Final Project

FA/DATT2400 A - Creative Coding I

Andrea Florez

218524801

12 / 06 / 22

Rough Overview of the project

Throughout this course two of the lessons that really stood out to me were Pixel Manipulations and Object Oriented Programming. For the final assignment I wanted to create something that incorporated both. I also really enjoy making programs that are interactive. So, I decided to create a program that allows a user to interact with pixels in a fun and interactive way.

The final project is basically an image rendered by a custom Pixel class that renders

ellipses as the pixels. The ellipses store the colour of the corresponding pixel to an image. The user can move the mouse around the image and the pixels will move away from the mouse as if they were hit by the mouse. The pixels will eventually return to their original position.

To make the program more interactive, the radius of the mouse position will increase over time if the mouse is pressed down. Resulting in more pixels being pushed away from the mouse.

To demonstrate pixel manipulations even further, you will also see an inverted image of the main image underlying the interactive pixels.

Development process

To develop this program I first had to review how to map pixels from an image to canvas. This was decently tricky because I wanted the image to be smaller than the canvas and also centered. I drew out how I wanted it to look on paper with rough measurements to help me figure out the offsets that I would need. Once I was able to map the image onto the canvas, I started to play around with the pixel colours. I liked how it looked inverted and inverting pixels is relatively easy because all you have to do is subtract the original pixel value for each r, g and b variable from 255. Similar to the code snippet shown below.

```
int c = photo.pixels[loc2];
pixels[loc1] = color(255 - red(c), 255 - green(c), 255 - blue(c));
```

After gaining more confidence playing with the pixels I began to think about my custom Pixel class. I knew I would need each ellipse (or Pixel) to store the colour of the corresponding pixel, and the location of where that pixel should be. So I started creating a simple version of the class that didn't have any moving functionality and would render an ellipse in the position where that pixel should be. I initially wanted to map each pixel of the image to an ellipse. However, that resulted in a very large array that required too much time to render and really slowed down the application. My solution was to only render every third pixel. Reducing the amount of pixels rendered to one third of the amount. Although the image resulting from this looks slightly blurry, the rendering appeared to be a lot more smooth.

To add movement to the Pixel class I had to review PVectors and decide how to bring

them into my class. I wanted the pixels to move away from the mouse as if they had been hit by them. So on each render I need to detect if the pixel is close to the mouse or not. If the mouse is close to the pixel I try to create an acceleration vector that will push the pixel away from the mouse in the direction that the mouse was traveling.

Once I had the pixels moving away from the mouse, I tried to push myself even further and get the pixels to return to their original location. This required me to update my Pixel class to store the initial pixel location when the object was constructed. I also made it so each Pixel stored its own timer. If the object is moved away from the mouse, I store the time and begin counting down. After 2 seconds have passed a try to revert the direction of the acceleration vector to point towards its original location. This took a significant amount of time for me and really stressed my knowledge of PVectors. Eventually I was able to piece together some code that moved the Pixels into the general area of their original position.

Techniques used

The main techniques I used to develop this application were Custom Classes for Object Oriented Programming (i.e: the Pixel Class), PVectors, Arrays and If/Else statements. I had to leverage PVectors to make the Pixels move in the desired direction. Arrays were used as a data structure to store the custom objects. If/Else statements were also used significantly throughout my code, mostly when detecting collisions with the mouse and deciding which direction the Pixel is moving.

Successes and weaknesses

Focusing on successes I think my application looks pretty cool, and I'm very proud of it! I learned a lot during this assignment and had a lot of fun doing it too, so I definitely consider that a success.

Some weaknesses would be the fact that I couldn't find an efficient way to map every single pixel of the image to a custom Pixel object. I'm still not sure how that could be done. Another weakness is with the PVectors themselves. I struggled a lot trying to understand the PVectors and I'm not confident I implemented the movement properly. I'm sure with more time, practice and understanding I could revise the code to handle the PVectors better.

All together though, I'm very satisfied with the result!