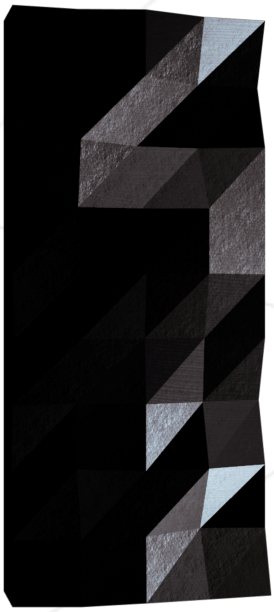
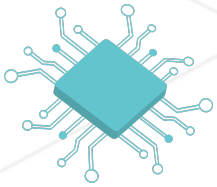


Portfolio Project

SideGoals

Kai kai@42.us.org

Summary:



HACK
HIGH
SCHOOL

Contents

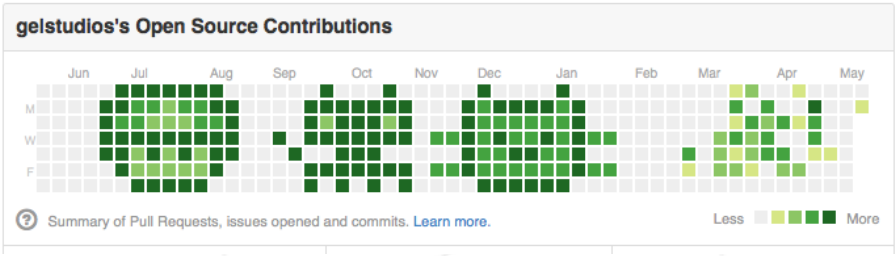
Chapter I

Forward

`Here is the README from the public Github repo, "Gitfiti":`

gitfiti *noun* : Carefully crafted graffiti in a github commit history calendar.

An example of gitfiti in the wild:



`gitfiti.py` is a tool I wrote to decorate your github account's commit history calendar by (blatantly) abusing git's ability to accept commits in the past.

How? `gitfiti.py` generates a bash script: `gitfiti.sh` that makes commits with the GIT_AUTHOR_DATE and GIT_COMMITTER_DATE environment variables set for each targeted pixel.

Since this is likely to clobber repo's history, I highly recommend that you create a new github repo when using gitfiti. Also, the generated bash script assumes you are using public-key authentication with git.

Pixel Art:

Included "art" from left to right: kitty, oneup, oneup2, hackerschool, octocat, octocat2

Usage:

1. Create a new github repo to store your handiwork.
2. Run gitfiti.py and follow the prompts for username, art selection, offset, and repo name.
3. Run the generated gitfiti.sh from your home directory (or any non-git tracked dir) and watch it go to work.
4. Wait... Seriously, you'll probably need to wait a day or two for the gitfiti to show in your commit graph.

User Templates The file format for personal templates is the following:

1. Each template starts off with a ":" and then a name (eg. ":foo")
2. Each line after that is part of a json-recognizable array.
3. The array contain values 0-4, 0 being blank and 4 being dark green.
4. To add multiple templates, just add another name tag as described in 1.

For example:

```
:center-blank
[[1,1,1,1,1,1,1],
[1,1,1,1,1,1,1],
[1,1,1,1,1,1,1],
[1,1,1,0,1,1,1],
[1,1,1,1,1,1,1],
[1,1,1,1,1,1,1],
[1,1,1,1,1,1,1]]
```

This would output a 7 x 7 light green square with a single blank center square.

Once you have a file with templates, enter its name when prompted and the templates will be added to the list of options.

Removal: Fortunately if you regret your gitfiti in the morning, removing it is fairly easy: delete the repo you created for your gitfiti (and wait).

License: gitfiti is released under The MIT license (MIT)

Todo:

- Remove 'requests' dependency thanks empathetic-alligator
- Web interface See several web-based things below
- Load "art" from a file thanks empathetic-alligator
- Load commit content from a file
- Text/alphabet option
- ...
- Profit?

Notable derivatives or mentions:

- Pikesley's Pokrovsky, which offers Github History Vandalism as a Service!
- github-board commits gitfiti from easy templates
- Gitfiti Painter visual drawing tool for artists to easily create templates
- git-draw a Chrome extension which will allow you to freely draw on your commit map(!)
- github-jack a pure bash version with space invaders and shining creepypasta
- github-graffiti a GUI editor with a bash script to allow custom designs on your commit map
- Seen something else? Submit a pull request or open an issue!

Chapter II

Introduction

As a HackHighSchool mentor we want you to establish expertise in Python, Javascript, Ruby, Java, or other funny languages (Lua? Go? Swift? Kotlin?) that teens coming to 42 may be interested to write some code in!

Your mission is to write a Tier 1 side project of your own invention and make a nicely documented Github repository for the world (employers and networkers!) to see. :)

This project must use a language that is currently offered in the HackHighSchool curriculum. If the language is not covered, you should instead look into your options for the "Project Authorship" project.

Chapter III

General Instructions

Your side project should take about one week of intense focused effort, or 10 days of 5-hour creative work blocks. (Every Sunday after HackHighSchool!)

You should be cautious to choose a topic that is both challenging and reasonable. Before you start, think about what prerequisite topics you know already; what technologies you are craving to learn; what end product would be fun to build; and what the minimum viable version of that product would be.

Go ahead and scout out some lessons or tutorials that will help get you started in the right direction. Consider applying the 80/20 rule and plan on following 10 hours of lessons, before branching out to develop your project creatively. The remaining 40 hours should involve thinking of something you want to do, researching how to do it, making attmpets, debugging, and dopamine-spiking "AHA!" moments.

Here's one list of ideas...!

Chapter IV

Mandatory Part

- The project code should run without errors and perform an interesting or useful task. It should handle bad input gracefully.
- The user interaction part of your program should be well explained or easy to understand.
- Your README should explain the purpose and/or use of the program, and include screenshots of it running.
- Your README should contain full instructions for installing and running the program on a blank MacOS system. Use a guest account on the HackHighSchool check-in computers for testing.
- Your Github repo should specify the license that clarifies what permissions you give for others to modify and reuse your work.
- There should be no unnecessary files in your Github repo that distract from the real content, or any that reveal secrets such as API keys.
- Each file containing code should start with a header block that introduces the origin and purpose of the code.
- Your code should be nicely formatted, and code/function/class/filenames should contribute to its overall readability.
- You should make multiple commits to your Github repository over the course of the project, and your commit messages should be dignified and descriptive.
- You should work socially to some extent by discussing your project with other cadets, whether it be to ask for advice or to show off your work.

Chapter V

Bonus Part

- Create a new git branch every time you start working on a new feature, and then merge it into the master branch when done.
- Make the program look beautiful with great front-end design or terminal colors and formatting.
- For maximum credit, add 5 additional "features" or improvements on top of the bare mininum functioning program.
- Use the gaming computers at the back of Zone 3 to ensure that your setup instructions also work on a guest account of a Windows operating system.

Chapter VI

Turn-in and Peer Evaluation

Upload to Vogsphere a link to your public facing Github repository. This project requires four corrections. You should invite your correctors to log into a blank guest account on the H2S Check-in Computers to test the setup instructions as they would work for computers outside our network. If you think you qualify for the Windows bonus, also test the setup instructions on one of the gaming section Windows computers.

