"Women think of all colors except the absence of color. I have said that black has it all. White too. Their beauty is absolute. It is the perfect harmony."

Chapter 00: Introduction

Color, "the property possessed by an object of producing different sensations on the eye as a result of the way the object reflects or emits light"--according to the Dictionary app on my laptop. Here's another: "one, or any mixture, of the constituents into which light can be separated in a spectrum or rainbow, sometimes including (loosely) black and white". Whatever that means.

It is believed that color 'ought' to be separate from language, and that the more we try to define or describe color the further we are from truly experiencing it. When I started writing this book it was never my intention to emerge with a definition of what color is--rather, I was interested in understanding color and seeing whether the things we see with our eyes can be reproduced systematically. Methodically. Algorithmically. But the more I tried to develop these algorithms, the more I realized that describing color was exactly what I was trying to do.

I am at a stage in my life where I am still trying to understand how things in the world work, still trying to figure out who I am, still suffering from the occasional existential crisis. Since I am probably unqualified to articulate something on behalf of other people, this book is not about how color works for everyone. This is about how color works for me.

This is The Colorist's Cookbook.

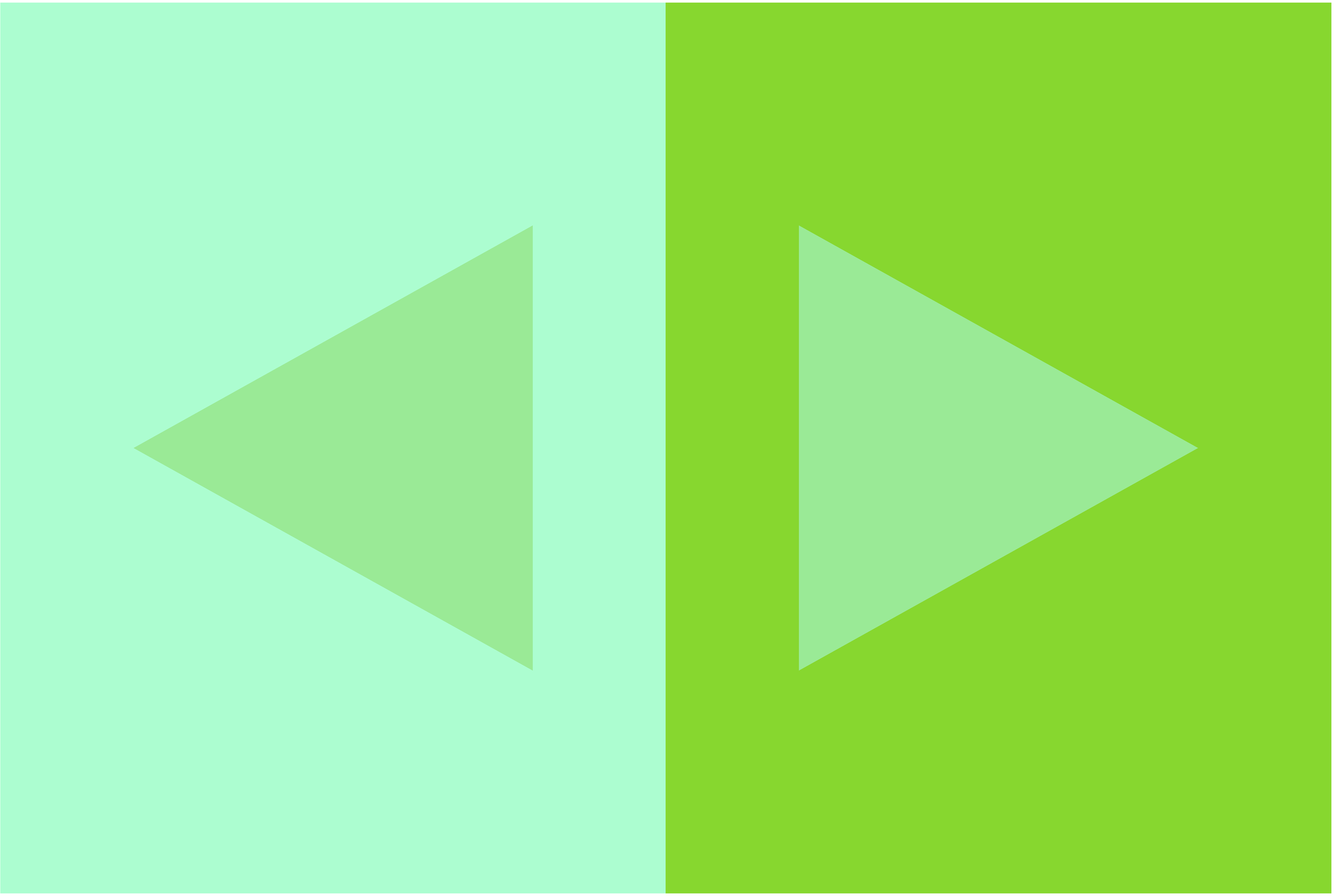
Chapter 01: 3 Colors Into 4 Part 1

Remember the dress that turned into a viral phenomenon? Not that thing Lady Gaga wore to the VMA's that one year--but the one that got famous just because people couldn't decide whether it was black and blue or white and gold? In February 2015, a woman took a picture of a dress she wanted to wear to a wedding, and released the photo on Tumblr after many of her friends disagreed over the color. The picture spread quickly across the Internet, and sparked an intense public debate about the color of the dress. It turns out that this phenomenon can be explained by the pure science of color vision. When scientists pitched in to provide insight into what was going on they found that if the dress was shown in yellow lighting the majority of people would see it as blue and black, while lighting with a blue bias caused people to view the dress as white and gold.*

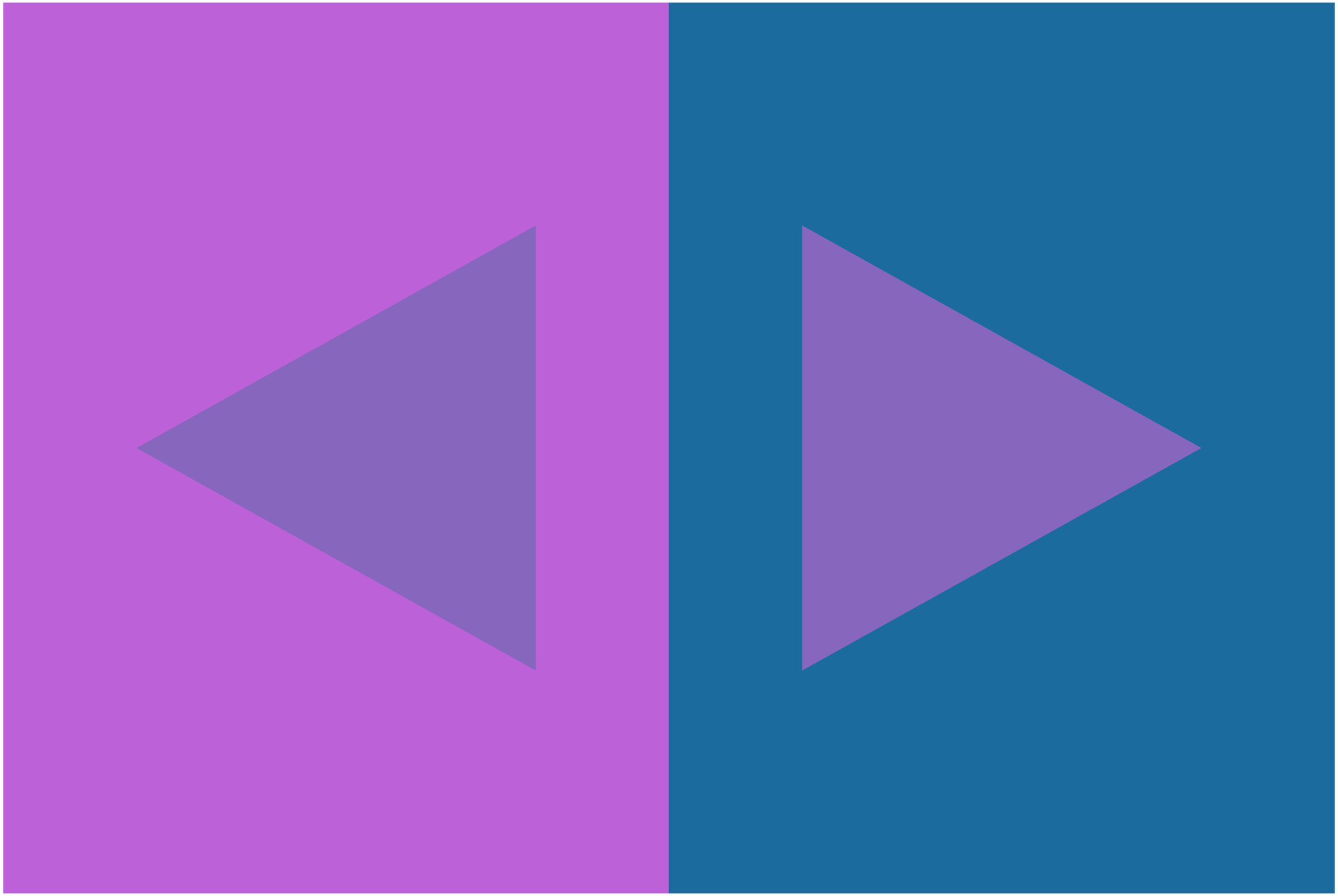
According to color theory, having a color in the background removes that color from the foreground--this is why the dress shown against yellow lighting would make it look less yellow and more blue. I won't get into the specifics of the science behind this because I'm running out of space on this page and frankly I don't fully understand it myself--but this is a very important and interesting concept in color theory that is the basis of many optical illusions. (It also really f**ked me up and made me question whether I was worthy of having perfect color vision.) This chapter is about using this very concept to make 3 colors look like 4. I hope it causes you to lose as much faith in your eyeballs as I did.

*Look up #thedress or Dressgate if my summary displeased you and you want to find out more

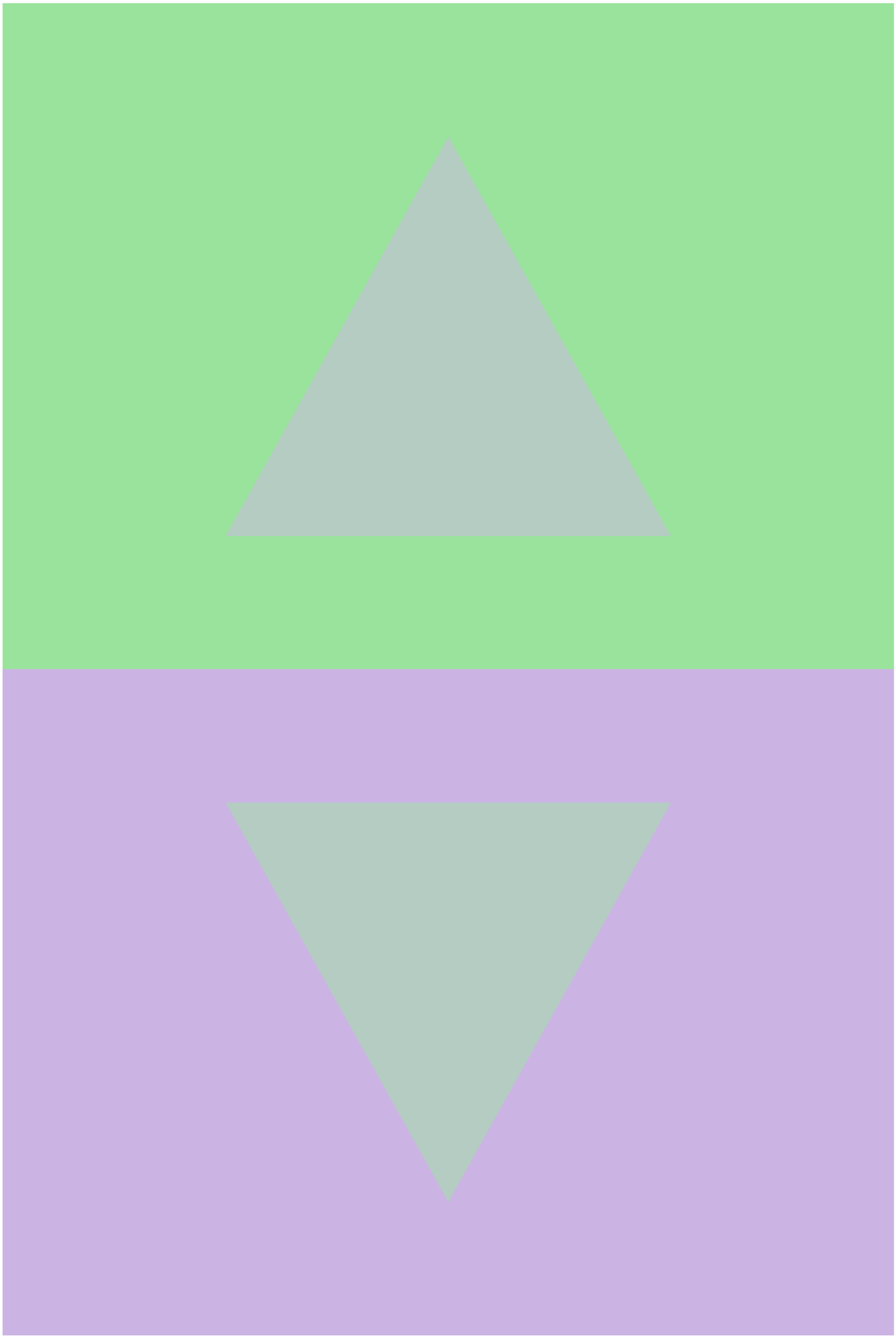
0xFFACFDD0 0xFF87D72F 0xFF9AEA96



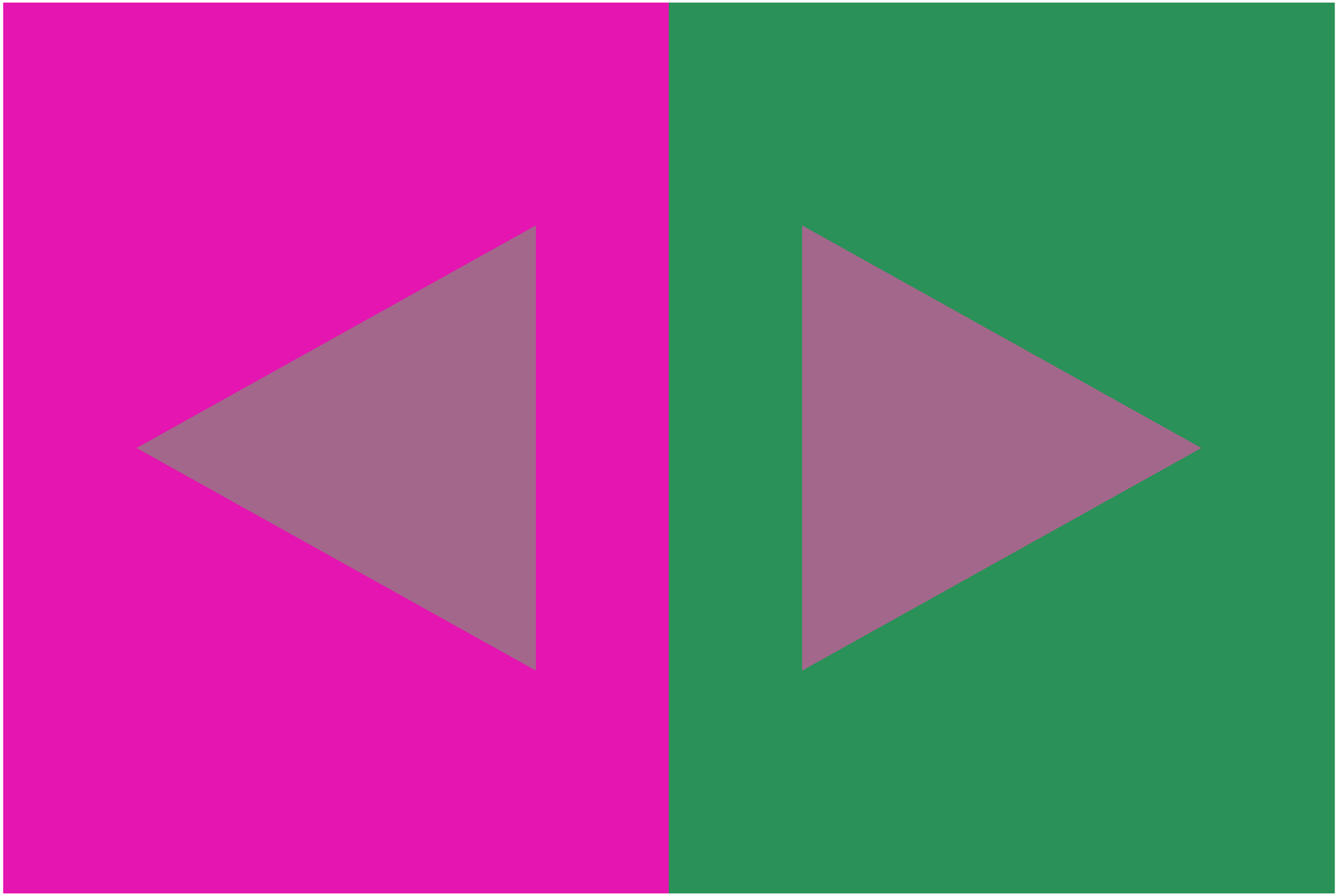
0xFFBD61D9 0xFF1C6B9E 0xFF8766BD



0xFF9AE39C 0xFFCBB4E3 0xFFB4CCC2



0xFFE415B0 0xFF2A9158 0xFFA3678B



```
// recipe for making 3 colors look like 4

// prepare the first color
SecureRandom random = new SecureRandom();

int min = 0;
int max = 255;
int r1 = random.nextInt(max-min+1)+min;
int g1 = random.nextInt(max-min+1)+min;
int b1 = random.nextInt(max-min+1)+min;
color col1 = color(r1, g1, b1);

// then prepare the second color
int r2 = random.nextInt(max-min+1)+min;
int g2 = random.nextInt(max-min+1)+min;
int b2 = random.nextInt(max-min+1)+min;
color col2 = color(r2, g2, b2);
// evenly mix the first two colors to create
// the 'middle' color
float mixedred = sqrt((sq(red(col1))*0.5 +sq(red(col2))*0.5));
float mixedgreen = sqrt((sq(green(col1))*0.5 +sq(green(col2))*0.5));
float mixedblue = sqrt((sq(blue(col1))*0.5 +sq(blue(col2))*0.5));

color mid = color(mixedred, mixedgreen, mixedblue);
// pre-translate the transformation matrix
// to the size of your margins
pushMatrix();
translate(margin, margin);
```

```
// arrange the first two colors beside each other
fill(col1);
stroke(col1);
rect(0,0,pgwidth/2f,pgheight);
fill(col2);
stroke(col2);
rect(pgwidth/2f,0,pgwidth/2f,pgheight);

// top with the middle color
fill(mid);
noStroke();

triangle(pgwidth*0.1,pgheight/2f,pgwidth*0.4,pgheight/4f, pgwidth*0.4,pgheight*0.75);
triangle(pgwidth*0.9,pgheight/2f,pgwidth*0.6,pgheight/4f, pgwidth*0.6,pgheight*0.75);

popMatrix();
```

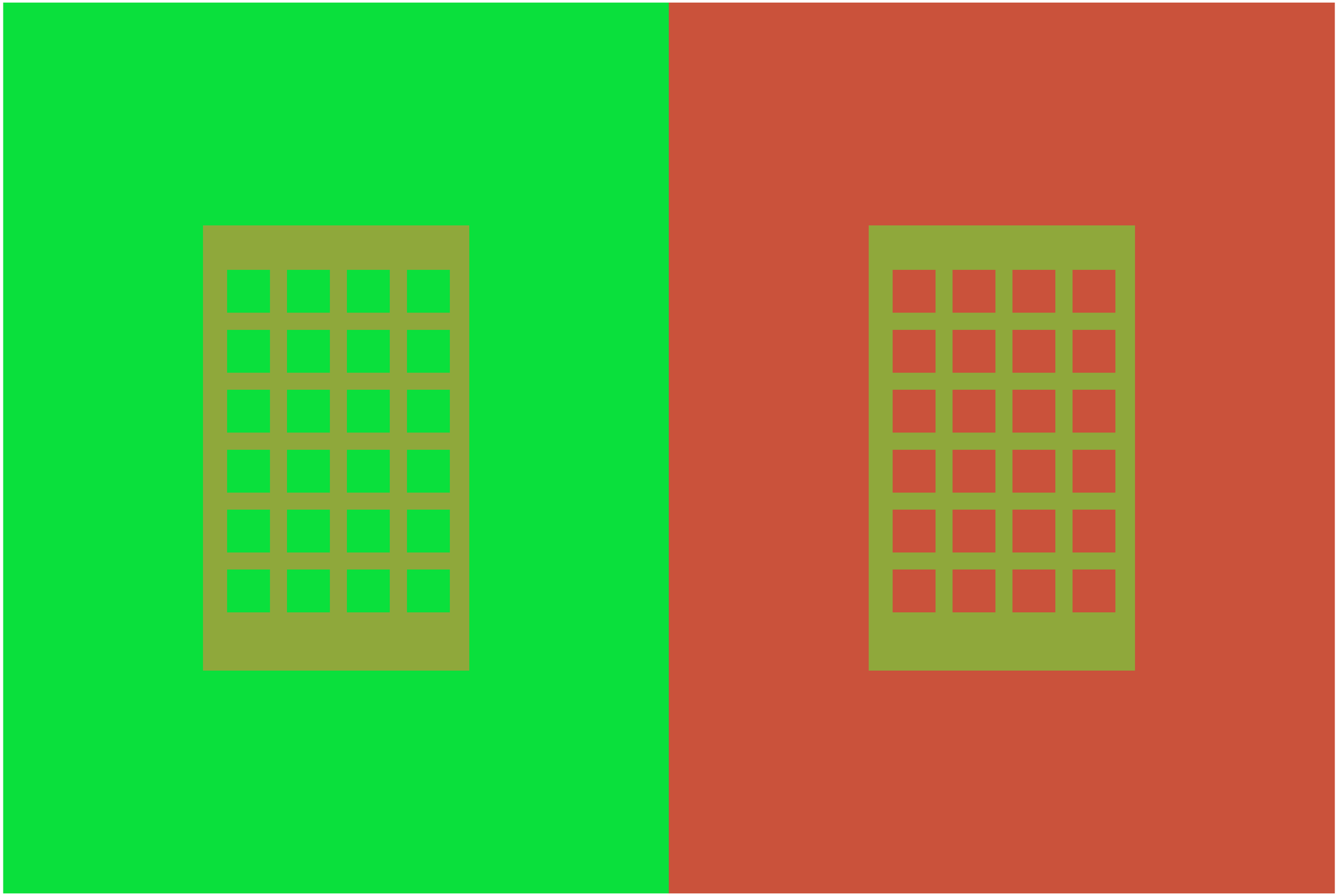
Chapter 02: 3 Colors Into 4 Part II

If you've been keeping up with the code (which you should--I painstakingly included it for a reason!), you could probably observe that everything boils down to just a bunch of math. This is because in computerland, colors (and all other things) are just numbers. Consider this: since every color is expressed as a number, then for a encoding scheme with 6 hexadecimal* numbers the value of colors ranges from 0 to 16,777,215--that gives us a total of 16,777,216 possible colors! Does that mean there are 16 million colors that exist in the world?

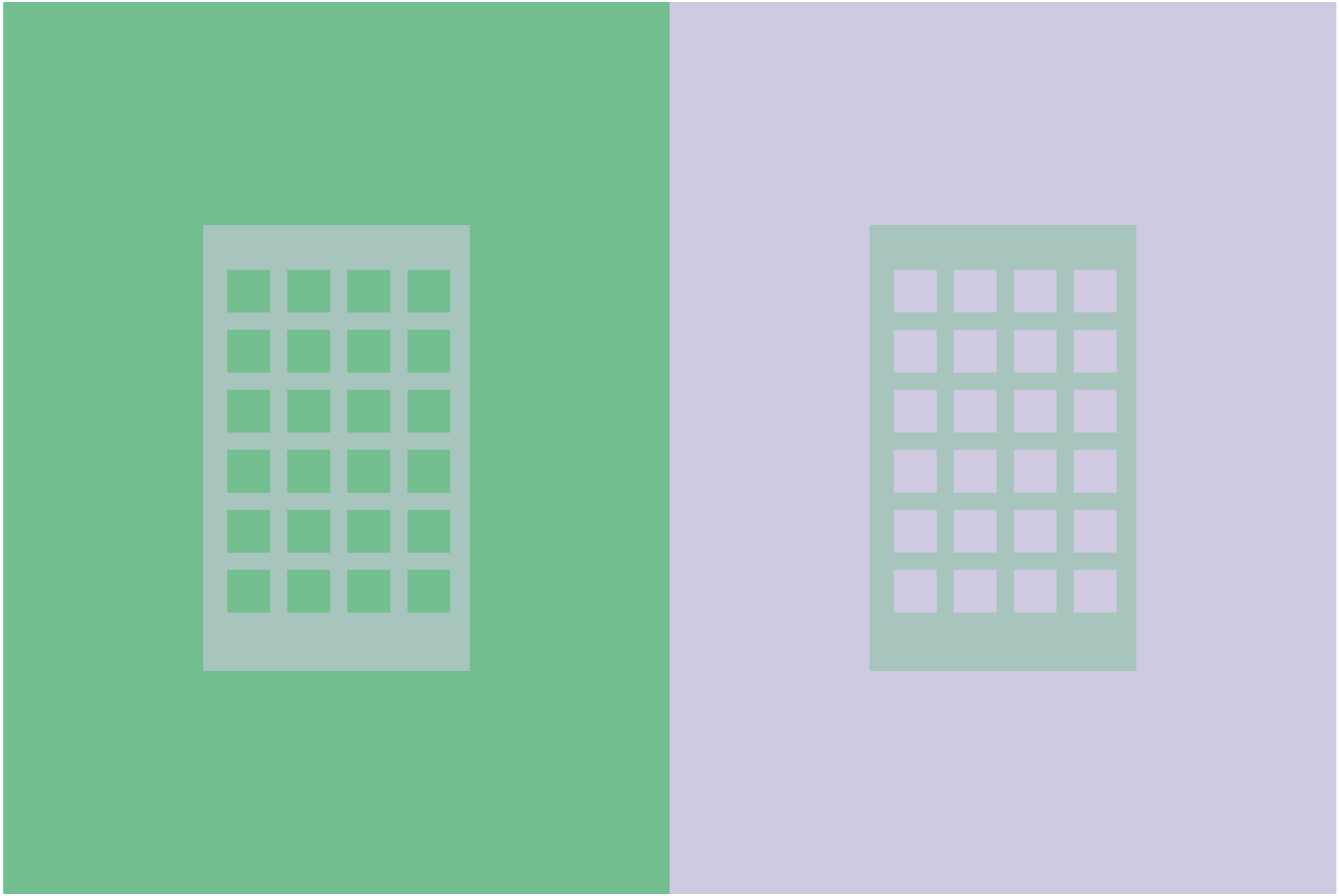
Well...no. It turns out that humans, on average, can only see about 10 million colors in a single viewing condition--so computers really encode 6 million more colors than necessary. So is the answer 10 million, then? Remember that the 10 million refers to the number of colors we can see in a SINGLE viewing condition. How many possible viewing conditions are there? And how do we take into account that color perception can differ from one person to another? That means the real answer is infinity. Now the question is: is the number of colors that exist in the world countably infinite, or uncountably infinite?** I suck at discrete math, so maybe someone else can answer this question for me.

*this is just a way of saying base-16, which is a very common numerical system for computers. Normal people think in base-10.
**for those who are not familiar with these terms: in mathland there are two infinities, the uncountable and countable. If that isn't weird enough, it has also been proven that the uncountable infinity is 'bigger' than the countable infinity.

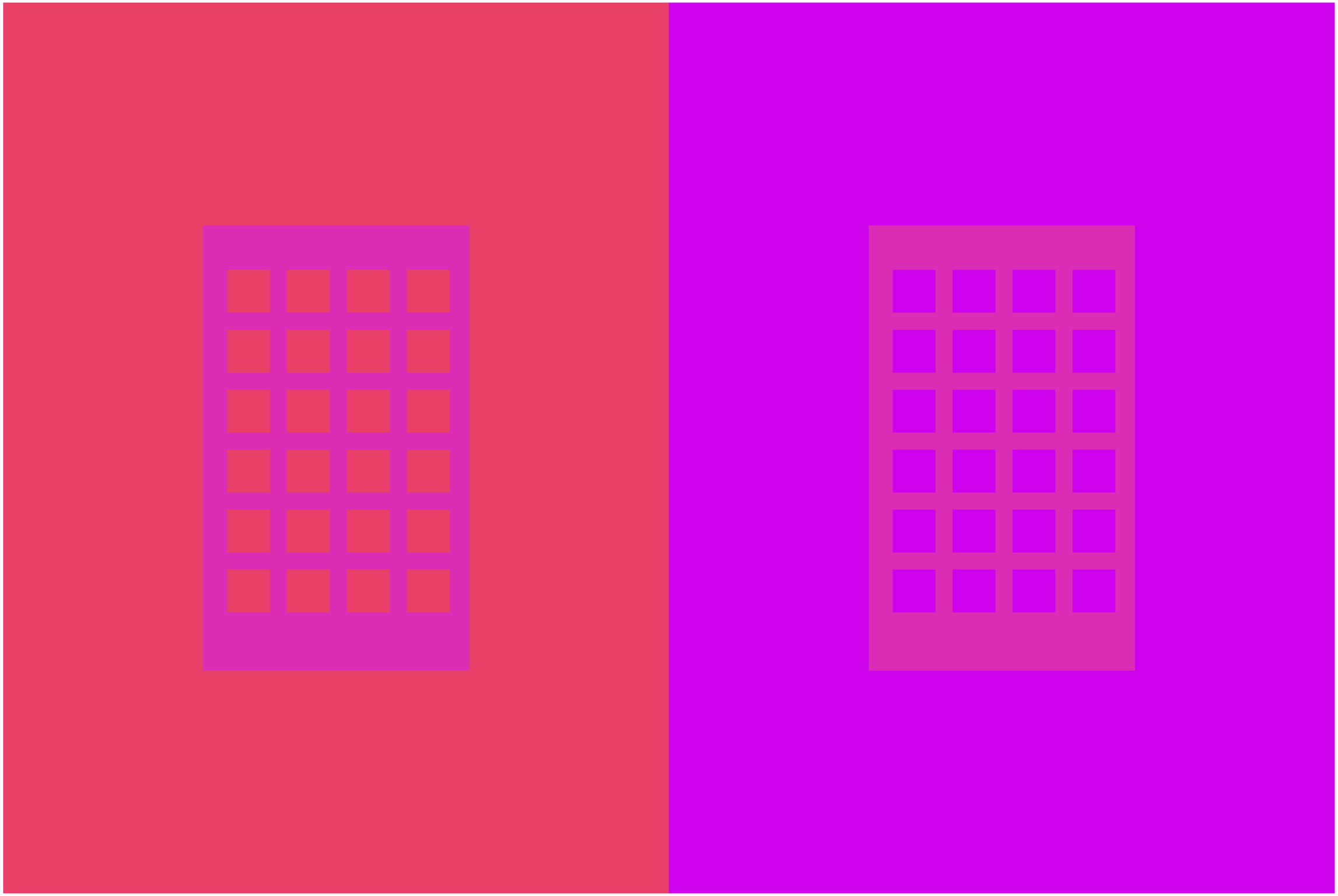
0xFF0AE03C 0xFFCA523B 0xFF8FA83B



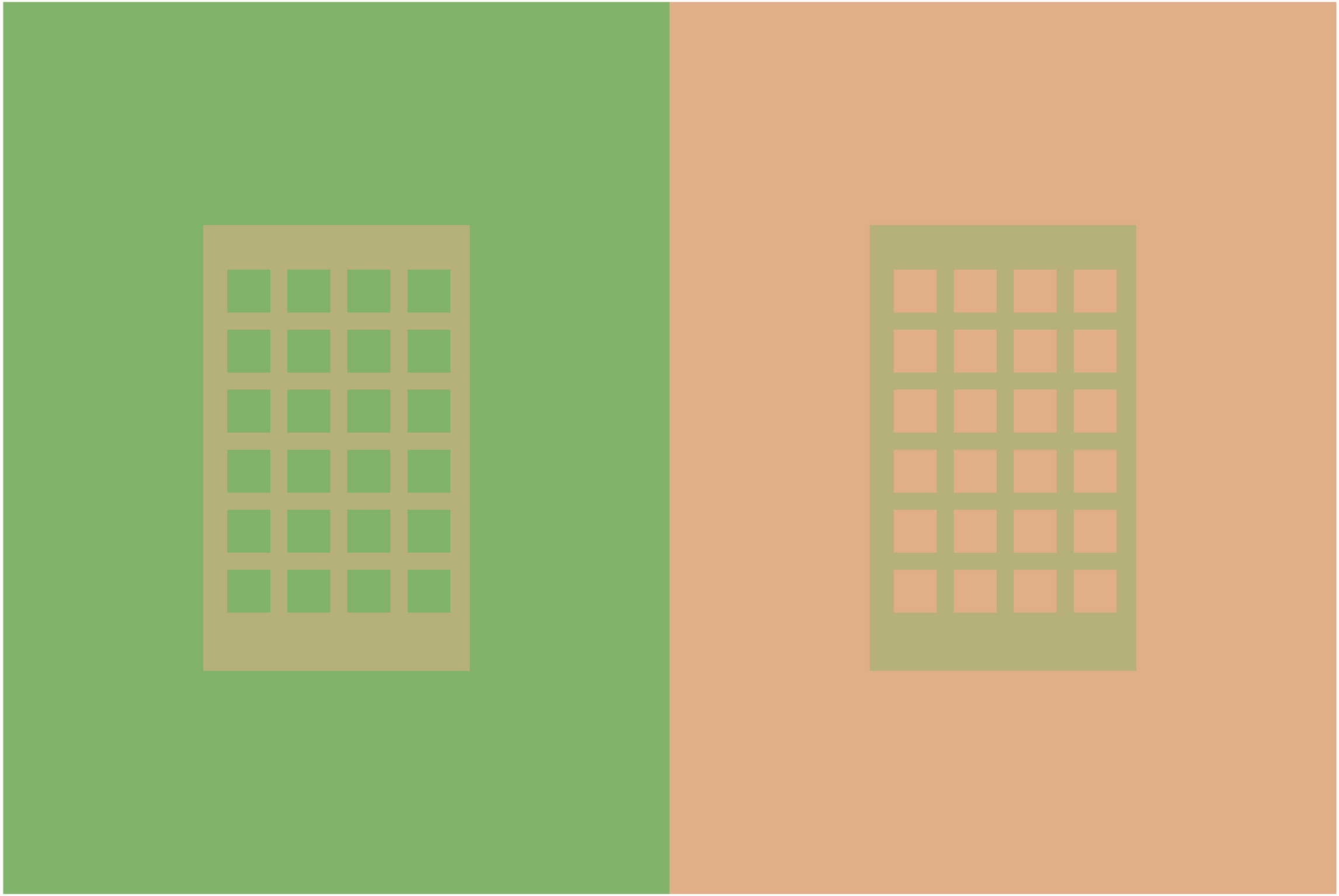
0xFF73BF8F 0xFFCFC9E2 0xFFA7C4BD



0xFFE84067 0xFFCE01ED 0xFFDB2DB6



0xFF80B369 0xFFE0AE89 0xFFB6B07A



```
// recipe for making 3 colors look like 4...with holes!

// prepare the first color
SecureRandom random = new SecureRandom();

int min = 0;
int max = 255;
int r1 = random.nextInt(max-min+1)+min;
int g1 = random.nextInt(max-min+1)+min;
int b1 = random.nextInt(max-min+1)+min;
color col1 = color(r1, g1, b1);

// then prepare the second color
int r2 = random.nextInt(max-min+1)+min;
int g2 = random.nextInt(max-min+1)+min;
int b2 = random.nextInt(max-min+1)+min;
color col2 = color(r2, g2, b2);
// evenly mix the first two colors to create
// the 'middle' color
float mixedred = sqrt((sq(red(col1))*0.5 +sq(red(col2))*0.5));
float mixedgreen = sqrt((sq(green(col1))*0.5 +sq(green(col2))*0.5));
float mixedblue = sqrt((sq(blue(col1))*0.5 +sq(blue(col2))*0.5));

color mid = color(mixedred, mixedgreen, mixedblue);

// pre-translate the transformation matrix
// to the size of your margins
pushMatrix();
translate(margin, margin);
```

```
rectMode(CORNER);

// arrange first two colors beside each other
fill(col1);
stroke(col1);
rect(0,0,pgwidth/2f,pgheight);
fill(col2);
stroke(col2);
rect(pgwidth/2f,0,pgwidth/2f,pgheight);

// top with the middle color
rectMode(CENTER);
fill(mid);
noStroke();
rect((pgwidth)/4f,pgheight/2f,pgwidth/5f,pgheight/2f);
rect((pgwidth*3f)/4f,pgheight/2f,pgwidth/5f,pgheight/2f);

// slice holes in the middle for a more dramatic effect
// waffle aesthetic
rectMode(CORNER);
fill(col1);
for(int rows = 0; rows < 6; rows++) {
  for(int cols = 0; cols < 4; cols++) {
    rect(pgwidth * 0.168 + 140*cols, pgheight * 0.3 + 140*rows,100,100);
  }
}

fill(col2);
float rand3 = random(-350,350);
```

```
pushMatrix();

for(int rows = 0; rows < 6; rows++) {
  for(int cols = 0; cols < 4; cols++) {
    rect(pgwidth * 0.668 + 140*cols, pgheight * 0.3 + 140*rows,100,100);
  }
}
popMatrix();

popMatrix();
```

Chapter 03: 4 Colors Into 3

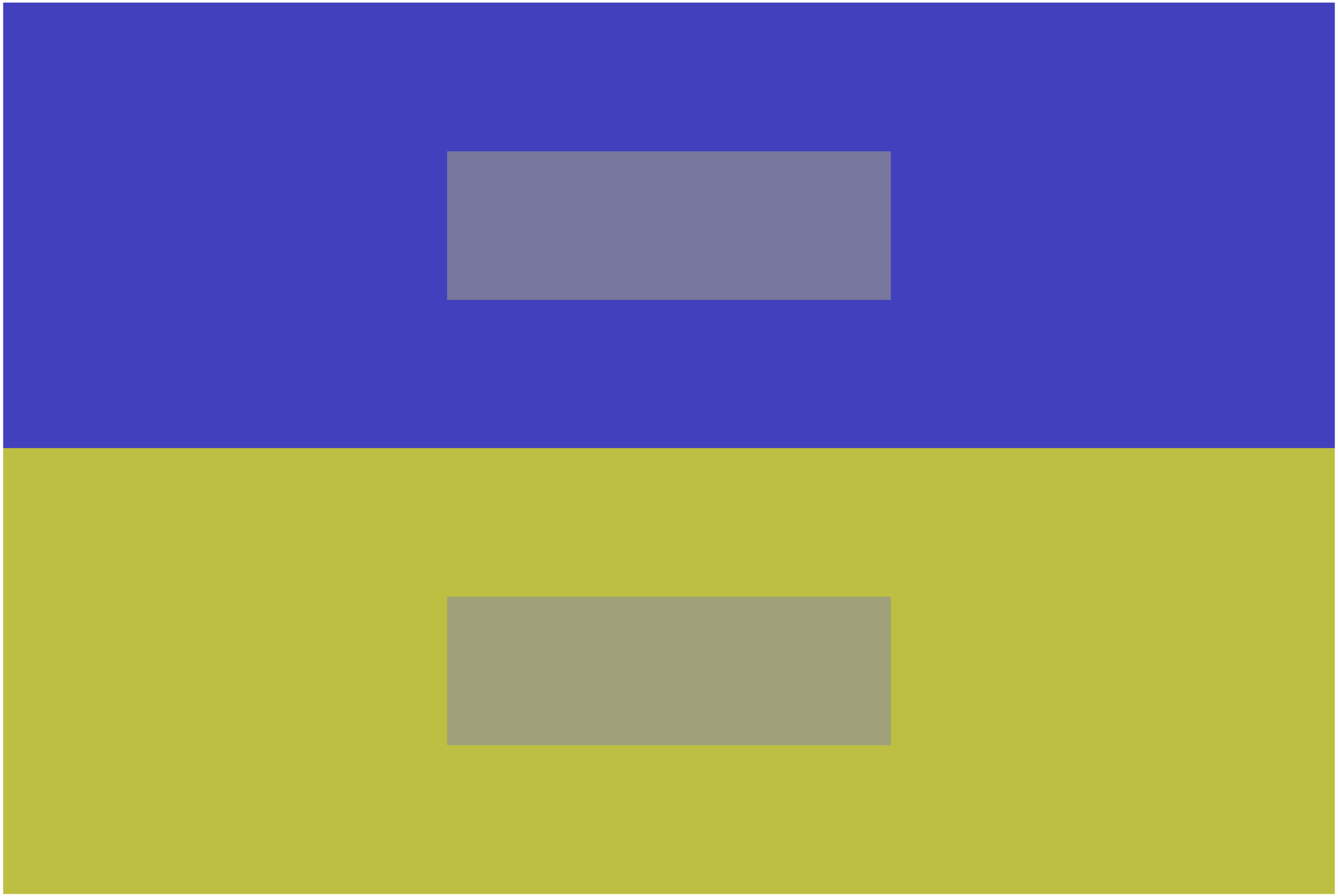
A guy named Pablo Picasso once said, "Why do two colors, put one next to the other, sing? Can one really explain this? No. Just as one can never learn how to paint." I'm sorry Pab, but I'm going to have to disagree with you on this one because:

1) I started learning how to paint the moment I taught myself how to hold a brush. I'm still learning.

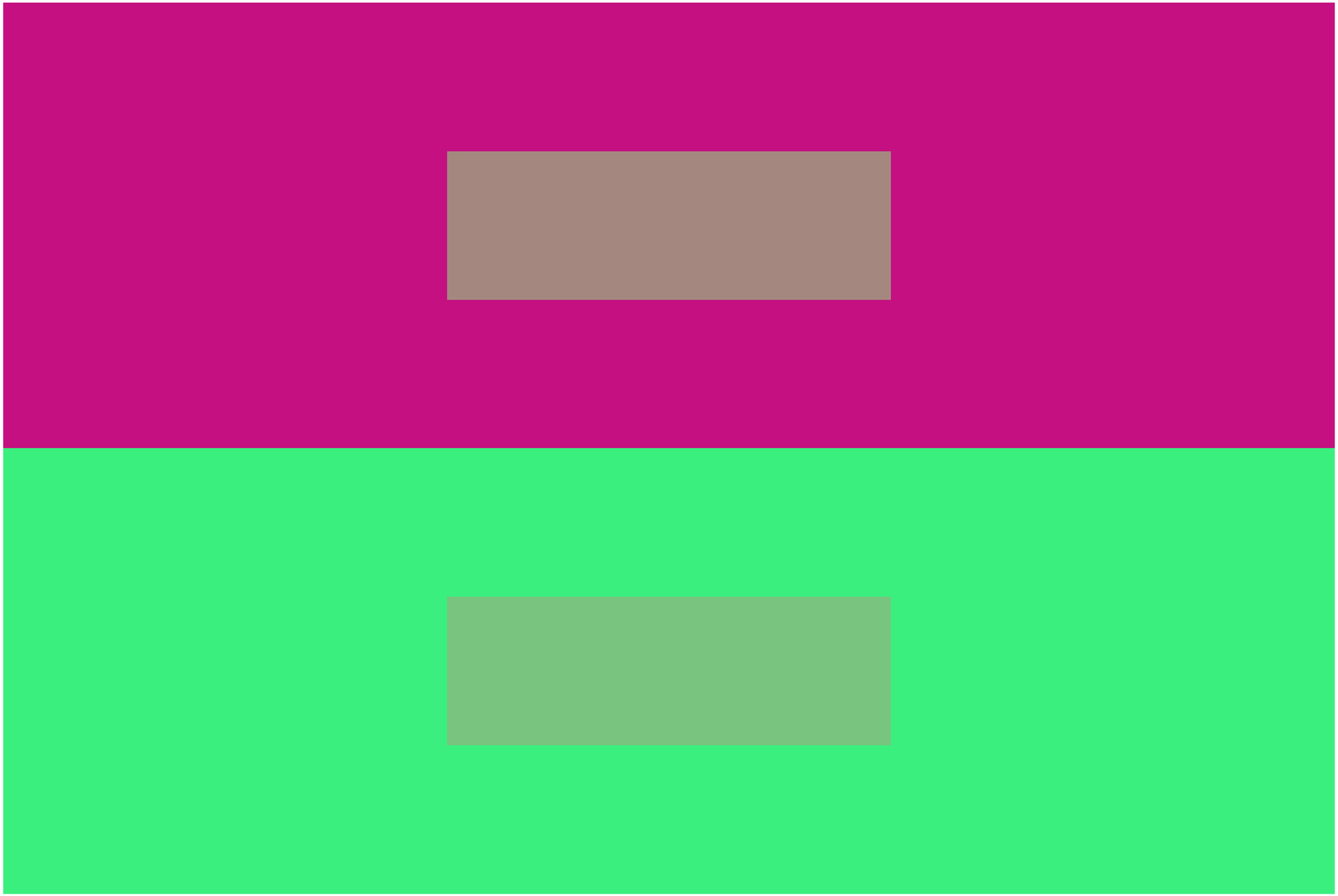
2) Although it's true that color harmony is complex and often tricky to articulate, it is not entirely inexplicable. Color theorists are better equipped to answer this than I am, but one reason why two colors may 'sing' when placed next to each other is because aesthetic responses can be evoked by complementary hues that are juxtaposed against each other, or analogous hues that are meshed with each other.

The logician in me believes that we can always find a reasonable explanation for almost all observable phenomena. The philosopher in me believes that, without the aid of some kind of 'oracle', there is no way for us to really confirm that our explanations are reasonable. I guess this is why I decided to write this book--because I wanted to see if I could prove my selves wrong (and therefore right) by turning the logician into a philosopher, and the philosopher into a logician.

0xFF4240BD 0xFFBDBF42 0xFF78789F 0xFF9FA078



0xFFC41080 0xFF3BEF7F 0xFFA4887F 0xFF79C47F



0xFF97BC29 0xFF6843D6 0xFF899F7E 0xFF7878B1

```
// recipe for making 4 colors look like 3
// prepare the first and second colors
SecureRandom random = new SecureRandom();

int min = 0;
int max = 255;
int r1 = random.nextInt(max-min+1)+min;
int g1 = random.nextInt(max-min+1)+min;
int b1 = random.nextInt(max-min+1)+min;
color col1 = color(r1, g1, b1);

// the second color is opposite from the first color
color col2 = color(255 - red(col1),
                   255 - green(col1),
                   255 - blue(col1));

// evenly mix the first two colors to create
// the 'middle' color
float mixedred = sqrt((sq(red(col1))*0.5 +sq(red(col2))*0.5));
float mixedgreen = sqrt((sq(green(col1))*0.5 +sq(green(col2))*0.5));
float mixedblue = sqrt((sq(blue(col1))*0.5 +sq(blue(col2))*0.5));

color mid = color(mixedred, mixedgreen, mixedblue);

float c1_weight = 0.35;
float c2_weight = 0.65;

// calculate a weighted average of the first color and middle color
mixedred = sqrt((sq(red(col1))*c1_weight +sq(red(mid))*c2_weight));
```

```
mixedblue = sqrt((sq(blue(col1))*c1_weight +sq(blue(mid))*c2_weight));

color mixed1 = color(mixedred, mixedgreen, mixedblue);

// calculate a weighted average of the second color and middle color
mixedred = sqrt((sq(red(col2))*c1_weight +sq(red(mid))*c2_weight));
mixedgreen = sqrt((sq(green(col2))*c1_weight +sq(green(mid))*c2_weight));
mixedblue = sqrt((sq(blue(col2))*c1_weight +sq(blue(mid))*c2_weight));

color mixed2 = color(mixedred, mixedgreen, mixedblue);

// pre-translate the transformation matrix
// to the size of your margins
pushMatrix();
translate(margin, margin);

rectMode(CORNER);

// arrange opposing colors beside each other
fill(col1);
stroke(col1);
rect(0,0,pgwidth,pgheight/2);
fill(col2);
stroke(col2);
rect(0,pgheight/2,pgwidth,pgheight/2);

// top with the middle colors
fill(mixed1);
noStroke();
```

```
fill(mixed2);
rect((pgwidth)/3, (pgheight*2)/3, pgwidth/3, pgheight/6);


popMatrix();
```

Chapter 04: 5 Colors Into 3

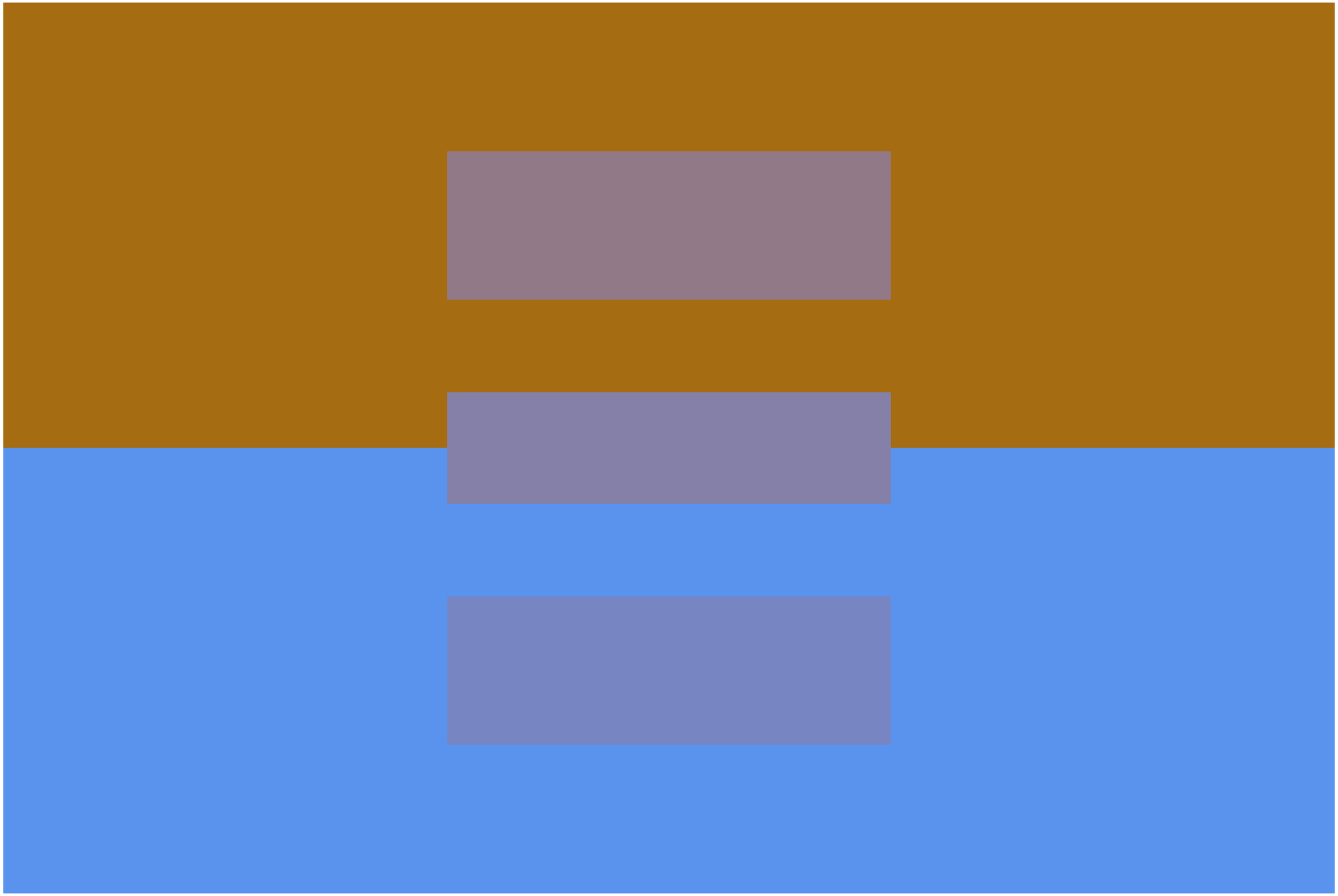
So I probably should be saying something about the next chapter, but I want to tell you a story instead. Last Tuesday, I had to go to my professor's office hours to prepare for an exam. My friend and I made our way to the back corner of the room and sat down, taking out our notes and note-taking instruments and whatever else people possess when they need to study. As more students shuffled into the small office with questions about the material, the afternoon proceeded unremarkably. Some time passed, and as the questions began to slow down I started to have a fairly unremarkable conversation with my friend about Windows computers:

"Oh, you're a Windows person?" I asked him.
"Yeah," he replied.
"Okay, I'm not judging," I lied.
And then:
"I am," chimed in my professor. "Macs are better."

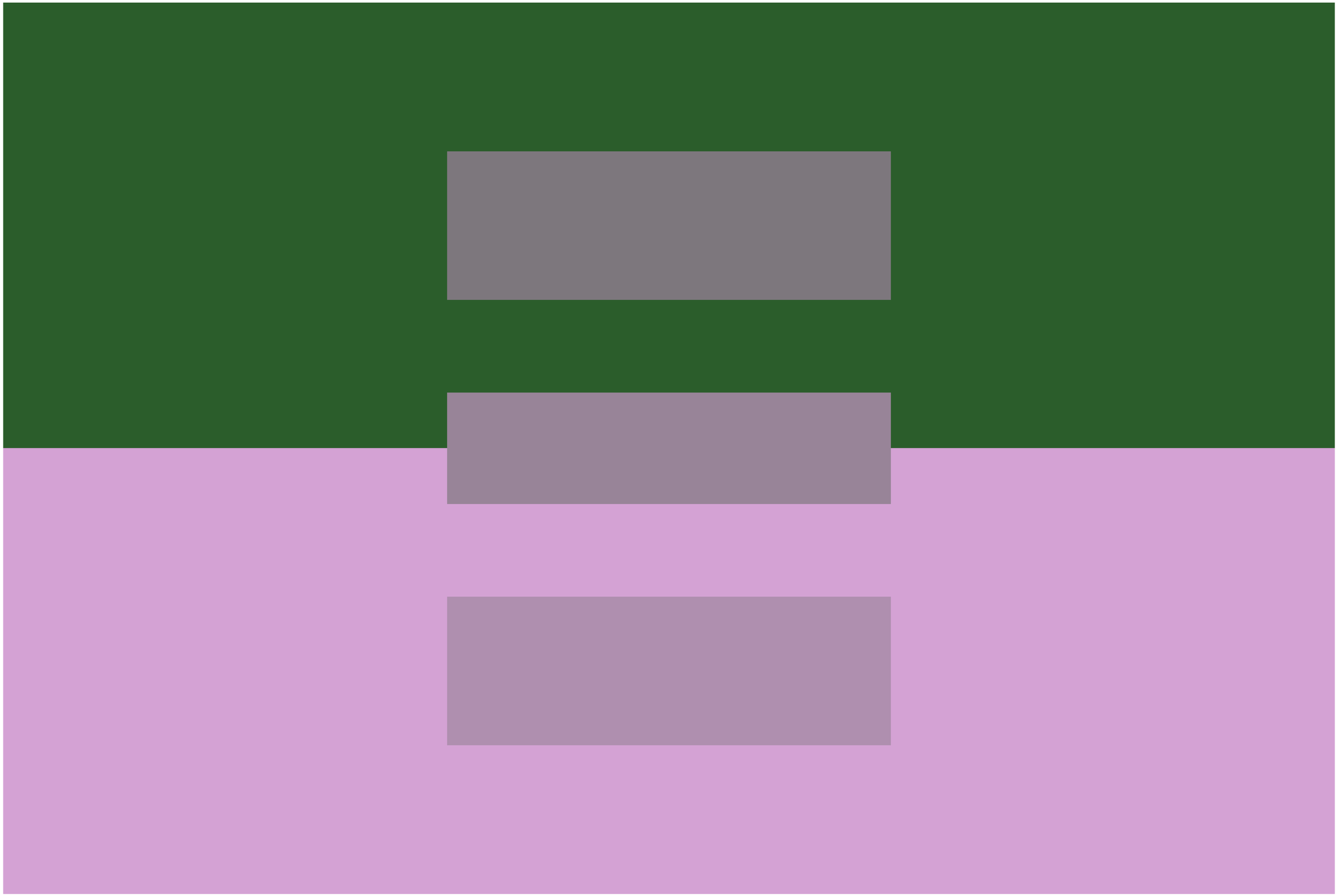
And that's when the room was thrown into a passive-aggressive debate about whether Macs were better than Windows machines (or rather, why Windows machines were better than Macs).

Now I have a confession to make: for the majority of public situations in which I am surrounded by people who are some combination of scientists/engineers/mathematicians, I generally avoid declaring myself as an art student--not out of shame but as a precautionary measure to avoid hostility and having metaphorical daggers thrown at my back. But for whatever reason, on that unremarkable Tuesday, during that unremarkable conversation, it was exactly what I did.

0xFFA66C12 0xFF5993ED 0xFF917987 0xFF8580A8 0xFF7786C2



0xFF2B5D2B 0xFFD4A2D4 0xFF7D777D 0xFF988498 0xFFAF8FAF



0xFF8A4CD6 0xFF75B329 0xFF8277B1 0xFF7F899A 0xFF7B997E

```
// recipe for making 5 colors look like 3
// prepare the first and second colors
SecureRandom random = new SecureRandom();

int min = 0;
int max = 255;
int r1 = random.nextInt(max-min+1)+min;
int g1 = random.nextInt(max-min+1)+min;
int b1 = random.nextInt(max-min+1)+min;
color col1 = color(r1, g1, b1);

// the second color is opposite from the first color
color col2 = color(255 - red(col1),
                   255 - green(col1),
                   255 - blue(col1));

// evenly mix the first two colors to create
// the 'middle' color
float mixedred = sqrt((sq(red(col1))*0.5 +sq(red(col2))*0.5));
float mixedgreen = sqrt((sq(green(col1))*0.5 +sq(green(col2))*0.5));
float mixedblue = sqrt((sq(blue(col1))*0.5 +sq(blue(col2))*0.5));

color mid = color(mixedred, mixedgreen, mixedblue);

float c1_weight = 0.35;
float c2_weight = 0.65;

// calculate a weighted average of the first color and middle color
mixedred = sqrt((sq(red(col1))*c1_weight +sq(red(mid))*c2_weight));
```

```
mixedblue = sqrt((sq(blue(col1))*c1_weight +sq(blue(mid))*c2_weight));

color mixed1 = color(mixedred, mixedgreen, mixedblue);

// calculate a weighted average of the second color and middle color
mixedred = sqrt((sq(red(col2))*c1_weight +sq(red(mid))*c2_weight));
mixedgreen = sqrt((sq(green(col2))*c1_weight +sq(green(mid))*c2_weight));
mixedblue = sqrt((sq(blue(col2))*c1_weight +sq(blue(mid))*c2_weight));

color mixed2 = color(mixedred, mixedgreen, mixedblue);

// pre-translate the transformation matrix
// to the size of your margins
pushMatrix();
translate(margin, margin);

rectMode(CORNER);

// arrange opposing colors beside each other
fill(col1);
stroke(col1);
rect(0,0,pgwidth,pgheight/2);
fill(col2);
stroke(col2);
rect(0,pgheight/2,pgwidth,pgheight/2);

// top with the middle colors
fill(mixed1);
noStroke();
```

```
fill(mixed2);
rect((pgwidth)/3,  (pgheight*2)/3, pgwidth/3, pgheight/6);

fill(mid);
rect(pgwidth/3, pgheight*0.4375, pgwidth/3, pgheight/8);

popMatrix();
```

Chapter 05: Color Modulation

"I think as an art person," I started boldly. "I am naturally inclined to like Macs more," I could feel the daggers digging into my spine. "It's just--" as the daggers made their way deeper into my skin, I found myself helplessly fumbling for words--"I don't know, the colors are just...nicer?"

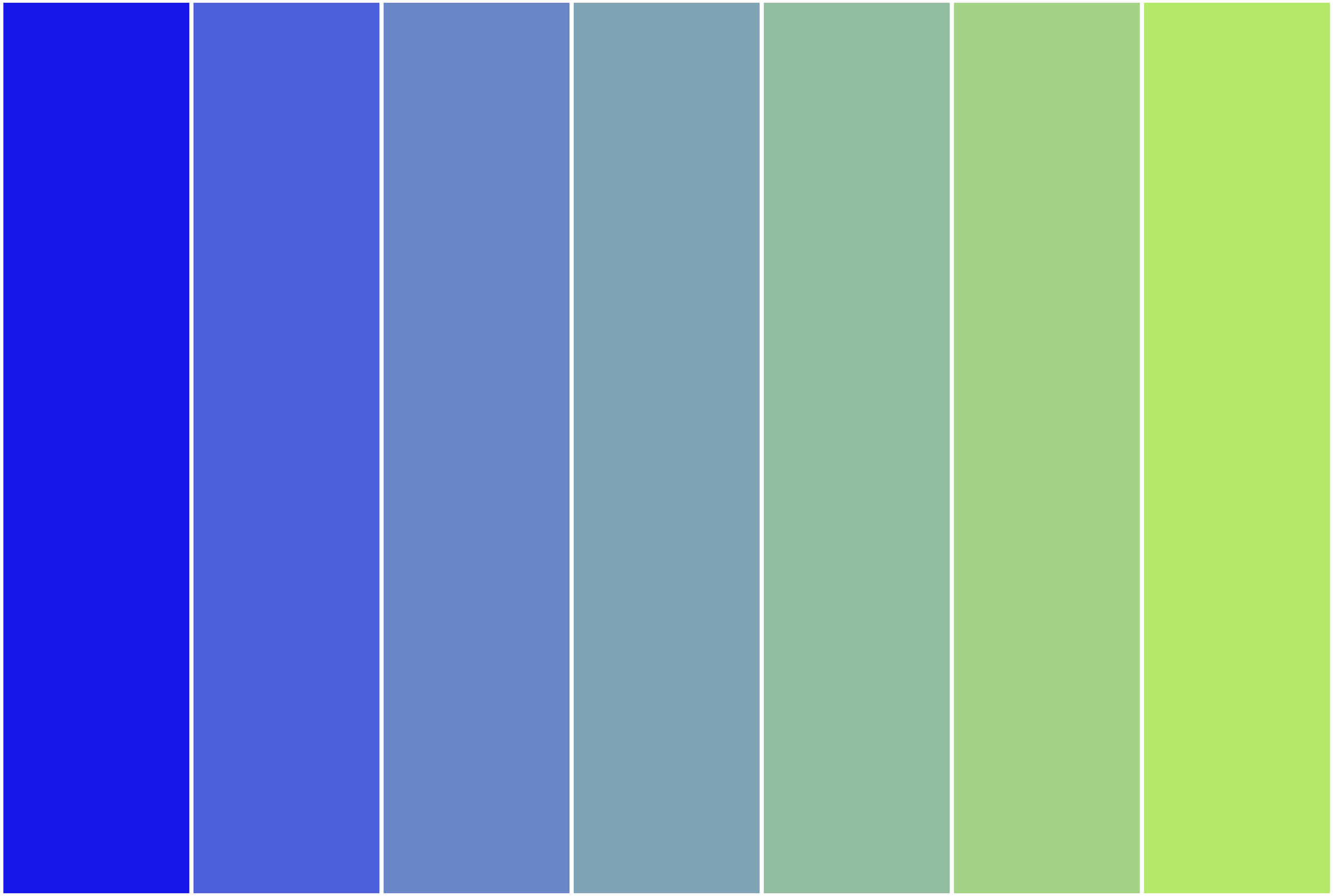
A heavy silence.

And then: "That's not a valid reason," argued the guy across the table.

Is it?

When were colors ever 'a valid reason'?

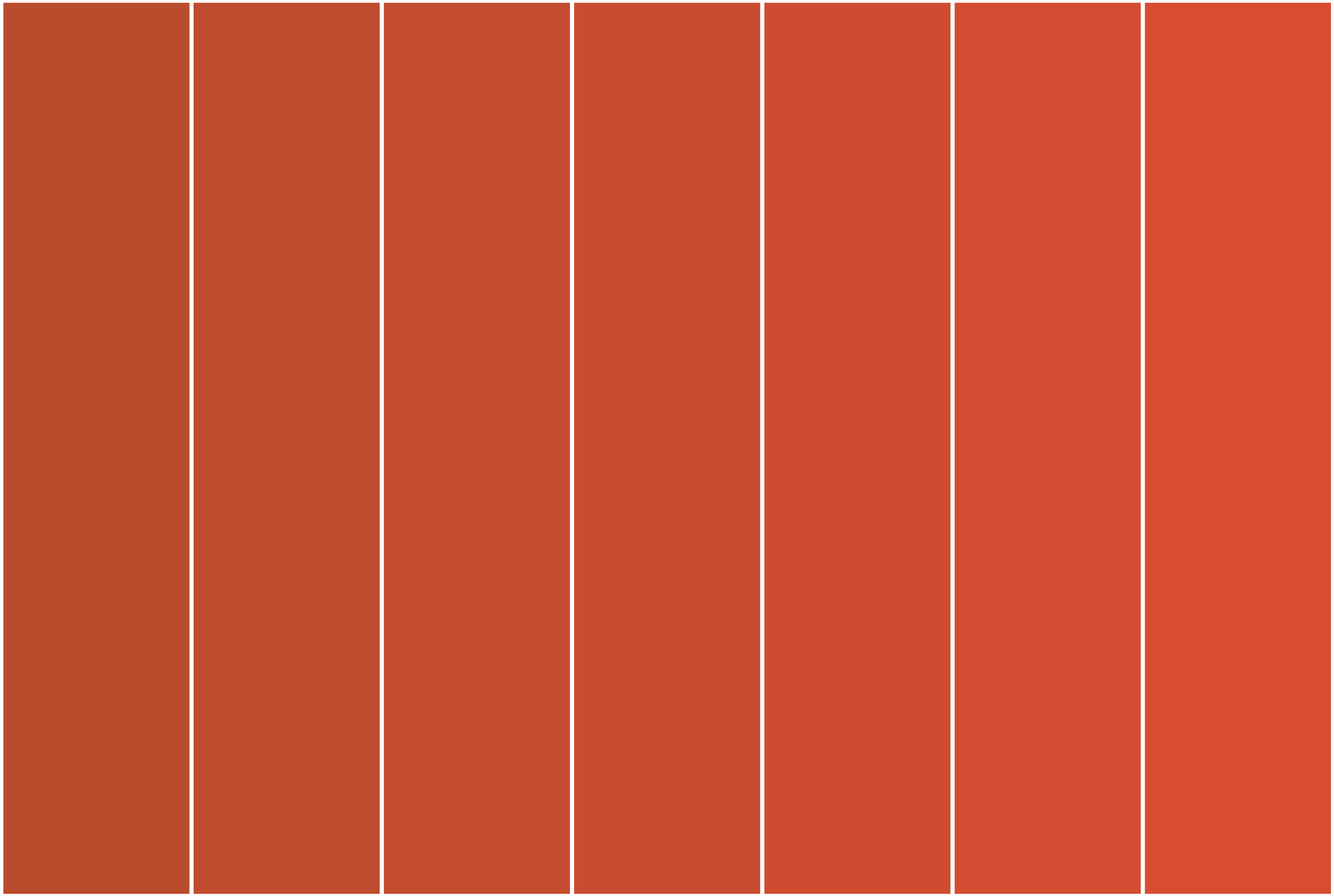
0xFF1817E9 0xFF4C61D9 0xFF6987C7 0xFF80A4B5 0xFF93BD9F 0xFFA4D387 0xFFB4E869



0xFFC6C28A	0xFFB5B17F	0xFFA39F73	0xFF8E8B66	0xFF767357	0xFF585445	0xFF261F2C

0xFF1B86BE	0xFF198AB6	0xFF178EAD	0xFF1592A5	0xFF13969C	0xFF119A92	0xFF0E9E87

0xFFBA4C2E 0xFFBF4C2E 0xFFC44C2F 0xFFCA4C30 0xFFCF4C30 0xFFD44C31 0xFFD84D31

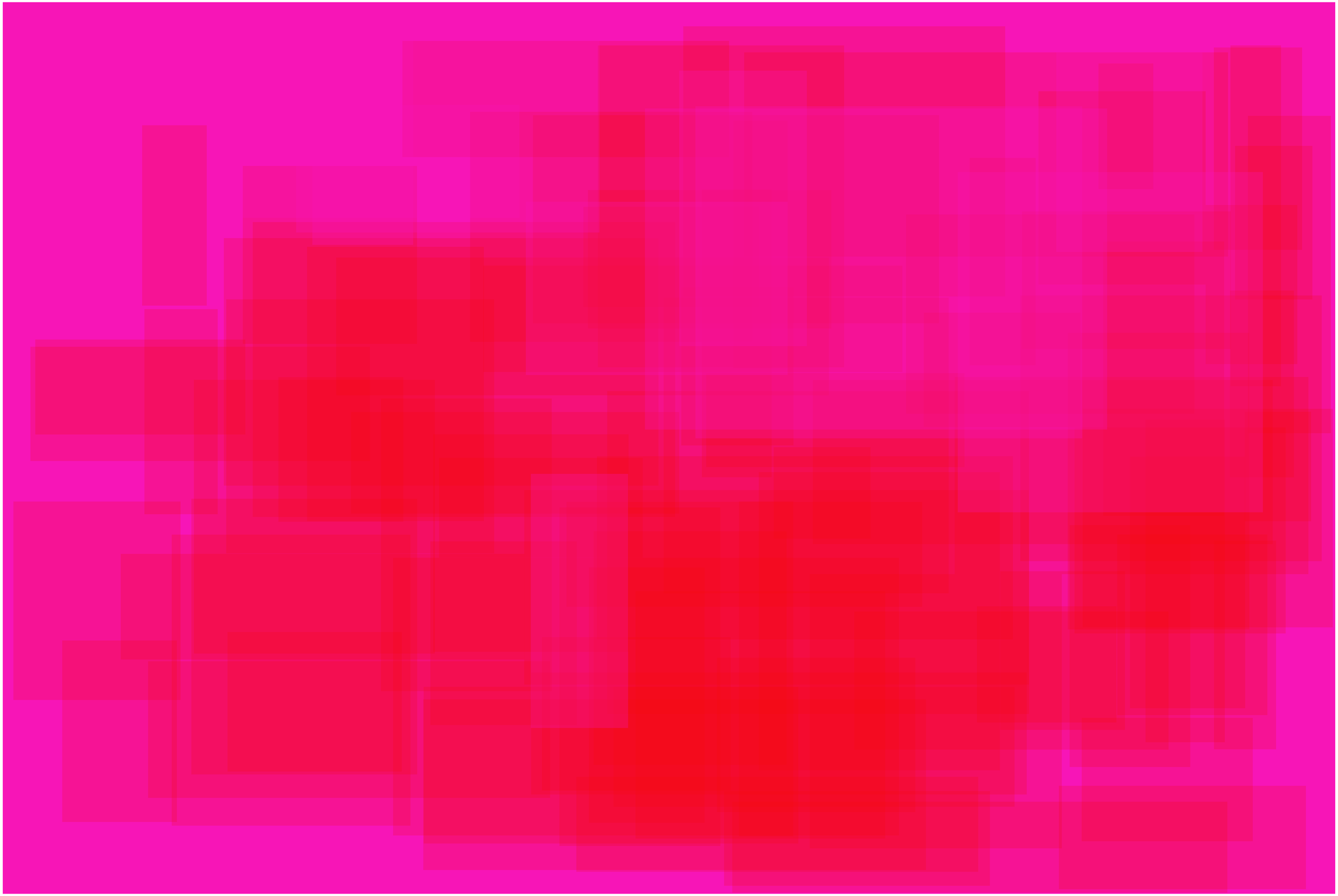


0xFF408952	0xFF53914C	0xFF629A45	0xFF70A23F	0xFF7CA937	0xFF86B12E	0xFF91B822

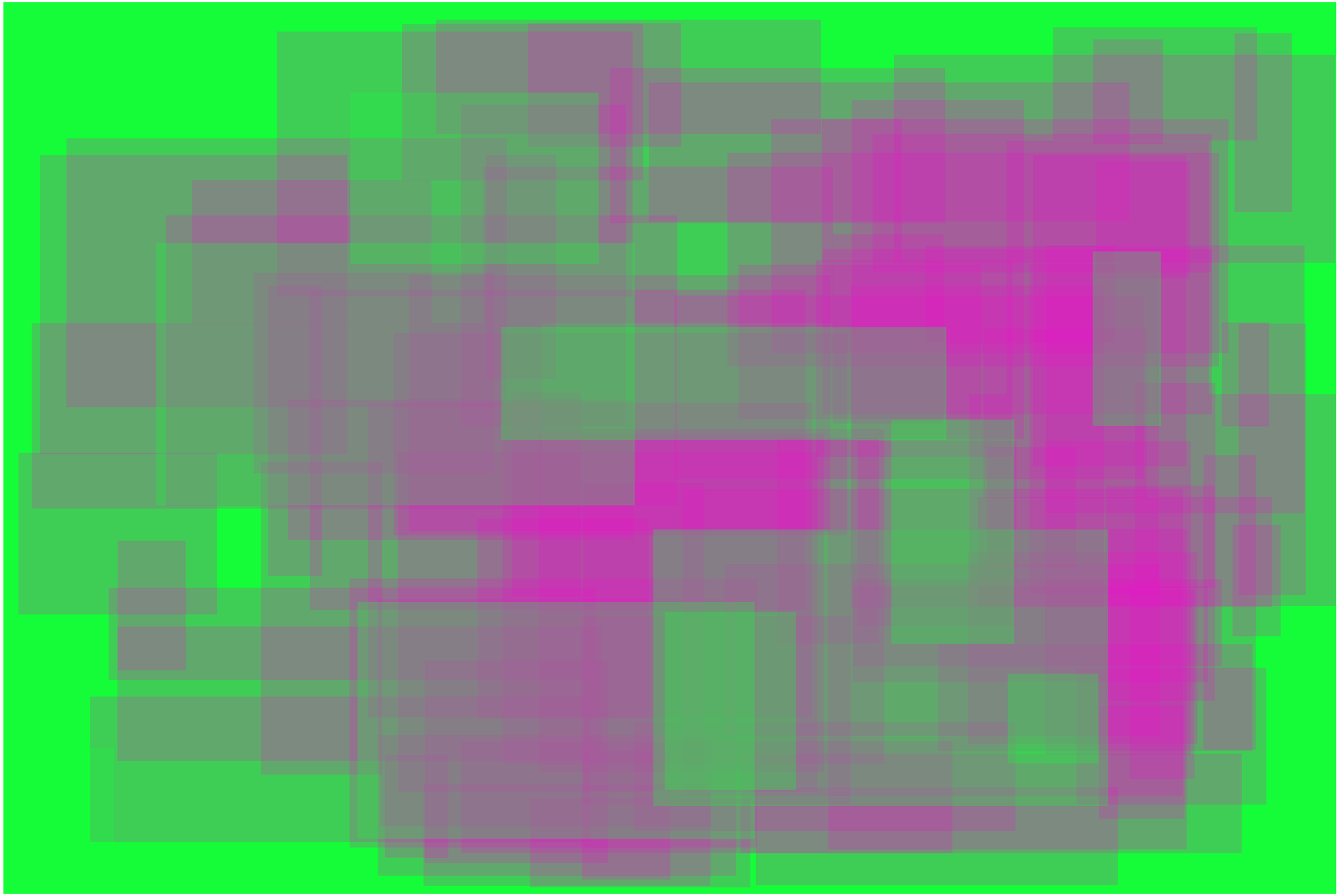
```
// recipe for color modulation

SecureRandom random = new SecureRandom();

// prepare the first color
int min = 0;
int max = 255;
int r1 = random.nextInt(max-min+1)+min;
int g1 = random.nextInt(max-min+1)+min;
int b1 = random.nextInt(max-min+1)+min;
color col1 = color(r1, g1, b1);

// then prepare the second color
int r2 = random.nextInt(max-min+1)+min;
int g2 = random.nextInt(max-min+1)+min;
int b2 = random.nextInt(max-min+1)+min;
color col2 = color(r2, g2, b2);

// make preliminary calculations
float numsteps = 7;
float step = (1.0)/(numsteps-1);
float c1_weight = 1; // the starting weight for the first color
float c2_weight = 0;  // the starting weight for the second color
float gapsize = 10;
float rectwidth = pgwidth/numsteps - gapsize;

pushMatrix();
translate(margin, margin);
noStroke();
```

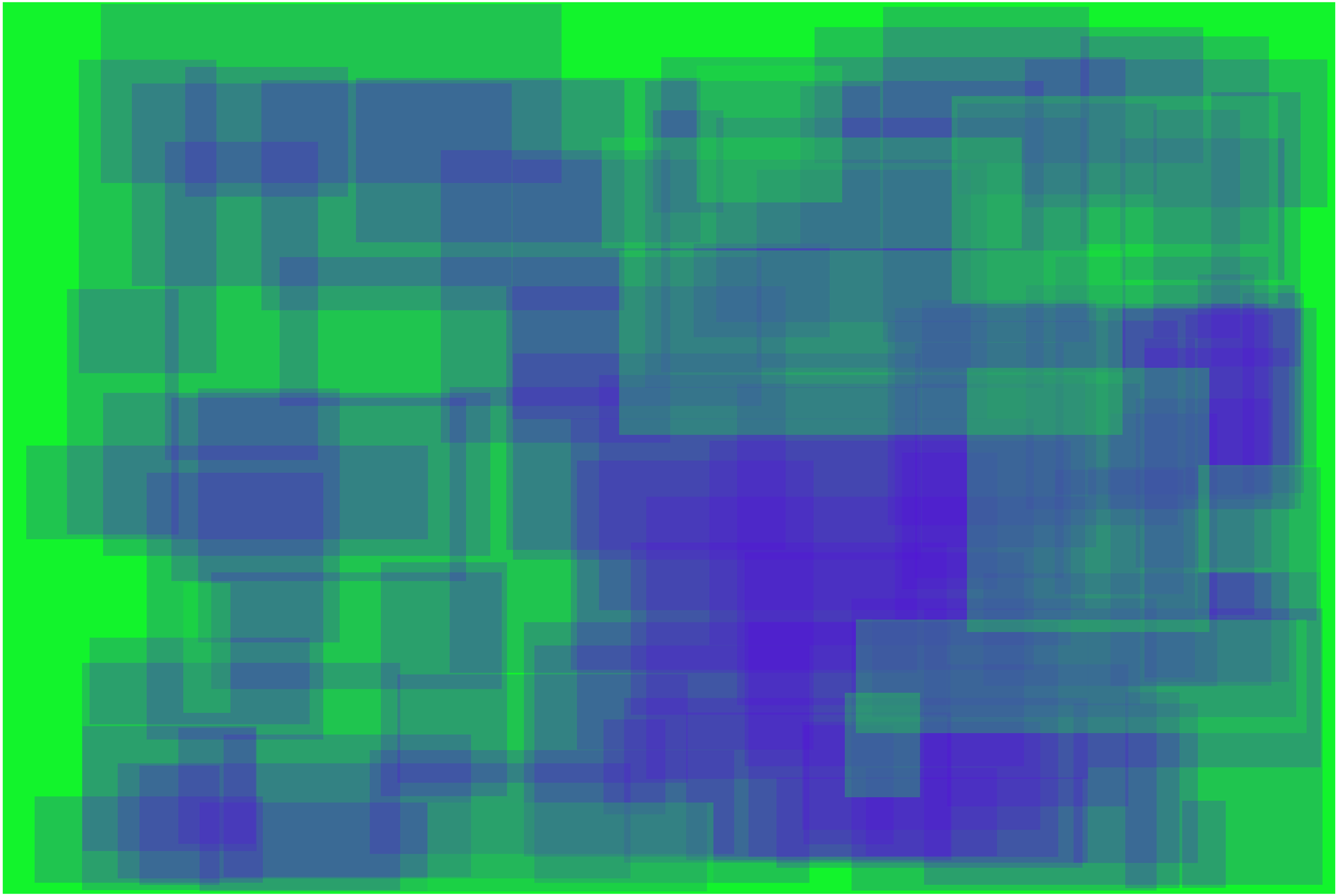
```
for(int i = 0; i < numsteps; i++) {
  // get a weighted average of the red, green, blue channels
  float mixedred = sqrt((sq(red(col1))*c1_weight +sq(red(col2))*c2_weight));
  float mixedgreen = sqrt((sq(green(col1))*c1_weight +sq(green(col2))*c2_weight));
  float mixedblue = sqrt((sq(blue(col1))*c1_weight +sq(blue(col2))*c2_weight));

  // prepare the color strip
  color stripcol = color(mixedred, mixedgreen, mixedblue);
  fill(stripcol);
  float posx = (rectwidth + gapsize)*i;

  // lay down the color strip
  rect(posx, 0, rectwidth, pgheight);

  text("0x"+hex(stripcol), posx, height*0.05 - margin);
  c1_weight -= step;
  c2_weight += step;

}

popMatrix();
```

Chapter 06: Bridging Colors



0xFFEF10CE 0xFF15FD38



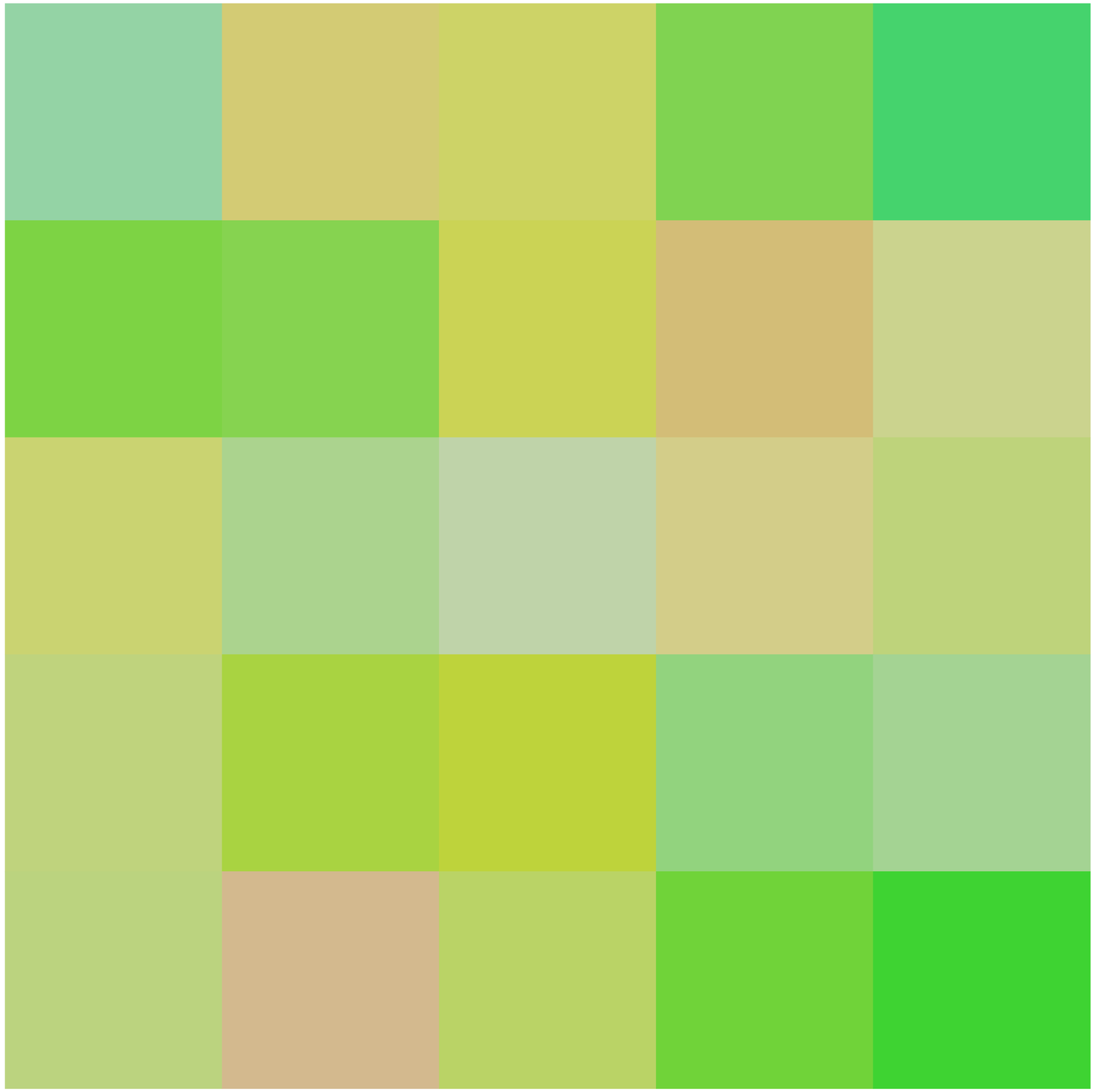


```
// recipe for using bridging colors

// switch working color mode to HSB
// before preparing the foreground color
colorMode(HSB,360,100,100);
SecureRandom random = new SecureRandom();

int min = 0;
int max = 360;
int h1 = random.nextInt(max-min+1)+min;

color col1 = color(h1, random(90,100), random(80,100));

rectMode(CORNER);
pushMatrix();
translate(margin, margin);
noStroke();

// glaze with the background color, which is opposite from the foreground color
// with a touch of randomness
color bg = color((hue(col1)+180+h2)%360, random(90,100), random(80,100));
fill(bg);
rect(0, 0, pgwidth,pgheight);

// randomly add 90 slices of the foreground color
for(int i = 0; i < 90; i++) {
  float posX = random(0, pgwidth-200);
  float posY = random(0, pgheight-200);
```

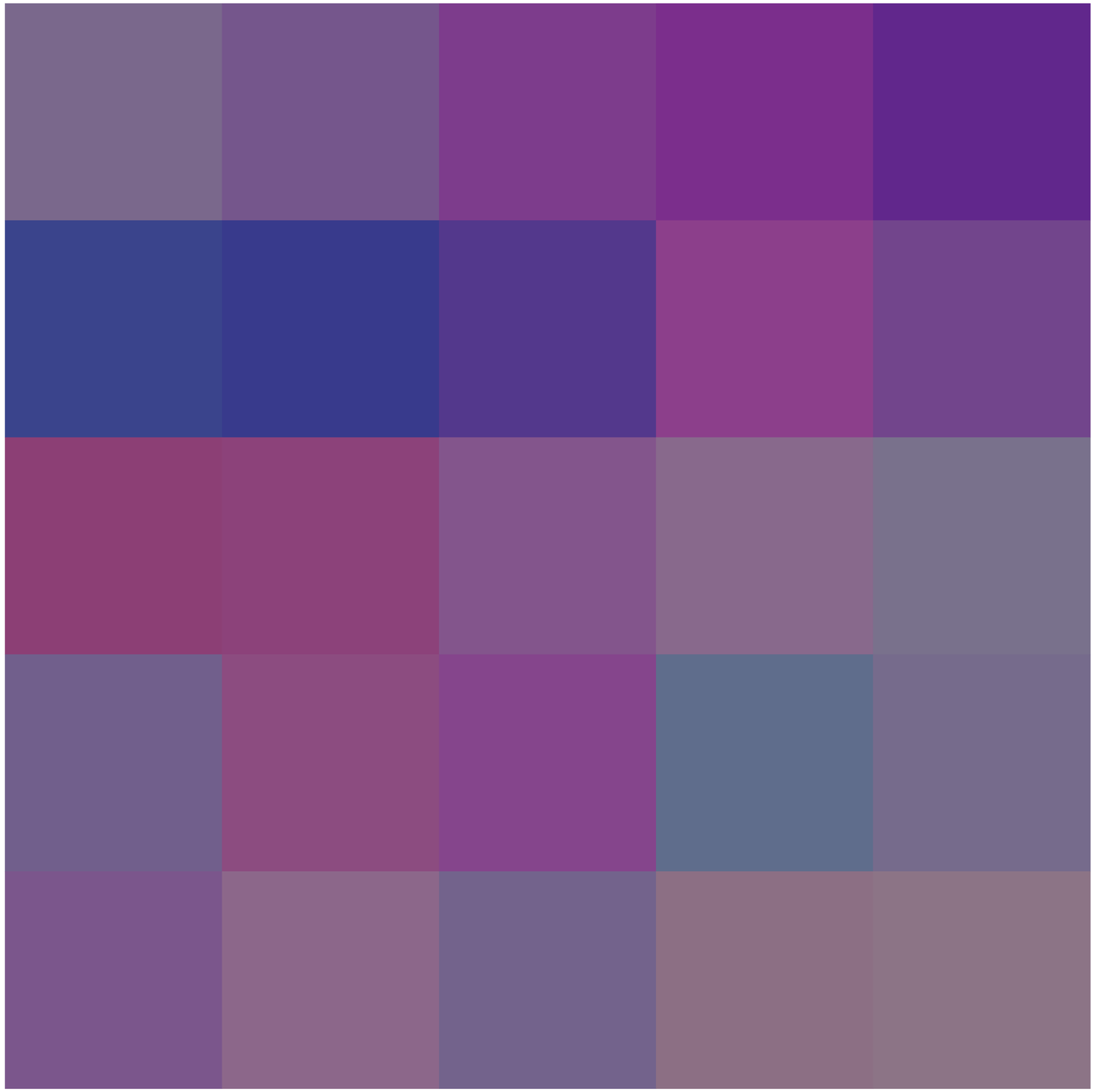
```
    float recth = random(200, min(800, pgheight-posY));

    // the slices can overlap with each other,
    // but they must be thin
    // like prosciutto
    fill(col1, 50);
    rect(posX, posY, rectw,recth);

  }

  // for balance, top with a few thin slices of the background color
  for(int i = 0; i < 10; i++) {
    float posX = random(0, pgwidth-200);
    float posY = random(0, pgheight-200);

    float rectw = random(100, min(1200, pgwidth-posX));
    float recth = random(200, min(800, pgheight-posY));

    fill(bg,70);
    rect(posX, posY, rectw,recth);

  }

  popMatrix();
  colorMode(RGB);
```

Chapter 01: Same Value Studies

value: 83



value: 55

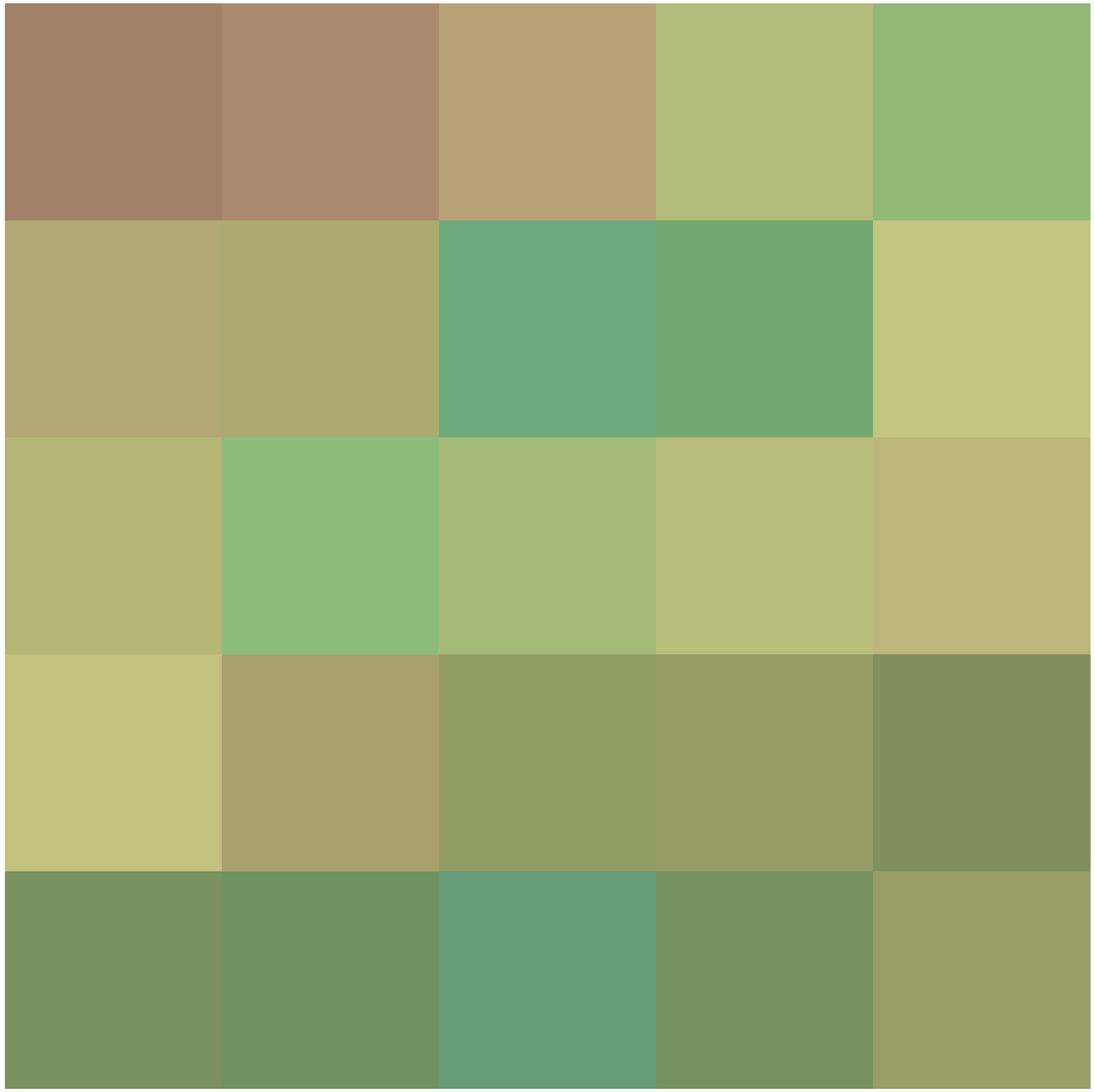


```
// recipe for same value studies (a la Johannes Itten)

// we're working in HSB space now!
// RGB is for posers
colorMode(HSB,360,100,100);
SecureRandom random = new SecureRandom();

int min = 40;
int max = 95;
int value = random.nextInt(max-min+1)+min;

// boundaries for saturation
int satmin = 0;
int satmax = 100;

// noise variables
int noisemin = 0;
int noisemax = 100;
float hoff = random.nextInt(noisemax-noisemin+1)+noisemin;
float soff = random.nextInt(noisemax-noisemin+1)+noisemin;

rectMode(CORNER);
pushMatrix();
translate(margin + 500, margin);
noStroke();

// sprinkle squares in a matrix
for(int row = 0; row < 5; row++) {
  for(int col = 0; col < 5; col++) {
```

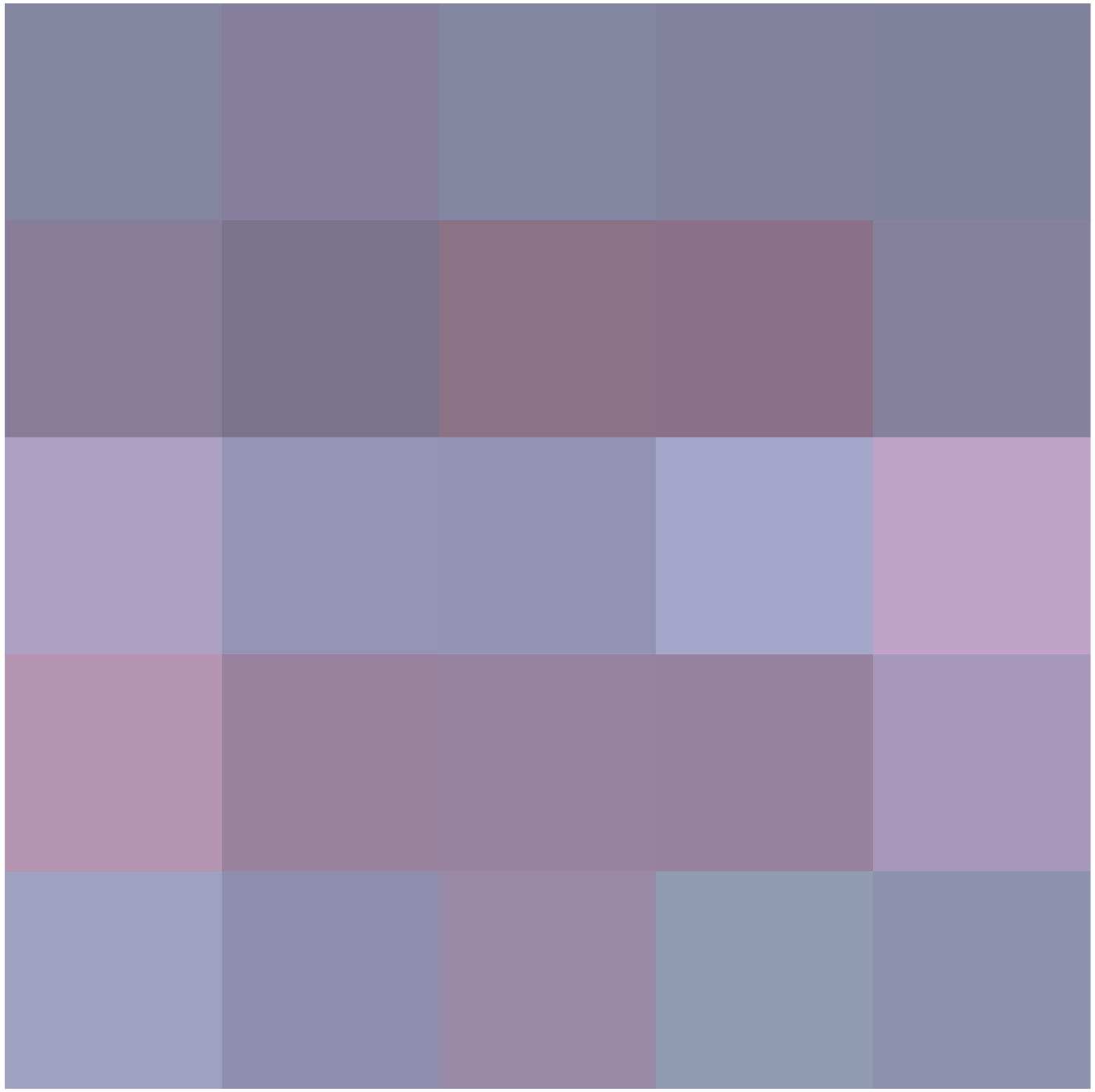
```
    // instead of using random, I'm mapping the colors to noise
    // noise is a bit more 'organic' than random
    float hue = map(noise(hoff), 0,1, huemin,huemax);
    float sat = map(noise(soff), 0,1, satmin,satmax);
    color squarecol = color(hue, sat, value);

    fill(squarecol);
    stroke(squarecol);
    rect(400*col, 400*row, 400, 400);

    // add a touch of magic numbers
    hoff += 0.55;
    soff += 0.3;
  }

  }

  popMatrix();
  colorMode(RGB);

  textFont(mainFont, 60);
  fill(0);
  text("value: " +value, width*0.05, height*0.05);
```

Chapter 08: Same Saturation Studies

saturation: 35



saturation: 18



Appendix: Palettes

artist:
James Jean

0xFFDCCECB

0xFFBBA49C

0xFF957668

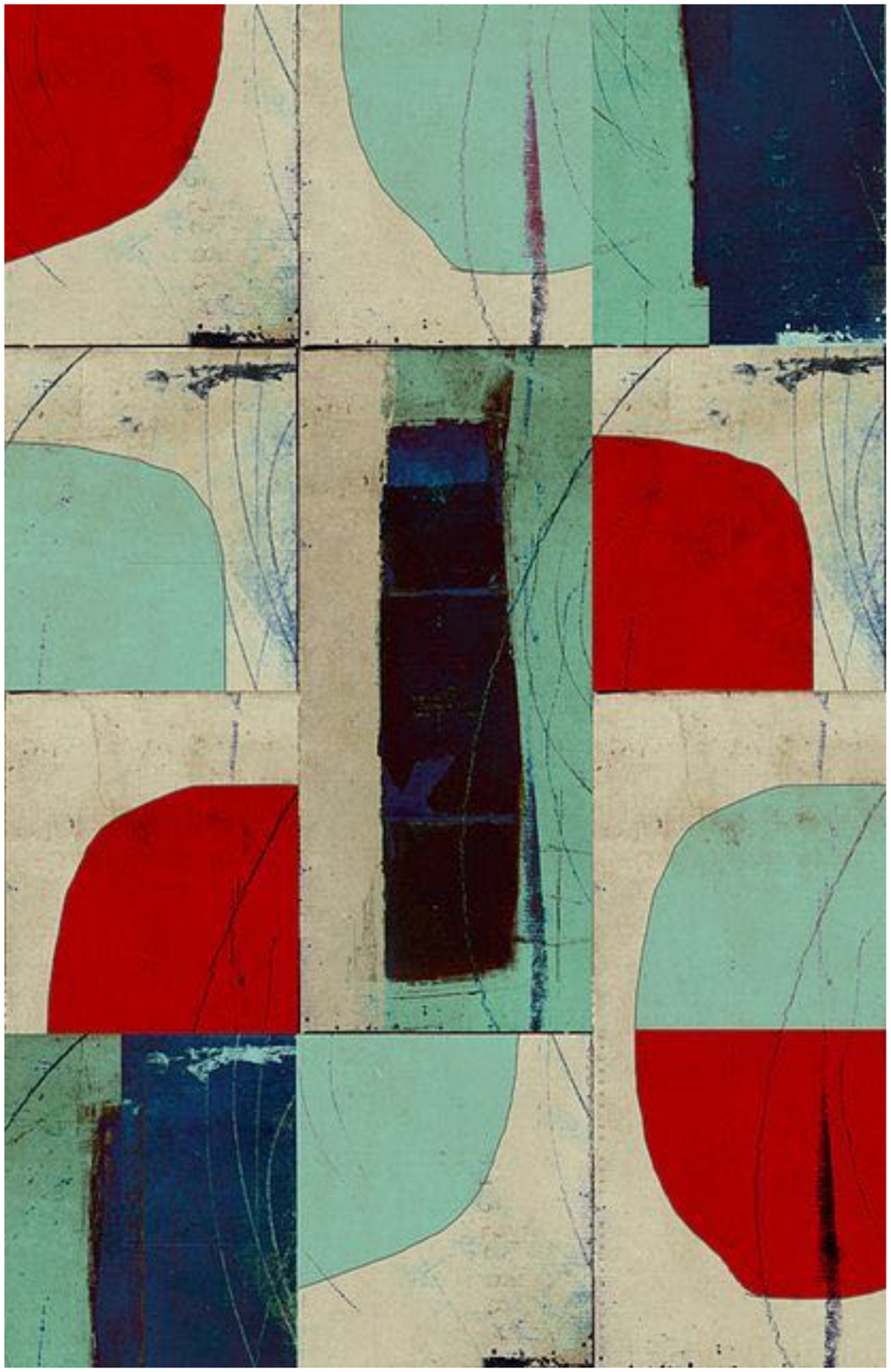
0xFF654339

0xFF3B221F

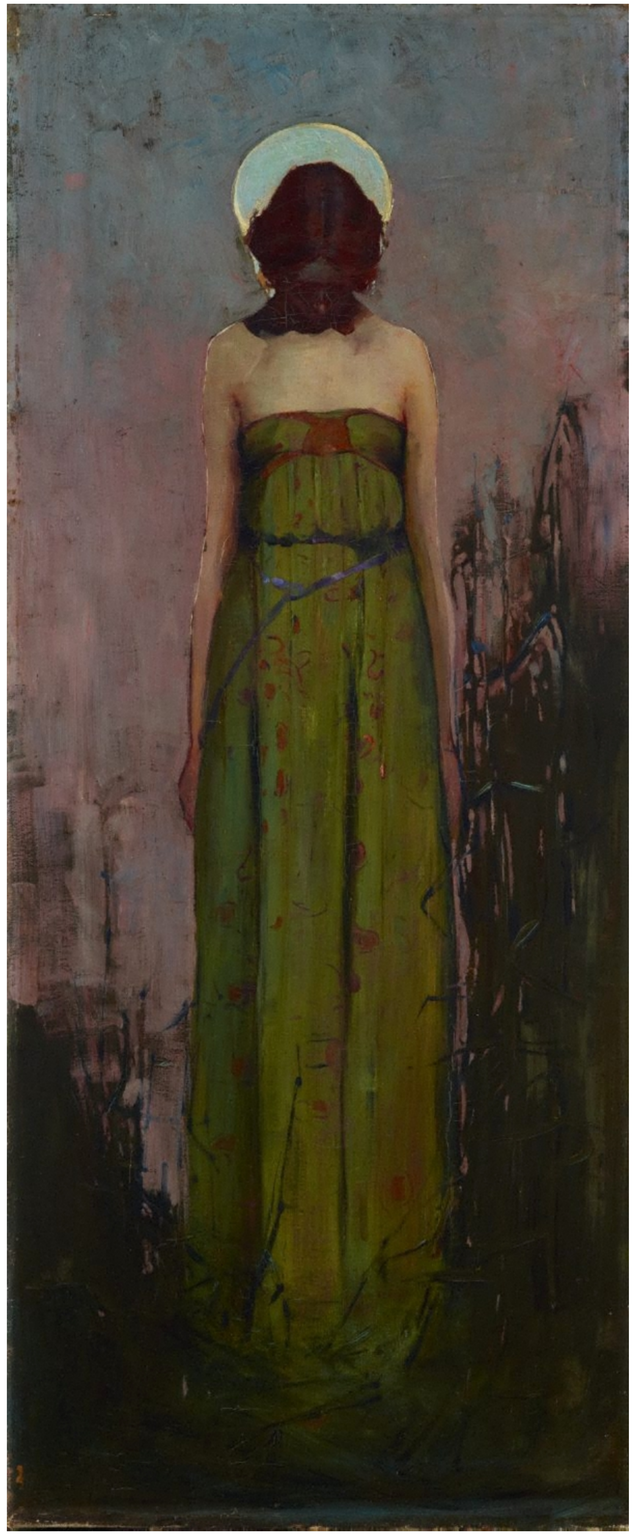
0xFF1B0F10

0xFFF2EDEB

artist:
Linda Vachon



artist:
Sydney Long



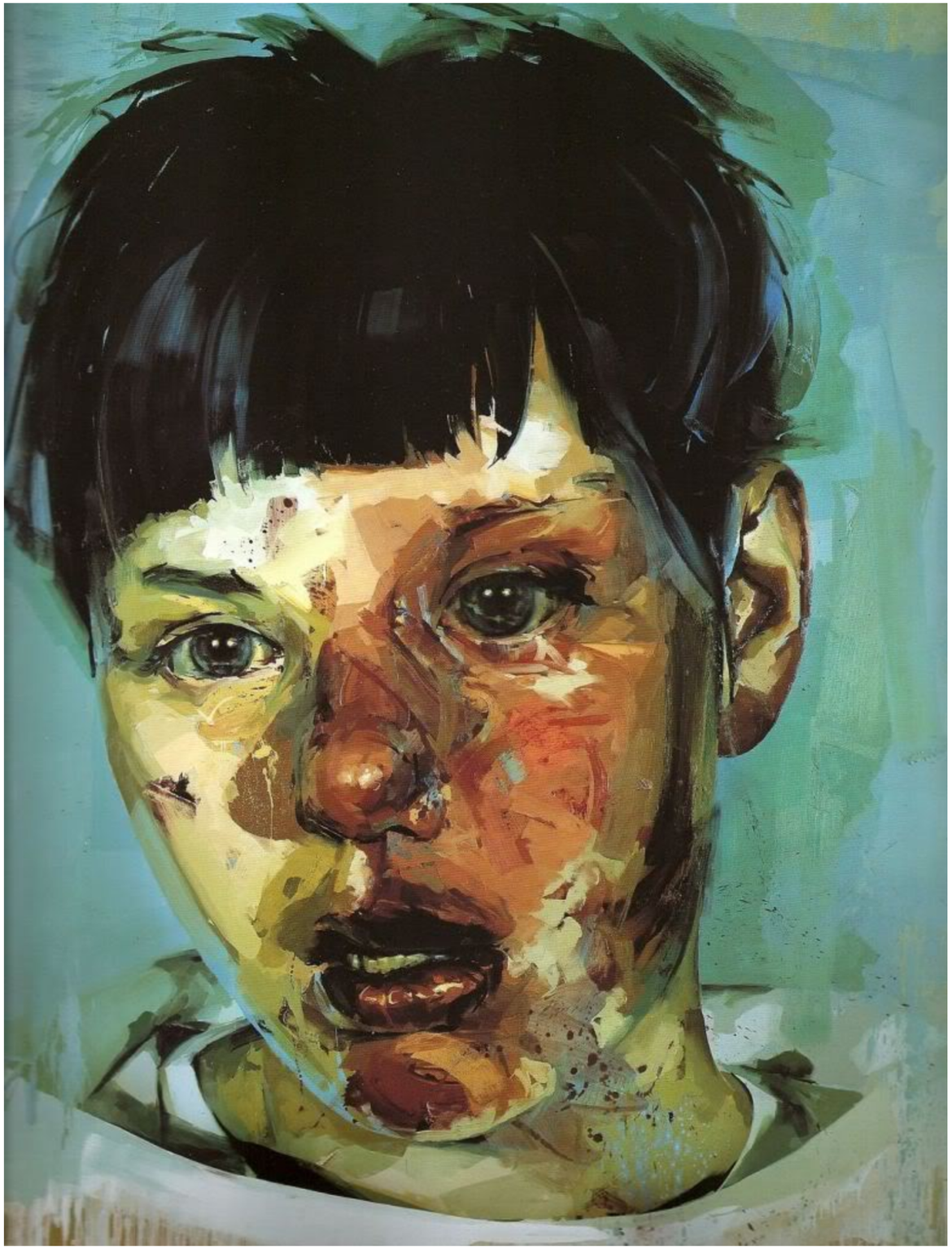
0xFF69605B

0xFF514E44

0xFF96876D

0xFF2F211A

artist:
Jenny Saville

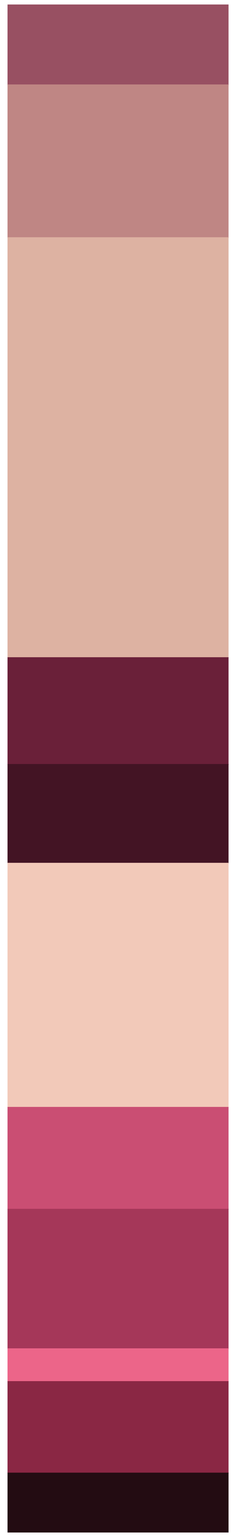


artist:
Deborah Zlotsky

artist:
Paul Klee

artist:
Alphonse Mucha

artist:
Shintaro Ohata



artist:
Tatsuyuki Tanaka

0xFFAF9888

0xFFD4C0B7

0xFF8B7667

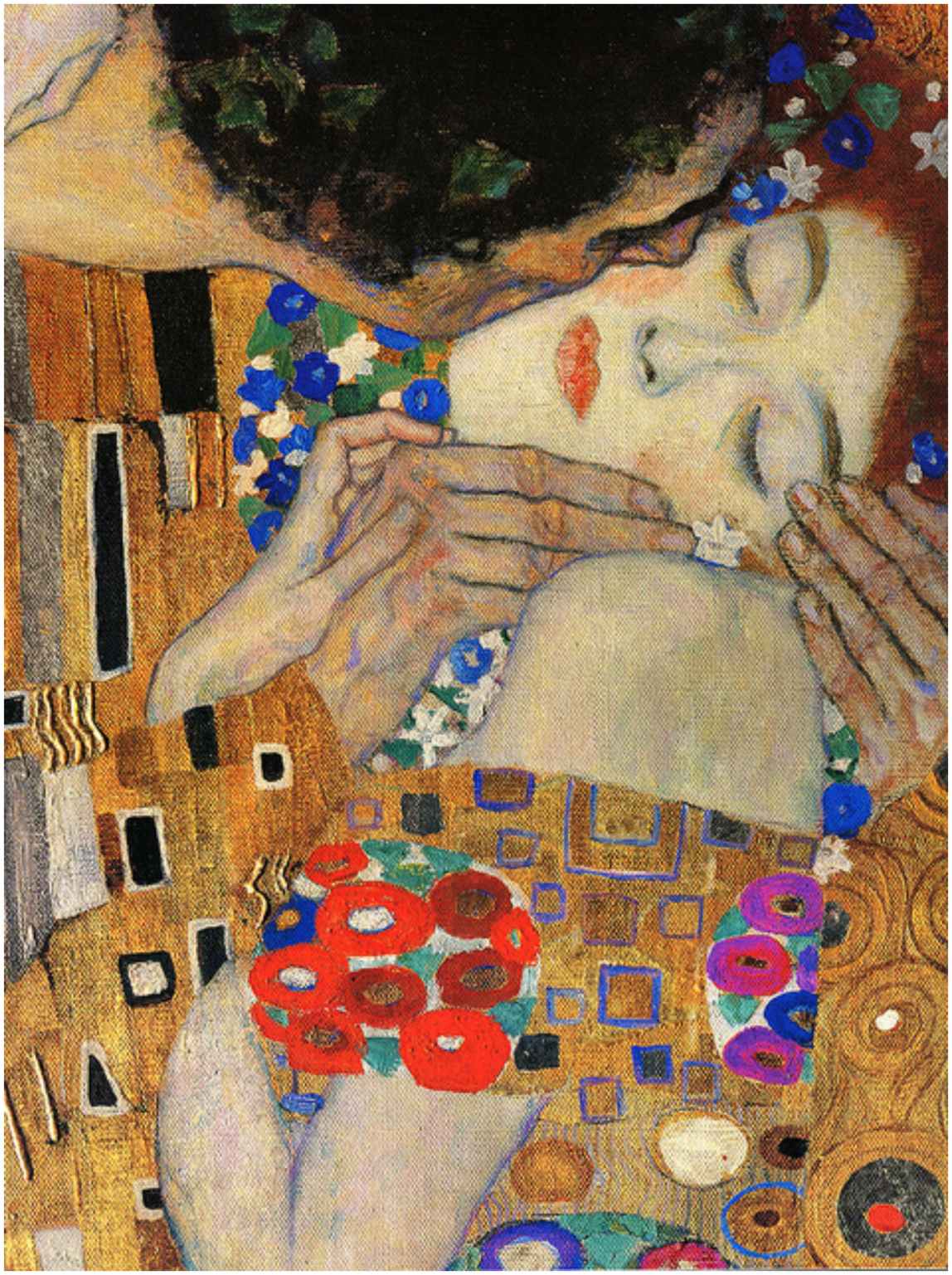
0xFF624E41

0xFF423126

0xFF211811

0xFF0F0E08

artist:
Gustav Klimt



```
ArrayList<Pixel> colors = new ArrayList<Pixel>();
float tolerance = 50.0;
int totalCount = 0;
float proptolerance = 0.005;
PImage[] images = new PImage[folderimages];
String[] names = new String[folderimages];

// recipe for palette grabber
public class Pixel {
  public ArrayList pixelgroup;
  public int count;
}

float colorDist(color c1, color c2) {
  float r = red(c1) - red(c2);
  float g = green(c1) - green(c2);
  float b = blue(c1) - blue(c2);

  return sqrt(sq(r) + sq(g) + sq(b));
}

void addPixel(color currpix) {
  Pixel p = new Pixel();
  // head of  the color group
  p.pixelgroup = new ArrayList();
  p.pixelgroup.add(currpix);
  p.count = 1;
  colors.add(p);
  totalCount++;
```

```
boolean notBlackorWhite(color col) {
  return ((col != 0.0) && (col != 255.0));
}

void palettePage(int imgindex) {
  totalCount = 0;

  // rinse palette before use
  for(int i = colors.size()-1; i >= 0 ; i--) {
     colors.remove(i);
  }
  PImage sourceimg = images[imgindex];

  sourceimg.resize(0, (int)pgheight);
  int iwidth = sourceimg.width;
  int iheight = sourceimg.height;
  color temp = sourceimg.get(0,0);
  float palettepos = sourceimg.width + 50;

  // extract colors from image
  // and add them to the palette
  for(int i = 0; i < iheight; i++) {
    for(int j = 0; j < iwidth; j++) {
      color currpix = sourceimg.get(i,j);

      if(notBlackorWhite(currpix)) {
        if(colors.size() == 0) {
          addPixel(currpix);
```

```
else {
  int idx;
  boolean foundMatch = false;
  int prevIdx = 0;
  float minDist = 0.0;

  // look through existing colors in the palette
  for(idx = 0; idx < colors.size(); idx++) {
    float currDist = colorDist(currpix, (Integer) colors.get(idx).pixelgroup.get(0));
    // if the color is in the palette
    if(currDist < tolerance) {
          // increase the count for the palette color closest to
          // the current pixel
          if(foundMatch && (currDist < minDist)) {
            colors.get(idx).count++;
            colors.get(prevIdx).count--;
            colors.get(idx).pixelgroup.add(currpix);

            prevIdx = idx;
            minDist = currDist;
          }
          else if(!foundMatch) {
            colors.get(idx).count++;
            colors.get(idx).pixelgroup.add(currpix);
            totalCount++;
            foundMatch = true;
            prevIdx = idx;
            minDist = currDist;
```

```
            }
        }

        if(!foundMatch) {
          addPixel(currpix);

        }

      }
    }
  }
}

int netCount = totalCount;
for(int idx = 0; idx < colors.size(); idx++) {
  float prop = colors.get(idx).count / ((float)totalCount);

  if(prop < proptolerance) {
    netCount -= colors.get(idx).count;
  }
}
rectMode(CORNER);
float ypos = 0f;

pushMatrix();
// place image a little to the right of your margins
translate(margin + 700, margin);
image(sourceimg, 0,0);
// place palette beside image
```

```
float prop = colors.get(i).count / ((float)netCount);

// if there is enough of the color in the image, display it
if(prop >= proptolerance) {
  float rheight = pgheight*prop;
  float totred = 0;
  float totgreen = 0;
  float totblue = 0;

  // calculate an average value of the colors in each group
  int groupsize = colors.get(i).pixelgroup.size();
  for(int j = 0; j < groupsize; j++) {
    Pixel thepixel = colors.get(i);
    color currcolor = (Integer) thepixel.pixelgroup.get(j);
    totred += red(currcolor);
    totgreen += green(currcolor);
    totblue += blue(currcolor);
  }
  color paletteColor = color(totred/groupsize, totgreen/groupsize,totblue/groupsize);

  stroke(paletteColor);
  fill(paletteColor);
  rect(palettepos, ypos, 300, rheight);

  // top it off with the hex value for the colors
  textFont(mainFont, 40);
  fill(paletteColor);
  text("0x"+hex(paletteColor), palettepos + 350, ypos+30);
```

```
    }
  }

  popMatrix();

}
```

A special thanks to
Clayton Merrell
and Bob Bingham
and Rafael Abreu-Canedo
for your guidance and support.