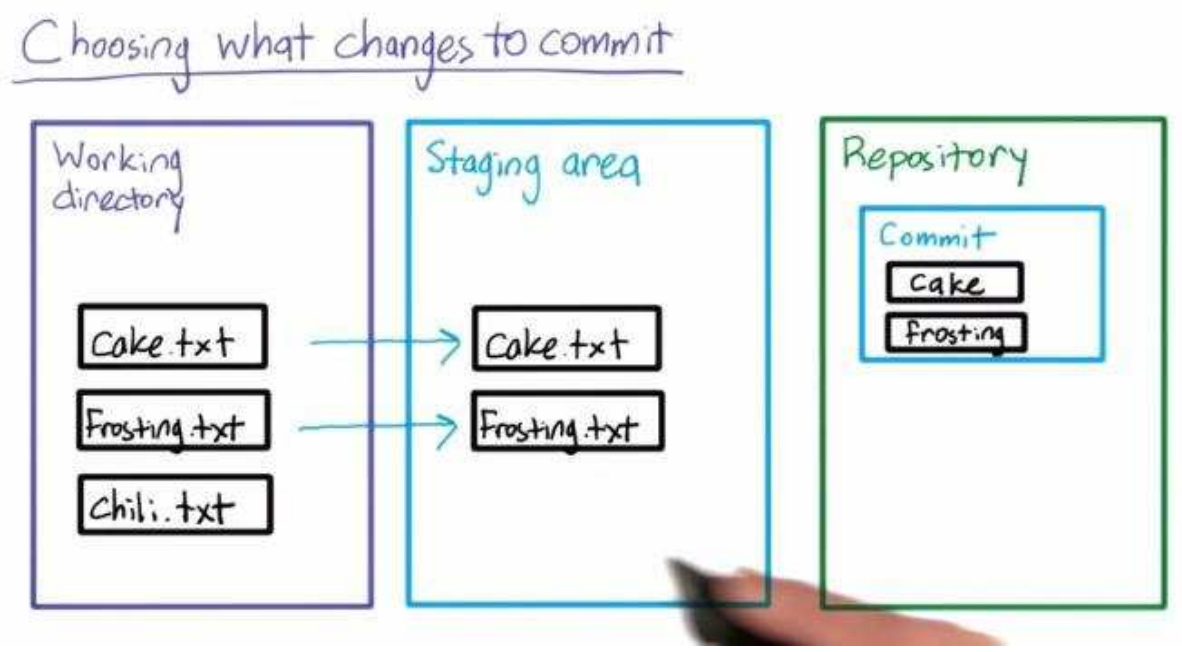
Udacity Git Lecture 2

Notebook: nikunibanka's notebook
Created: 05-01-2015 20:18 **Updated:** 06-01-2015 15:50

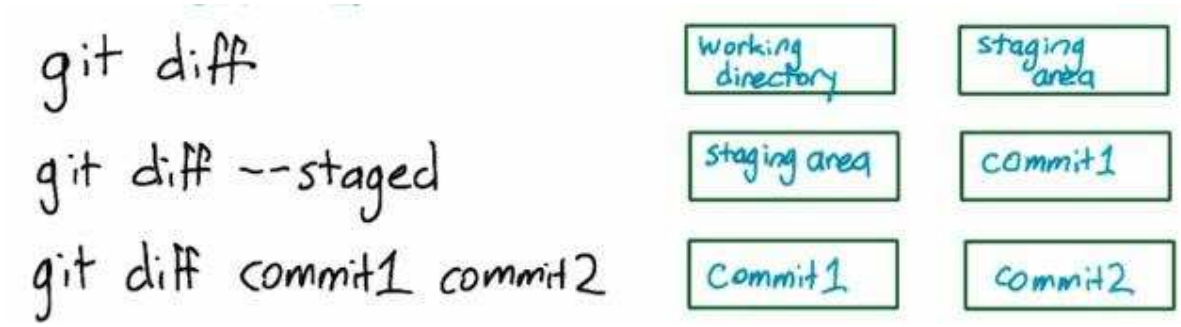
- git init
  creates the .git metadata file and makes the directory a git directory.

Instructor Notes

If you accidentally add a file to the staging area, you can remove it using `git reset`. For example, if you accidentally add lesson_2_reflections.txt, but don't want it to be committed yet, run `git reset lesson_2_reflections.txt` and the file will be removed from the staging area, but it will still be in your working directory.



- There are 3 stages.
  1. Files that have been just created.
  2. Files that are in staging area.
  3. Files in repository.
- git add will add files to the staging area.
- git commit will commit all the files that are in the staging area together.



- **git diff**
  WIll give the changes that we have made but are not in the staging area.
- **git diff --staged**
  will give the changes b/w the staging area and repository.
- **git reset --hard**
  will discard all the changes in the working directory and the staging area.
  - But be very careful!!
  - This command is irreversible.
  - So whatever changes that werent committed will be lost.
-

Instructor Notes

If you are following along, you should run `git checkout master` before you commit. This is because your HEAD is still 'detached' from Lesson 1 when you checked out an old commit, and running this command will fix that.

- git checkout master
  will bring to the tip of the master branch.

- Master is the main branch that is created when we start working on a rep.
- Earlier message of 'Detached head' was: 'Git was warning that we were looking at a commit that was not labeled with a branch name.'
- Tip of that branch: Current last commit made on that branch.
- Committing will only commit the work of the branch that we are currently on.

- git branch
  above command, ie without any args will show the current branches.

Checking out a branch made by someone else

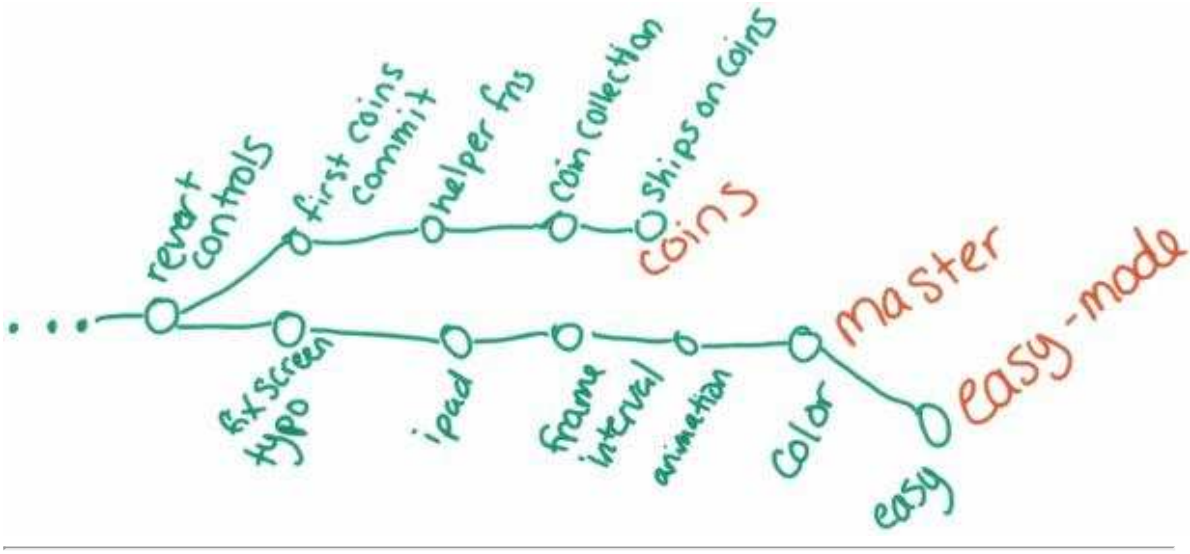
```
Laptop (easy-mode) asteroids $ git checkout coins
Branch coins set up to track remote branch coins from origin.
Switched to a new branch 'coins'
Laptop (coins) asteroids $
```

- Branch coins set up to track remote branch coins from origin.
- This means that coins is a remote branch that we are accessing and it has been set up for us.

- git log --graph --oneline master coins
- creates a graph about the log history giving one line to each commit. And showing the log for the master and coins branches.

```
Laptop (coins) asteroids $ git log --graph --oneline master coins
* f4ce745 Add delay back to bullets
* 3884eab Add color
* 3e42136 now using requestAnimationFrame
* 4035769 frame interval was set wrong after game was paused
* 25ede83 a couple missing ends with the ipad version
* df03538 I can't spell 'screen' apparently :)
| * 354dfdd Make ships able to spawn on coins
| * 0c6daf1 Make it possible to collect coins
| * a3c0ae4 Create helper functions
| * 656b02e First pass at adding coins
|/
* b0678b1 Revert controls
* f19cb1b Fix typo in space
* 75928a9 Use space for movement and enter for shooting
* ac83b72 mostly finished ipad version
* 7ca4826 trying to get div touch controls to work
* 3cb406e first pass at ipad controls
* fe7993e now capturing down key so people don't accidently scroll the page
* 343935f added license
* 80f3bc7 increased canvas size
* 7dc3de0 made the framerate counter dumb but more accurate
* f077ea3 added pointInPolygon method for moz
* 5cb7bed added reverseTransform method
* a53b00a trying to get ff collision detection working
* 1e47136 matrix really only needs to be a 2x3
* 186453f small changes
* c0b670e you get 200 Quatloos for shooting a big flying saucer
* c808f10 bigalien now crosses and disappears
* c588345 move lives out of Ship
* 4bb6bf0 whole lot of refactoring
* 5a54ab9 got the flying saucer working
* f421d99 made ship rotate faster
* 649ccbb collisions now work on dupes
* 1c3ea2d no longer a border, duping rendering on the edges
```

Current History Diagram

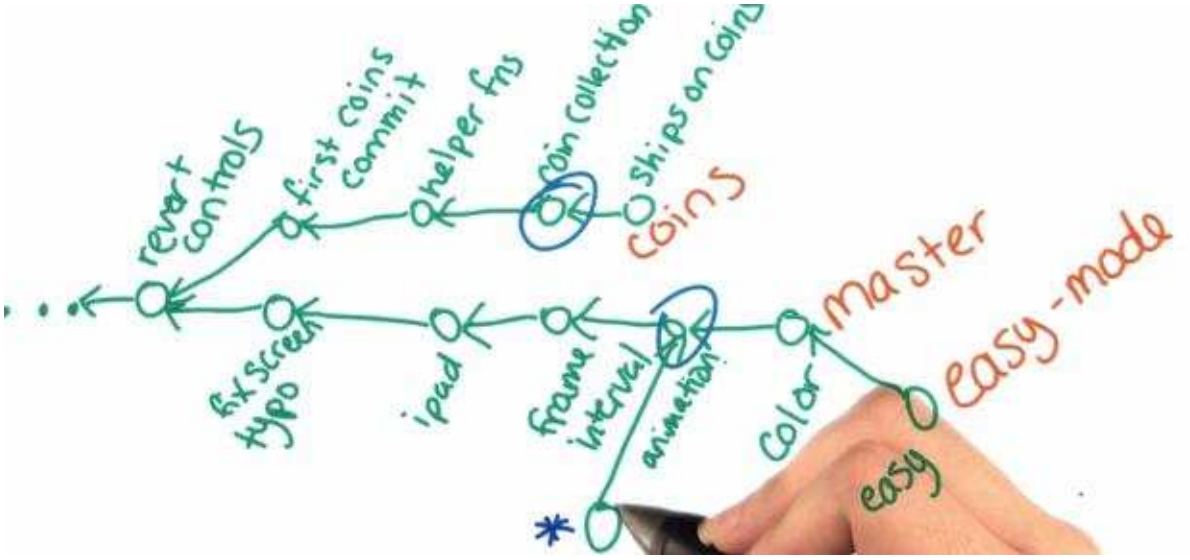


- git log doesnt show all the commits.
- It just shows the commits that are ancestors of the current head.

Revisiting Detached head

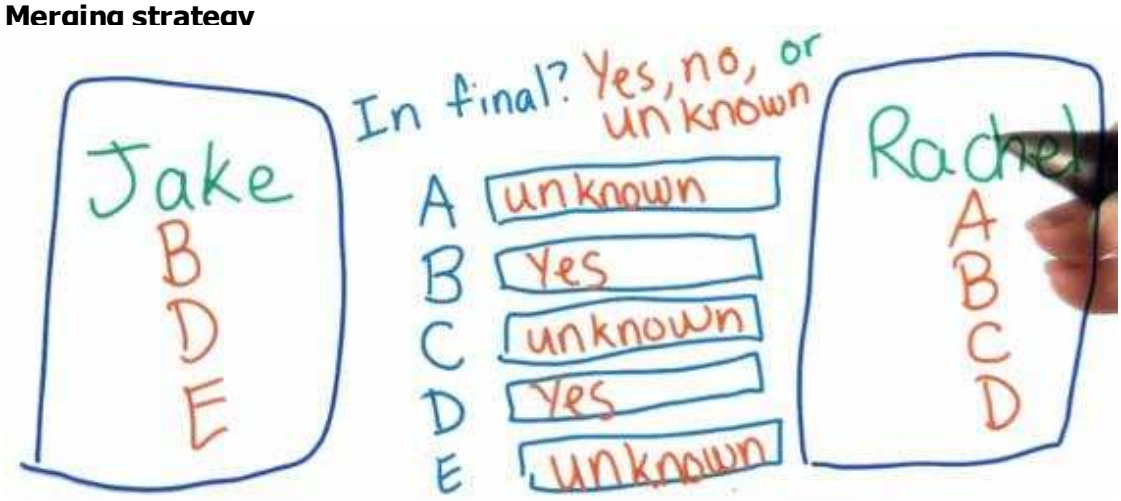
```
asteroids (master)$ git checkout 3e42136a76cf78c6c421cd720427bf6337c2d623
Note: checking out '3e42136a76cf78c6c421cd720427bf6337c2d623'.

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 3e42136... now using requestAnimationFrame
asteroids ((3e42136...))$ git branch new_branch_name
asteroids ((3e42136...))$ git checkout new_branch_name
Switched to branch 'new_branch_name'
asteroids (new_branch_name)$
```

- Detached head message comes when we checkout on a commit instead of a branch.
- It means we are now not on a branch head.
- If we do any commits after making changes on the code now, then a new branch would be created. With the parent = the commit to which we checked out.
- **git checkout --b new_branch_name**
  will create a new branch with the name new_branch_name and commit the changes we've made.
-



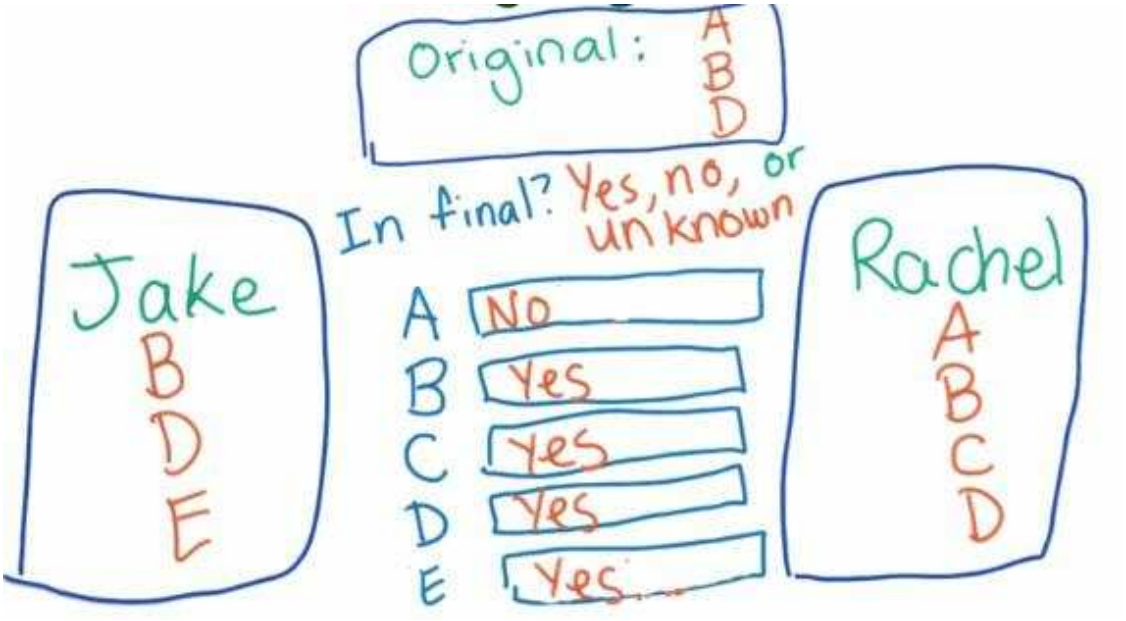
- In this the asterick marked node indicates the new branch that would be created if we commit some changes on a commit id that is not at the head.

Merging strategy



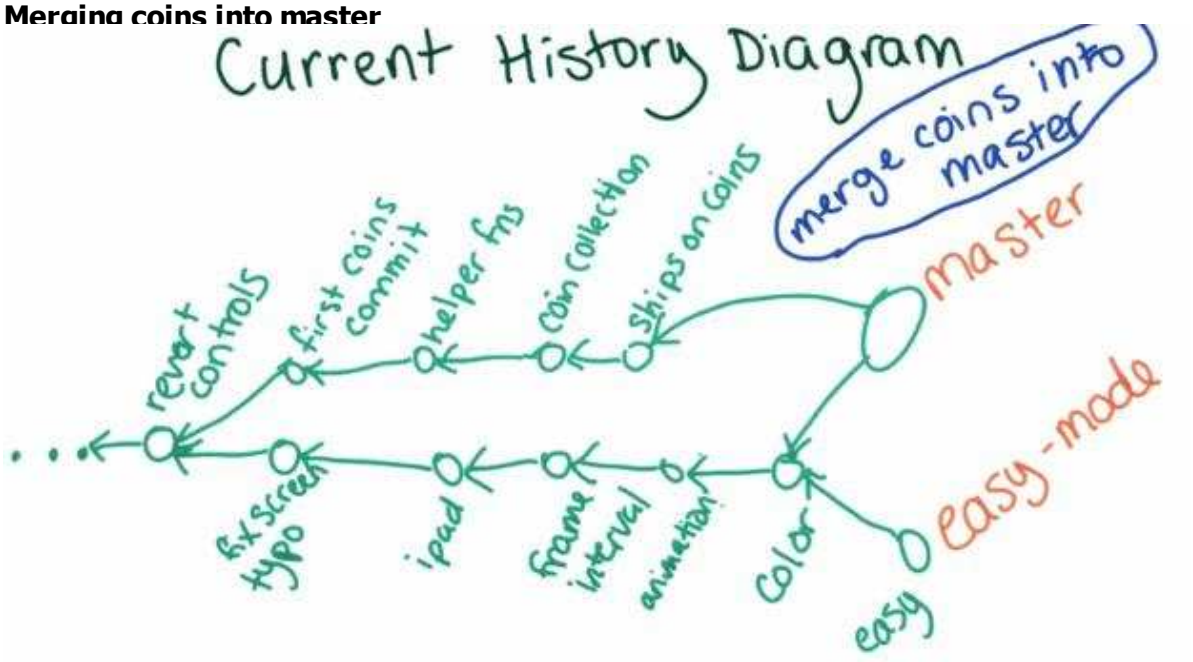
- Since B and D are both in Jake and Rachel's files it must be present in the combined file. (Maybe they were present in the root copy OR maybe they both added the same lines.)
- We aren't sure about A, C, E as we aren't using any information about the root file right now.

Merging strategy when we have the root file also



- Since A was in the original file but jake deleted it. Hence it must not be in final file.
- B and E will be in the final file because they weren't deleted by anyone and have been only added.

Merging coins into master



Points to note when merging branches

1. The parent of the new commit will be both of the branches that have been merged.
2. We can refer to the new node created as the head of the master branch.
   And can **delete the coins branch.**
   - Deleting a branch just deletes the name.
   - The commits stay.
   - But if the commits are unreachable then we would have essentially deleted the commits

also.

3. So after updating the head to be referred as that of master, we will call this as 'merging coins into master branch.'
4. If we do a git log now, then all the branches except the head of the easy-mode branch would be shown sorted by the time. So the coins and master branches would essentially be interleaved in the log.

Instructor Notes

If a branch is deleted and leaves some commits unreachable from existing branches, those commits will continue to be accessible by commit id, until Git's garbage collection runs. This will happen automatically from time to time, unless you actively turn it off. You can also run this process manually with `git gc`.

- git show id
  will show the differences b/w the commit_id and the parent of that commit_id
- git branch -d coins
  will delete the branch coins.

- Note that always checkout in one of the branches that you are merging. Because if you are on branch A and issue
  git merge B C
  then it will merge A, B and C.
- The final name of the branch that you want should be the branch that you are on when issuing the merge command.

A note about `git merge`

`git merge` will also include the currently checked-out branch in the merged version. So if you have branch1 checked out, and you run `git merge branch2 branch3`, the merged version will combine branch1 as well as branch2 and branch3. That's because the branch1 label will update after you make the merge commit, so it's unlikely that you didn't want the changes from branch1 included in the merge. For this reason, you should always checkout one of the two branches you're planning on merging before doing the merge. Which one you should check out depends on which branch label you want to point to the new commit.

- If while merging you get an error about merge conflict.
- Then you can abort the merge with
  git merge --abort

Merge conflict

If you get a message like this

```
Auto-merging game.js
CONFLICT (content): Merge conflict in game.js
Automatic merge failed; fix conflicts and then commit the result.
```

then your files were not in the same state as Caroline's when you started the merge. To fix this, complete the following steps:

1. Restore your files to their state before you started the merge by running `git merge --abort`

Merge conflict (Newline characters between Windows and Unix systems)

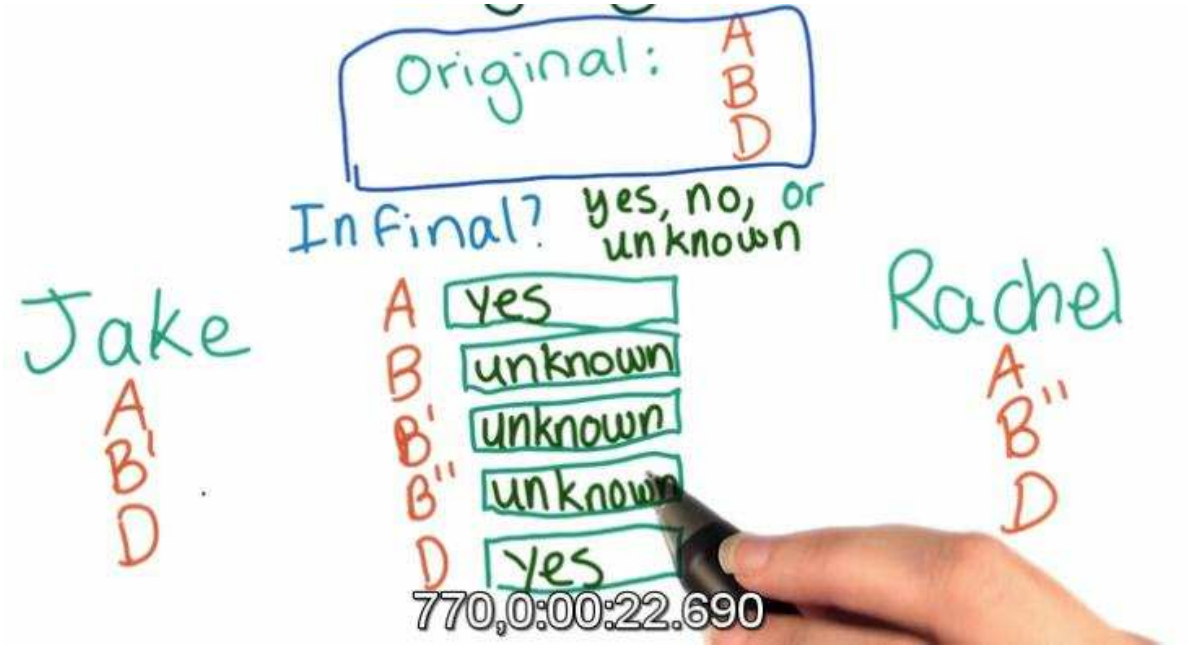
Context: Whenever we hit the "Enter" key on the keyboard, we are actually telling the computer to insert an invisible character into our text file to indicate to the computer that there should be a new line. Unix systems adds one character called the "line feed" character or LF or \n while Windows systems adds two characters, "carriage return" and "line feed" or CRLF or \r\n.

Caroline's files have LF because her files were edited on Mac OSX, which uses LF. If a Windows user were to edit Caroline's files, the Windows text editor might convert all LF to CRLF to make editing files possible. When the Windows user merges her file with Caroline's files, a merge conflict will result due to the different LF and CRLF characters.

To fix this, Windows users should set the global autocrlf attribute to true: `git config --global core.autocrlf true`. More information can be found here: https://help.github.com/articles/dealing-with-line-endings/#platform-all

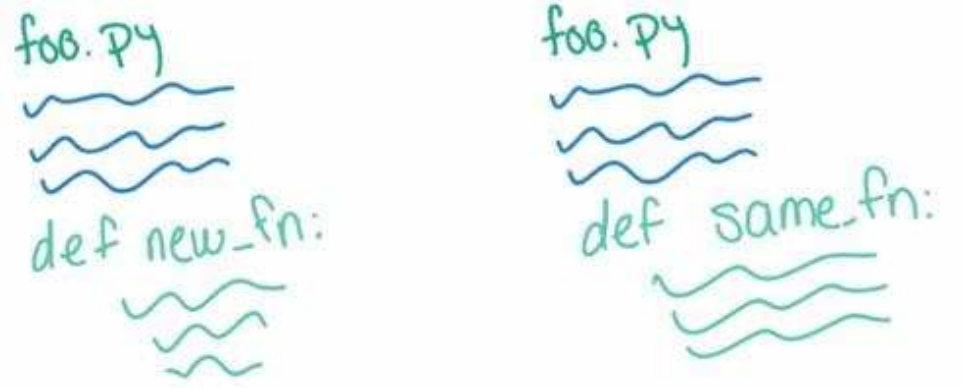
---

Conflict Detection



- A and D will definitely be in the final file.
- We arent sure about B and B'.
- We arent sure about B either because maybe both Jake and Rachel agree that they have the common version. As they have made very minor changes.
- eg Case 1. 2 implementors add 2 functions. But they both are different. So we must add both functions to the final version.

- Case 2. they add functions but both the functions do the same job. But have different names. So maybe we will agree on adding one of the functions that is better implemented.



Resolving merge conflicts in game.js

```
<<<<<<< HEAD
      // break into fragments
      for (var i = 0; i < 2; i++) {
        var roid = $.extend(true, {}, this);
        roid.vel.x = Math.random() * 6 - 3;
        roid.vel.y = Math.random() * 6 - 3;
        if (Math.random() > 0.5) {
          roid.points.reverse();
        }
        roid.vel.rot = Math.random() * 2 - 1;
        roid.move(roid.scale * 3); // give them a little
        Game.sprites.push(roid);
      }
||||||| merged common ancestors
      // break into fragments
      for (var i = 0; i < 3; i++) {
        var roid = $.extend(true, {}, this);
        roid.vel.x = Math.random() * 6 - 3;
        roid.vel.y = Math.random() * 6 - 3;
        if (Math.random() > 0.5) {
          roid.points.reverse();
        }
        roid.vel.rot = Math.random() * 2 - 1;
        roid.move(roid.scale * 3); // give them a little
        Game.sprites.push(roid);
      }
=======
      this.breakIntoFragments();
>>>>>>> master
```

- After issuing
  git merge easy-mode Master
- we get merge conflict in game.js
- So open game.js file and it looks like above.
- The error is b/w the <<<<<< and the >>>>>>. with a divider in b/w ||||||||||
- Top part is code of easy-mode.(when there are 2 frags.)
- Middle part is original code.
- Bottom part is master code.

- So now modify the game.js file and remove the conflicts and extra lines added by git.

- After changing the game.js file. We do a git status
- And get
  both modified: game.js
- This is because

both branches modified the file and there was a conflict.

- Also notice that after committing the changes, only 1 new entry in the log is created. ie. no entries for the conflict and its resolution.

- git log -n 1
  will show 1 last commit.
-
