Generative Design – Manipulating Images

Step 1

'use strict';

​

var img;

​

function preload() { // function runs and code runs after the image is loaded

  img = loadImage('data/pic1.jpg');

}

​

function setup() {

  createCanvas(600, 600);

  noCursor();

  noStroke();

​

  // Useful function to stop loop function of the draw function

  noLoop();

}

​

function draw() {

  // Loads the image into the pixel[] array

  img.loadPixels();

  console.log(img.pixels[0]); // Retrieves the information of the first color in the pixel array

​

  console.log(img.pixels);

}

Step 2

'use strict';

​

var img;

// Create an empty colors array to store your color values

var colors = [];

​

function preload() { // function runs and code runs after the image is loaded

  img = loadImage('data/pic1.jpg');

}

​

function setup() {

  createCanvas(600, 600);

  noCursor();

  noStroke();

​

  // Useful function to stop loop function of the draw function

  noLoop();

}

​

function draw() {

  // Defines number of tiles

  var tileCount = 2;

​

  // and the width of each tile

  var rectSize = width / tileCount;

​

  img.loadPixels();

  // Empty our colors array each loop

  colors = [];

​

  // If our image is 400px wide and high and our tile count is 20

  // which will mean our tile width is also 20px. We will want to grab

  // the colors at 0, 20, 40, 60 .. and store them

​

  for (var gridY = 0; gridY < tileCount; gridY++) {

    // For each gridX value we need to work out a color to be stored

    for (var gridX = 0; gridX < tileCount; gridX++) {

      var px = int(gridX \* rectSize); // pixel value in x location

      var py = int(gridY \* rectSize); // pixel value in y location

​

      // Convert this to the appropriate index value in the pixel array

      var i = (py \* img.width + px) \* 4;

      // find the hsba locations meaning the color of the pixel and create color object

      var c = color(img.pixels[i], img.pixels[i + 1], img.pixels[i + 2], img.pixels[i + 3]);

      // push that color to an array

      colors.push(c);

    }

  }

  console.log(colors);

}

​

​

Step 3

​