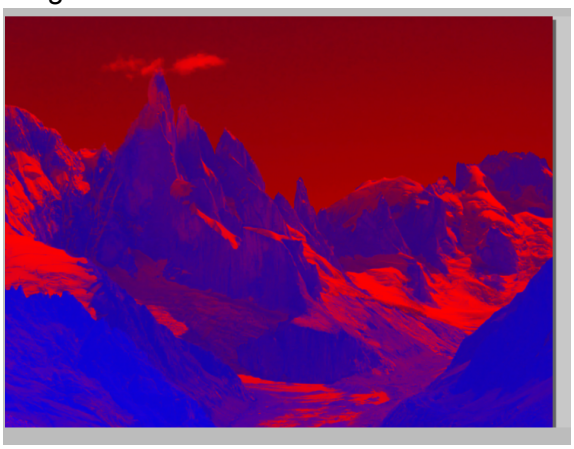
# *OnRamps Digital Manipulation Filters Project*

https://www.openprocessing.org/sketch/1117170

Effect 1

Image:



Code:

```
14          float r,g,b;
15        int indexF1 = 0;
16        while (indexF1 < img.pixels.length)
17            {
18                color cF1 = img.pixels[indexF1];
19
20                r = red  (cF1);
21               g = green(cF1);
22               b = blue (cF1);
23
24                //Add the key word of Or
25            //Check if blue pixel is over 150 or check if
red pixel is less than 200.
26            if ( b > 150 || r < 200)
27            {
28                //Invert my blue and green pixel to create
my new color
29                color nextColor = color(r, g-255, 255-b);
30             }
31             else
32             {
33                 //Invert my red pixel to create new color
34                 color nextColor = color(r-225,g,b);
35              }
36
37                 img.pixels[indexF1] = nextColor;
38               indexF1 = indexF1 + 1;
39            }
40
41        img.updatePixels();
42        image(img, imgOffset, imgOffset);
43 }
```

**Effect 1:** Within the code for effect one, we focused on modifying our image through the use of rgb, as we can see in lines 20,21,and 22.  In line 24 we added a note to define the symbol "||" in line 26 that represents the option or. As noted in line 25, we used an if statement to check if the blue pixel in the image is over 150, and if it was it

would invert the image to create a color. Although, since we also included an or statement, it would do the first task or it would check if the red pixel is less than 200. If the statements were true, the code would invert the blue and green pixel to create a new color.Although, we added an else statement if it was not true so it would invert the red pixel to then create a new color.

Effect 2

Image:

Code:

```
51          float r,g,b;
52        int indexTwo = 0;
53        while (indexTwo < img.pixels.length)
54        {
55          color colorTwo = img.pixels[indexTwo]
56
57              r = red  (colorTwo);
58              g = green(colorTwo);
59              b = blue (colorTwo);
60
61          //Update new color for every 10th pixel
62              if (indexTwo % 10 == 0 || indexTwo % 4
== 0)
63          {
64              //Double the red pixel in value
65              color newColor = color(r*2, g, b);
66           }
67           else
68           {
69               //Change colors for red and blue if not
divisible by 10
70               color newColor = color(r*2, g, b*2);
71            }
72
73               img.pixels[indexTwo] = newColor;
74               indexTwo = indexTwo + 1;
75           }
76
77        img.updatePixels();
78        image(img, imgOffset, imgOffset);
79 }
```

**<u>Effect 2</u>:** In this second effect we also coded in order to change up the rgb using if and else statements. We wanted to update the color to a new one and we created statements that would use the pixels. For this pixel we picked the value of 10 % which would translate to every 10th pixel. Now our first statement says that if it is divisible by 10 then every 10th pixel would change color. For this first statement the 10th pixel

would change to the rgb being r*2, g, and b. Technically the value for red doubles. Now for the "else" statement, if it is not divisible by 10 then the rgb changes to something else. the new color will be r*2, g, b*2. The red and blue doubles. As we can see with this code our image is not divisible by 10 therefore it picked the second one in which the red and blue pixel double in value choosing the second new color.

Effect 3

Image:

Code:

```
89          //****flipped image****
90      color[] tempPixels = new color[img.width *
img.height]; // create a temporary pixel array
91      int index = 0;
92      while (index < img.pixels.length)
93      {
94          tempPixels[index] = img.pixels[index];
95          index = index + 1;
96      }
97      index = 0;
98      while (index < img.pixels.length)
99      {
100         if (index < img.pixels.length*3/5 &&
index>img.pixels.length*2/5)
101         {
102             img.pixels[index] =
tempPixels[img.pixels.length-1-index];
103         }
104         if (index > img.pixels.length*4/5 && index
< img.pixels.length*5/5)
105         {
106             img.pixels[index] =
tempPixels[img.pixels.length-1-index];
107         }
108             index = index + 1;
109     }
110
111     //****mirror image****
112     int flippedIndex;
113     index = 0;
114     while (index < img.pixels.length)
115     {
116         int x = index % img.width;
117         int y = int(index/img.width);
118
119         if (index < img.pixels.length*2/5 && index
> img.pixels.length*1/5)
120         {
121             flippedIndex = y*img.width + (img.width
- 1 - x);
122             img.pixels[index-img.width/2] =
```

```
tempPixels[flippedIndex];
123             }
124             if (index < img.pixels.length*4/5 && index
> img.pixels.length*3/5)
125             {
126               flippedIndex = y*img.width + (img.width
- 1 - x);
127                img.pixels[index-img.width/2] =
tempPixels[flippedIndex];
128               }
129               index = index + 1;
130     }
131
132     img.updatePixels();
133     image(img, imgOffset, imgOffset);
134 }
```

**Effect 3:** For the last effect we mirrored, flipped, and shifted certain parts of our image. We made sure to divide our image into five sections so we used the fractions ranging from ⅕ through 5/5. We made statements that if our index was either less than or equal to, depending on the statement it would choose that part and either mirror or flip. We chose the part of ⅖ and ⅘ to be mirrored and ⅗ and 5/5 flipped. The statements were made that if the index was between ⅕ and ⅖ or ⅗ and ⅘ then it would get mirrored. Now if it was between ⅖ and ⅗ or ⅘ and 5/5 it will get flipped. In order to shift our image in lines 122 and 127 we subtracted the width of the image divided by 2 in order to slice the image and shift it to the right.