**Images in Processing**

**Carson Foster**

Now, we’ll discuss how to use images in our Processing sketches, mainly through the PImage class. We’ll look at how to load, display, and write our sketches to images.

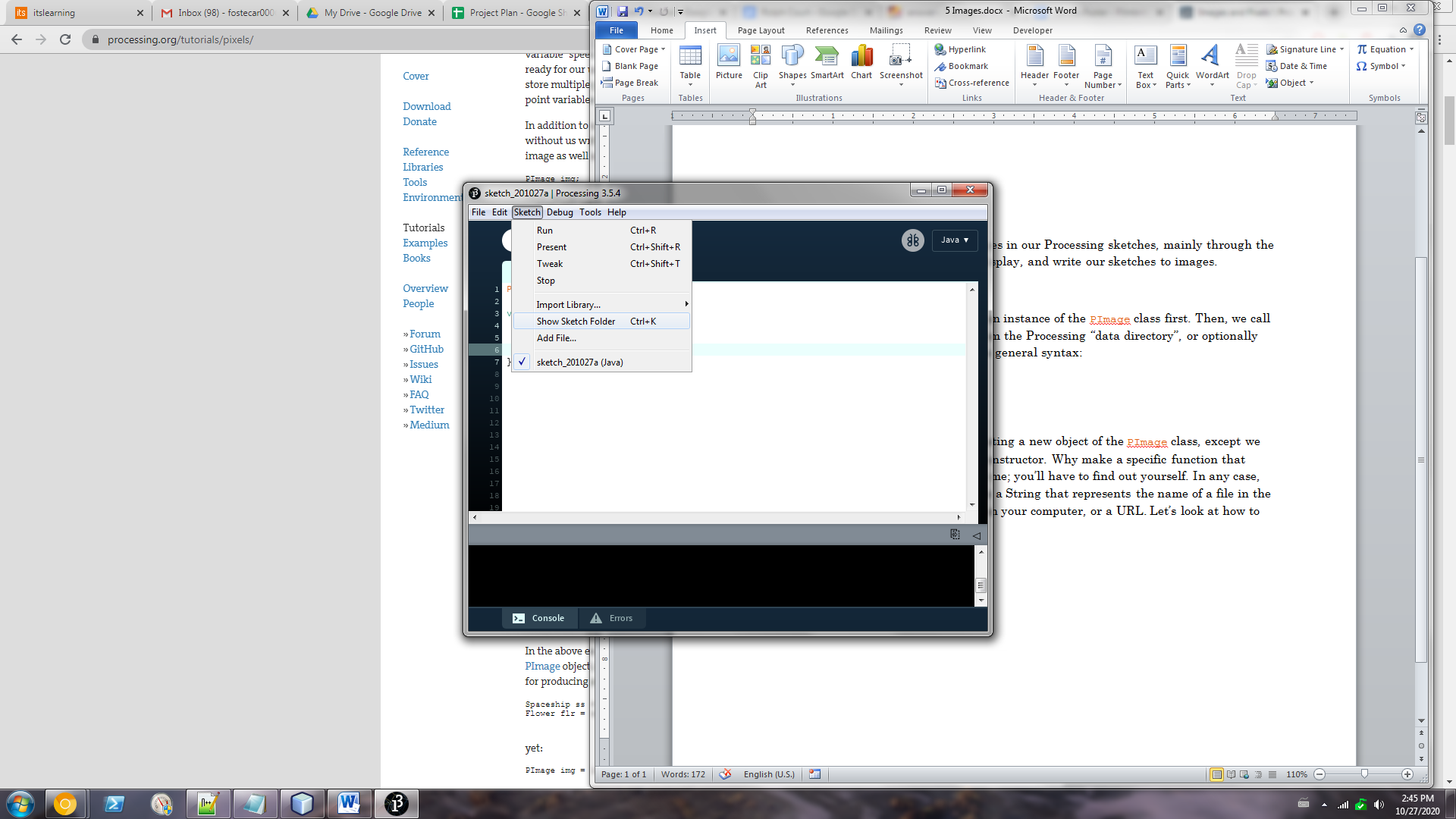
Loading Images

Currently, Processing supports loading GIFs, JPEGs, or PNGs. When loading images, we declare an instance of the PImage class first. Then, we call a specific function that loads an image from the Processing “data directory”, or optionally an absolute path or URL. Let’s look at the general syntax:



As you can see, the syntax is just like creating a new object of the PImage class, except we use the loadImage function instead of a constructor. Why make a specific function that could be replaced by a constructor? Beats me; you’ll have to find out yourself. In any case, the one argument that loadImage takes is a String that represents the name of a file in the data directory, an absolute path to a file on your computer, or a URL. Let’s look at how to access the data directory next.

While in the PDE, select “Sketch” and then “Show Sketch Folder” to open up an Explorer window.



This is not the data directory. Now, inside the sketch folder, create a new folder named “data” (hence the name, data directory). If there is already a folder named “data” there, then that is your data directory. Any images in there can be recognized by name only (e.g. “image.png” instead of “C:\Users\cwf\Processing\[...]\image.png”).

An absolute path is, conversely, the latter example. An absolute path tells you exactly how to get to a file from the root directory of a specific drive (for example, the C drive). I suspect that everyone reading this knows what a URL is, so I’ll skip that topic.

Lastly, whenever you are loading images with the active mode in Processing, be sure to actually load the image in the setup function. This means that you might have to declare the image first as a global variable (put its declaration outside of any function). Otherwise, if you load an image multiple times (e.g. in the draw function), you will have terrible performance and may run out of memory. Here’s a simple example:



Displaying Images

Displaying images that have already been loaded is very simple in Processing. Here’s the syntax:



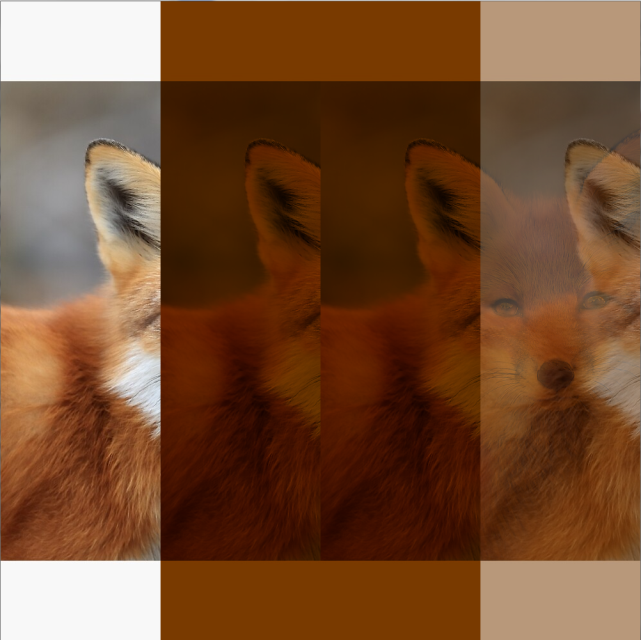
The img parameter must be an object of the class PImage. You must specify the x and y coordinates of the image’s upper left hand corner (think of it as a rectangle). You can also specify, optionally, the width and height of the image. However, this does not crop the image, instead, it stretches or squishes it.

Additionally, Processing has some basic image manipulation functions, and we’ll look at the most widely used one in this section, which is used to manipulate the color of the image. Here’s the syntax:



This is similar to the syntax of the stroke and fill functions of the past. The tint function can take a few different sets of arguments: a color variable, a color and an alpha, a greyscale value, a greyscale value and an alpha, RGB, and RGBA. The tint function applies this tint to all later image calls. However, tint function calls are not additive like transformations; one tint call completely replaces the previous call. In addition, calling tint with an argument representing white (such as tint(255)) essentially removes tint.

Let’s look at an example:



The first fox is normal, with no tint on it. The second and third foxes are both orange-tinged, but notice how I only have to call tint once. The last fox has no color tint on it, but it is only half opaque, so some of the orange tint from the underlying third fox shows through. Finally, draw your attention to the loadImage statement. This is an absolute path, but note the double backslashes. This is the same as in Java. If I used single backslashes instead of doubles, then Java would think this string was riddled with escape characters, so we use double backslashes to escape the single backslashes.

Writing to an Image or PDF

Within Processing, you can natively export a sketch to either a PDF or an SVG image. Unfortunately, there is no built in way to export to a JPEG or PNG, but you can always take a screenshot or convert the generated SVG file. However, there are caveats to both methods. First, you cannot use images in a program that exports to an SVG file. This is because displaying an image is a raster operation (pixel-based), while SVG files are vector images (do not use pixels). Although you can use images while exporting to PDF, I have discovered that no tints will be visible in the resulting PDF. Finally, although it is possible to save a single frame from an animation, we won’t be discussing that here, as we haven’t learned how to animate yet. Both of these methods are similar in syntax:



And also:



Notice that in both, the size call changes. Also note that you must call exit when using the SVG exporter, otherwise nothing will be output to the SVG file. In addition, the file created by Processing will be in the sketch folder.