Lecture 01

Introduction

5 Prairial, Year CCXXX

Song of the day: ***hello world*** *by Louie Zong*.

Hey, my name is Sebastián, and my last names are Romero Cruz. I teach about programming for a living and, incredibly, sometimes even enjoy programming itself.

I'm actually very excited to teach this class; I have done both game development on Unity and interactive computer graphics with OpenGL before. Having the previlege of putting these two together by teaching this class and learning with everyone is something I am super exited about.

Anyway, other things I enjoy are:

- **Slice of life anime**: my favourite is ***Hibike! Euphonium***
- **French history**: anything starting with the French Revolution of 1789 to the end of the de Gaulle presidency in 1969.
- **Literature**: my current favourite is ***Paris is a Party, Paris is a Ghost***.
- **Playing music**: I play a **Rickenbacker 4001C64**.

That's all I want to flex at the present moment. Let's talk about the course (see syllabus).

Sections

Part 1: *Working with GitHub*

The way we're going to be managing our work in this class via **GitHub**. Follow **these instructions** to install Git onto your computer.

If you have never used this site before, it's essentially a way for us to do our work on our own computer, and then asking GitHub to keep track of those changes.

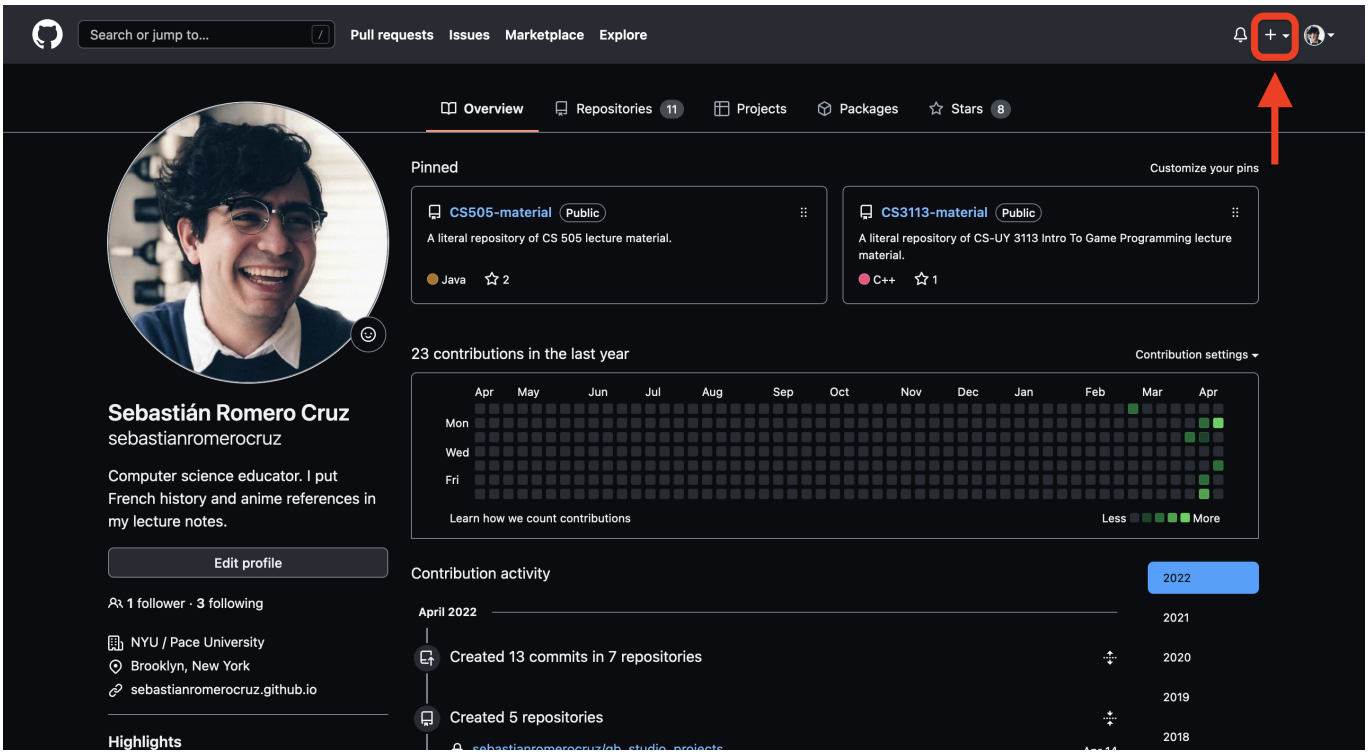
The reason why this is helpful is because, as you're making progress in, say, one of your projects, you can ask GitHub to upload those changes onto your account. This way, if you have a question for me, I can take a look at your latest "upload," download it myself, and suggest some changes.

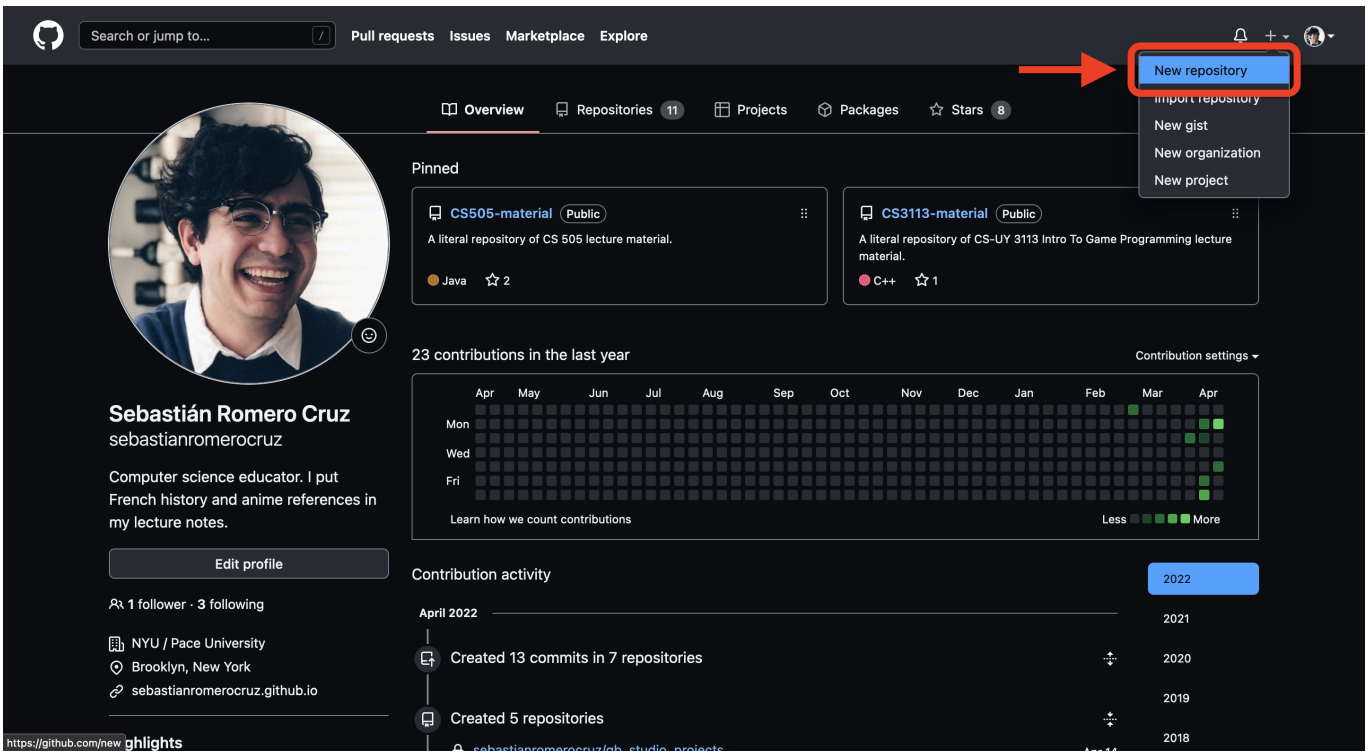
Moreover, **all of your finished projects must be successfully uploaded to GitHub to be considered on-time**. So let's spend some time in learning how to this. I'm going to show you an easy method to creating a GitHub repository that you do in order to get youself set up for the rest of the summer:

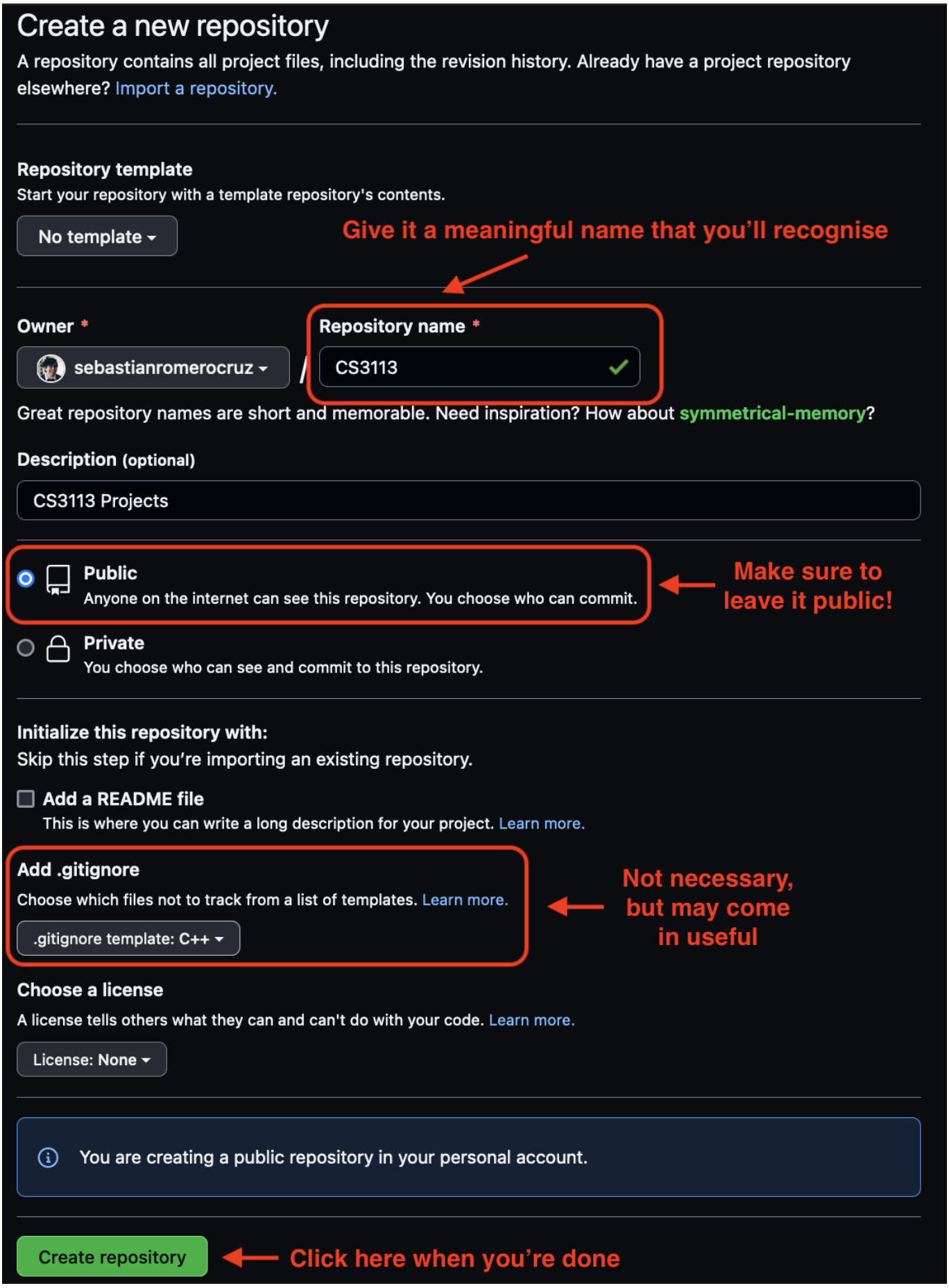
Note: If you already know how to handle yourself in GitHub, you can skip this section entirely.

Step 1: *Create a new repository through GitHub*

After you've installed gotten yourself a GitHub account and installed Git, your first step is to create a new public repository in your account:

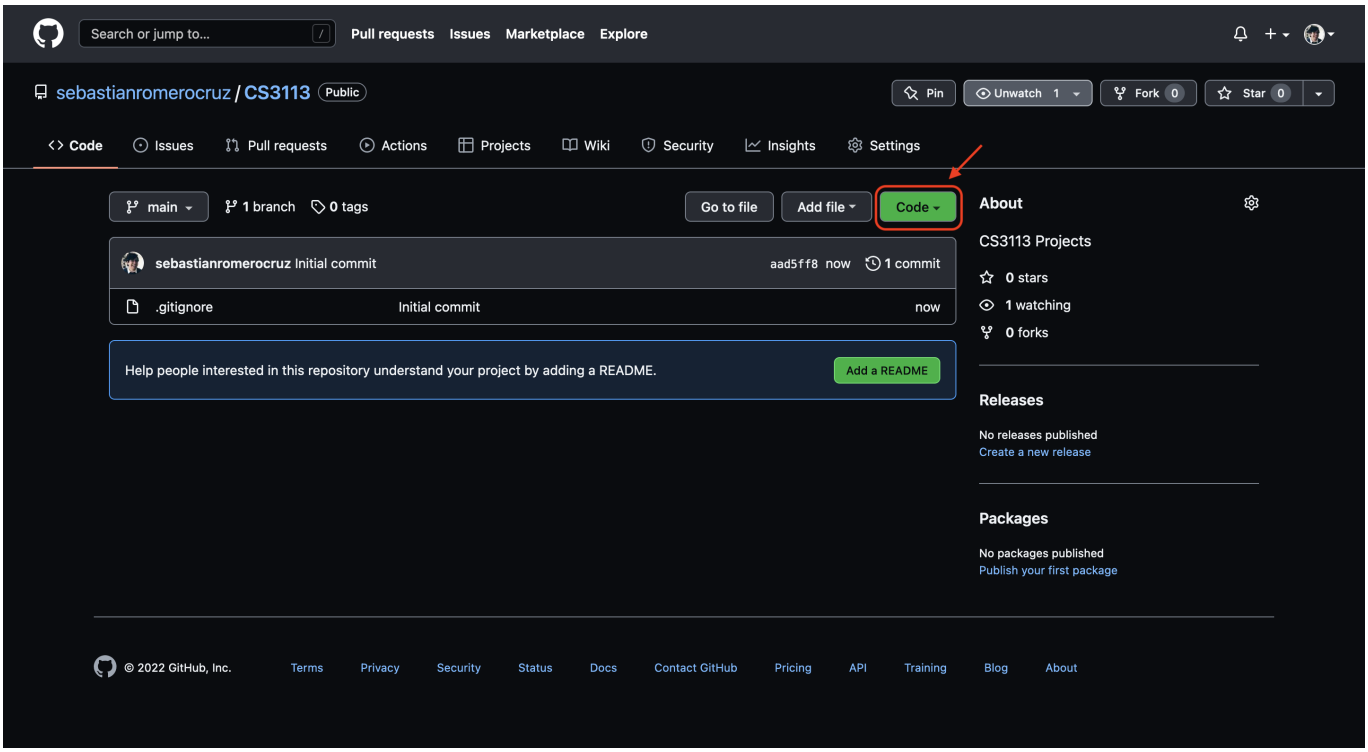

Search or jump to...
Pull requests
Issues
Marketplace
Explore
Overview
Repositories 11
Projects
Packages
Stars 8
Pinned
Customize your pins
CS505-material Public
A literal repository of CS 505 lecture material.
Java 2
CS3113-material Public
A literal repository of CS-UY 3113 Intro To Game Programming lecture material.
C++ 1
23 contributions in the last year
Contribution settings
Apr May Jun Jul Aug Sep Oct Nov Dec Jan Feb Mar Apr
Mon
Wed
Fri
Learn how we count contributions
Less More
Sebastián Romero Cruz
sebastianromerocruz
Computer science educator. I put French history and anime references in my lecture notes.
Edit profile
1 follower · 3 following
NYU / Pace University
Brooklyn, New York
sebastianromerocruz.github.io
Highlights
Contribution activity
2022
April 2022
2021
Created 13 commits in 7 repositories
2020
2019
Created 5 repositories
2018

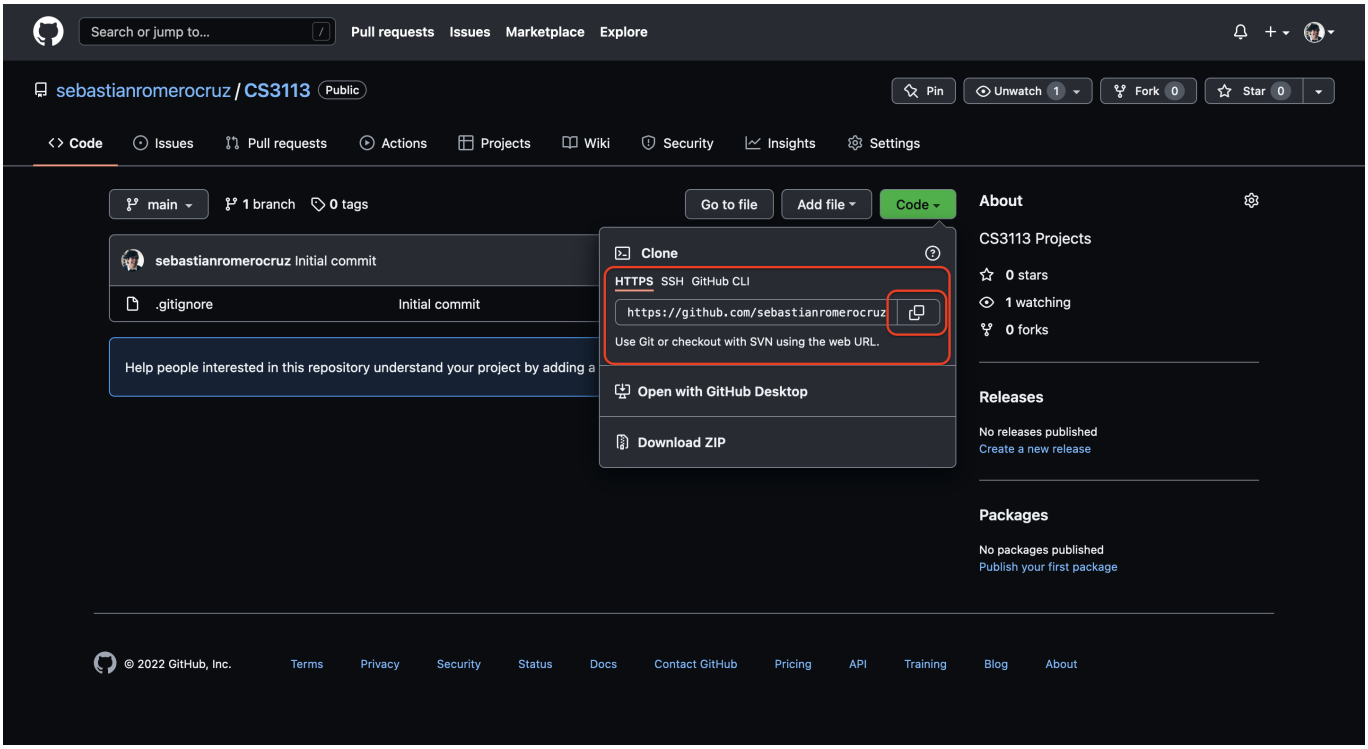

Search or jump to...
Pull requests
Issues
Marketplace
Explore
New repository
New gist
New organization
New project
Overview
Repositories 11
Projects
Packages
Stars 8
Pinned
CS505-material Public
A literal repository of CS 505 lecture material.
Java 2
CS3113-material Public
A literal repository of CS-UY 3113 Intro To Game Programming lecture material.
C++ 1
23 contributions in the last year
Contribution settings
Apr May Jun Jul Aug Sep Oct Nov Dec Jan Feb Mar Apr
Mon
Wed
Fri
Learn how we count contributions
Less More
Sebastián Romero Cruz
sebastianromerocruz
Computer science educator. I put French history and anime references in my lecture notes.
Edit profile
1 follower · 3 following
NYU / Pace University
Brooklyn, New York
sebastianromerocruz.github.io
https://github.com/new
Contribution activity
2022
April 2022
2021
Created 13 commits in 7 repositories
2020
2019
Created 5 repositories
2018

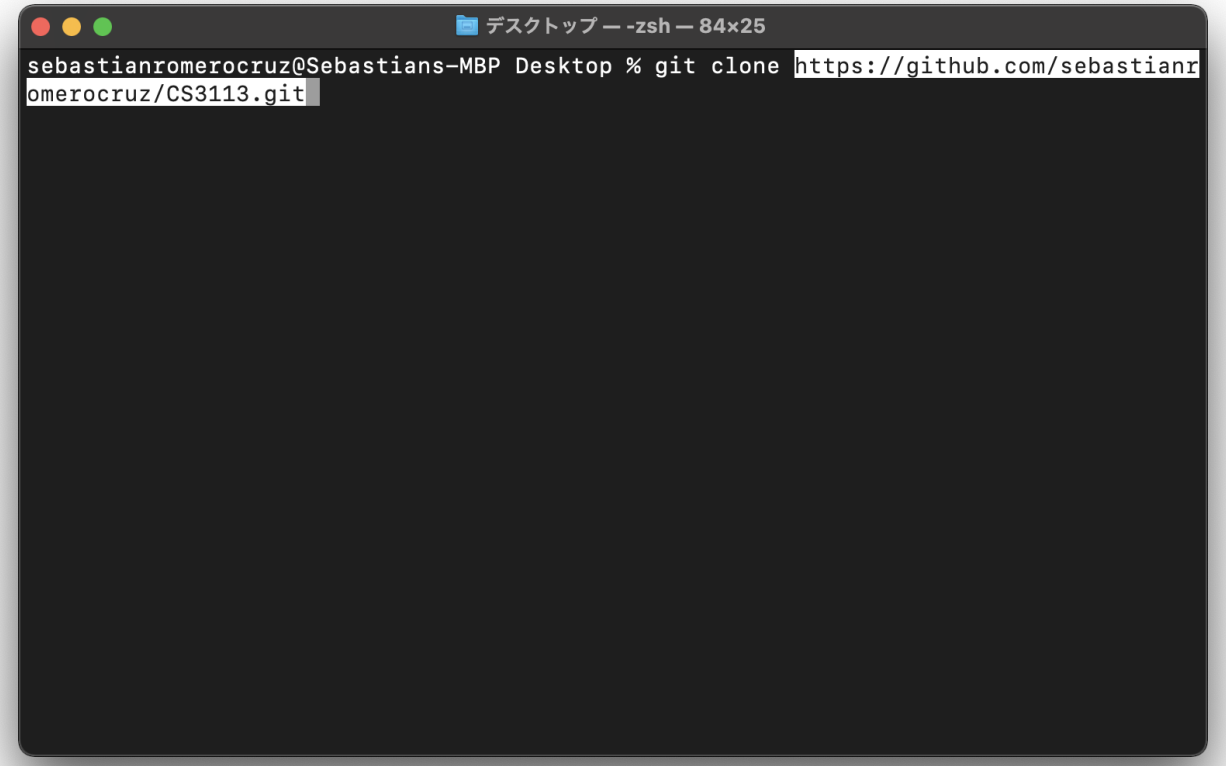


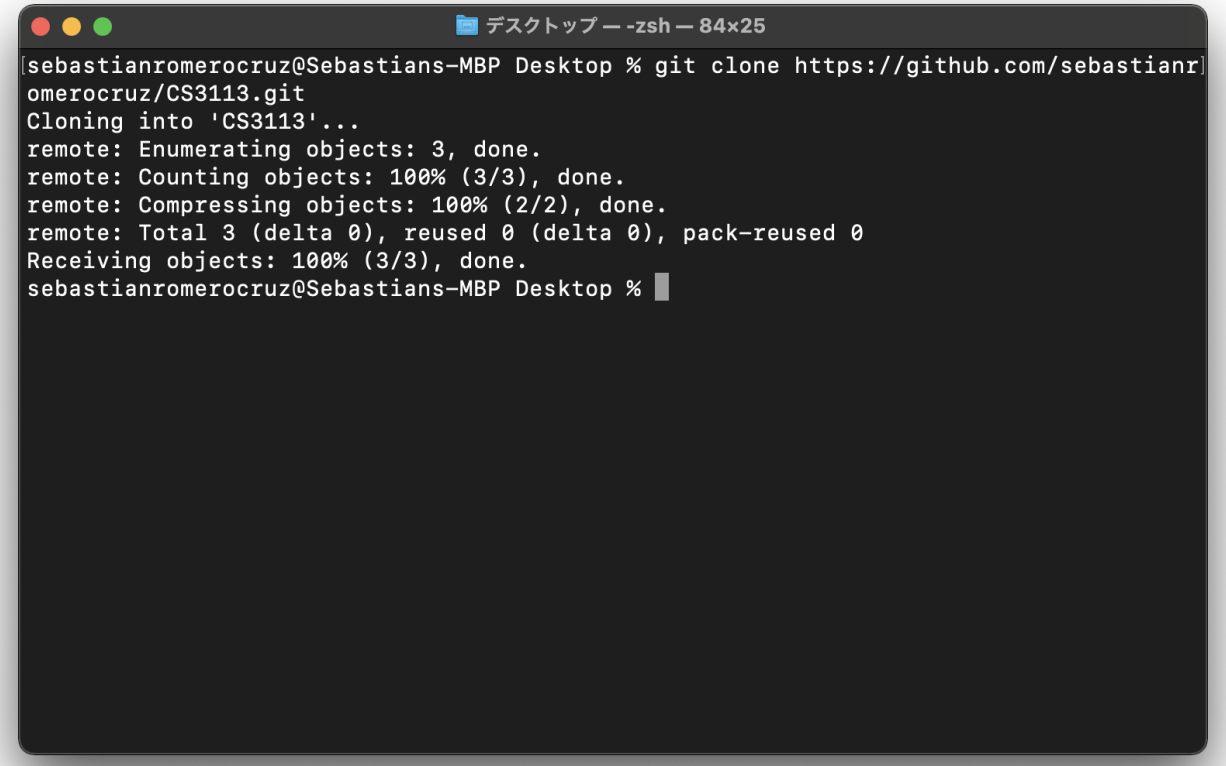
Figures 1-3: Creating a new public GitHub repository (or *repo*).

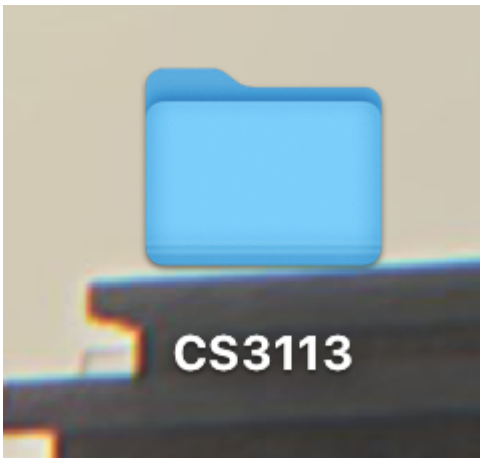
Step 2: *Cloning your repo*


Search or jump to...
Pull requests
Issues
Marketplace
Explore
sebastianromerocruz / CS3113
Public
Pin
Unwatch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
main
1 branch
0 tags
Go to file
Add file
Code
sebastianromerocruz Initial commit
aad5ff8 now
1 commit
.gitignore
Initial commit
now
Help people interested in this repository understand your project by adding a README.
Add a README
About
CS3113 Projects
0 stars
1 watching
0 forks
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
© 2022 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact GitHub
Pricing
API
Training
Blog
About


Search or jump to...
Pull requests
Issues
Marketplace
Explore
sebastianromerocruz / CS3113
Public
Pin
Unwatch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
main
1 branch
0 tags
Go to file
Add file
Code
sebastianromerocruz Initial commit
.gitignore
Initial commit
Help people interested in this repository understand your project by adding a
Clone
HTTPS SSH GitHub CLI
https://github.com/sebastianromerocruz
Use Git or checkout with SVN using the web URL.
Open with GitHub Desktop
Download ZIP
About
CS3113 Projects
0 stars
1 watching
0 forks
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
© 2022 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact GitHub
Pricing
API
Training
Blog
About


デスクトップ — -zsh — 84×25
sebastianromerocruz@Sebastians-MBP Desktop % git clone https://github.com/sebastianr
omerocruz/CS3113.git

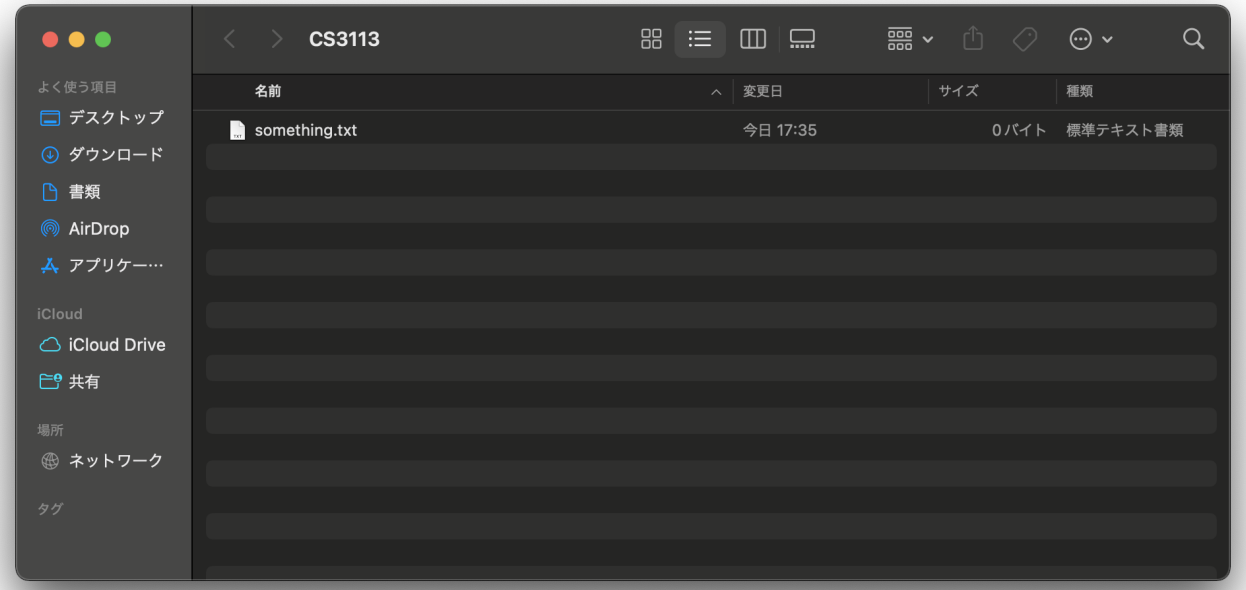

デスクトップ — -zsh — 84×25
sebastianromerocruz@Sebastians-MBP Desktop % git clone https://github.com/sebastianr
omerocruz/CS3113.git
Cloning into 'CS3113'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
sebastianromerocruz@Sebastians-MBP Desktop %

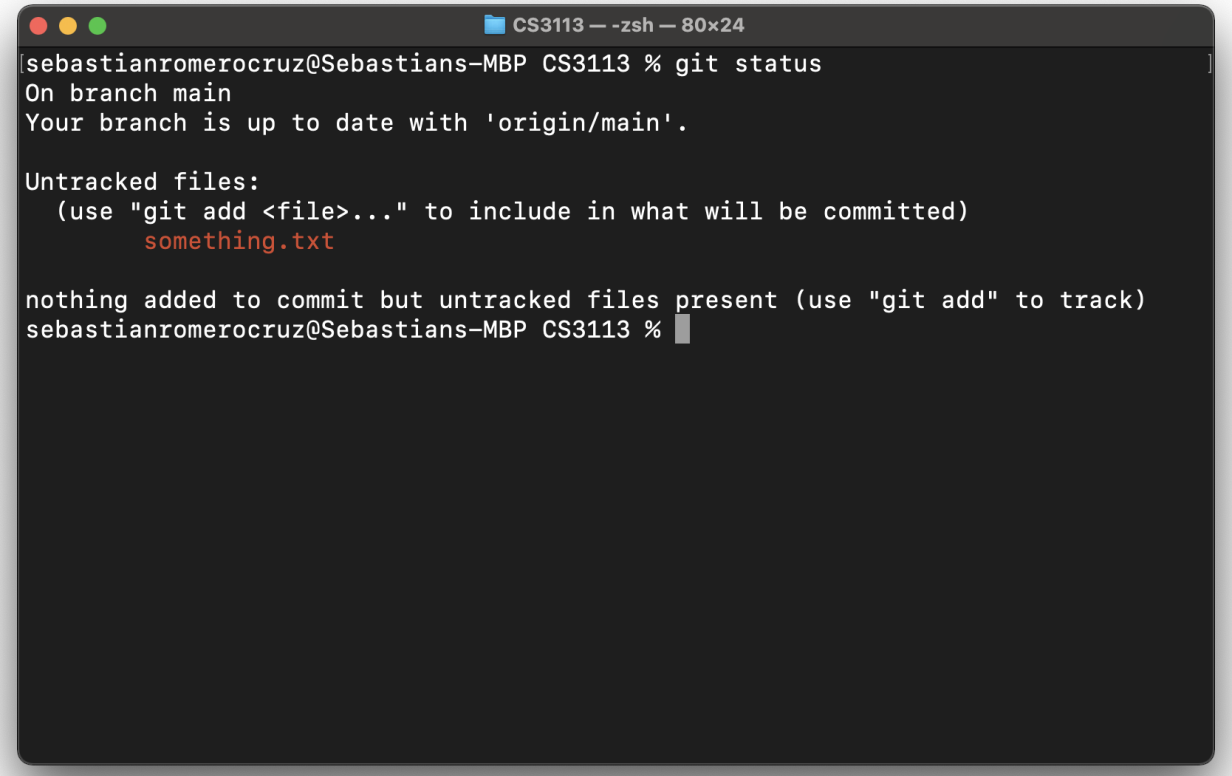


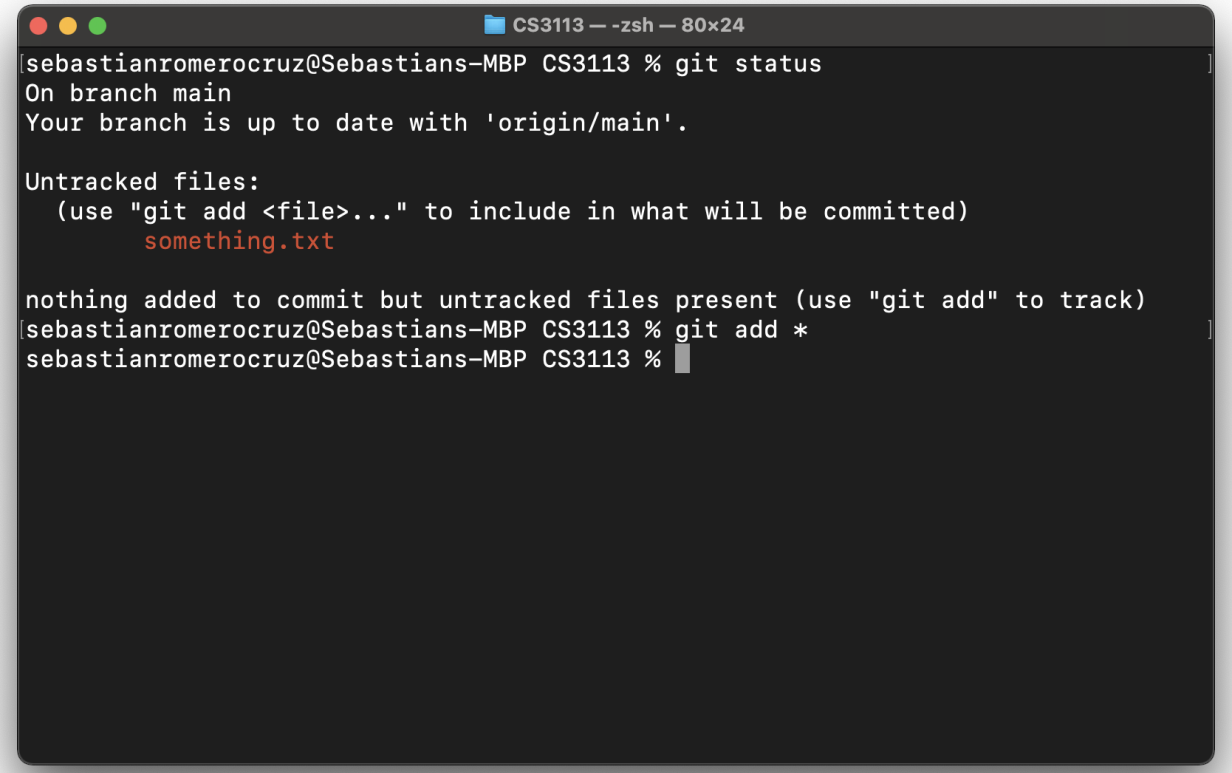
Figures 4-7: Getting the repo locally onto your computer.

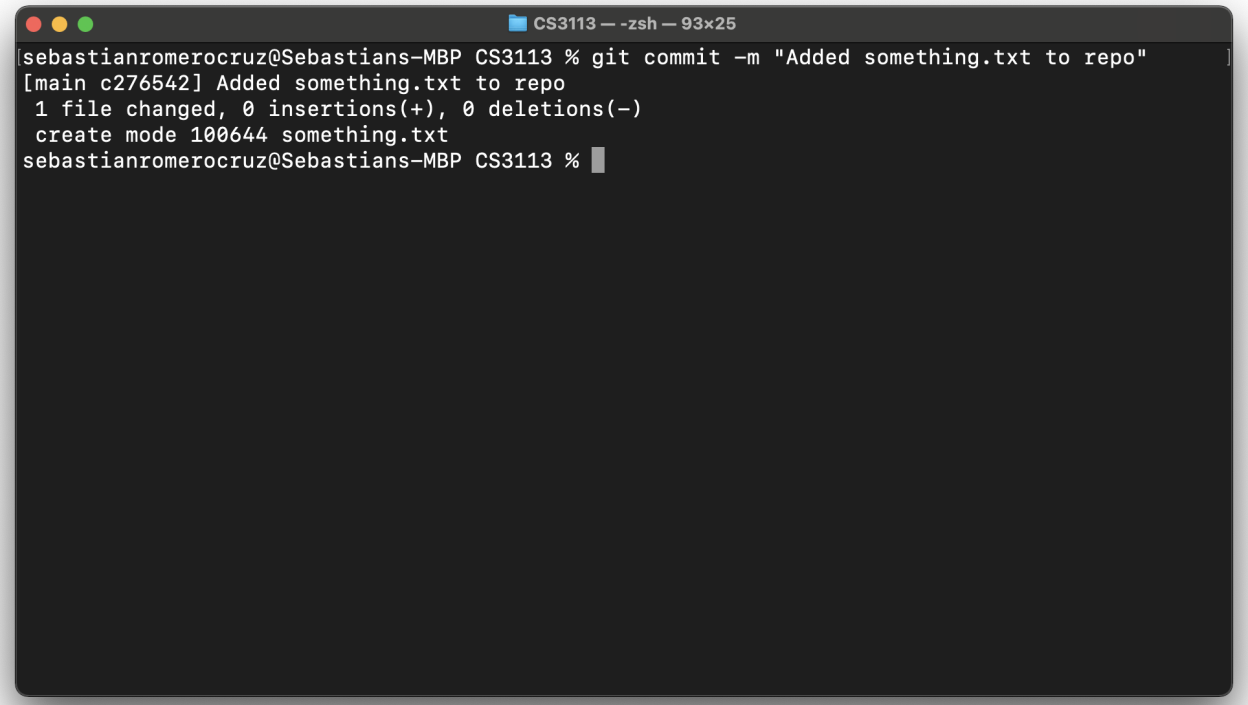
Step 3: *Making and saving changes into your repo*

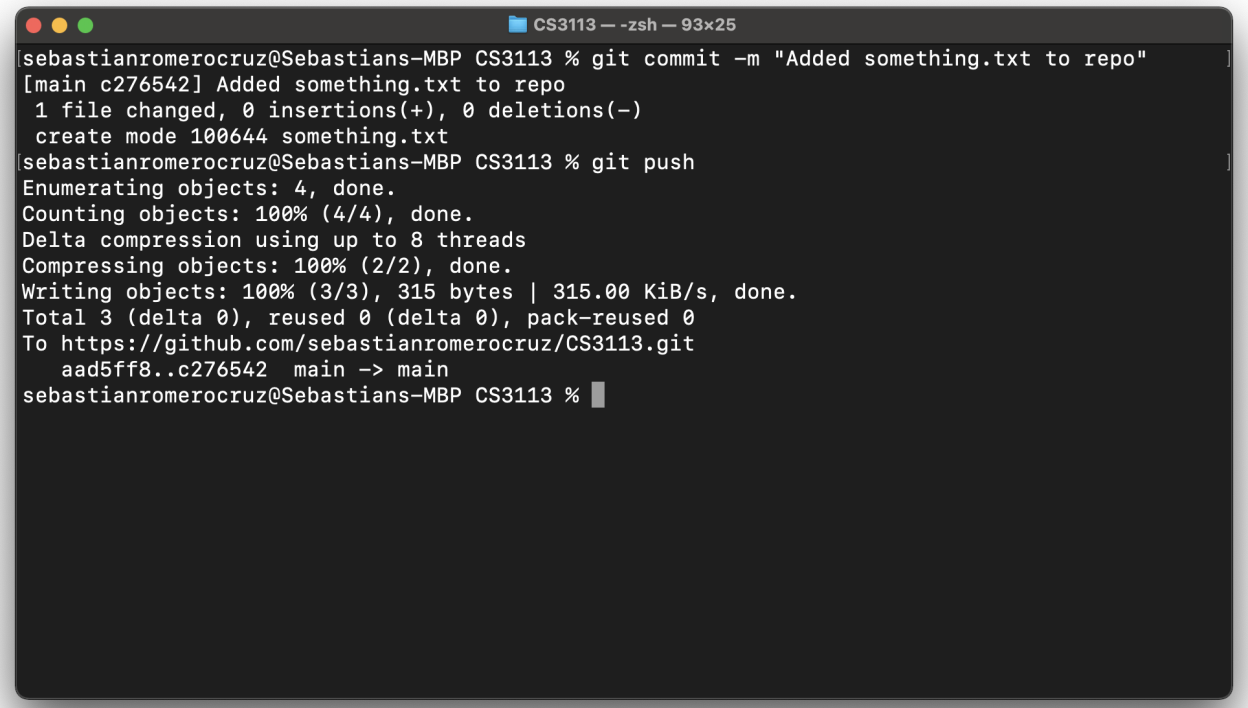
Let's make a simple change to your repo—create a `txt` file called `something.txt` inside your repo—and save (i.e. push) them up to GitHub.

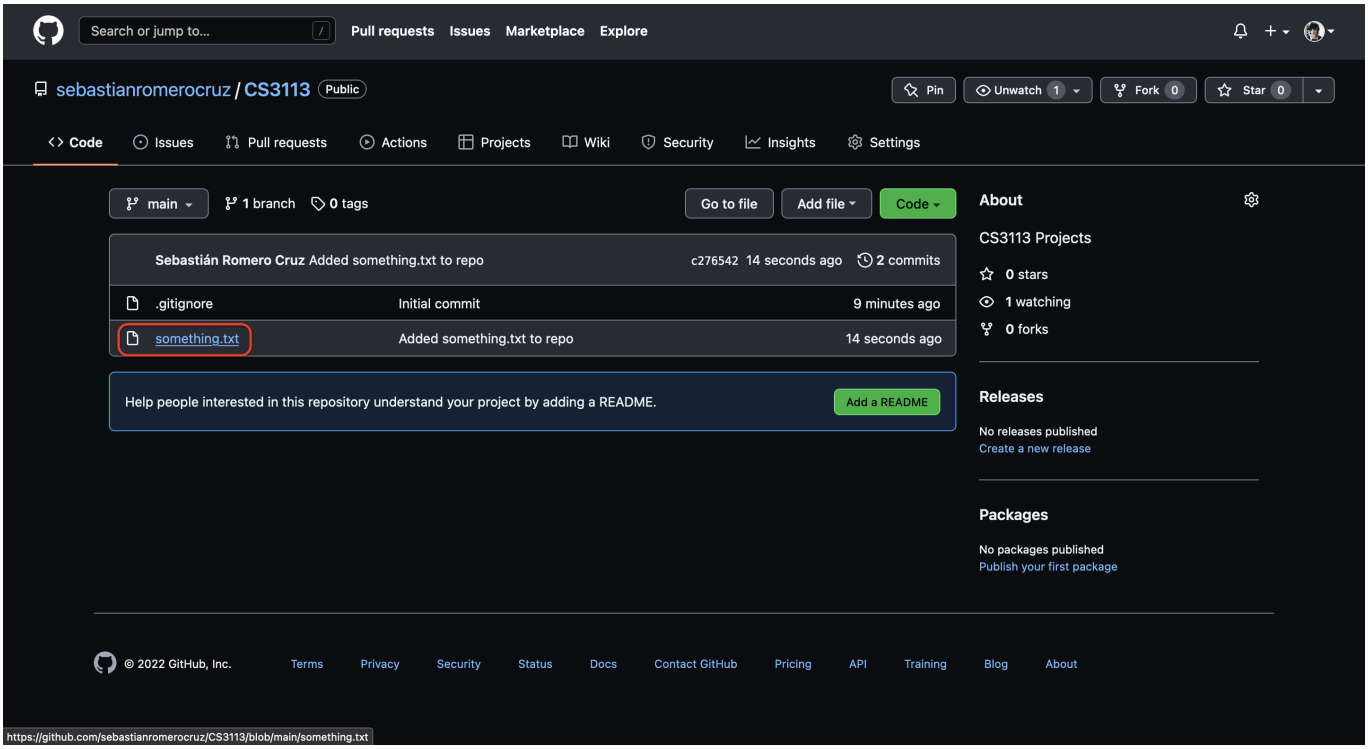

CS3113
よく使う項目
デスクトップ
ダウンロード
書類
AirDrop
アプリケー…
iCloud
iCloud Drive
共有
場所
ネットワーク
タグ
名前
変更日
サイズ
種類
something.txt
今日 17:35
0バイト
標準テキスト書類


CS3113 — -zsh — 80×24
sebastianromerocruz@Sebastians-MBP CS3113 % git status
On branch main
Your branch is up to date with 'origin/main'.
Untracked files:
(use "git add <file>..." to include in what will be committed)
something.txt
nothing added to commit but untracked files present (use "git add" to track)
sebastianromerocruz@Sebastians-MBP CS3113 %


CS3113 — -zsh — 80×24
sebastianromerocruz@Sebastians-MBP CS3113 % git status
On branch main
Your branch is up to date with 'origin/main'.
Untracked files:
(use "git add <file>..." to include in what will be committed)
something.txt
nothing added to commit but untracked files present (use "git add" to track)
sebastianromerocruz@Sebastians-MBP CS3113 % git add *
sebastianromerocruz@Sebastians-MBP CS3113 %


CS3113 — -zsh — 93×25
sebastianromerocruz@Sebastians-MBP CS3113 % git commit -m "Added something.txt to repo"
[main c276542] Added something.txt to repo
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 something.txt
sebastianromerocruz@Sebastians-MBP CS3113 %





Figures 9-14: Making, adding, committing, and pushing changes onto GitHub.

Part 2: *Opening a sample project*

Unfortunately, setting up OpenGL takes some setting up. A former instructor of the course and friend of mine, Carmine Guida, lent me these guides for each operating system, so please follow **these instructions** if you are a Windows user or **these instructions** if you are a Mac user. Once you've finished, come back to this `README`.

Alright, supposing that everything has been set-up correctly, the following code should compile without any errors:

```
#include <iostream>

#define GL_SILENCE_DEPRECATION

#ifdef _WINDOWS
#include <GL/glew.h>
#endif

#define GL_GLEXT_PROTOTYPES 1
#include <SDL.h>
#include <SDL_opengl.h>

const int WINDOW_WIDTH = 640;
const int WINDOW_HEIGHT = 480;
const float BG_RED = 0.1922f, BG_BLUE = 0.549f, BG_GREEN = 0.9059f;
const float BG_OPACITY = 1.0f;

SDL_Window* display_window;
bool game_is_running = true;

int main(int argc, char* argv[]) {
    SDL_Init(SDL_INIT_VIDEO);  // Initialising

    display_window = SDL_CreateWindow("Hello, World!",
SDL_WINDOWPOS_CENTERED, SDL_WINDOWPOS_CENTERED, WINDOW_WIDTH,
WINDOW_HEIGHT, SDL_WINDOW_OPENGL);

    SDL_GLContext context = SDL_GL_CreateContext(display_window);

    SDL_GL_MakeCurrent(display_window, context);

#ifdef _WINDOWS
    glewInit();
#endif

    glClearColor(BG_RED, BG_BLUE, BG_GREEN, BG_OPACITY);

    SDL_Event event;
    while (game_is_running) {
        while (SDL_PollEvent(&event)) {
            if (event.type == SDL_QUIT || event.type ==
SDL_WINDOWEVENT_CLOSE) {
                game_is_running = false;
            }
        }

        glClear(GL_COLOR_BUFFER_BIT);
        SDL_GL_SwapWindow(display_window);
    }
```

```
    SDL_Quit();
    return 0;
}
```

And the result should be the following window popping up on your screen:

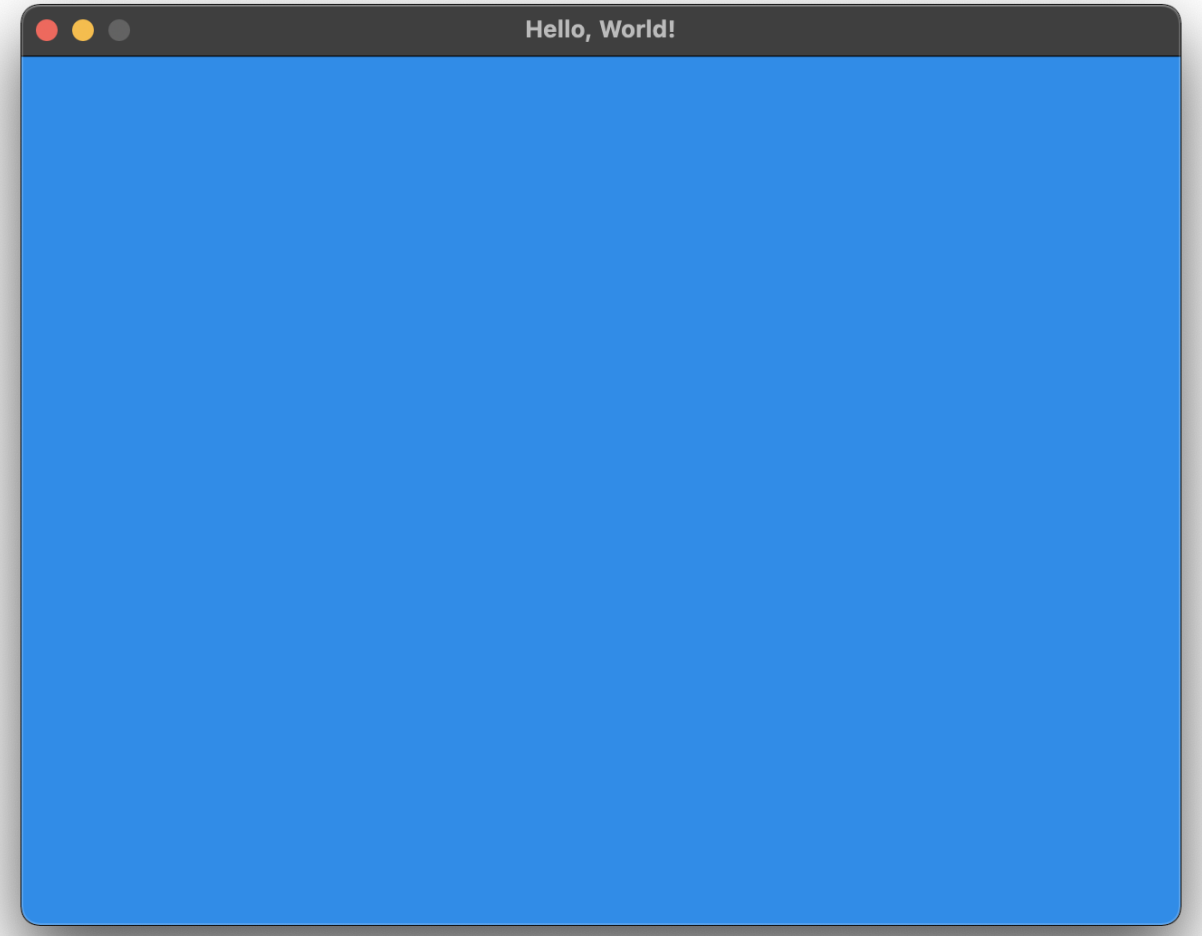


Figure 15: A lovely blue screen.

That's nice, isn't it? It may not seem like much, but getting pixels onto the screen using OpenGL is no small feat. Let's walk through the code, section by section, and understand what is going on here.

Part 3: *Understanding the basics*

Housekeeping

The first few lines are fairly standard for OpenGL programs; we are basically just importing the necessary libraries for everything to run smoothly:

```
#define GL_SILENCE_DEPRECATION

#ifdef _WINDOWS
#include <GL/glew.h>
#endif

#define GL_GLEXT_PROTOTYPES 1
#include <SDL.h>
#include <SDL_opengl.h>
```

Constants and (global) variables

Next up are some constants that, while not strictly necessary, may make your programs easier to read and follow. For example, I define the dimensions of my display window, as well as the RGB values of my background, ahead of time. That way, if I have to change them on the fly, I can do so in one place only and not have to fish around for other instances of them. In short programs like these, it may not seem like a very big deal, but once you start making actual games, you're going to want to know what each value means, and to have an easy way to change them:

```
// Some constants necessary to get started
const int WINDOW_WIDTH = 640;
const int WINDOW_HEIGHT = 480;

// The background colours may change in the course of our games, so they
can also be variables
const float BG_RED = 0.1922f, BG_BLUE = 0.549f, BG_GREEN = 0.9059f;
const float BG_OPACITY = 1.0f;
```

We also need to instantiate a couple of variables that are of supreme importance: our display window and our game-loop flag:

```
SDL_Window* display_window;
bool game_is_running = true;
```

`display_window` is simply where our game will be displayed, and `game_is_running` will keep track of whether our game is on or off. Naturally, it starts out as being on (or `true`), but if the player performs an action like closing the window, we will need a mechanism to flip this switch to `false`. We will talk about this in the sections that follow.

Starting our driver program

Just like in every other C++ program, we will need a `main`, or driver, function. Our game loop will reside in here:

```
int main(int argc, char* argv[])
{
    // The rest of the code described in this README will go in here
}
```

The first two things we have to do are initialise OpenGL and our `display_window` object:

```
SDL_Init(SDL_INIT_VIDEO);  // Initialising

display_window = SDL_CreateWindow("Hello, World!",
                                  SDL_WINDOWPOS_CENTERED,
SDL_WINDOWPOS_CENTERED,
                                  WINDOW_WIDTH, WINDOW_HEIGHT,
                                  SDL_WINDOW_OPENGL);
```

Note: For those of you who are curious, the parameters of `SDL_CreateWindow` are, in order: its title (`title`), its x-position (`x`), its y-position `y`, its width and height (`w` and `h`, respectively), and its flag (`flag`). Don't worry too much about what this flag means—all that matters is that it is `SDL_WINDOW_OPENGL` because we are working in an OpenGL context.

Setting up our context

Speaking of working in an OpenGL context, that's another thing we always have to create and set up:

```
    SDL_GLContext context = SDL_GL_CreateContext(display_window);  //
Create an OpenGL context for an OpenGL window...
    SDL_GL_MakeCurrent(display_window, context);                   //
...and make it the context we are currently working in
```

An OpenGL context is many things, but you can think of it as an object that stores all of the "states" associated with this instance of OpenGL. Think of a context as an object that holds all of OpenGL; when a context is destroyed, OpenGL is also destroyed.

The game loop

And now, the moment of truth.

The function call making our background blue is the following:

```
glClearColor(BG_RED, BG_BLUE, BG_GREEN, BG_OPACITY);
```

Note: Remember our RBG values were defined in the constants section.

`glClearColor` specifies the red, green, blue, and alpha values used by glClear to clear the color buffers (kind of like a computer screen might clear to black whenever it is inactive). In other words, we are making our screen 19.22% red, 54.9% green and 90.59% blue (a.k.a. **bleu de France**) with an opacity of 1.0 (a solid colour).

Having done this, we're ready to implement our game loop. For this, we use a `while`-loop that will run so long the `game_is_running` condition remains `true`. The mechanism we use to change this condition to `false` is by "polling", or waiting for, a user event (like a screen click or key press). For this simple program, the only event we will be polling for will be the user exiting the game:

```
SDL_Event event; // The SDL_Event type is a union that contains structures
for the different event types, like quitting or closing
while (game_is_running)
{
    /**
     Our games will go here, eventually!
        */
    while (SDL_PollEvent(&event))
    {
        /**
         Basically, while OpenGL is polling, or expecting, an event from
the user, don't do anything.
            */
        if (event.type == SDL_QUIT || event.type == SDL_WINDOWEVENT_CLOSE)
        {
            /**
             The only types of events we are expecting thus far are simple
those that end our OpenGL program. This will change soon.
                */
            game_is_running = false;
        }
    }

    glClear(GL_COLOR_BUFFER_BIT); // Quite simply: clear the space in
memory holding our colours
    SDL_GL_SwapWindow(display_window); // Update a window with whatever
OpenGL is rendering
}
```