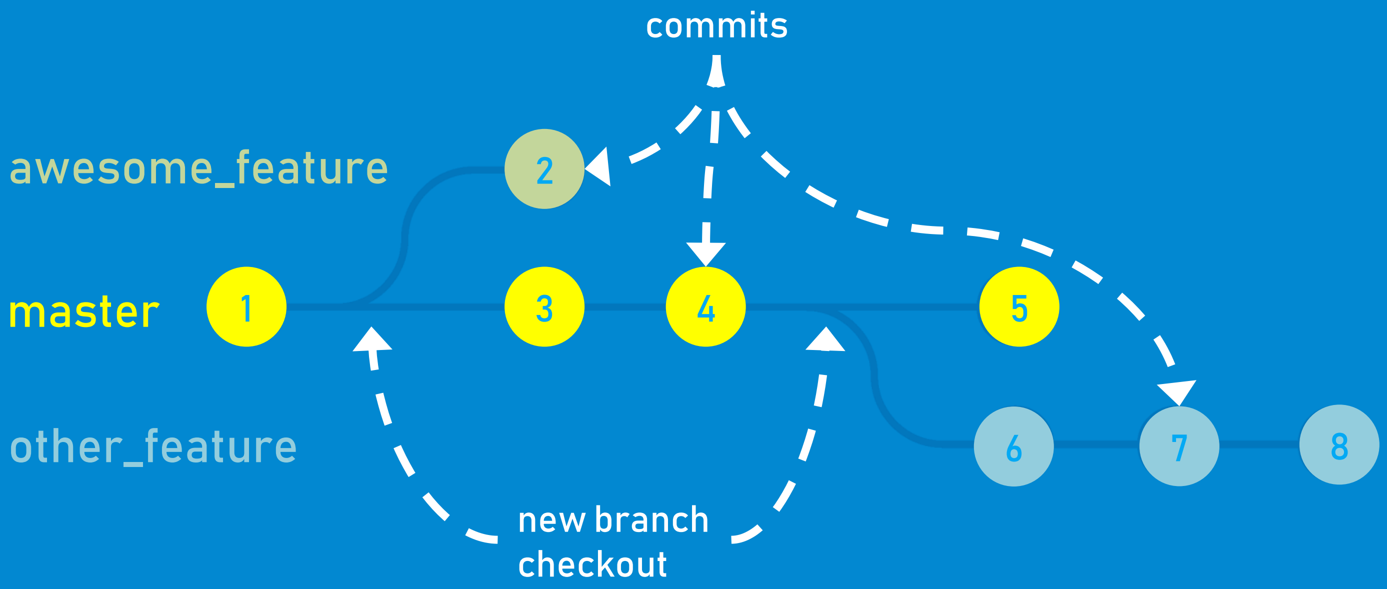
Dude… check out 'git checkout'

- Enables changing between views of different objects in the repository
- git checkout works for single files, commits and branches

```
git checkout -b <branch>
```

- Short form for creating a branch and switching to it immediately

Branching visualized



Staying up-to-date

How to integrate the new bugs my colleague just programmed?

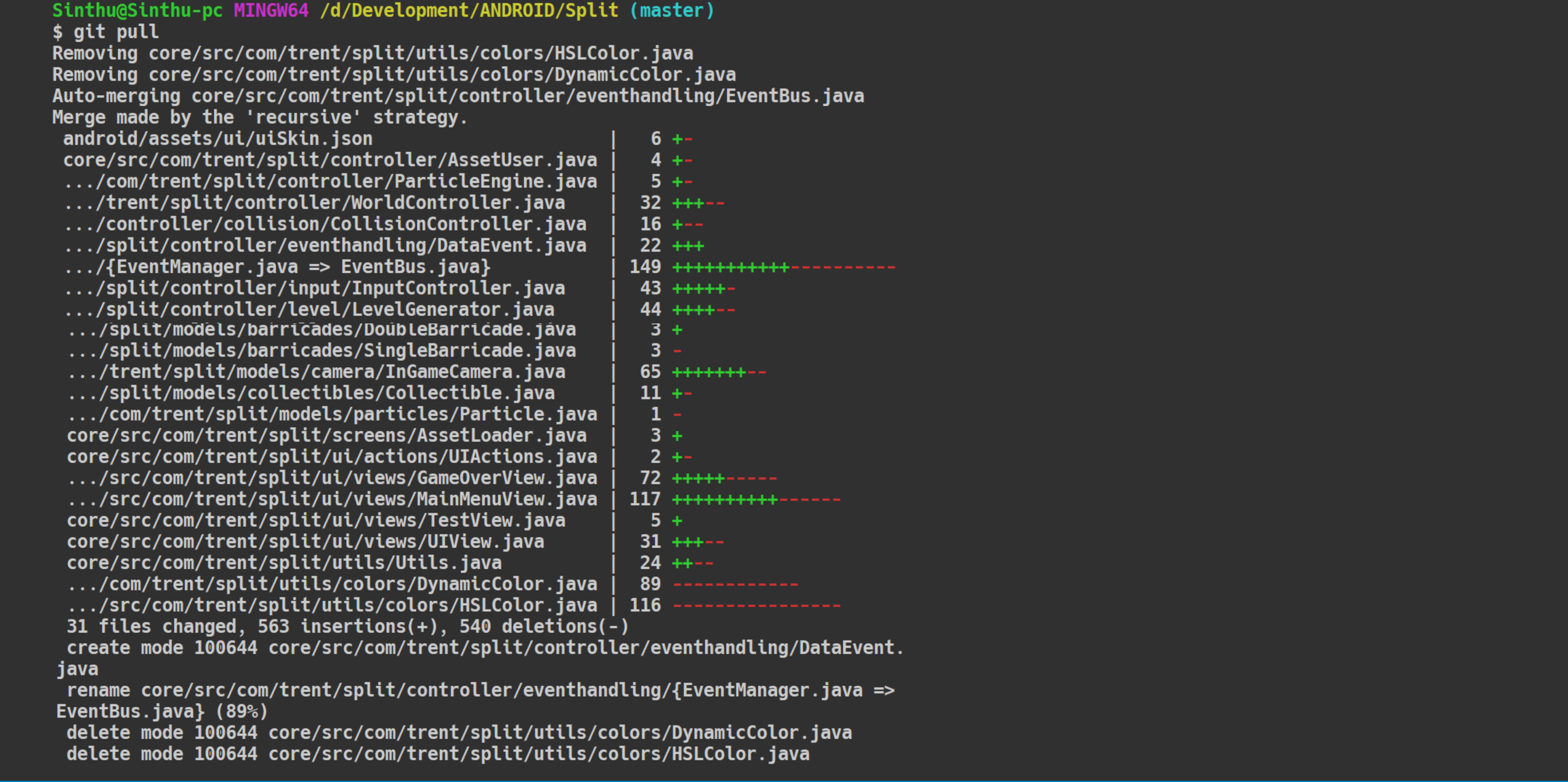
I SEE THAT YOU COMMIT YOUR CODE WITHOUT DOING A PULL FROM GIT TO GET THE LATEST UPDATES
I ALSO LIKE TO LIVE DANGEROUSLY
quickmeme.com

git pull

```
git pull
```

- Fetches the most recent state of the remote repository
- Local view of the repository gets updated
- Short hand for fetching remote state and auto merging
- git tries to auto-merge both versions of the codebase

What could possibly go wrong?



```
Sinthu@Sinthu-pc MINGW64 /d/Development/ANDROID/Split (master)
$ git pull
Removing core/src/com/trent/split/utils/colors/HSLColor.java
Removing core/src/com/trent/split/utils/colors/DynamicColor.java
Auto-merging core/src/com/trent/split/controller/eventhandling/EventBus.java
Merge made by the 'recursive' strategy.
 android/assets/ui/uiSkin.json                      |   6 +-
 core/src/com/trent/split/controller/AssetUser.java |   4 +-
 .../com/trent/split/controller/ParticleEngine.java |   5 +-
 .../trent/split/controller/WorldController.java    |  32 +++--
 .../controller/collision/CollisionController.java  |  16 +--
 .../split/controller/eventhandling/DataEvent.java  |  22 +++
 .../{EventManager.java => EventBus.java}           | 149 +++++++++++----------
 .../split/controller/input/InputController.java    |  43 +++++-
 .../split/controller/level/LevelGenerator.java     |  44 ++++--
 .../split/models/barricades/DoubleBarricade.java   |   3 +
 .../split/models/barricades/SingleBarricade.java   |   3 -
 .../trent/split/models/camera/InGameCamera.java    |  65 +++++++--
 .../split/models/collectibles/Collectible.java     |  11 +-
 .../com/trent/split/models/particles/Particle.java |   1 -
 core/src/com/trent/split/screens/AssetLoader.java  |   3 +
 core/src/com/trent/split/ui/actions/UIActions.java |   2 +-
 .../src/com/trent/split/ui/views/GameOverView.java |  72 +++++-----
 .../src/com/trent/split/ui/views/MainMenuView.java | 117 ++++++++++------
 core/src/com/trent/split/ui/views/TestView.java    |   5 +
 core/src/com/trent/split/ui/views/UIView.java      |  31 +++--
 core/src/com/trent/split/utils/Utils.java          |  24 ++--
 .../com/trent/split/utils/colors/DynamicColor.java |  89 ------------
 .../src/com/trent/split/utils/colors/HSLColor.java | 116 ----------------
 31 files changed, 563 insertions(+), 540 deletions(-)
 create mode 100644 core/src/com/trent/split/controller/eventhandling/DataEvent.
java
 rename core/src/com/trent/split/controller/eventhandling/{EventManager.java =>
EventBus.java} (89%)
 delete mode 100644 core/src/com/trent/split/utils/colors/DynamicColor.java
 delete mode 100644 core/src/com/trent/split/utils/colors/HSLColor.java
```

git merge

TO GIT MERGE!
PAWN SHOP
THE CAUSE OF - AND SOLUTION TO - ALL GIT PROBLEMS!
imgflip.com

git merge – The integration step

```
git merge <otherbranch>
```

- Integrates state of <otherbranch> into current branch
- Is always triggered on a pull
- If merge fails (and boy they do), you are presented with merge-conflicts

That sounds terrifying, because at first, it is!

A simple file in conflict

conflict resolution markers

```
<<<<<<< HEAD:hello_world.py
print(“Hello World!")
=======
print(“Hello!")
print(“Good Bye, World!")
>>>>>>>24b9b78:hello_world.py
```

hello_world.py

local version

remote version

Resolve the conflict by hand

- Remove the conflict resolution markers
- Craft a solution of the code that fits your needs
- Solution can be a mix of the local and remote state
- Add the resolved files, commit and push them

Faith in humanity restored

```
print(“Hello World!")
print(“Good Bye, World!")
```

hello_world.py

local version

remote version

merged version

Merge conflicts

- Conflicts occur if…
  - Remote changes were not integrated properly
  - Remote and local state of files or branches differ “too much”
- So please…
  - Always pull remote changes <u>before</u> you start working
  - Communicate in your team to avoid merge conflicts
  - Use branches to divide work
  - Avoid multiple people working on the same file

Undoing your screwing

```
git reset
```

- Unstages a commit in progress but your local changes stay

```
git reset --hard
```

- Unstages all files and deletes all changes since last commit
- Do not do this if there is work you want to keep!

Almost done

Any general questions before we tackle details to git during RoboLab2018?

?

Hints regarding the internship

- The use of git is mandatory
- Commit regularly: avoid 1000 line monolithic commits
- Commit quality and frequency can and will influence your grade
- Commits should be evenly distributed between all members
- On exam day, the last state of the master branch counts

Gettin' started

1. Create a Bitbucket account & tell your tutor your username, also install git (͡° ͜ʖ ͡°)
2. You will be granted access to the robolab-template-lab repository
3. Only one of the team members clones the repository using the following instruction (recursively!!)

```
git clone --recursive https://bitbucket.com/SE-
Robolab/robolab-template-lab.git
```

Gettin' started

4. Change the remote URL to point to your own team repository on Bitbucket that we prepared for you, using

```
git remote set-url origin https://bitbucket.org/robolab-autumn-18/group-<id>
```

   Replace <id> with your group id with leading zeros!

5. Verify that the remote now points to your team repo with

```
git remote -v
```

Getting' started

6. Perform an initial commit and push it using

```
git push origin master
```

7. All remaining team members clone the freshly populated team repo using:

```
git clone https://bitbucket.org/robolab-autumn-
18/group-<id>
```

Last advice for working with git

- Be careful!
- Read the documentation!
- Do not blame the software, blame yourself!
- Talk to each other to avoid conflicts! Communication is key!

And of course: Have fun!

Done. It’s time to GIT GUD

Any questions remaining?

Please ask!

?