

Introduction to Version Control using Git



What is Version Control?

- Version Control - A system for managing changes made to documents and other computer files
- What kinds of files can we use it with?
 - Computer Source code
 - Documentation
 - Short stories
 - Binary files (music and pictures)
- What should we use it for?
 - Text files
 - Projects that have lots of revisions (changes)
 - Computer Source code

Many of us have our own version control systems:

Resume_8292014_22.docx

MyMemoirs_RainbowEditV3_112.doc

Program_v1

Many of us have our own version control systems:

Resume_8292014_22.docx

MyMemoirs_RainbowEditV3_112.doc

Program_v1

When we're trying to do this with many files it becomes difficult to manage and error prone.

**Software developers realized this and
created version control systems for
computer source code.**

What does it look like?

The screenshot displays the GitHub web interface for a repository. The top section shows a commit history sidebar with a list of commits and their authors. The main area shows a diff view for the file 'build/shared/lib/languages/PDE_es.properties'. The diff highlights changes in the 'menu.debug' section, with green lines indicating additions and red lines indicating deletions. The commit is by Federico Bond, dated 2015-07-17. The interface includes a search bar, a diff view with line numbers, and a commit history sidebar.

Details for each change including who made it, when it was made and what the changes were.

The screenshot shows the gitk GUI for the 'Processing' repository. The top panel displays a commit graph on the left and a list of commits on the right. A large black arrow points from the top of the commit list down to the commit details panel at the bottom.

Commit List (Right Panel):

Author	Commit Hash	Date
Jakub Valtar	<jakub.valtar@gmail.com>	2015-08-05 16:01:21
codeanticode	<andres.colubri@gmail.com>	2015-08-05 13:01:41
Ben Fry	<fry@processing.org>	2015-08-05 09:34:20
Ben Fry	<fry@processing.org>	2015-08-05 09:31:03
Ben Fry	<fry@processing.org>	2015-08-05 09:29:50
Ben Fry	<fry@processing.org>	2015-08-05 09:28:45
Ben Fry	<fry@processing.org>	2015-08-05 09:17:52
Ben Fry	<fry@processing.org>	2015-08-05 09:05:35
Ben Fry	<fry@processing.org>	2015-08-05 08:36:49
Ben Fry	<fry@processing.org>	2015-08-05 08:33:16
codeanticode	<andres.colubri@gmail.com>	2015-08-05 08:15:46
codeanticode	<andres.colubri@gmail.com>	2015-08-05 07:52:25
Ben Fry	<fry@processing.org>	2015-08-05 06:28:23
Ben Fry	<benfry@users.noreply.github.com>	2015-08-05 05:53:03
Akarshit Wal	<akarshitwal@gmail.com>	2015-07-16 10:27:51
Ben Fry	<benfry@users.noreply.github.com>	2015-08-05 05:52:32
Federico Bond	<federicobond@gmail.com>	2015-07-17 13:05:06
Ben Fry	<fry@processing.org>	2015-08-05 05:50:43
Ben Fry	<fry@processing.org>	2015-08-05 05:48:53
Ben Fry	<fry@processing.org>	2015-08-05 05:40:28
Ben Fry	<fry@processing.org>	2015-08-05 05:40:18
Ben Fry	<fry@processing.org>	2015-08-05 05:33:34
codeanticode	<andres.colubri@gmail.com>	2015-08-04 07:30:41

Commit Details (Bottom Panel):

SHA1 ID: 65598bdbc5d2243d0eee7e1ea25314d5f338521 | Row 238 / 10848

Find | next | prev | commit | containing: | Exact | All fields

Search | Patch | Tree

Diff | Old version | New version | Lines of context: 3 | Ignore space change | Li

Author: Federico Bond <federicobond@gmail.com> 2015-07-17 13:05:06
Committer: Federico Bond <federicobond@gmail.com> 2015-07-17 19:06:19
Parent: 9bef0e72eb625dca721b1aade0b689ab3f857882 (rolling over for the next release)
Child: 5710eba048771a5c01f6839c6cb66bb4d0333cc7 (Merge pull request #3480 from federicobond/es-trans)
Branches: master, remotes/origin/master
Follows: processing-0238-3.0a11
Precedes: processing-0239-3.0b1

Update spanish translation

----- build/shared/lib/languages/PDE_es.properties -----
index 0f36fab..272a023 100644
@@ -66,27 +66,30 @@ menu.sketch.add_file = Añadir archivo
| File | Edit | Sketch | Debug | Tools | Help |
| | | | Debug | |
menu.debug = Depuración
-menu.debug.show_debug_toolbar = Mostrar barra de herramientas de depuración
-menu.debug.debug = Depuración
+menu.debug.enable = Activar depurador
+menu.debug.disable = Desactivar depurador
+menu.debug.show_debug_toolbar = Mostrar barra de herramientas de depuración

Comments
build/shared/lib/languages/PDE_es.properties

SHA-1 Checksum for each change

The screenshot shows the gitk GUI for the 'Processing' repository. A large black arrow points from the title 'SHA-1 Checksum for each change' to the SHA-1 ID field, which displays '65598bdbc5d2243d0eee7e1ea25314a15f338521'. The commit history on the right lists various contributors and their commit times. The diff view at the bottom shows the changes in the file 'build/shared/lib/languages/PDE_es.properties', including updates to the menu.debug section.

gitk: Processing

SHA1 ID: 65598bdbc5d2243d0eee7e1ea25314a15f338521

Row 238 / 10848

Find next prev commit containing: Exact All fields

Search

Diff Old version New version Lines of context: 3 Ignore space change Li

Author: Federico Bond <federicobond@gmail.com> 2015-07-17 13:05:06
Committer: Federico Bond <federicobond@gmail.com> 2015-07-17 19:06:19
Parent: 9bef0e72eb625dca721b1aade0b689ab3f857882 (rolling over for the next release)
Child: 5710eba048771a5c01f6839c6cb66bb4d0333cc7 (Merge pull request #3480 from federicobond/es-trans)
Branches: master, remotes/origin/master
Follows: processing-0238-3.0a11
Precedes: processing-0239-3.0b1

Update spanish translation

----- build/shared/lib/languages/PDE_es.properties -----
index 0f36fab..272a023 100644
@@ -66,27 +66,30 @@ menu.sketch.add_file = Añadir archivo
| File | Edit | Sketch | Debug | Tools | Help |
| | | | Debug |
menu.debug = Depuración
-menu.debug.show_debug_toolbar = Mostrar barra de herramientas de depuración
-menu.debug.debug = Depuración
+menu.debug.enable = Activar depurador
+menu.debug.disable = Desactivar depurador
+menu.debug.show debug toolbar = Mostrar barra de herramientas de depuración

Comments
build/shared/lib/languages/PDE_es.properties

Each change is also marked with
a descriptive comment

The screenshot shows the gitk GUI for the 'Processing' repository. The top panel displays a commit history with a list of commit messages on the left and a list of commit details (author, date) on the right. A vertical arrow points from the text 'Each change is also marked with a descriptive comment' to the commit messages in the history. The bottom panel shows a detailed view of a specific commit, including the commit message, the commit hash, and the diff of the files changed.

Commit history (SHA1 ID, Row, 238 / 10848):

- Do not filter Ctrl+Alt+? out as menu mnemonics
- any lighting change triggers flush
- todo notes
- notes on implementation
- remove dead (hopefully) code, start on contrib counter
- remove dialog box about contrib updates
- remove spacing performance art to normalize w/ the guidelines
- Q: Who likes to break the build? A: Who doesn't.
- rework WebFrame implementation so that images work (fixes #3494)
- Merge branch 'master' of github.com:processing/processing
- use dot literal to split device id string
- use different device parsing method on Linux
- Merge branch 'master' of github.com:processing/processing
- Merge pull request #3470 from Akarshit/gsoc-CMtable
- Adding ellipses only when text is long
- Merge pull request #3480 from federicobond/es-translation
- Update spanish translation
- more notes about fixes
- fix button firing so that mouse movement does not interfere (fixes #3529)
- add notes for recent updates
- deal with the empty sketch case
- change up how extra statements from size() are handled (fixes #3531)
- Merge pull request #3528 from jakubvaltar/bunfly-nehama-rahmanov

Commit details (SHA1 ID: 6558bdbc5d2243d0eee7e1ea25314a15f338521):

Author: Federico Bond <federicobond@gmail.com> 2015-07-17 13:05:06
Committer: Federico Bond <federicobond@gmail.com> 2015-07-17 19:06:19
Parent: 9bef0e72b625dca721b1aade0b689ab3f857882 (rolling over for the next release)
Child: 5710eba88771a5c81f6839c6cb66bb4d0333cc7 (Merge pull request #3480 from federicobond/es-trans)
Branches: master remotes/origin/master
Follows: processing-0238-3.0a11
Precedes: processing-0239-3.0b1

Update spanish translation

Diff (Patch view):

```
index 0f36fab..272a023 100644
@@ -66,27 +66,30 @@ menu.sketch.add_file = Añadir archivo
 # | File | Edit | Sketch | Debug | Tools | Help |
 # | Debug |
 menu.debug = Depuración
 -menu.debug.show_debug_toolbar = Mostrar barra de herramientas de depuración
 -menu.debug.debug = Depuración
 +menu.debug.enable = Activar depurador
 +menu.debug.disable = Desactivar depurador
 +menu.debug.show_debug_toolbar = Mostrar barra de herramientas de depuración
```


The Processing project has tracked 10848 changes(additions, deletions, modifications) to it's source code since it's inception(2001), for example.

gitk: Processing

- made buzz.pl support multiple subdirectories
- fixed nasty OutOfMemoryError bug. removing basicstamp dir
- some tweaking on the serial stuff
- updating the todo list
- oops
- cleanup after japan
- cleaning up this town
- lots of changes and fixes to pde, can now export
- lots of changes from sunday/monday morning in japan
- updating changes and tastiness from the plane flight
- file remover and update tested on the mac, working with minor tweaks
- remove double thread crap (make mac happy), fix full screen for mrj, auto-updater code
- cleanup in pde properties, turn off lighting by default in ProcessingApplet
- realigned buttons at top, moved help text message for buttons ot upper area
- play button stays lit, applet window comes forward when fullscreen toggled, more small changes
- making processingapplet run smoothly, properly destroy old ones (though no gc), proper file cleanup
- automatic generation of g. code for processingapplet
- working on improving kjc engine
- getting kjc to work, adding python support files
- python runs, but doesn't quit
- full screen toggle works
- Initial revision

SHA1 ID: beddf8abdd328653ef9b4979c4104312af5897f3

Row 10848 / 10848

Find next prev commit containing: Exact All fields

Search

Diff Old version New version Lines of context: 3 Ignore space changes

Author: benfry <fry@processing.org> 2001-07-26 10:09:40
Committer: benfry <fry@processing.org> 2001-07-26 10:09:40
Child: [0819b2743df0b4f5dc3a01557228e85a9531d45c](#) (full screen toggle works)
Branches: [master](#), remotes/origin/master
Follows:
Precedes: [processing-0093@1666](#), [processing-0216-2.0b8](#), [processing-2.0b8](#)

Initial revision

Patch Tree

Comments

Why would use version control for software based art?

- Languages change, source code needs to be updated to run on current machines and operating systems.
- Maybe we want to comment the source code and track the progress of that documentation.
- Perhaps the artist used version control, the history would give us insight into the creation of the work.

For example..

What is Git

- An open source Version Control System(VCS) designed for speed and efficiency
- Better than competing tools
- Created by Linus Torvalds in 2005 (for source code for the Linux operating system)
- Your best insurance policy against:
 - Accidental mistakes like deleting work
 - Remembering what was changed, when, why and by whom
 - Your hard drive blowing up, *if you sync with a external server

Where do you get it:

<https://git-scm.com/>

(The official site)

-Has installers for OSX, Windows and Linux

-It's free and open source

-Awesome free book online for going further.

<https://git-scm.com/book/en/v2>

The Command Line

There are a lot of different ways to use Git.

There are the original command line tools and there are many graphical user interfaces(GUI).

For this workshop, we will be using Git on the command line.

The command line is the only place you can run all Git commands – most of the GUIs only implement some subset of Git functionality for simplicity.

Teaching Style

I'm going to walk through 2-3 slides of commands that we will type.

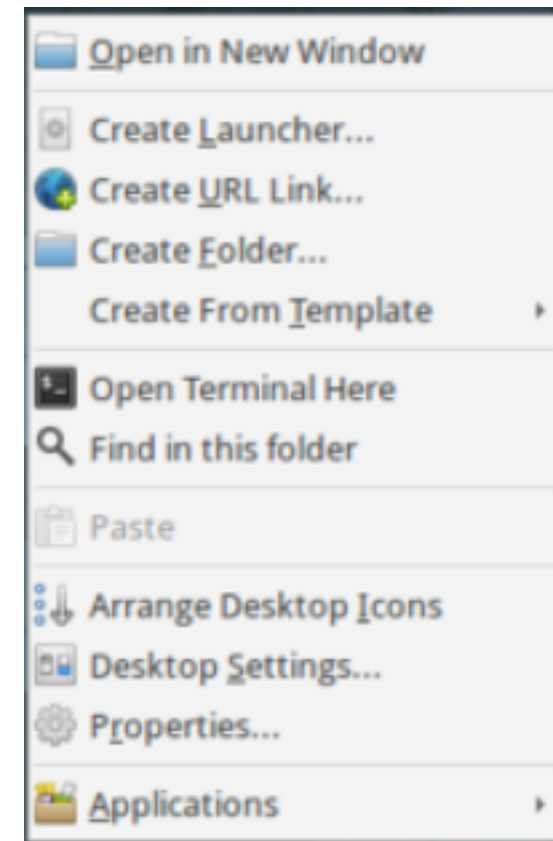
Then I will switch to the computer so that we can type them in together.

I encourage you to keep this PDF open on your own computer as a reference
(it's on the thumb drive)

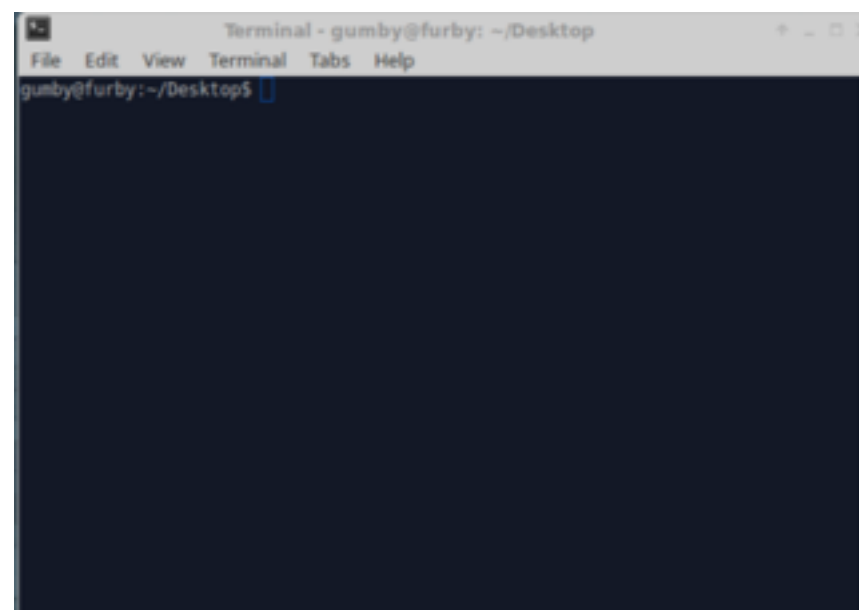
If you need assistance, raise your hand and one of the excellent TA's will come to help you.

Let's open the terminal now to get started:

1. On the desktop right click, or for Mac users CTRL+Click
2. Click on the option 'Open Terminal Here'



You should now have a black terminal window.



Setting Up Git

First-Time Git Setup

Before we start using Git to track out source code we need to do a little set up. Here we will tell git about ourself, so it knows who is making the changes

first lets type:

git config --global user.name "Your Name"

A screenshot of a terminal window titled "Terminal - gumby@furby: ~/Desktop". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The terminal shows the command "git config --global user.name 'Mark Hellar'" being entered and executed. The prompt "gumby@furby:~/Desktop\$" is visible before and after the command. A green cursor is positioned at the end of the second prompt line.

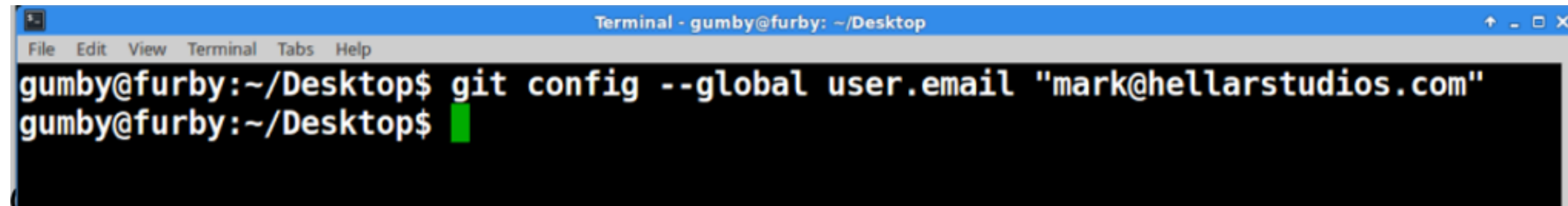
```
Terminal - gumby@furby: ~/Desktop
File Edit View Terminal Tabs Help
gumby@furby:~/Desktop$ git config --global user.name "Mark Hellar"
gumby@furby:~/Desktop$
```

You wont get any feedback, but we will check the setting in just a bit.

First-Time Git Setup

Now enter your email by typing:

git config --global user.email johndoe@example.com

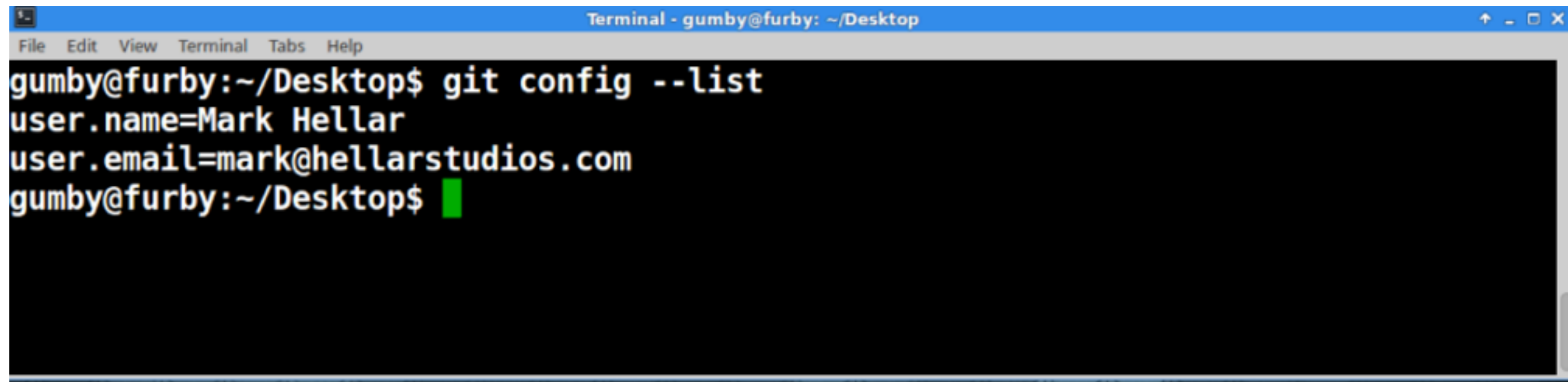
A screenshot of a terminal window titled "Terminal - gumby@furby: ~/Desktop". The terminal has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The prompt is "gumby@furby:~/Desktop\$". The command "git config --global user.email \"mark@hellarstudios.com\"" has been entered and executed. The prompt "gumby@furby:~/Desktop\$" is shown again with a green cursor.

```
Terminal - gumby@furby: ~/Desktop
File Edit View Terminal Tabs Help
gumby@furby:~/Desktop$ git config --global user.email "mark@hellarstudios.com"
gumby@furby:~/Desktop$
```

First-Time Git Setup

We can check our settings by typing:

git config --list

A screenshot of a terminal window titled "Terminal - gumby@furby: ~/Desktop". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The terminal shows the command "git config --list" being executed, with the output "user.name=Mark Hellar" and "user.email=mark@hellarstudios.com". The prompt "gumby@furby:~/Desktop\$" is visible at the end of the output lines.

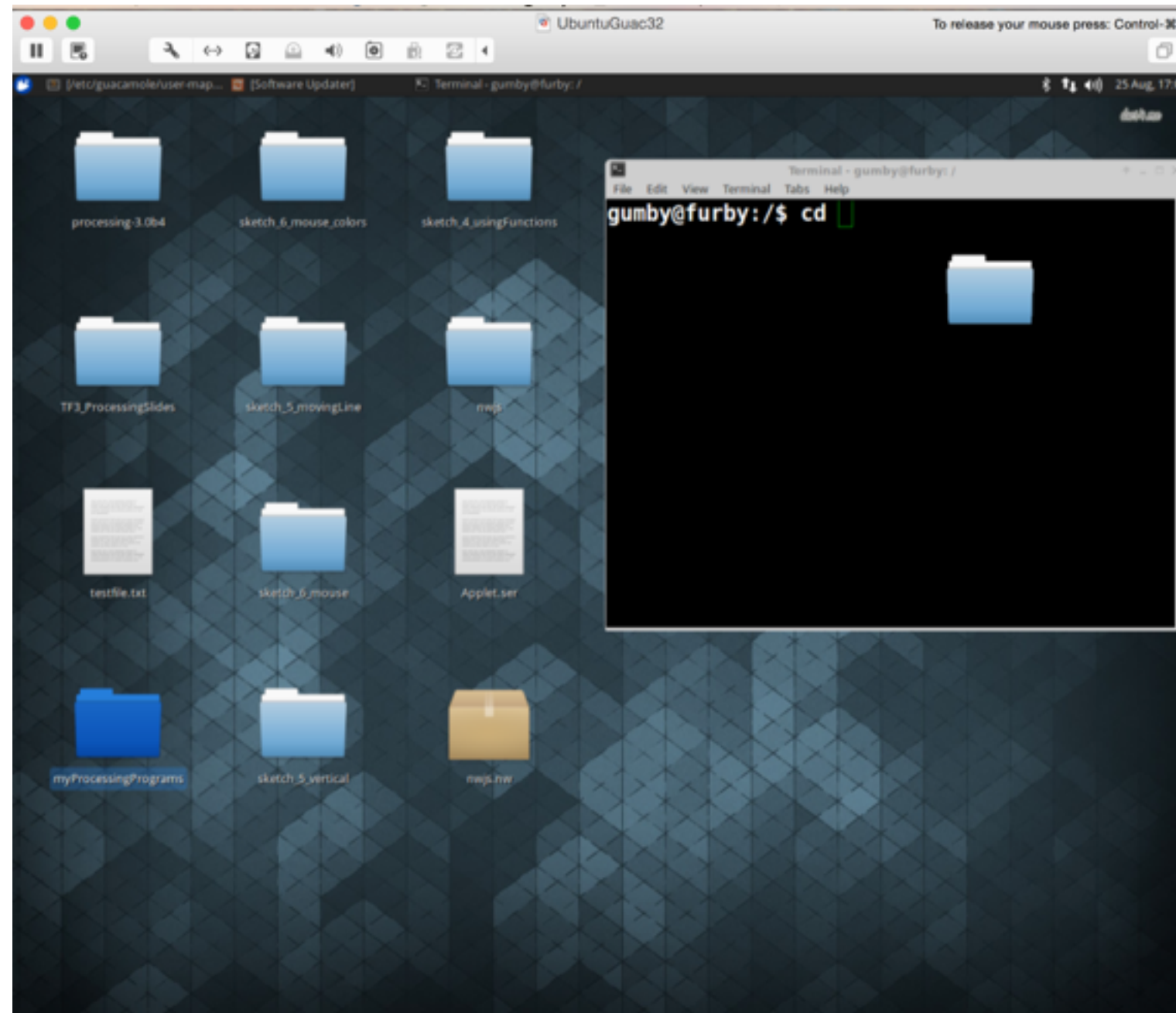
```
Terminal - gumby@furby: ~/Desktop
File Edit View Terminal Tabs Help
gumby@furby:~/Desktop$ git config --list
user.name=Mark Hellar
user.email=mark@hellarstudios.com
gumby@furby:~/Desktop$
```

If you see a mistake, you can correct it by re-entering any of the commands from the last 2 slides.

Creating our first repository

Type:
cd

Hit space and drag **myProcessingPrograms** on the terminal



Your terminal should now look like this:
Hit Return

A screenshot of a terminal window titled "Terminal - gumby@furby: /". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The terminal content shows the command "cd /home/gumby/Desktop/myProcessingP" followed by "rograms" on the next line. A green cursor is positioned at the end of the second line, ready for a return key press.

```
Terminal - gumby@furby: /
File Edit View Terminal Tabs Help
gumby@furby:/$ cd /home/gumby/Desktop/myProcessingP
rograms
```

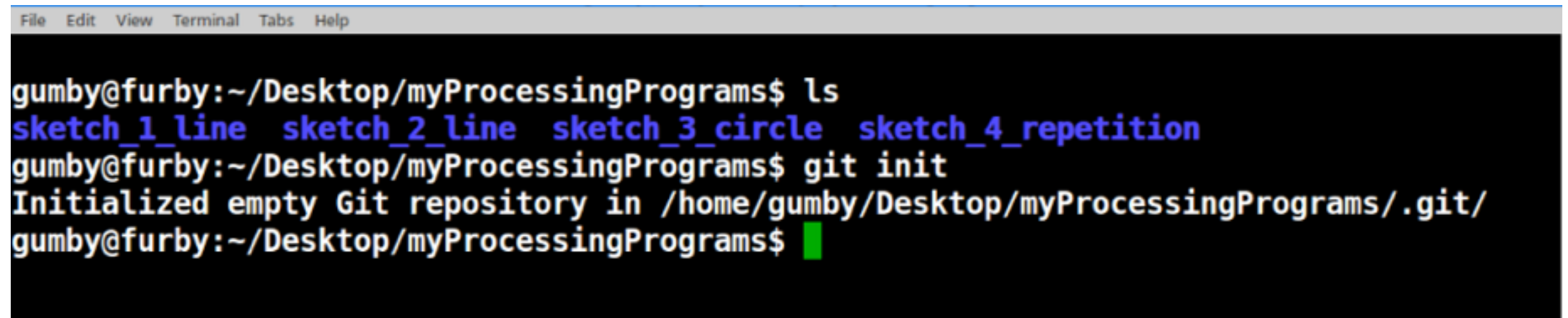
cd means Change Directory
since we dragged our processing directory onto it
we have moved into that directory

Creating an initial source code repository:

to do this type:

git init

and hit return

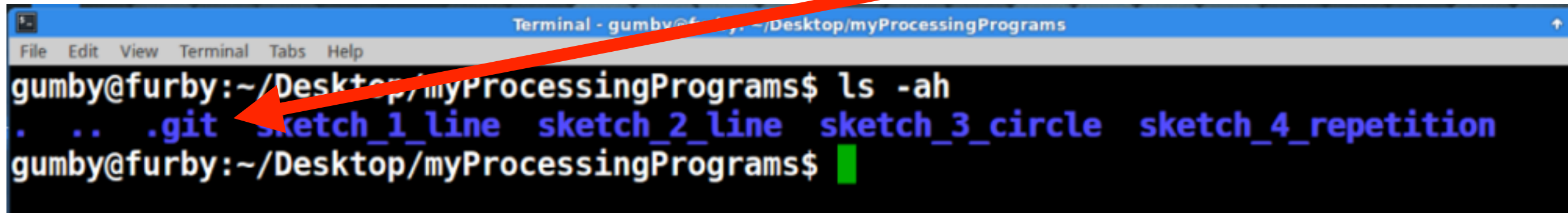
A screenshot of a terminal window with a menu bar at the top containing 'File', 'Edit', 'View', 'Terminal', 'Tabs', and 'Help'. The terminal shows a user named 'gumby' at a machine named 'furby' in the directory '~/Desktop/myProcessingPrograms'. The user runs 'ls' and lists four files: 'sketch_1_line', 'sketch_2_line', 'sketch_3_circle', and 'sketch_4_repetition'. Then, the user runs 'git init', and the terminal outputs 'Initialized empty Git repository in /home/gumby/Desktop/myProcessingPrograms/.git/'. The prompt returns to the user's shell.

```
gumby@furby:~/Desktop/myProcessingPrograms$ ls
sketch_1_line sketch_2_line sketch_3_circle sketch_4_repetition
gumby@furby:~/Desktop/myProcessingPrograms$ git init
Initialized empty Git repository in /home/gumby/Desktop/myProcessingPrograms/.git/
gumby@furby:~/Desktop/myProcessingPrograms$
```

I get some feedback telling me that a empty repository has been created.

Creating an initial source code repository:

When we ran '**git init**' in the last step, the Git program created a special folder called `.git`

A screenshot of a macOS Terminal window. The title bar reads "Terminal - gumby@furby: ~/Desktop/myProcessingPrograms". The menu bar shows "File", "Edit", "View", "Terminal", "Tabs", and "Help". The terminal text shows the user running the command "ls -ah" in the directory "~/Desktop/myProcessingPrograms". The output lists the files: ".", "..", ".git", "sketch_1_line", "sketch_2_line", "sketch_3_circle", and "sketch_4_repetition". A red arrow points from the ".git" folder in the text above to the ".git" folder in the terminal output. The prompt "gumby@furby:~/Desktop/myProcessingPrograms\$" is visible at the end of each line.

```
gumby@furby:~/Desktop/myProcessingPrograms$ ls -ah
.  ..  .git sketch_1_line sketch_2_line sketch_3_circle sketch_4_repetition
gumby@furby:~/Desktop/myProcessingPrograms$
```

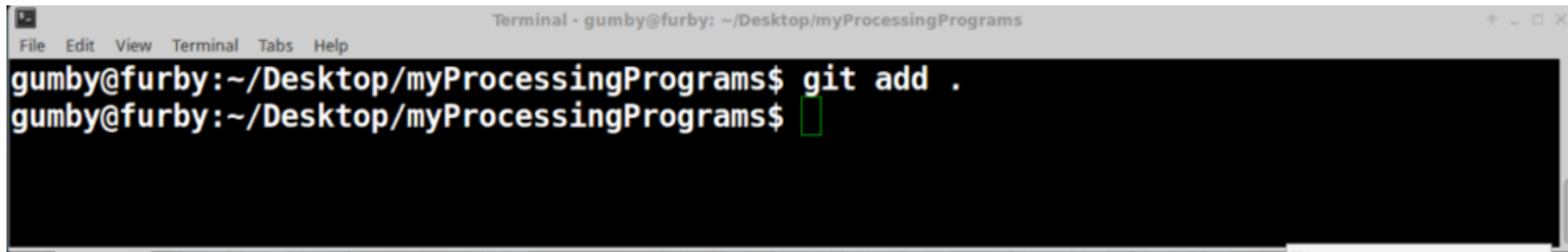
This folder is called a repository, it is where Git will store a copy of the source code and all cumulative changes.

The new repository is currently empty so now we have to add some files to it.

I can add all the files in the directory by typing

git add .

and return

A screenshot of a terminal window titled "Terminal - gumby@furby: ~/Desktop/myProcessingPrograms". The terminal shows the command "git add ." being entered and executed. The prompt "gumby@furby:~/Desktop/myProcessingPrograms\$" is visible before and after the command. The command is highlighted in yellow in the original image. The terminal background is black, and the text is white. The window has a standard macOS-style title bar with a menu bar (File, Edit, View, Terminal, Tabs, Help) and window control buttons (back, forward, close).

```
Terminal - gumby@furby: ~/Desktop/myProcessingPrograms
File Edit View Terminal Tabs Help
gumby@furby:~/Desktop/myProcessingPrograms$ git add .
gumby@furby:~/Desktop/myProcessingPrograms$
```

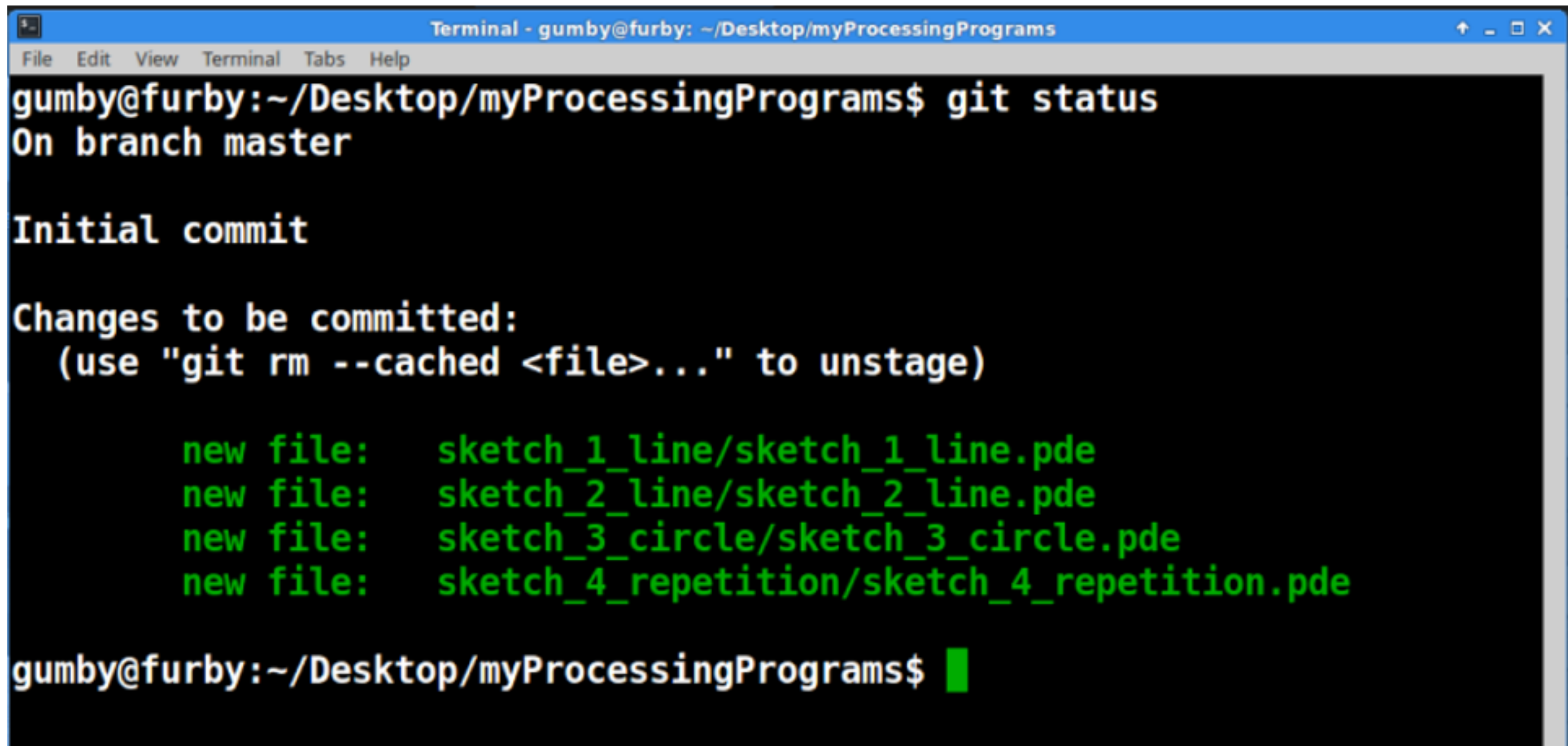
The '**git add**' command places files in a **staging** area that we will be **commit** to the repository

The period in this statement is shorthand for all files and directories

Now if I type :

git status

Git tells me that a number of files are ready to be committed

A screenshot of a terminal window titled "Terminal - gumby@furby: ~/Desktop/myProcessingPrograms". The terminal shows the command "git status" being executed. The output indicates an initial commit on the master branch with four new files staged for commit. The files are listed in green text: sketch_1_line/sketch_1_line.pde, sketch_2_line/sketch_2_line.pde, sketch_3_circle/sketch_3_circle.pde, and sketch_4_repetition/sketch_4_repetition.pde. The prompt "gumby@furby:~/Desktop/myProcessingPrograms\$" is visible at the bottom with a green cursor.

```
Terminal - gumby@furby: ~/Desktop/myProcessingPrograms
File Edit View Terminal Tabs Help
gumby@furby:~/Desktop/myProcessingPrograms$ git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

    new file:   sketch_1_line/sketch_1_line.pde
    new file:   sketch_2_line/sketch_2_line.pde
    new file:   sketch_3_circle/sketch_3_circle.pde
    new file:   sketch_4_repetition/sketch_4_repetition.pde

gumby@furby:~/Desktop/myProcessingPrograms$
```

I will explain Staging Vs. Committing in depth in just a bit

type:

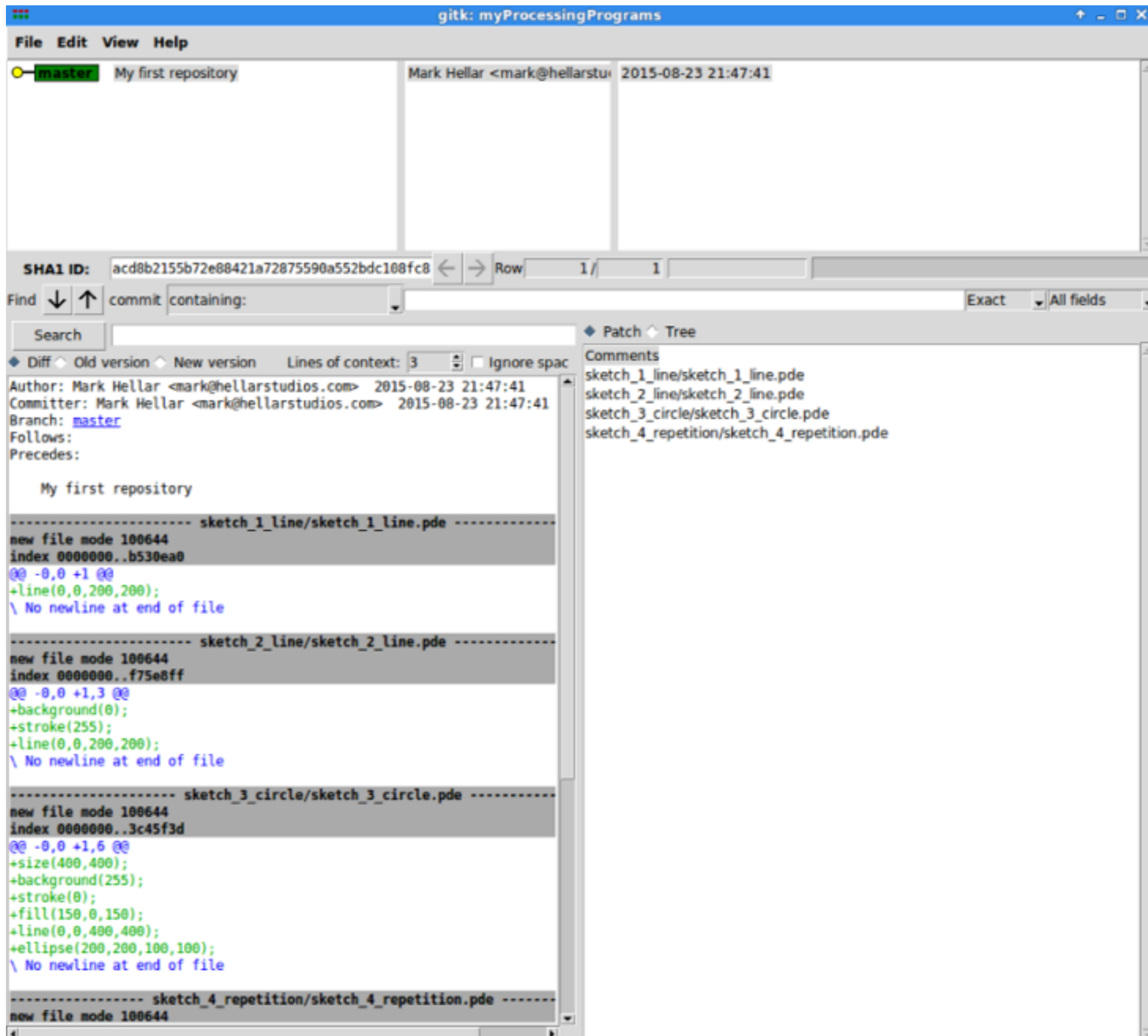
git commit -m “My First Repository”
and hit return

```
gumby@furby:~/Desktop/myProcessingPrograms$ git commit -m "My first repository"
[master (root-commit) acd8b21] My first repository
4 files changed, 29 insertions(+)
create mode 100644 sketch_1_line/sketch_1_line.pde
create mode 100644 sketch_2_line/sketch_2_line.pde
create mode 100644 sketch_3_circle/sketch_3_circle.pde
create mode 100644 sketch_4_repetition/sketch_4_repetition.pde
gumby@furby:~/Desktop/myProcessingPrograms$
```

We got a lot of feedback here, this is Git telling us that the files have been added to our repository.

-m is an option or flag that means message, the text in quotes is the commit message

So what does the repository look like?
Type **'gitk'** and hit return



I know there were a few steps here.

But, once the initial repository is created and the files have been added it becomes very easy to track your changes

I'm going to hand you back to Deena now so that you can write some more code.

After that we'll commit your new code into the repository

Leave the terminal window open, we'll come back to it later

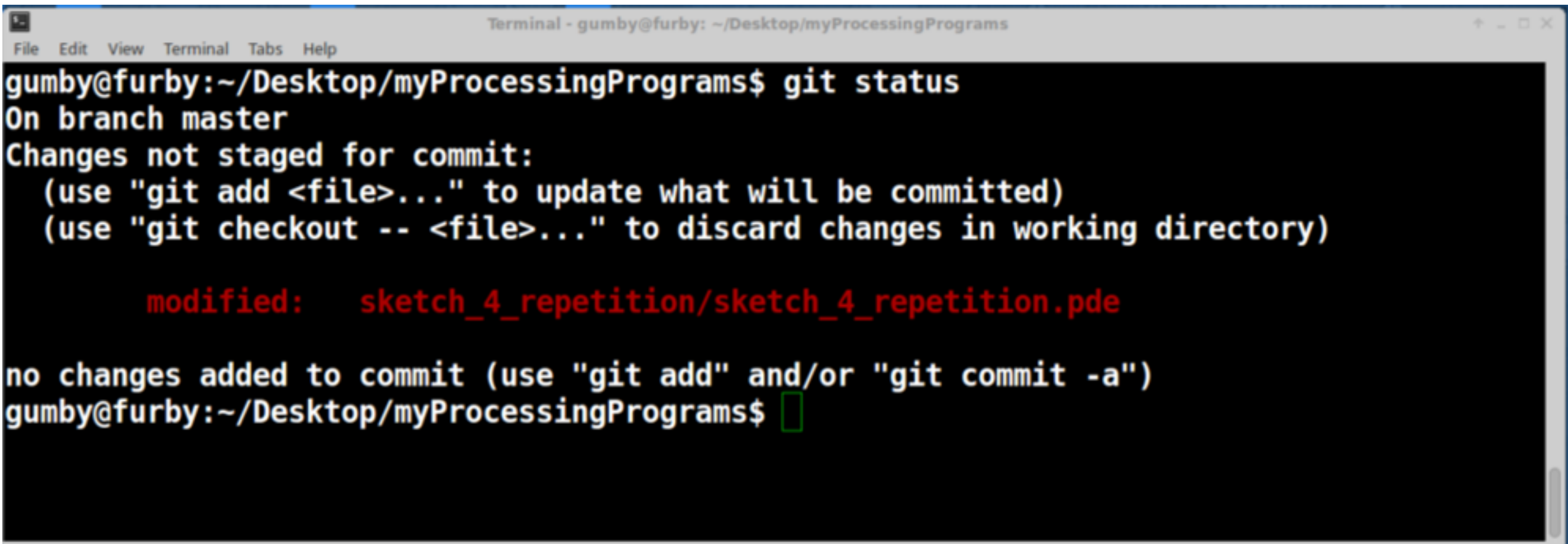
Tracking Changes

Git workflow

Ok so now we have a repository let's define a workflow to track changes:

1. Edit, add or delete files
2. Stage the changes
3. Review your changes
4. Commit the changes

Edit, add or delete files at the terminal Type: **git status**

A screenshot of a terminal window titled "Terminal - gumby@furby: ~/Desktop/myProcessingPrograms". The terminal shows the command "git status" being executed. The output indicates that the user is on the "master" branch and that there are changes not staged for commit. Specifically, the file "sketch_4_repetition/sketch_4_repetition.pde" is listed as "modified". The terminal also provides instructions on how to stage changes using "git add" or "git commit -a".

```
gumby@furby:~/Desktop/myProcessingPrograms$ 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:   sketch_4_repetition/sketch_4_repetition.pde

no changes added to commit (use "git add" and/or "git commit -a")
gumby@furby:~/Desktop/myProcessingPrograms$
```

Git lets you know that there have been changes to your
program

Git knows that a file has changed, now we need stage it

Stage the changes

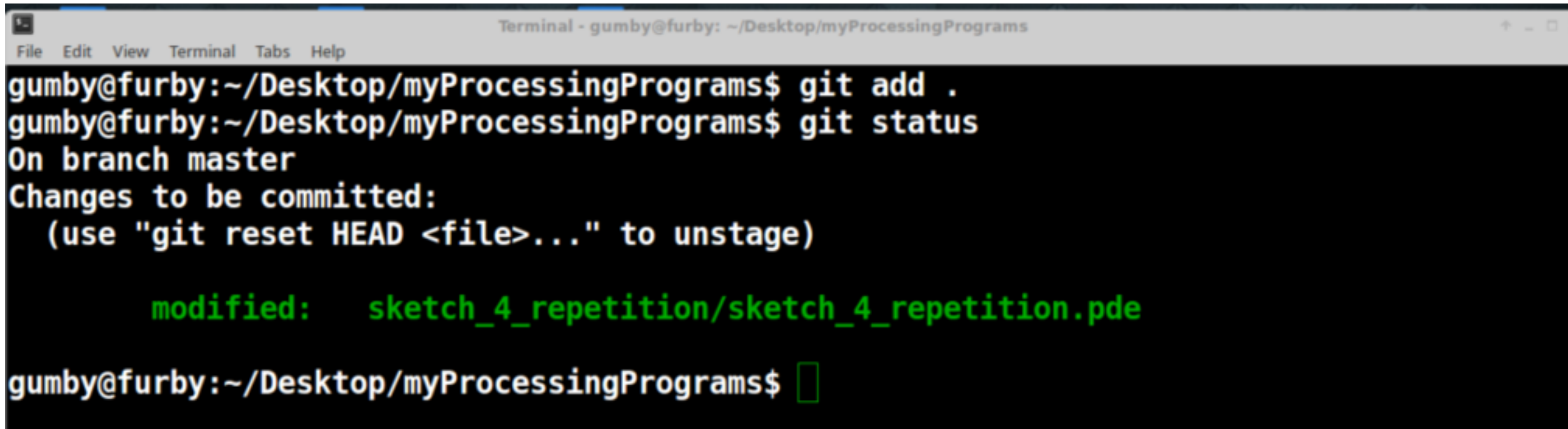
at the terminal Type:

git add .

hit return

git status

hit return

A screenshot of a terminal window titled "Terminal - gumby@furby: ~/Desktop/myProcessingPrograms". The terminal shows the following commands and output:

```
gumby@furby:~/Desktop/myProcessingPrograms$ git add .
gumby@furby:~/Desktop/myProcessingPrograms$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

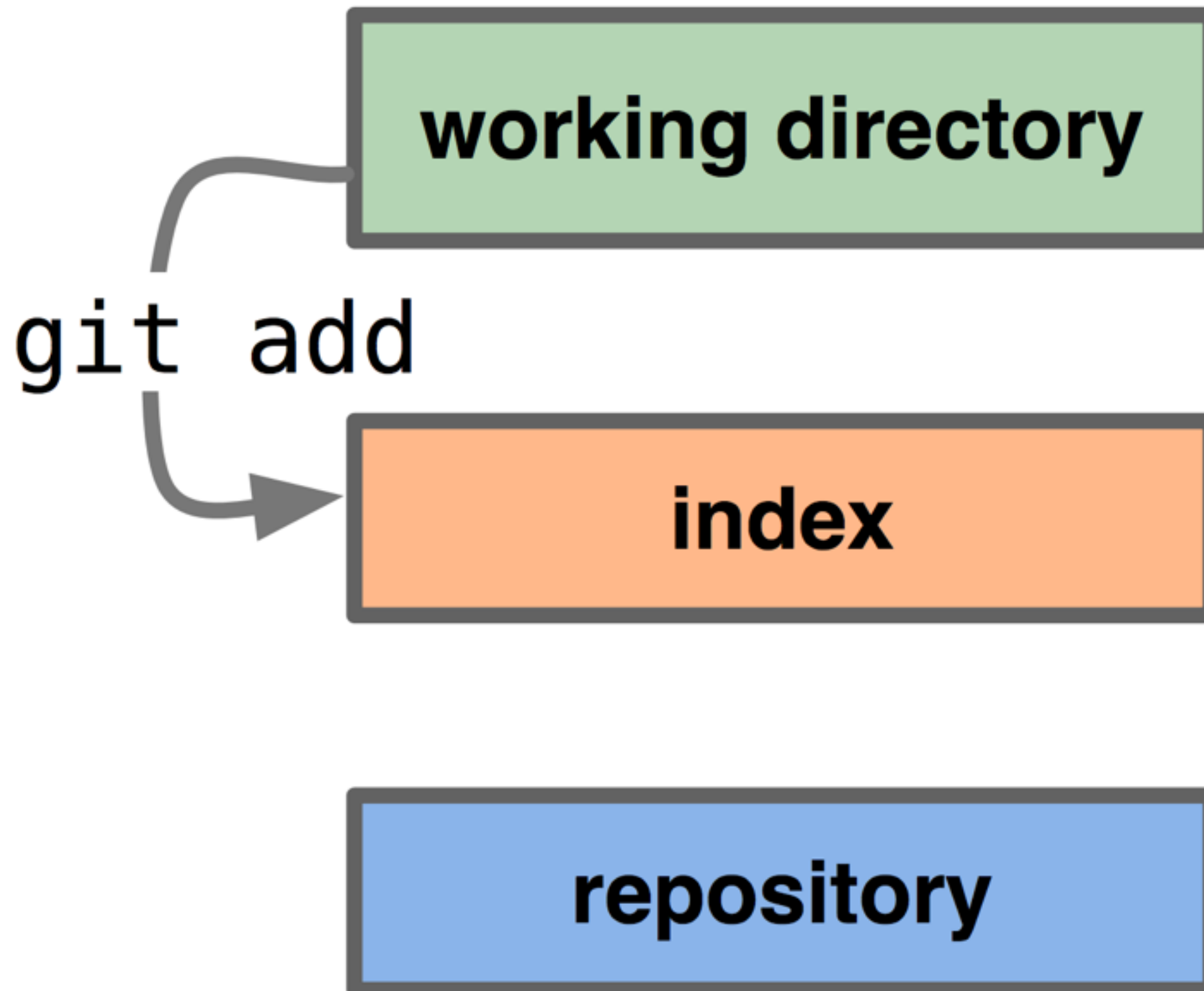
    modified:   sketch_4_repetition/sketch_4_repetition.pde

gumby@furby:~/Desktop/myProcessingPrograms$
```

git add . sends all files to the staging area

git status confirms that

Stage the changes



Review your changes

With our files in the staging area, we can compare differences with files in the repository before we commit them:

git diff —staged

A screenshot of a terminal window titled "Terminal - gumby@furby: ~/Desktop/myProcessingPrograms". The terminal shows the command "git diff --staged" and its output. The output indicates a change in the file "a/sketch_4_repetition/sketch_4_repetition.pde" to "b/sketch_4_repetition/sketch_4_repetition.pde". The change is a modification of the "size(400,400);" line, changing the fill color from red to green. The terminal also shows a comment about setting up ellipse parameters as integer variables and the initialization of x and y variables. The output ends with "(END)".

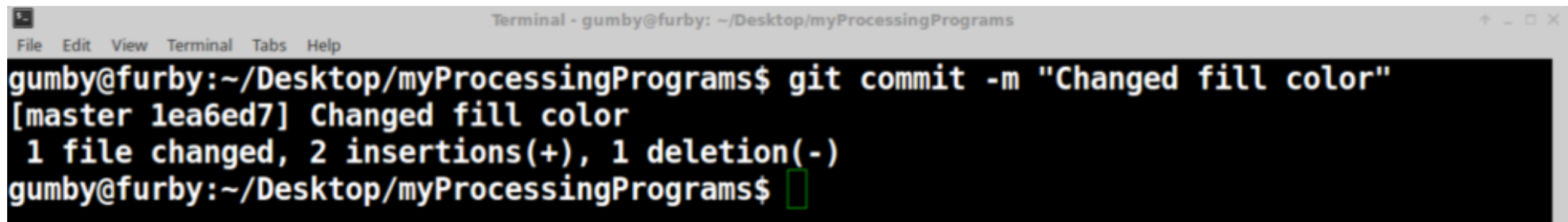
```
gumby@furby:~/Desktop/myProcessingPrograms$ git diff --staged
diff --git a/sketch_4_repetition/sketch_4_repetition.pde b/sketch_4_repetition/sketch_4_repetition.pde
index ebf1143..0f746b4 100644
--- a/sketch_4_repetition/sketch_4_repetition.pde
+++ b/sketch_4_repetition/sketch_4_repetition.pde
@@ -2,8 +2,9 @@ size(400,400);
  background(255);
  stroke(0);
  line(0,0,400,400);
- fill(150,0,150);
+ fill(150,255,150);
  ellipse(200,200,100,100);
+
  /* setting up the ellipse parameters as integer variables: */
  int x = 200;
  int y = 200;
(END)
```

(hit q to quit)

Commit the changes

If we are happy with our changes we can now commit them into the repository:

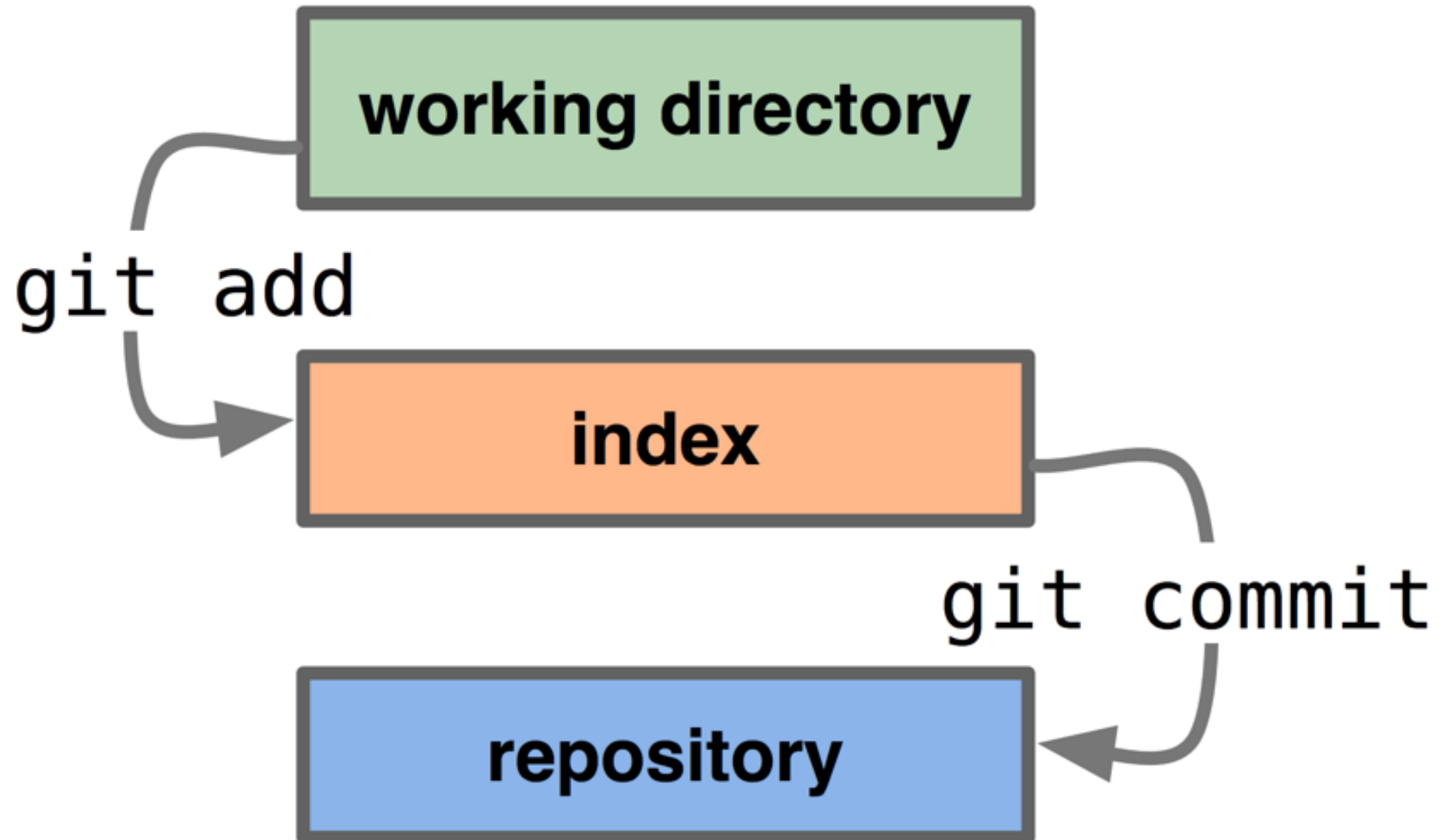
git commit -m “changed fill color”

A screenshot of a terminal window titled "Terminal - gumby@furby: ~/Desktop/myProcessingPrograms". The terminal shows the command "git commit -m 'Changed fill color'" being executed. The output indicates a successful commit on the master branch with hash 1ea6ed7, describing the change as "Changed fill color". It also shows that 1 file was changed with 2 insertions and 1 deletion. The prompt returns to the shell.

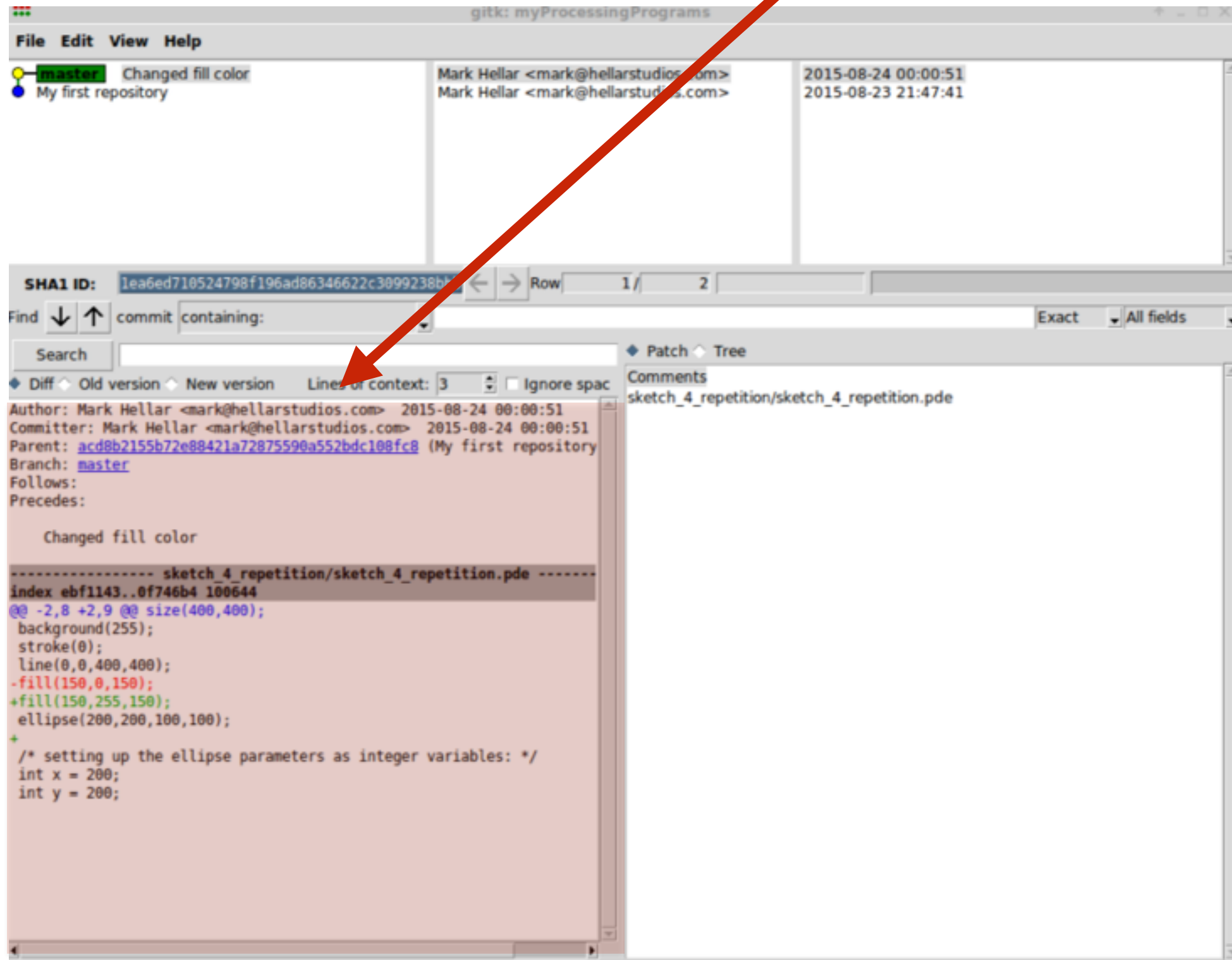
```
gumby@furby:~/Desktop/myProcessingPrograms$ git commit -m "Changed fill color"
[master 1ea6ed7] Changed fill color
1 file changed, 2 insertions(+), 1 deletion(-)
gumby@furby:~/Desktop/myProcessingPrograms$
```

The **-m** option is for message it allows us to provide a commit message in quotes describing changes

Commit the changes



If I run **gitk** i can see my change



Git workflow

Edit, add or delete files

Modify your files

Stage the changes

git add

Review your changes

git status

Commit the changes

git commit

Easier Git workflow

Edit, add or delete files

Modify your files

Commit and add the changes
(with one command)

git commit -a

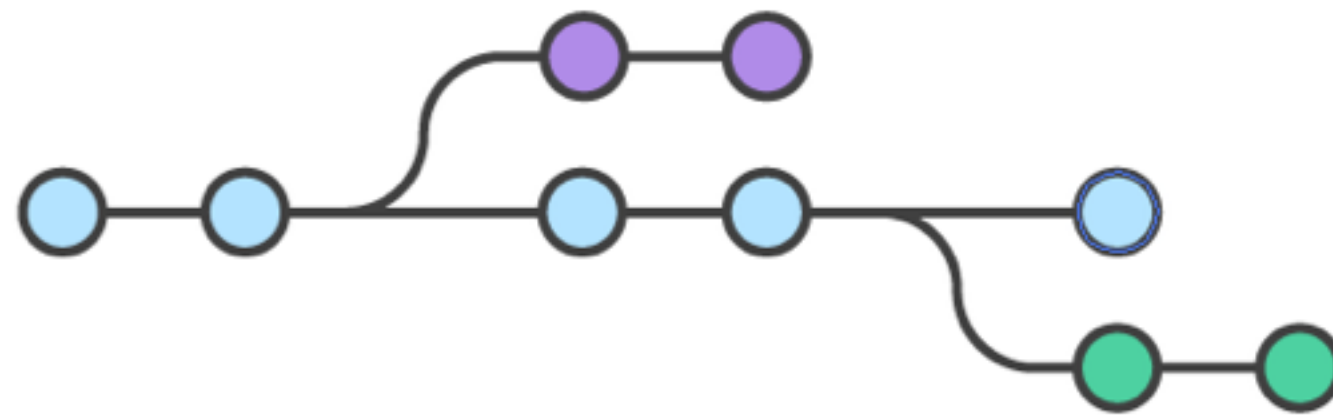
Branching

Right now our repository is storing everything in a master '**branch**'. Branching allows you to make a derivative of that branch and work on it independently.

Some possible reasons to create a **branch**:

- You want to make a research copy
- An artist is going to make updates and you want to keep a copy of the original code.
- The language the code was written in is no longer supported and you want to attempt a migration

Branching



git branch **<branch>**

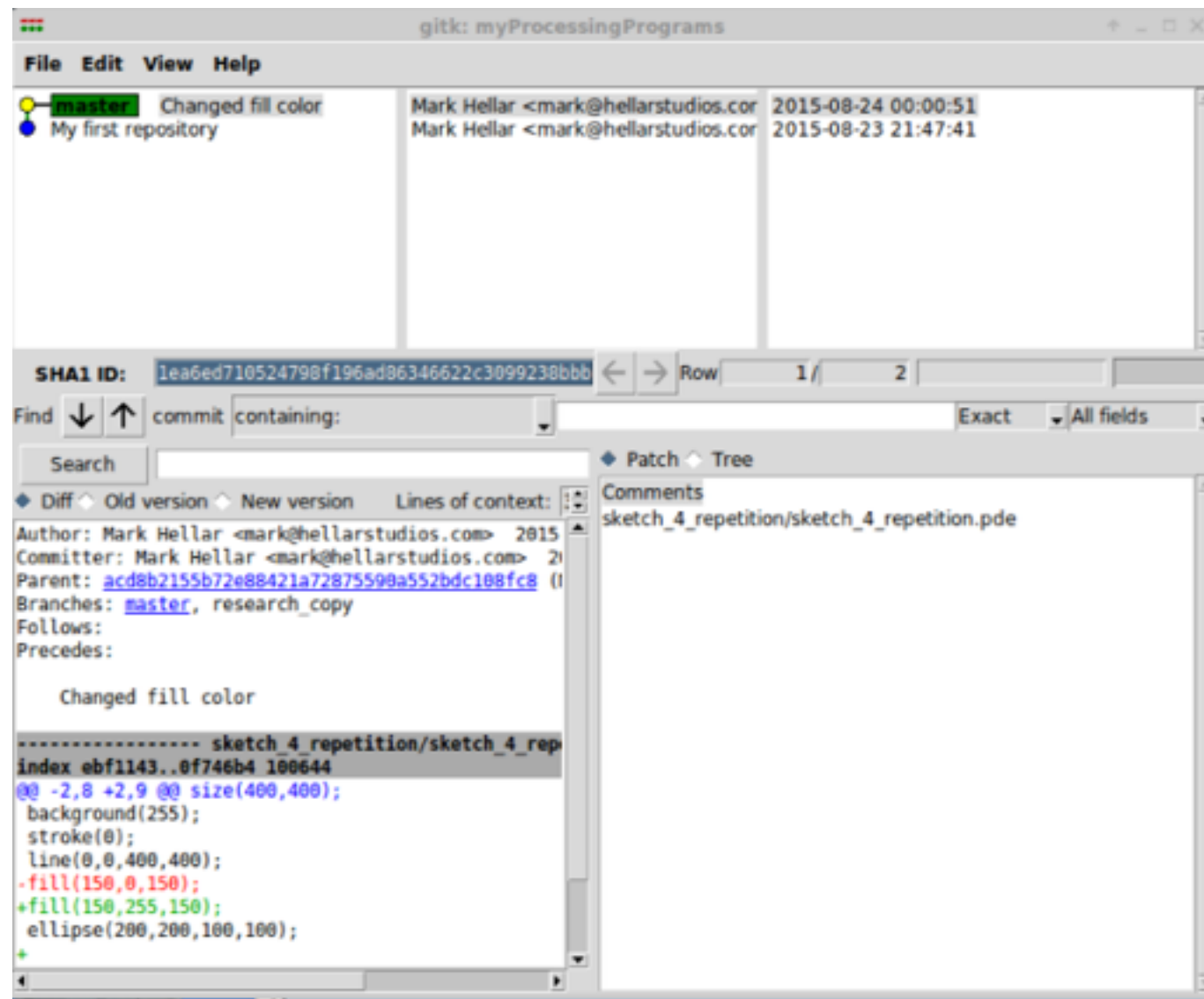
Git branch creates a new <branch>

git checkout **<branch>**

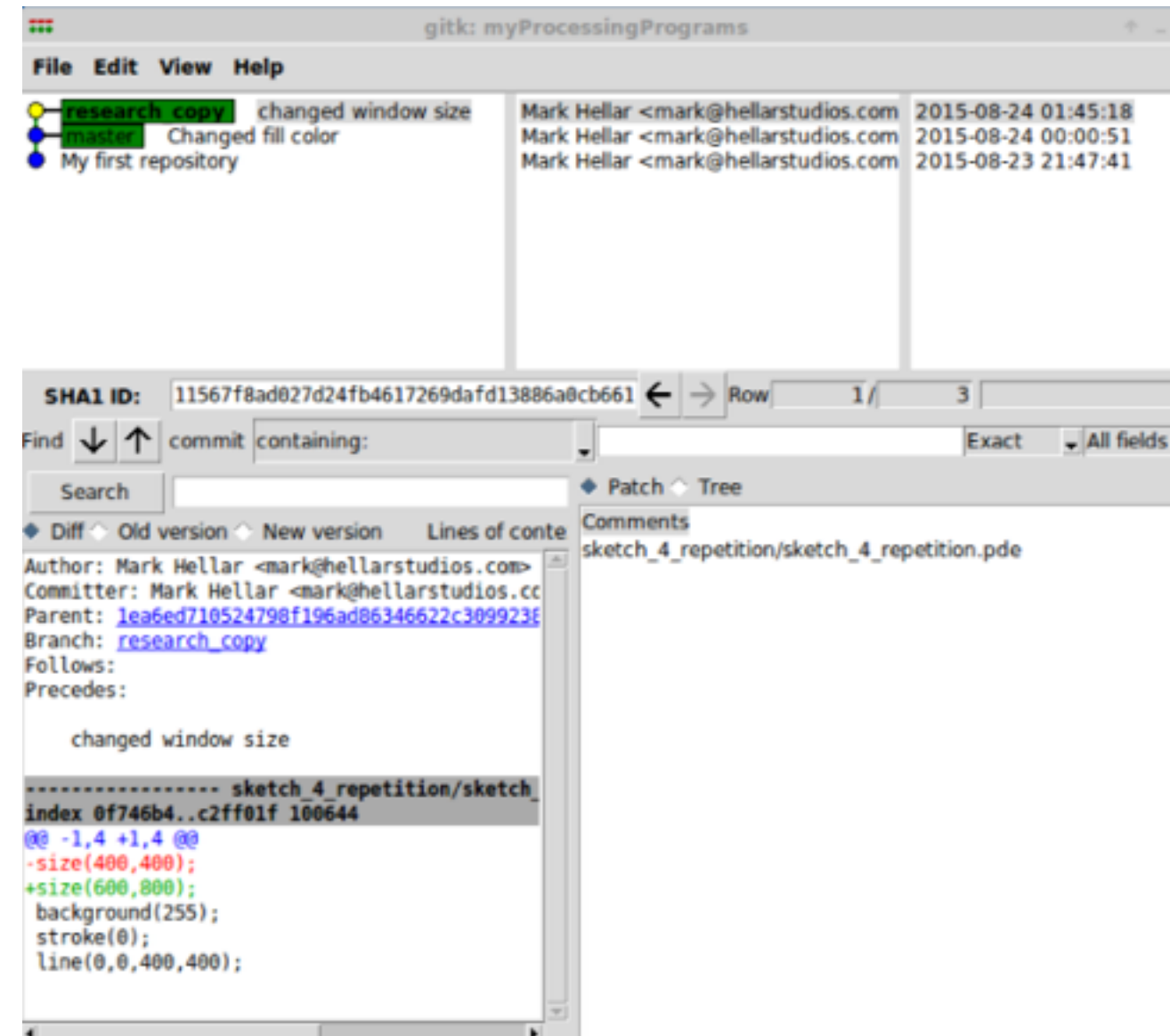
Git checkout switches the working folder to
a new <branch>

Branching

git checkout master



git checkout research_copy



Today we covered:

- Creating a repository
- Workflow for tracking changes
- Introduction to branching

There's more to Git, but these cover the common operations

Git and GitHub

Git is a version control system; think of it as a series of snapshots (commits) of your code. You see a path of these snapshots, in which order they were created. You can make branches to experiment and come back to snapshots you took.

GitHub, is a web-page on which you can publish your Git repositories and collaborate with other people.

<https://help.github.com/articles/set-up-git/>

Git and GitHub



Git and GitHub

cooperhewitt / Planetary

the all new clean and UI callback enabled app formerly known as Kepler (AKA pollen-planets)
<http://planetary.bloom.io/>

882 commits 9 branches 10 releases 5 contributors

Branch: master Planetary / +

File	Description	Time
bin	OCRA; landscape mode and legal size to account for linewraps and page...	2 years ago
blocks	bump BloomScene	4 years ago
resources	removing unused images and rearranging image folders	4 years ago
src	spelling is hard!	2 years ago
xcode	change name and bump version to 2.0	4 years ago
.gitignore	add first pass at src2pdf script	2 years ago
.gitmodules	added BloomScene back as a submodule block	4 years ago
LICENSE	replace license boilerplate with SI	2 years ago
Makefile	add first pass at src2pdf script	2 years ago
README.md	ootd reference	2 years ago

Code

Issues 10

Pull requests 0

Wiki

Pulse

Graphs

HTTPS clone URL

<https://github.com/cooperhewitt/Planetary>

You can clone with [HTTPS](#), [SSH](#), or [Subversion](#).

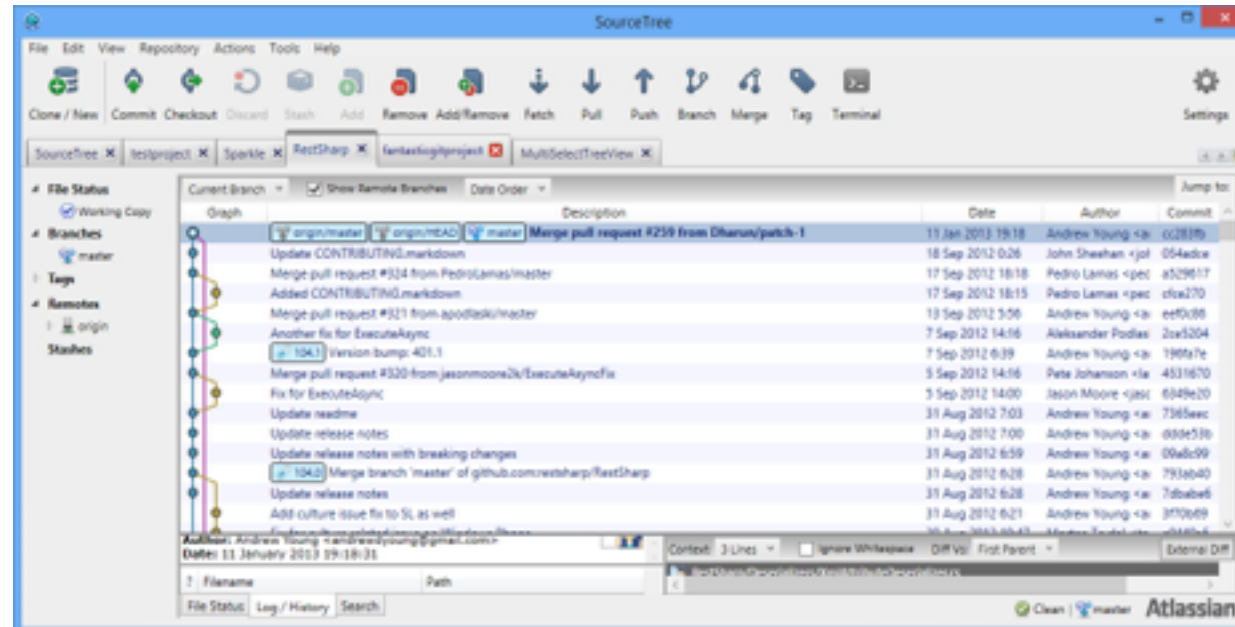
Clone in Desktop

Download ZIP

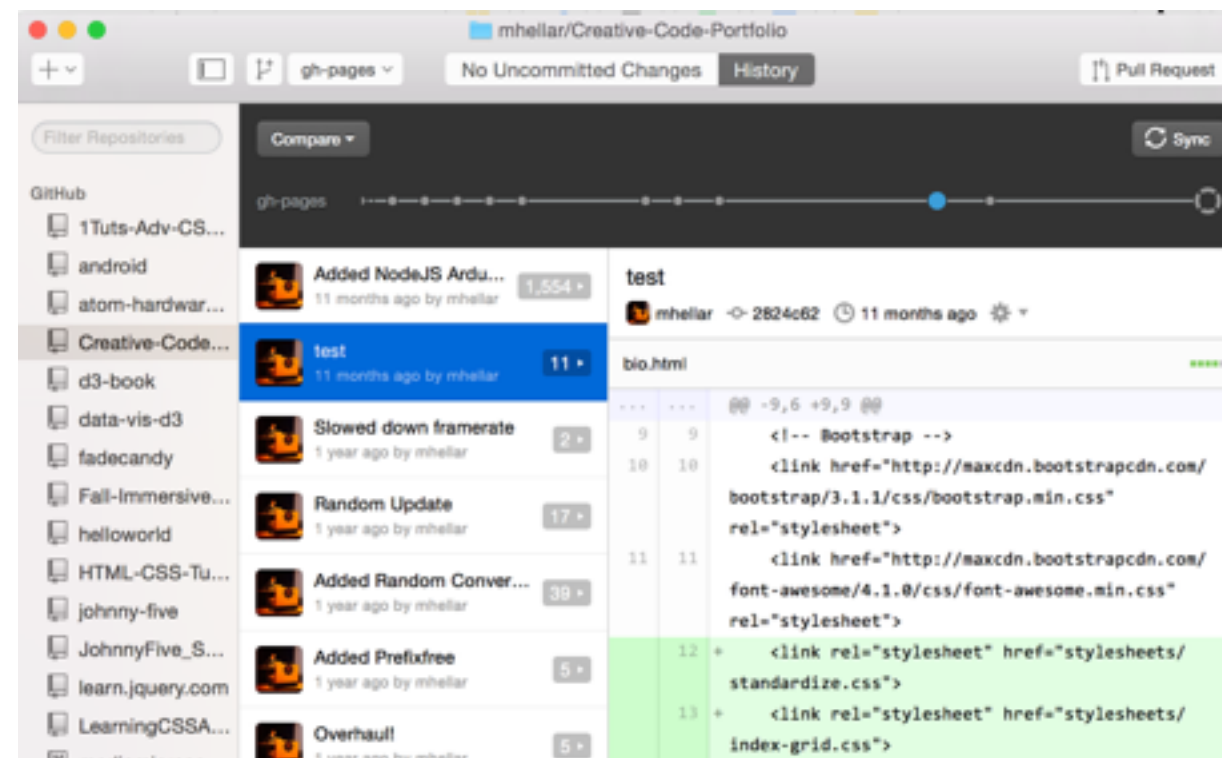
git clone <https://github.com/cooperhewitt/Planetary.git>

GUI Versions!

<https://www.sourcetreeapp.com/>



<https://desktop.github.com/>



Git Book

<https://git-scm.com/book/en/v2>

(copy is on the thumb drive)

Getting Started

<https://git-scm.com/doc>

Online Tutorial

<https://try.github.io/levels/1/challenges/1>

Thank You!

mark@hellarstudios.com