Pixtory

PROJECT**INSPIRATION**

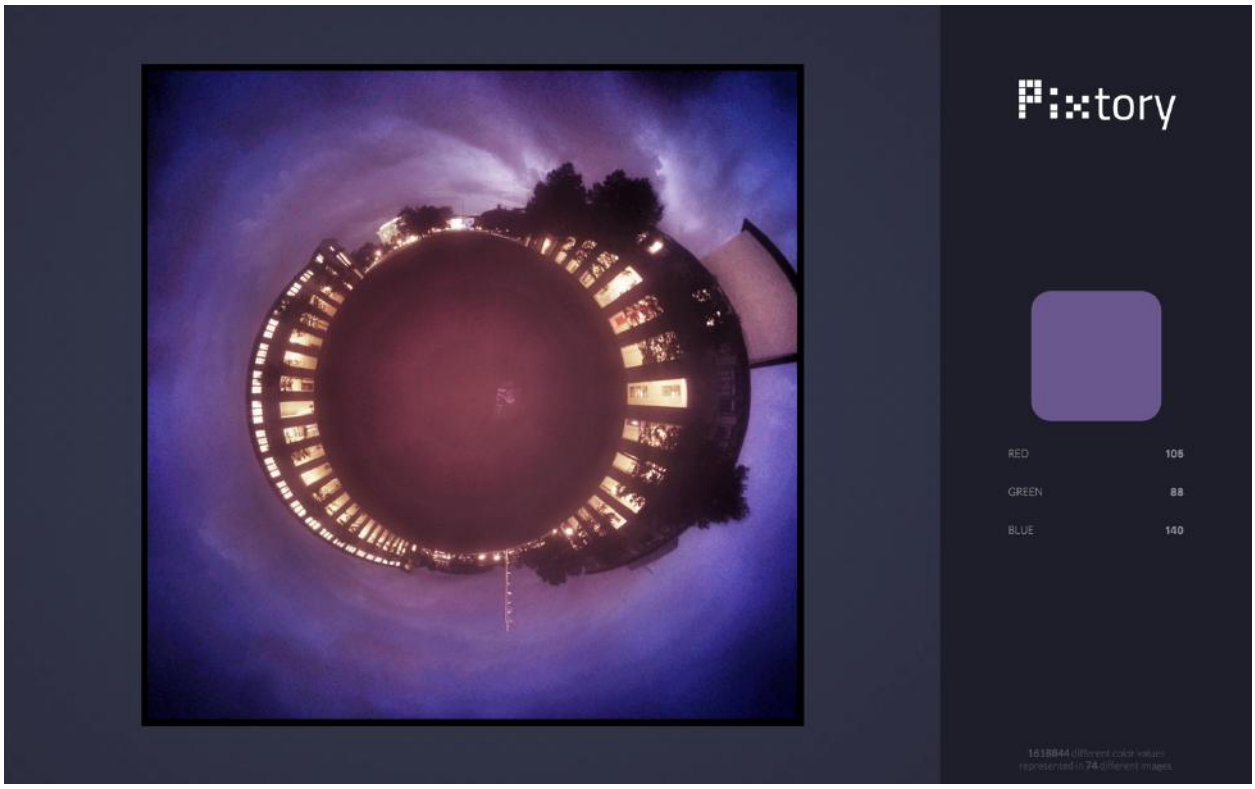
This project was the result of a rather long train of thought that started with the analysis of our project prompt, mapping interactions. I had thought *"what is the thing I interact with the most?"* and after an initial idea of "money," I realized the one thing I look at, listen to, touch and use the most every day is my laptop. This beautiful MacBook has been my tool and canvas for all things art- from computational generative works to meticulously detailed PowerPoint paintings. I thought, digging deeper into the idea of "interacting with", I thought to myself "*what PART of the computer do I interact with the most?"* More than the keyboard or trackpad, I interact with the screen, or rather, the individual pixels that each display a single color, where when working in harmony they create the image that is displayed and relayed into my eyes. Thus working with pixels, and the single colors they display, became the basis for my project. I wanted to map the history of the colors each pixel displayed.

The idea of the project is to *map a history of the colors pixels have displayed* on my screen. I wanted to create an interface where someone could choose a color and it would show the state of my screen at some point in time that, somewhere on the screen, contains that specific color. After some iterations of planning and testing, the final project uses every photograph on my computer rather than my original idea of screen captures. This is mainly because of the abundance of photos (I'm in charge of my family's photos, and do professional photography) and the abundance of colors in photos compared to often static screen states. Deciding to go with the photographs on my computer also added a second level of conceptual thought into the project, one where I'm not only mapping colors of the pixels on my display, but also *mapping my life, through photos, by color*, something that I don't think I've heard of as being done before.

PROJECT**DESCRIPTION**

Pixtory ("pixel" + "history") is the final product of all this testing and designing. It uses a python script to analyze every pixel in every image in a designated folder (in my case, a folder with every image on my computer, involving family, personal and professional photographs). As it analyzes each pixel, it reads the color value in the form of RGB (red/green/blue) and checks a database of every color to see if it has found an image with that RGB value before. If it hasn't script records information about the image file and the (x,y) coordinate at which the color is, and adds it to the database. If the color had already been found in a previous image file, it moves on to the next pixel. I also have some logic

that prevents one image from “hogging” too many colors at once, and the script also randomly chooses images from the folder, thus resulting in a different list of color-to-image values with every time the script is run. The resulting data is huge, because the amount of colors a pixel can display is exactly **16,777,216** (the range of the red, green and blue values are between 0 and 255, so for example the color white is [255,255,255]).

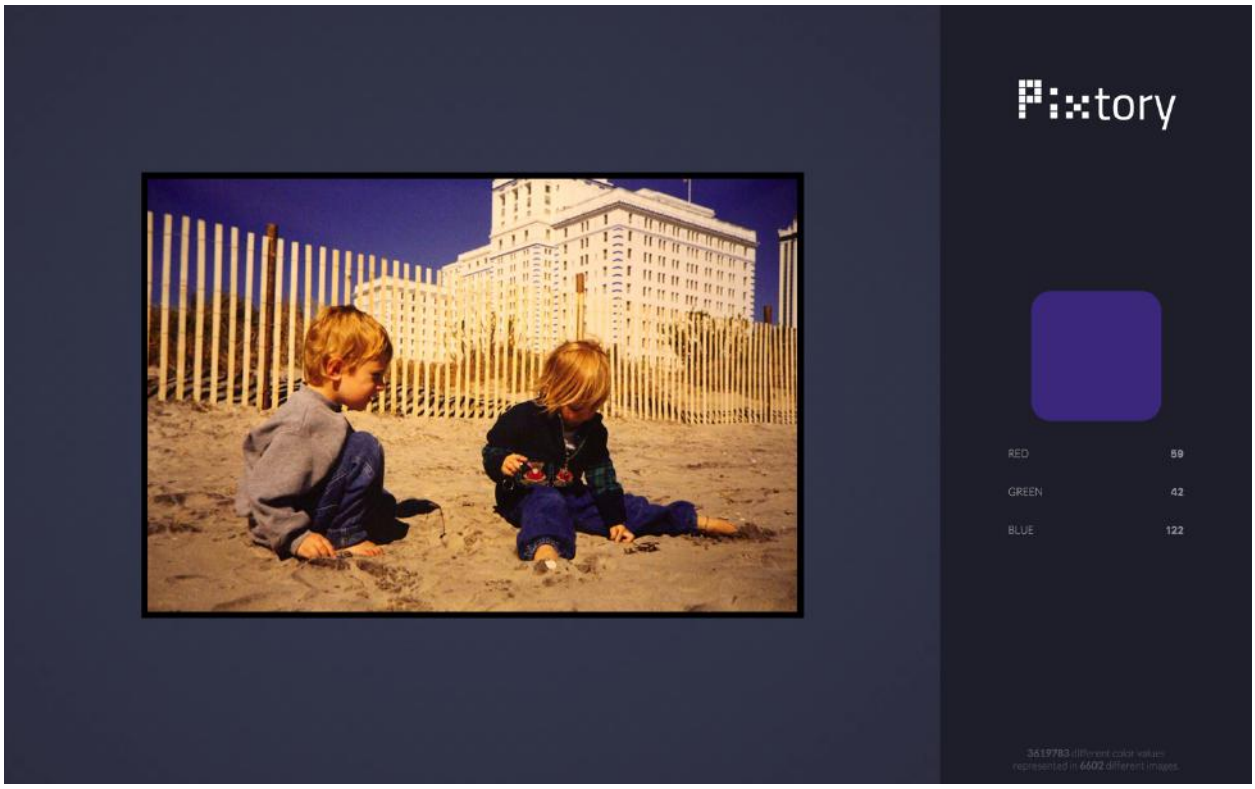


The project then has a webpage component for the interactive user interface that displays an image with the selected color. The page is created with HTML (and CSS) and uses Javascript for the logic and interactivity. A user can manipulate the selected color by using the “color menu” or the individual red, green and blue text inputs. As the user changes the values, an image that contains that color is displayed, and if the user moves the mouse over the image, it will display a mask highlighting the exact area where the pixel with that color is located. All this information is initially loaded from the data produced by the python script.

PROJECT**REFLECTION**

While creating this project, I had one major disappointment, and one major development issue. Remarkably, the development of the analysis algorithm wasn’t the hardest part of the project, it was actually efficiently and quickly loading the analyzed data into the webpage. For two whole days I spent my time doing nothing but trying new and different methods of loading data into a webpage and parsing it into the data structures Javascript could recognize and work with. Initially, it would take about 15 minutes to load data with

100% color completion (meaning, every color possible of the 16.7 million had an image file associated with it). During that 15 minutes, the webpage would freeze up. The final version of my project now can load data with 100% completion in about 14.5 *seconds*, a huge improvement in time. And it does so with minimal freezing, and with a useful loading screen to see how much data is left to load.



The disappointing part of my project was actually color completion (how many colors were represented in my folder of images). Originally, I resized my images to a maximum width/height of 1000px for accurately displaying the pixels in the web browser. This means that if a color was found at x=30, y=30, then you could count the pixels in your browser and find that color too. I had also originally planned to only scan the images I took since the day I bought my MacBook, which was June 22, 2012. That resulted in about 27% color completion with the 6,600 images I had. Big disappointment (I thought I had more color representation in my photography than that! It's like if you were a painter, and you found out you've been using one shade of color in all your paintings). So I expanded my project scope to include *every* photograph I've ever had in my life, which was about 25,000 photographs on my computer. Once resized down to max 1000x1000, it resulted in 31.1% completion. Wow! Quadrupling the amount of photos, and only a 4% color representational increase. However, to picture what that 4% really is, you must realize that it is *an additional 671,088* colors. So, in an effort to increase the color representation even more, I decided to analyze full-size images and scale them to 1000x1000 post-analysis, for webpage displaying purposes only. The mask that shows where the pixel is located then may not actually be displaying that color, but where that color would be in the full size image file. For reference, my photographs generally are

greater than 5000x3000 pixels. This decision to analyze 25,000 full-size images was a costly one, so as of the time of this writing, I am still analyzing the photos, with 20,900 images more to go (it has been two days so far). However, there is an upside. After analyzing the 4,900 images so far, I've already reached 31% color completion, so only time will tell how far up we go (I'd ideally like to hit 50%, although 38% might be more realistic).

In regards to the mission of this project, and the "rubric/goals" associated with it, I think I have succeeded quite well. I think this project does critically evaluate a "portion of the world around me" in an interesting, unique and compelling way. I took a few risks with this project, such as applying techniques and technologies I've never used before (such as image analysis) and the possibility of it not working out (one thing I was worried about was if the web browser could display potentially 16 million different images without crashing, or if the coding languages could handle 16-million-sized objects). I believe I thoroughly researched this project, from a conceptual and a technical standpoint. I think this project is both a very personal project (it involves photos of my life, after all, and feels like a randomized slideshow of my life) and still intriguing for others to use and see. If it were possible, I could port this into an app others could use on their own family photos with. I believe I successfully chose the right medium and comprehensively developed it within it. I don't think an idea like this would work in another medium (such as print), although variations of this concept might work (such as those photo collages you see, where an image is made up of lots of tiny images). It may be possible to come up with another approach to the project within the digital medium, too. I do believe that I also developed a design and aesthetic to the user interface that is both simple and effective. Even the analysis print-out results may please the inner hacker in some people.

Update: *The processing has been complete. The result is a 34.2% completion for full size images, however I figured that if you vary the RGB values of a color by up to 3 digits, the resulting color is identical to the human eye to the original color, so by applying this logic to "empty neighbors" of assigned colors, I managed to achieve 100% completion. However, I realized there was a miscalculation in my original analysis, one where I limited images to 0.3% color spectrum. But that means 50,000+ color representations which is still a lot, so I will work on this project in the future to limit images even more and allow a broader range of images (for reference, this latest analysis used only 3,500 of the 25,000 images for color representations, so reducing the limit will increase number of images represented in the final presentation).*

PROJECT**DOCUMENTATION**

Final project video documentation can be found here: https://vimeo.com/184268677

PROJECT**NOTES**

Interaction Project IDEAS

- money
  - → where has ~~the~~ a single bill been (tracking geolocation around world (?)
  - → all the things a single bill has been spent on (from bank to ...)
  - → see same item purchased in all decades since 1900 (to see inflation)
- concrete (where did the sand/rocks you're standing on come from? Are you standing on a canadian beachfront?
- computer screen (history of screen ... scrolling through timeline of screen recordings?)

> create function to analyze all pixel colors on a screen at one time

> analyze all pixels

[r][g][b] = Pixel

↳ obj
- int r
- int g
- int b
- int x
- int y
- str. filename

> 2012 – 2016 all photos (at some point were displayed on screen)

> management functions:
- check for unused colors
- check for unused images (delete?)
- get num images used, num colors done, num not done

JS: - can read file

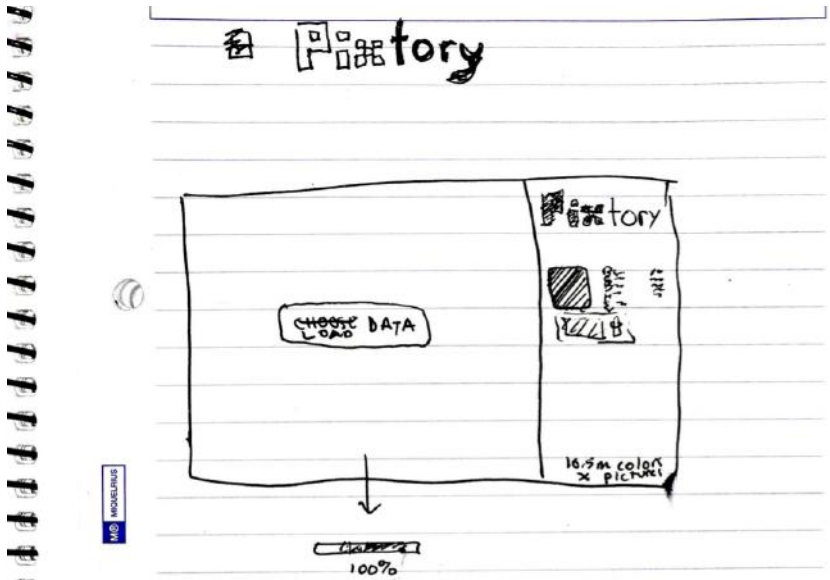
add lazyload to library.md

3 folders:

> batch resize (PS)

PY
> get list of images in "analyze"
> choose randomly from list
> analyze all pixels. If any, move to "used" once done. Else, to unused. write findings to file (object to json)
> run tests

> read file into array, JSON str to dict (loading sign)
> read r/g/b value from inputs
> find [r][g][b]
> display image, position viewfinder/loupe



To Do
- thread python
- set observer on data-progress (for "100%"
- figure out progress/non-freeze for loading .js. file
- make a run-all-in-one file (server, etc)