Assignment #0

Winter 2018

CS 174A: Introduction to Computer Graphics

Overview

This assignment is designed to introduce the libraries and development tools that will be used throughout much of this course. While this assignment does not contribute to your grade in any way, completing this is highly recommended as all future assignments use some version of these technologies.

All assignments for this class heavily utilise **tinygraphics,** a modern javascript library that provides a simple abstraction layer on top of **webgl**. It was implemented by Garett Ridge, a former TA of this course, as part of the work for his PhD dissertation.

As an aside, we will be using Piazza as a forum for questions outside of class/office hours. If you have any questions for either the TAs or your fellow students, please ask on this course's site on Piazza.

Step 0: Install the prerequisite software

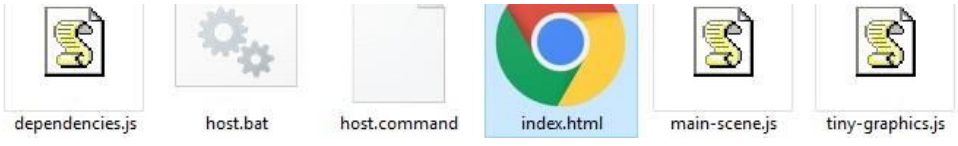
This assignment requires you to have **chrome** and **python** installed on your machine before you begin.

The code we will be using relies on relatively modern browser tools. We recommend using an updated version of Chrome (at time of writing the latest version is v71), though some older versions will work fine. While some other browsers (eg. firefox) may be compatible with out code, all developer tool instructions given here are written explicitly for chrome.

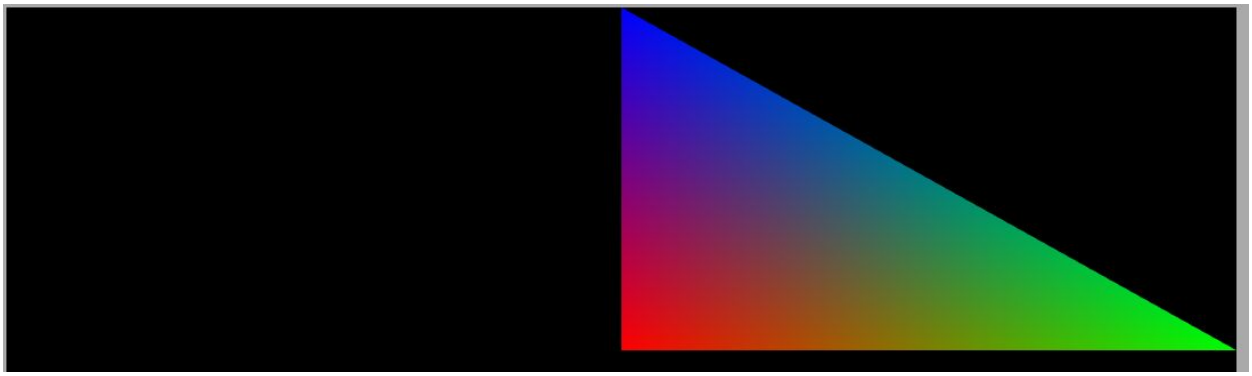
Installing python, and ensuring it is in your PATH, is required for the local server you will run in step 1 below. Either python 2 or python 3 should work (though python 3 will require 'python' to refer to 'python3' in your PATH).

Step 1: Run the code

Extract the contents of `a0_w19.zip` into a folder on your computer. You should see the following files. Don't worry if you see different icons for each file.



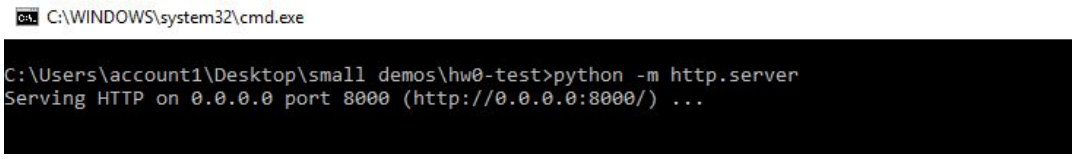
You should see the file index.html in your folder. You can already try clicking that open to see the code run on your machine … mostly. This is a start; you'll see an animation. But this isn't good enough. Your animation is still unable to load local files (texture images, sounds, models) out of your own file-system, due to safety precautions built into your web browser to prevent XSS attacks.



Run a fake server. which circumvents those security protections. Do this by opening the file we gave you called "host" - "host.bat" if you're Windows, "host.command" if your Mac. On Windows you can just double click the file to run.

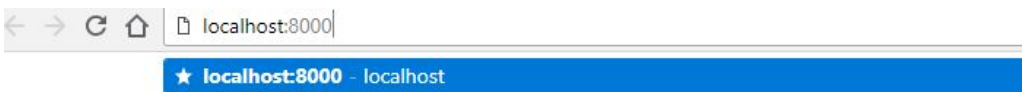
On Mac, you might get a security warning if you double-click. Instead, right click the files, then choose Open, or you can go into System Preferences/Security & Privacy/General and click 'Open Anyway'. You may possibly need to check the file permissions and set them to 744.

Look in the resulting console window. If you can't find a message starting with "Serving HTTP on ..." (or if the window abruptly closes), you may not have python correctly installed -- use Google for help on that, then try our files again. For Mac users, installing Python involves some minor usage of shell commands, unfortunately.



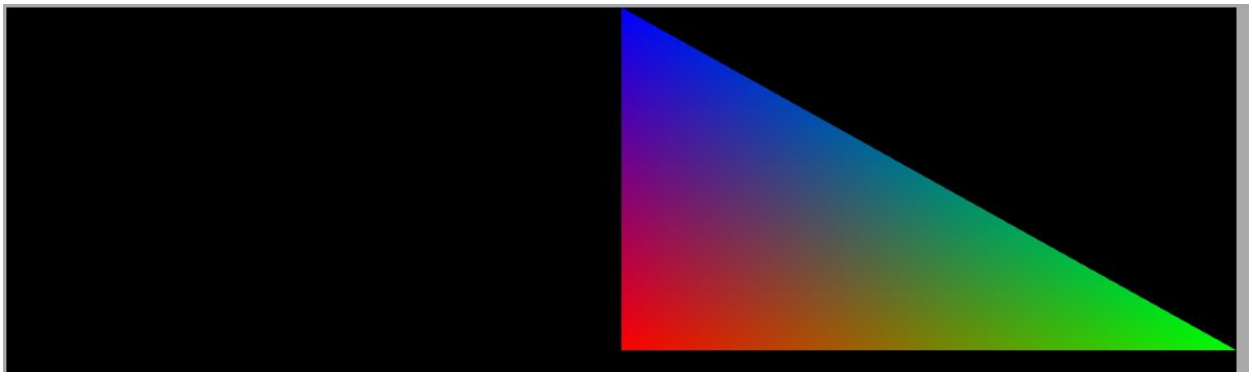
Now you're hosting. Keep that window open.

Open a new window of Google Chrome. Download it first if needed.



Navigate Chrome to the url http://localhost:8000/ That assumes that step 5's message said port 8000 - otherwise change the number in the URL to match.

Observe that your project shows up at this new URL. That's where you'll access it from now on.

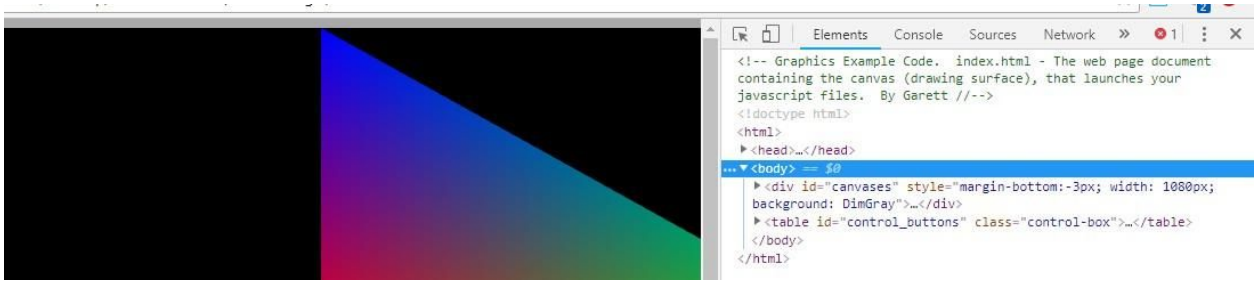


Unfortunately, web developers in practice have to do that fake server thing pretty often to be able to work on their files locally. **Keep the .bat or .command program open while you work; do not close the window.**

Step 2: View the source code

Although any text editor will work on our files, for this class you'll want to use the editor inside of Chrome, because of its debugging tools.

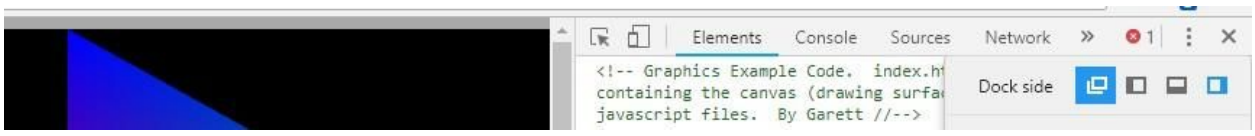
Resume with the open Chrome window from the previous step 7.



Press F12 (Windows) or Cmd+Option+i (Mac) to open the Chrome developer tools panel (DevTools).

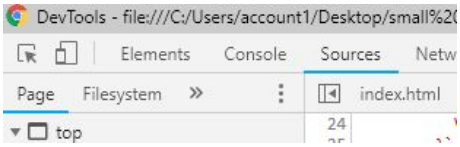
Note: When you open Chrome DevTools, you might be greeted with an error about "fav.ico" not being found. That's ok, you didn't mess up, we just omitted that file for simplicity.

You want DevTools to be able to take up the whole screen. Undock it from your web page window. Do this by clicking the ellipsis at the upper right corner, and selecting the first choice under "Dock Side".

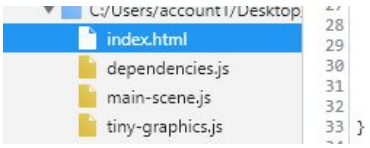


Maximize both your web page window and DevTools windows. Use the keyboard shortcut Alt+tab (Windows) or Cmd+~ (Mac) to switch between them quickly.

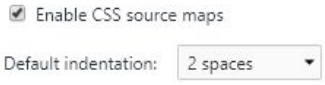
Click the "Sources" tab of the DevTools panel, towards the top of the screen.



Without leaving the "Sources" outer tab, look at the navigator panel on the left. This might be collapsed in the upper corner. Regardless open the "Page" inner tab underneath it.



You should see all the files you downloaded from GitHub here. Click them open to make sure you can see the code. Now you can read it all here.

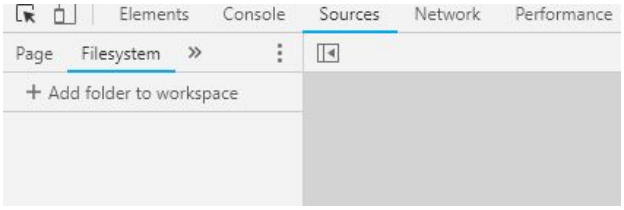


Press F1 to open settings, and choose "Default indentation: 2 spaces". Close settings.
This is just so you won't be prevented from matching our formatting.

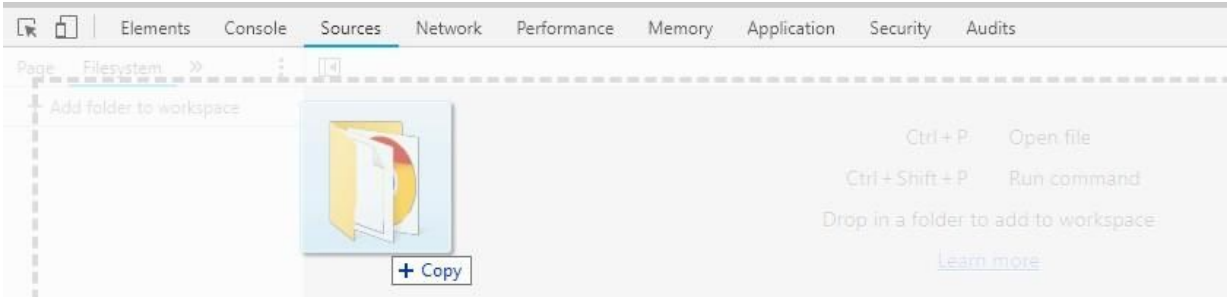
These steps, and the following ones, may seem like a lot of work but they are part of becoming a real web developer with a good workflow, as opposed to someone who just knows the language. The biggest key of all to becoming a good developer is actually going be mastering the **debugger** feature, but first for this assignment let's just take it slow and set up our editor.

Step 3: Begin modifying the code

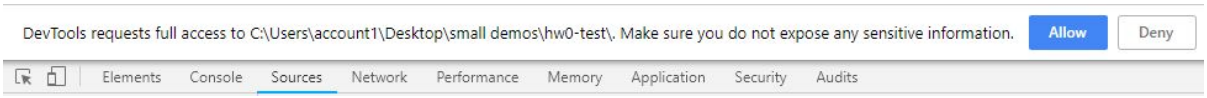
Change from the "Page" inner tab to the "Filesystem" inner tab, which might be collapsed behind the arrow. This one should be empty.



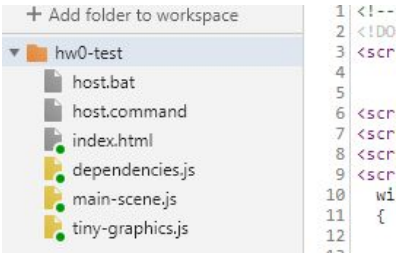
Drag and drop your local file folder from your computer's folder navigator straight into the middle of the DevTools window. If you can't figure out how to drag between maximized windows (you can), just use the manual "add folder to workspace" button and choose your folder. Either way this will complete the mapping between your real local files and the fake ones over the network.



It's going to ask you for permission to modify your local files. Hit yes.



If this doesn't happen as described, try to find help on setting your local folder as a workspace.



Observe the little green dots next to each file in the "Filesystem" subtab. These green dots verify that your Chrome has matched your fake server to your local files.

Sometimes a green dot is missing -- especially on index.html. That is dangerous; the file is not mapped right and any changes you make to it will be lost. When green dots are missing, hit ctrl+F5 (Windows) or cmd+F5 (Mac) to do a hard refresh. This re-loads them from your local files and re-maps them again.

Be aware that for as long as you have DevTools open, back at browser window you have unlocked some new ways to refresh: Right-click the refresh button to see them.

If the green dots still don't show up, delete what's in the workspace area and try again until they appear.

Now you can edit the files directly inside Chrome, in the DevTools "Sources" tab.

As long as you make changes under "Sources" and not "Elements", your changes will now persist in your own local files even after page refreshes. You should avoid ever accidentally typing in the "Elements" tab. That's only for temporary HTML stuff your code generates.

Editing directly in Chrome like this is the workflow you should use. One reason is that your code

immediately changes its behavior as you type. Even when it's in the middle of running, or as soon as you un-pause it in the debugger. Elements will move around on the page immediately when you make changes. This allows you to you dynamically test new code without restarting your whole animation and losing your place -- without having to wait for your timed scenes to progress to that point again -- or without having to enter the right inputs again.

Step 4: Try chrome developer's *power* tools

If you've never learned your way around an IDE for editing code, now is the time to. Chrome's code editor is kind of in- between in terms of quality: Better than Windows Notepad or TextEdit, but not quite as good as Notepad++ or Microsoft Visual Studio. In order for it to be better than crudely opening your code in notepad, you need to know what basic features to expect from a text editor. Let's learn a few of these *power* tools.

Find and try each of the following code editing commands once. They're found in that DevTools Sources tab.

Block indent / unindent (Tab and Shift+Tab)
Block comment / uncomment (Ctrl+/ or Cmd+/)

** For both of the above bullet points, try it on multiple highlighted lines at once.

Zoom in/out while reading

** Hold down Ctrl (Windows) or Cmd (Mac) and then press plus, minus, or zero to adjust.
** Use this fit a comfortable amount of code on-screen for you to read at once.

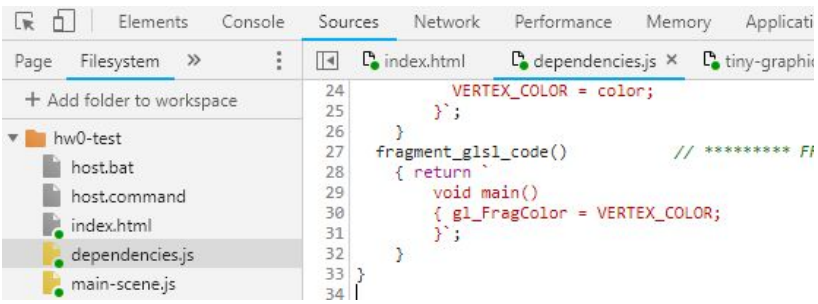
Find (Ctrl+f or Cmd+f)
Find-and-replace

** For both of the above bullet points, note that you don't have to find specific or exact strings; Chrome supports matching **regular expressions**, for finding all text of a more general pattern. That's the .* button.

Press the "{}" button in the bottom left of the window to auto-magically format your code after making changes.

Step 5: Check that your code changes save successfully

With our animation running in Chrome, with DevTools still open to the Sources tab. Open the file "dependencies.js". This is under the "Filesystem" tab of the navigator panel, which might be collapsed in the upper corner.



If there's no green dot next to "dependencies.js", fix it as described above.

The code is divided up into two JavaScript classes. The one at the bottom of the file makes the triangle. Let's edit it.

```
56        // Describe the where the points of a triangle are in space.
57        this.positions = [Vec.of(0, 0, 0), Vec.of(1, 0, 0), Vec.of(0, 1, 0)];
58        // Besides a position, vertices also have a color.
59        this.colors = [Color.of(1, 0, 0, 1), Color.of(0, 1, 0, 1), Color.of(0, 0, 1, 1)];
60        // With this turned off, every three vertices will be interpreted as one triangle.
61        this.indexed = false;
```

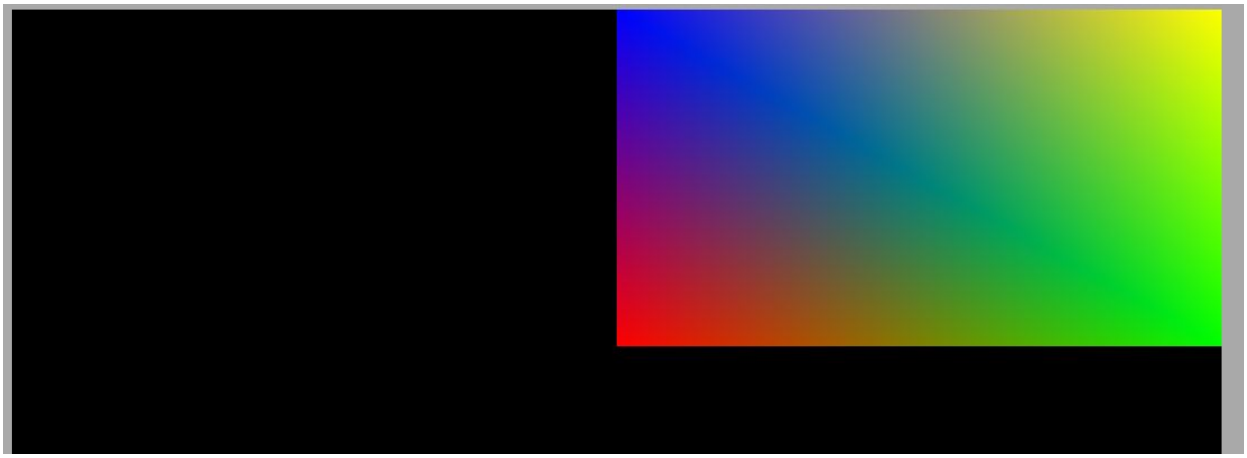
On **line 57**, add the following three items to the JavaScript array, which is all the text enclosed by square brackets [ ]. Add a comma to separate from previous items in the array.

```
Vec.of(0,1,0), Vec.of(1,0,0), Vec.of(1,1,0)
```

On **line 59**, add the following three items to the JavaScript array:

```
Color.of(0,0,1,1), Color.of(0,1,0,1), Color.of(1,1,0,1)
```

Save the file, and reload the page (using `Ctrl+Shift+r` for Windows, `Cmd+Shift+r` for Mac). Switch back to look at your web page window. The triangle should be a square now, because you just attached a second triangle to it. If so, your edit worked and your file is saved. If not, check for green dots and fix it as per above.



If you typed the wrong thing, you could get console errors, a blank web page, or missing triangles. Later on we'll show you how to use the debugger and the console together to approach errors in a smart way. For now, just type it right.

And with that, you're done with assignment 0!

As this assignment does not contribute to your grade in any way, you do not need to submit anything. By the magic of learning, you will have succeeded in this assignment simply by going through these steps. Enjoy!