Process Book: GLITCH

Emmett McCann, February 28th, 2019

CS480X DataViz Final Project

For my final project, I decided to explore some of the data behind my own artwork! I do a lot of algorithmic artwork for fun and have never had a good way to articulate how the art is actually made. Most of my recent work has been with image processing via standard array functions in Python. I really like exploring how chaining together simple steps can result in very complex and interesting art pieces.

The art project I explored in this visualization is called GLITCH. Written in Python, it takes in an image and creates a corrupted-looking, colorful, sharpened art piece. My goal with this visualization was to articulate the relatively simple process that underlies this seemingly complex program. I wanted to take the image state at different points of the program and find some way to visualize each state and the transitions between them.

Another aspect that I really wanted to explore in this project was 'narrative visualization.' I chose to create a 'scrollytelly' visualization that lends itself well to storytelling and describing a process.

Related Work

I was introduced to scrollytelly vizzes by R2D3.org. Written by a few JavaScript developers in their free time, the site has a few articles explaining processes like machine learning classifier training. The narrative structure and opportunities of scrollytelly's really stood out to me here as they were able to explain complex computational tasks with simple visual idioms.

Another popular data visualization site that inspired this project is Pudding.cool. This is an actual, maintained semi-news site that presents interesting visualizations about pressing issues. A lot of their work uses scroll based interaction because of how simple it is for users and how it forces a sort of linear, narrative timeline for exploring a set of data.

Questions

The interesting thing about designing this kind of visualization is that I already know everything going on behind the scenes of the data. I am not looking for certain patterns in the data because I already know every pattern that is there.

Therefore, my main question was: "How can I articulate this image generation process in a way that anyone can understand?"

A question that developed in the background of the project was "how much information about the process is needed to claim 'understanding'?" For instance, I could just show the entire image at every major step, or I could show every pixel at every instruction in the script. I settled on a midpoint, showing some larger overview shots of the image at key steps and zooming into a pixel by pixel level for some subprocesses and analysis.

Data

Since my data source was my own Python code (createData.ipynb), I was able to extract whatever data I wanted at any step in the process. Within the image generation code, I inserted save functions that exported the overall image and slowly assembled a large CSV with values for one specific column of the image that I wanted to focus on. This allowed me to show both overviews (images) and do some very interesting graphs and animations with a single pixel-column worth of data.

Exploratory Data Analysis:

I mostly stuck with my initial plan for the visualization with regard to viz structure. I had a sort of narrative planned out that stemmed directly from the Python code. Most of my exploration was actually in the color space. Since the overall function of the GLITCH script is to change each pixels' color slightly, finding where to show that color change was an interesting task.

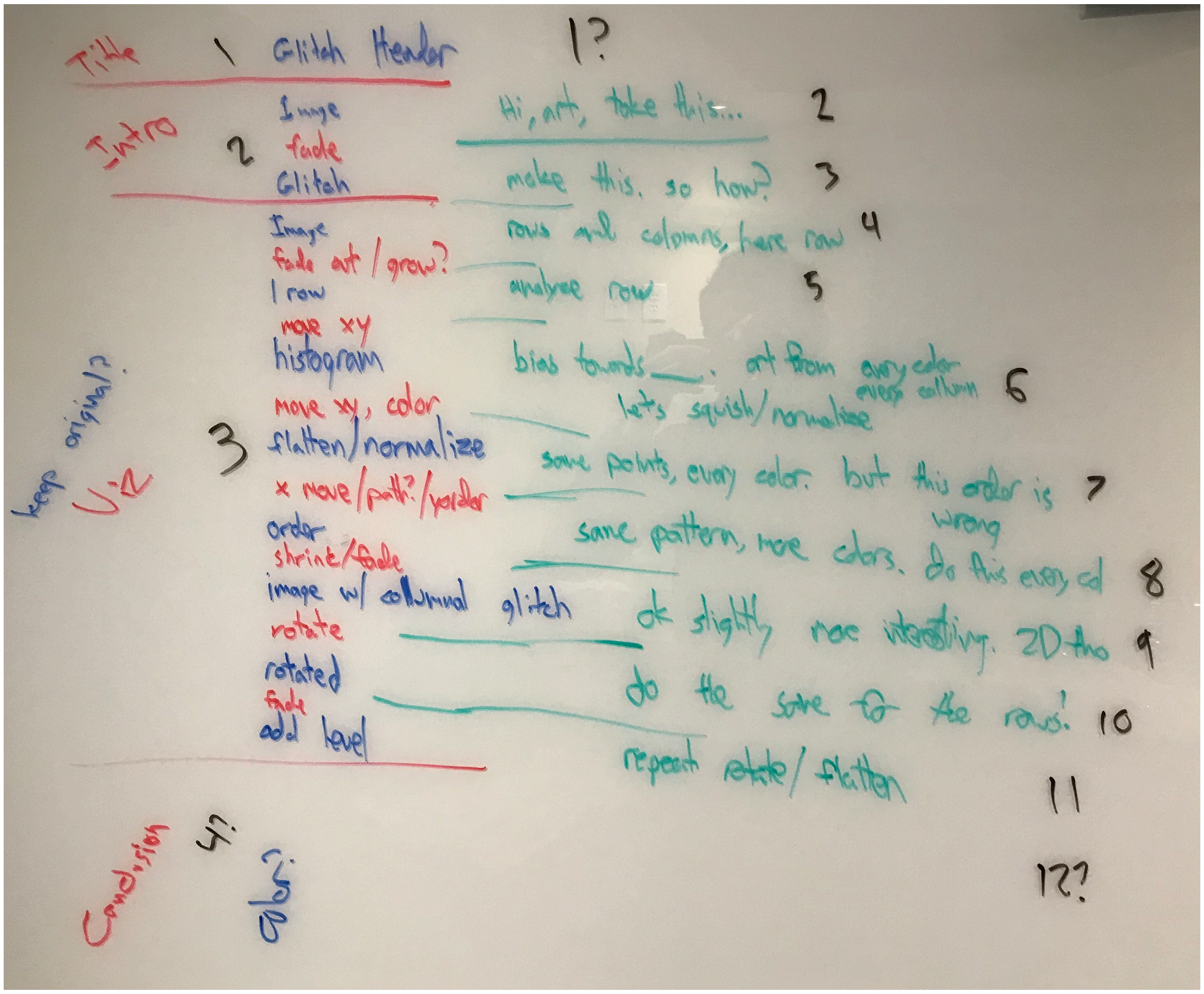
I originally started off analyzing and telling the story of GLITCHing a black and white image where I only had to deal with grayscale. Once this was partially working, I moved to a multicolor image. In order to properly show the true process behind the code, I had to split the single column of pixel data into red, green, and blue color channels. I then focused in on one of them because my attempts at creating some sort of 3-color viz with independent channels just got too complicated and busy to fulfill my goal of making the process understandable.

Design Evolution

I started out my design process on paper, sketching a few different visualizations for different stages in the GLITCH code. I settled on a few layouts early on that carried throughout the process, including the histogram and the zippering value arrangement.

My next real step was laying out the narrative. Luckily, for a piece like there where I am describing a process, most of the narrative is already laid out in the GLITCH code. However, there are a few sections that I gloss over — data IO, array-to-image conversion, and some math/mapping functions to name a few. I felt that these parts didn't translate well to visuals and are almost assumed by users. For example, you don't really need to know the underlying array structure of the image, you just need to see that you can select a column of pixels. Your brain kind of puts the rest together.

After this, I took to a whiteboard for the main portion of the initial design process: pairing visualizations to the narrative. I ended up with the below diagram showing page sections, visualizations, transitions, and narrative. This was what really allowed me to create the strucrure of the web page and actually translates almost exactly to my content arrays in the code.



One thing that I really wanted to do was create narrative through linked elements/transitions. I think the smooth scroll-based transitions allow a user to really explore the data without throwing too many options or controls at them.

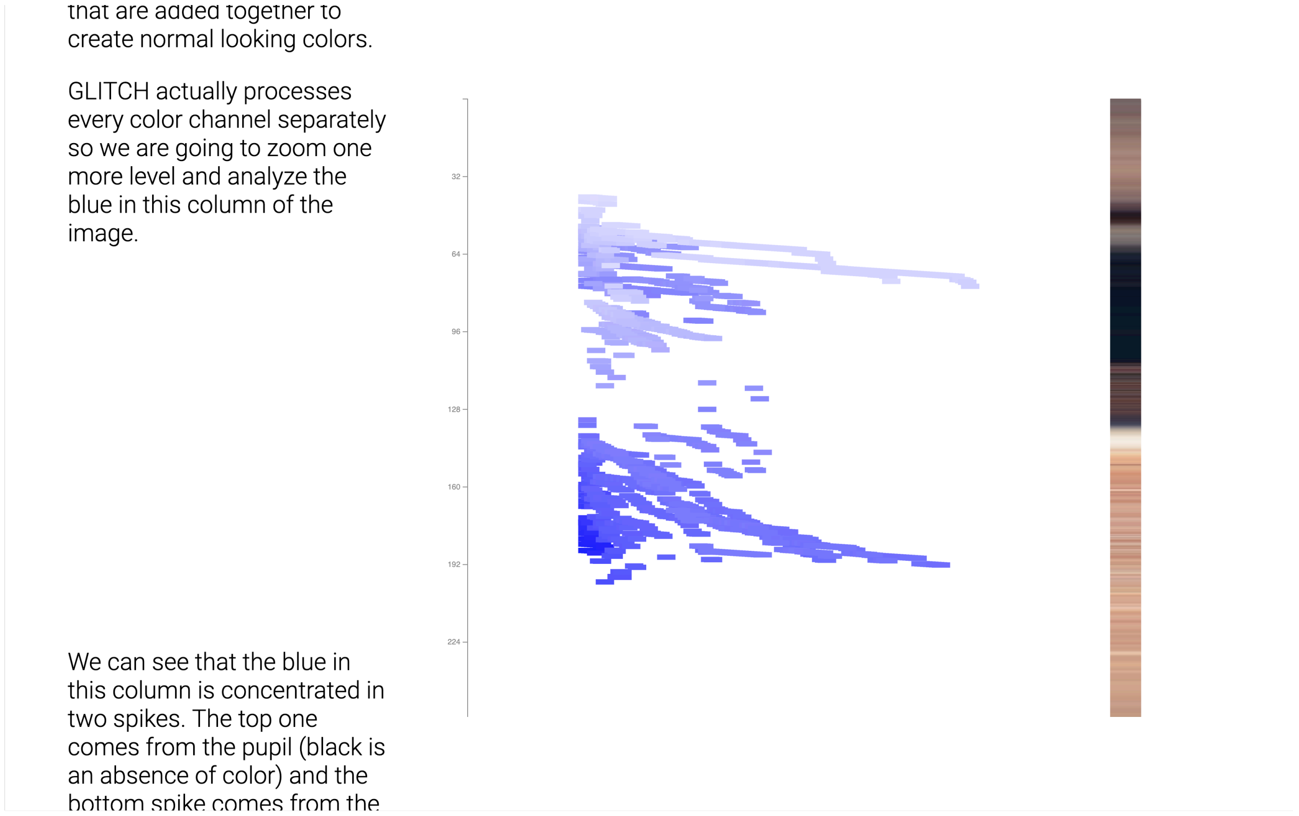
Implementation

The end result of my project is a scroll-based narrative visualization. I wanted to keep it very clean so that the narrative and the data could really shine through. This was also inspired by sites like R2D3 and The Pudding that have very simple visualizations with smart spatial encoding to make the narrative engaging and not overpowering.

In order to get the scroll-interaction specifically how I wanted it, I actually wrote my own scrollytelly system that basically converts two object arrays (narrative sections, and visualization charts) to a structured, scrolling webpage where sections are linked to specific charts and transitions within charts. This allowed me to essentially copy over the strucure laid out in the above picture and easily change the ordering and narrative structure of the page. I will be using this system in future vizzes and plan to standardize it to some sort of library at some point (still some improvements to make in it).

I think the most interesting section of the site is once the user gets down to just looking at the blue channel for one column. With a ton of moving parts, this was where it was crucual to get the animations and transitions between vizzes correct so that a user understands what is going on.

My favorite examples of this are the position to histogram transition and the newColor to original position transition (zippering). The position to historgram transition moves from showing the blue pixels in their image location, to showing them in the histogram. By scrolling up and down bit by bit, the user can really gain an understanding of what is happening. By taking the raw data (position) and switching to an analysis state (histogram) with animation, each datapoint is visually linked to its effect on the histogram. As a transition, it is best represented in the actual website, but a screencapture of it partway through is below.


that are added together to
create normal looking colors.

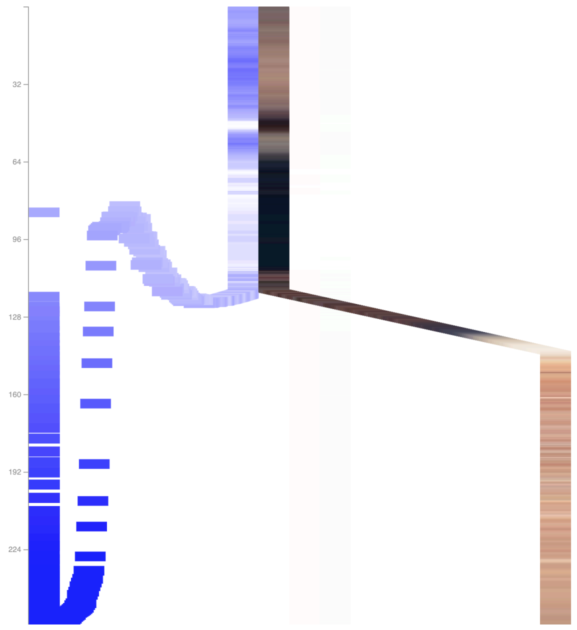
GLITCH actually processes every color channel separately so we are going to zoom one more level and analyze the blue in this column of the image.

We can see that the blue in this column is concentrated in two spikes. The top one comes from the pupil (black is an absence of color) and the bottom spike comes from the

The zippering transition later in the page does a great job of again tying a somewhat abstract state (sorted by color) to a very real state (pixel positions in an image). I tried a few different versions of this and eventually settled on the zipper effect to drive home the point that we are matching some sort of value. The whole middle section of the process essentially shows a "thing" (the column) being taken apart, altered, and then put back together. This zippering motion for "putting the column back together" created a really nice visual idiom that I think should be fairly intuitive. Again, check this out in action to get a good idea of the process.

covering the full range of blue values. However, this doesn't look anything like the original column on the right!

The final step here is to put the blue pixels back in the right order for the image.



We did the same thing for the red and green channels and now it looks like the colors are matched up again!

Evaluation

Overall, I think this was a great project. I love the final site, even though it is far from complete in my mind. It does what I set out to do, but through the course of the project, I had so many more ideas that I would like to implement. One of the main things I want to implement is linked-hover selection on pixels. I want to be able to hover over one of the blue-value points and see what point it corresponds to in the image and in each other visible form of that pixel.

There are still a few bugs, mostly with the scrolling system I came up with. They aren't hard to avoid, but can cause some problems when you scroll too fast. I think the viz really cleanly captures the process of the GLITCH art project and provides a very interesting window into a field that most people never consider to be a form of art. I am super excited about extending my knowledge of D3 to create more of these kind of sites for different projects and algoritms.