Assignment 0: JS Paint

Due: Tue, Feb 9 at 11:55pm

Overview

In this assignment you will create a simple paint program. The purpose of this assignment is to ease you into both JavaScript programming and computational photography as comfortably and as painlessly as possible. Additionally, this assignment serves to familiarize you with the programming environment and interactive GUI that you will see in future assignments. Think of this as the walk before the run.

For this simple warm-up assignment, you will implement the fill and brush tools that you might be familiar with from MSPaint, Adobe Photoshop, or GIMP.

Getting Started

The JavaScript based image processing program has two modes: (1) an interactive mode where you can enable/disable various filters, adjust parameters to these filters, and see the result right away; and (2) a batch mode where all the filters and parameters are fixed via the URL string. In general, you will find the interactive mode more useful for testing your programs during development, whereas you will find the batch mode more useful for generating output images while composing the writeups for your assignments.

To get started, download this zip file and unzip it on your computer.

Setting Up a Development Webserver

Next, change to the subdirectory `COS-426-Assignment-0` and run the command `python3 -m http.server` in the terminal. (That command is for python 3. In python 2.7 it should be `python -m SimpleHTTPServer`. Note that there are many other ways to start a web server, for example using `php -S localhost:8000` or by installing WAMP, MAMP, or Node. Various other options are discussed here). A web server will be launched locally on your computer, rooted at this directory. Note that our assignments will not work if you simply open the html file in the browser directly using `file://...` due to the same-origin security policy. Therefore we run a local web server.

Once you have started the local web server, direct your browser to `http://localhost:8000` (8000 is the default port used by the python web server). If you have done everything correctly up to this point, you should see a picture of a flower, which is the web page for Assignment 0. If you see something else, please reach out to your friends or to an instructor for help.

Before you begin, it is necessary to temporarily disable browser caching, or else changes you make to your code during development may not necessarily be reflected on the web page even after a hard refresh! **Many students skim over this step and struggle to get started.** For Google Chrome, you can disable cache via the DevTools: first right-click the web page background and choose `Inspect Element`; then, left-click `Network` in the toolbar to open the network pane. Finally, check the `Disable cache` checkbox at the top (you may need to scroll horizontally to find it).

Keep in mind, this setting is **only active while the DevTools are open.** In general, we strongly recommend Google Chrome for development.

UI Tour

On the right hand side of the web page, you can find two control menus (created using the lightweight dat.GUI library). The left panel is the feature menu, which contains a list buttons that, when pressed, apply a filter/tool/operation to the active canvas. Note that it is the goal of the assignment for you to implement most of these features, almost all of which are not implemented in the starter code.

The right panel is the history menu, which displays all operations that are actively applied to the canvas in the order that they were applied.

Take a moment to play around with the menu and to familiarize yourself with its behavior. For example, when "Fill" is clicked, a new operation is pushed to the bottom of the history menu, and now you can change the fill color using the color picker. Note that nothing happens right now since part of the fill filter is commented out in `js/filters.js`.

You will notice that a warning box pops up when you click on unimplemented features. You will need to comment out or delete the warning box code after you implement the features. All the operations you apply will appear in sequence in the history menu and you can delete the past operations if you want. Observe what happens when you push a number of images onto the history stack, and then click `Delete Below` on the topmost history element. For other operations, only the selected operation will be removed when its delete button is clicked. **Also note that all the history operations are encoded in the URL, so feel free to refresh the webpage and everything will still be there.** This allows you to change your code and then get back to the same place. It also allows you to change the values directly in the URL rather than in the GUI!

To make the assignment debugging and linking friendly, a "Batch Mode" is provided. Once you are satisfied with the parameters in the interactive GUI, click `Batch Mode` and the image output will be loaded in a new tab with all the parameters fixed. Then you can simply refresh when you make changes to your code. Note that when filters that are slow to process, the Batch mode will say "Processing" instead of immediately showing the result.

How to Program the Features

To make your first edit use your favorite text/code editor (we recommend VSCode and Atom) to edit the file `js/student.js` and fill in your name and NetID. Reload the web page in your browser, and now your information should appear above the image. **You should do this for every assignment.**

To implement the image processing features listed below, **you only need to edit the file `js/filters.js`.** Nevertheless, before starting on that, we recommend you take a quick look at the file `js/image.js` because it has some important helper code relating to images and pixels. You are welcome to look at any of the other files, but it should not be necessary and some of them have some pretty byzantine JavaScript magic.

The first filter you should look at is `Filters.fillFilter( image, color )`. If you print the `color` to the developer console by adding the line `console.log(color);` to the function, and then applying the filter, you will see that the `color` parameter holds a `Pixel()` object that has the same RGB values as specified in the UI. It will be important to understand the pixel data structure for the next two filters, but for now, you can complete the fill filter by uncommenting `image.setPixel(x, y, color);` within the for loop. Play around with this filter to ensure it works!

The remaining filters are for you to solve and implement. Before steamrolling ahead, you might want to take the time to read, understand, and test out some of the `Pixel()` and `Image()` member functions implemented in `js/image.js`.

Note that it is not necessary to make deep copies of images since the html canvas always displays the returned image; it doesn't matter whether it's a new copy or the original image. Generally speaking, in-place image processing is the better solution due to significant speedup.

A Note on Grading

Your assignment submissions for this course will be graded by manual inspection, not by scripts. **As such, do not stress about or waste your time striving for "pixel-perfect" accuracy.** For instance, if your Brush implementation draws a circle with a radius less-than-or-equal-to the radius parameter, whereas our reference example uses the strict less-than radius, you will still receive full-credit! We care far more about your understanding, and likely won't even notice small discrepancies so long as your result is generally correct.

Debugging Tips

In general, graphics programs are very challenging to debug. Thankfully, modern browsers contain a built-in JavaScript debugger (under the sources pane in Chrome) that will allow you to set execution breakpoints and to trace variables. You can also pre-set breakpoints by inserting the line `debugger;` into your solution code — this trick will likely prove invaluable. Students also find print statements via the `console.log()` prove helpful.

Hints

A few hints:

- This assignment is meant to be an easy warm-up. If you are struggling, seek help now, or consider taking COS 426 next year.
- Please make a public post on Piazza if you have a question.
- Take note of the late policy and the collaboration policy.

FAQ

Here are some answers to frequently asked questions. Check back here occasionally, as we may add FAQs to this list:

- **How do I add my own images or .json files?**

  The file lists are hardcoded in coursejs/guiConfig.js because javascript does not have access to the filesystem. Please modify this file when needed.

Deliverables

Submitting

You should submit your solution via google classroom. The submitted zip file should preserve the directory structure of the skeleton code we provided in the zip file above. If you like to include larger files, that exceed the file size limitations, you can put these files in your own web space, Google Drive, Dropbox, etc. and then include a link to that in the write-up.

The `writeup.html` file should be an HTML document demonstrating the effects of the features you have implemented and would like scored. For the features you would like to demonstrate, make sure that you include the required results by

An image filled with Princeton Orange (generate)

replacing `placeholder.png(s)` with your results. You are encouraged to include more representative results, but extra results only affect your score when your implementation is partially correct. You don't have to show the input images for the required results. ***Please indicate late day usage at the top of the write-up.***

You should start from the the example `writeup.html` provided. At the top of that file are a list of features that you might implement, linking to the section where you talk about them. Please remove any features that you do not implement from the list as well as the corresponding sections, but otherwise leave this header section intact. When you include an extra result, also include a link to the `batch.html` command that creates that image. Please put effort into your `writeup.html` as this is the file we spend the most time grading.

Do not, under any circumstances, share any component of your writeup (text and images) with another student, even if you partnered with the student when composing your JavaScript solutions.

Scoring

This assignment is worth 5 points. The list of features that you need implement is provided below (roughly ordered from easiest to hardest). The number in front of each feature corresponds to how many points the feature is worth for the full implementation. Partial or partially-correct solutions will receive partial credit. For this assignment, all features are required. Example images of expected output are provided for your reference.

You are encouraged to use the implemented features or the custom filter to create an art piece to participate in the art contest (which yields one point for participation and two for winning). Your final score will be calculated by adding:

- Your score on the required features (up to 5 points).
- Your participation in the art contest (up to 2 bonus points). You are welcome to use the `customFilter()` to make your art project, although this is not required.

Assignment Features

(1.0) Fill

Fill the whole image with a given color.

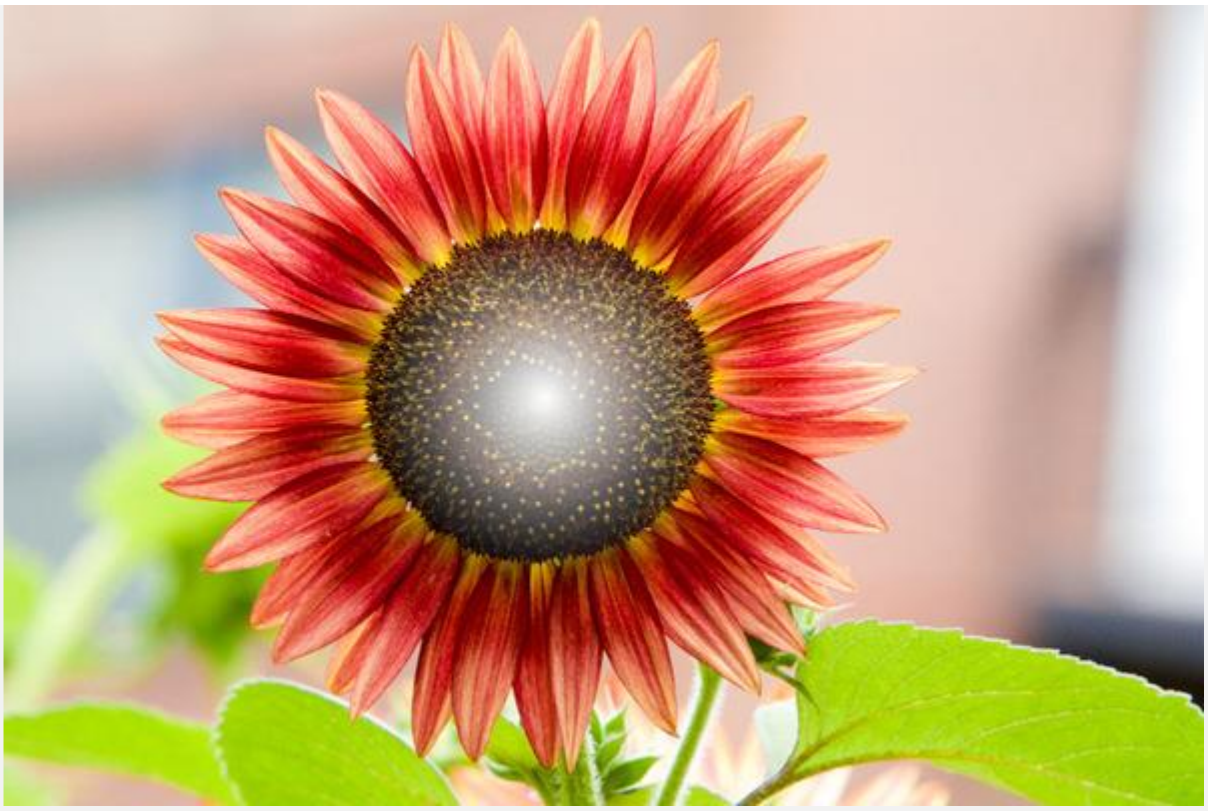
(2.0) Brush

On mouse click, draw a colored circle with a given radius that is centered at the mouse position. Note that the provided code both captures the click and computes the mouse position in image-coordinates for you. Your task is to figure out how to fill-out pixels in a circle of radius `r` that is centered at some point `(x, y)`.

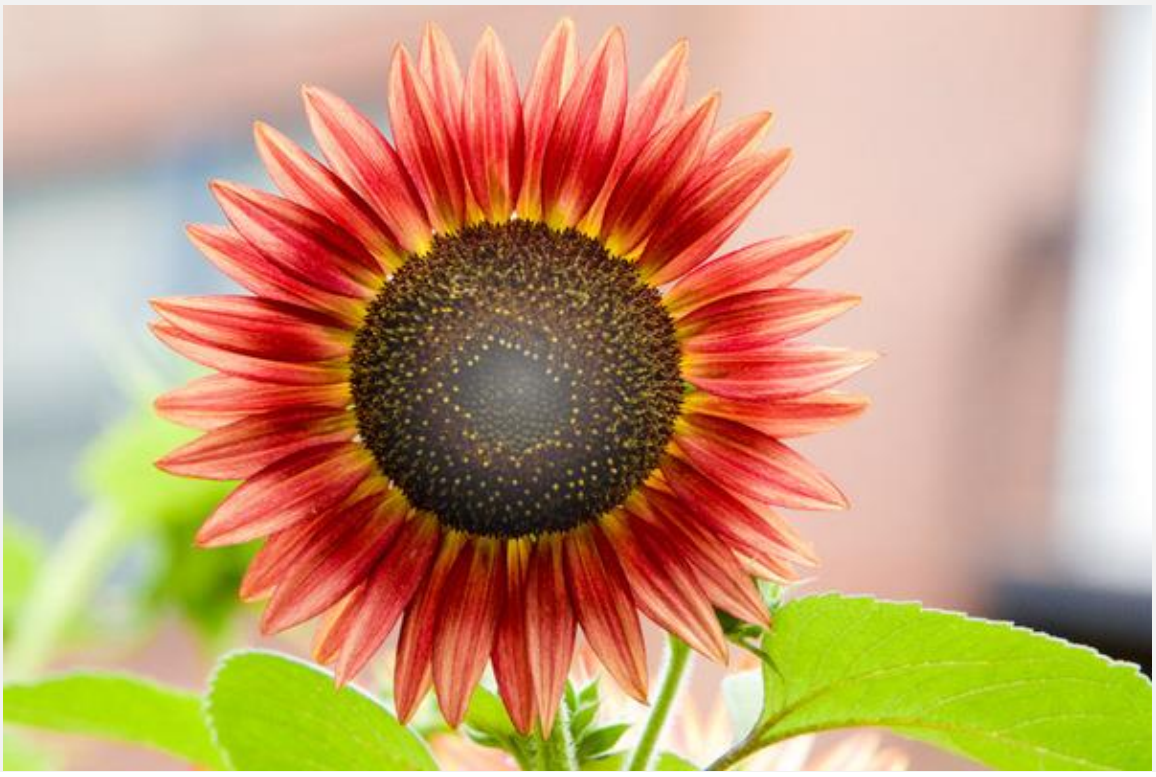
A painted course logo (generate)

(2.0) Soft Brush

On mouse click, draw a colored circle with opacity decreasing from center to the edge. You will likely want to read up on alpha blending. Note that you should not be touching the alpha channel of the `Pixel()` class to accomplish the blended compositing effect.



A circle with center opacity 1.0 painted on top of the flower image (generate)



A circle with center opacity 0.5 painted on top of the flower image (generate)

Clouds painted over the Mesa (generate)