The original plan for my design was to start off with lots of colour and move on from there. As you can see in the picture on the right I originally wanted to use a gradient of some sort as my background. I took this particular gradient from the Processing textbook. It is a shift from blue to green in RGB colour mode.

I encountered an immediate problem however when I tried to apply this to my program. The error occurred due to the size of my window. I was trying to increase the size of the gradient so that it would cover my (600, 200) size window. The height posed no problem as it stretched. However when I tried to increase the width of the gradient it produced the blue to green for the first 100 pixels and then the remaining 500 pixels were a bright yellow as you can see below.

The problem may have been in the maths. The code used was from the textbook and I’ve included part of the code below:

\[
\text{float } r = 61 + (i*0.92); \\
\text{float } g = 156 + (i*0.48); \\
\text{float } b = 204 - (i*1.43);
\]
The problem with that section of the code is that the RGB values are calculated extremely specifically to pass from a pure RGB blue to a pure RGB green at a very specific rate and passing through a very specific amount of shades. I realised that to stretch it out would require me to recalculate all of those value and I knew I wouldn’t have a clue how to do that. So instead I tried fiddling about with another section of the code hoping I could salvage something and make it look half decent. The section of code I isolated was this: line(i, 0, i, 100);

The code itself was part of a for loop so basically all it was doing is repeatedly drawing a vertical line and changing it’s colour fractionally each time. This was much simpler than trying to change the colour values. By changing the last value to 200 I lengthened the height then I multiplied the first I value by 2 and the second I value by 3. This changed the x co-ordinates of the first and second point respectively. The end result of which is that I ended up with the background you see below. This ended up being one of three backgrounds that I have kept in my final program.

![Background Image](image_url)

With that done I began to experiment with how I could start manipulating my backgrounds. Originally I didn’t want to have a static background with simple shapes like lines or circles appearing on top. I wanted to have a series of fluid backgrounds instead. I didn’t implement this as my background at first. I just kept it on the side.
As I experimented with different ways of manipulating this background I began to find it more difficult than I first expected. Eventually I decided that I would have to have a static background and play around with what kinds of shapes that I could use to perhaps give the illusion of a changing background. Being relatively new to the idea of drawing using a programming language I started off with basic shapes and I never really left basic shapes. Below is the first example of what I was doing with various shapes. I decided to try squares and I found the random() method extremely useful. I used random() to decide the top left corner points of the squares using the height and width as maximum values so the square would be at least partially inside the 600 by 200 frame.

I also used random to decide the colours of the squares. I began to notice that it reminded me of old video games. The effect was reminiscent of back when video games were made in low resolution and the pixels were large and blocky. However I quickly discarded this idea as I felt the blocks were too basic and old fashioned especially noticeable on the particular background I had chosen. They were large and clumsy which created an unfavourable contrast with the sleek background, shown below:
I discarded the squares but kept the background. Working with the background I began to manipulate it. Realising that I didn’t know enough about the specific colour values to manipulate the numbers directly I quickly realised I didn’t have to. I used the random method. I have included an excerpt of my code showing this below:

```java
float r = random(a) + (random(i) * 0.92);
float g = random(c) + (random(i) * 0.48);
float b = random(d) - (random(i) * 1.43);
stroke(r, g, b);
line(i*2.5, 0, i*2.5, 200);
```

Again the whole code involves a for loop but I chose to focus on this part of the code. If you look at the first few lines you can see it’s the same gradient code I used from the processing textbook with minor alterations. I have used the random() method to create a multi coloured strip. The variables a, c & d are floats that are declared earlier in the code but basically they are random values with a max of 255 in order to be a valid fill value. Likewise the value i is the float I declared in the for loop earlier.

Having completed this new background I still knew that having a series of multicoloured backgrounds would not have enough variety so I returned to basic shapes. I was still reluctant to use big chunky or clumsy shapes. So I ended up choosing lines. Using a for loop similar to what I’d used for the background I created 3 random variables with a max of 255 and used them to create a series of lines that also used random variables to determine their location all to some degree within the 600 by 200 frame. On the next page I have included the code.
stroke(r, g, f);
line(random(200), random(a), random(a * 5), random(a));
exit();

Of those 3 lines the last caused me plenty of problems until I figured it out. Originally the for loop continued to create line upon line until you couldn’t see the background. I couldn’t figure out why this was. Eventually I decided to try using an exit() method that I’d seen before in Java. I am still uncertain as to why without the exit() method the loop creates an infinite amount of lines because the for loop is set to only run 5 time. The loop has conditions int i = 0;  i < 5; i++. Still using exit() worked and I ended up with the pattern you see below.

I felt that this was definitely something I wanted so I resolved to keep this and try to make only minor alterations. Keeping with that idea I drew inspiration from the lecture slides. The picture on the right drew my attention I wanted to create a background like that but coloured. What eventually transpired was that I created exactly what you see to the right but with a much lighter strokeWeight() of 1. I then layered it atop my multicolour striped background.
Below is the result of layering the thin white stripes atop the multi coloured background. It became my second out of three backgrounds.

Finally I decided to use my earlier blue to green Gradient as my third and final background as shown below with coloured lines.

Once again the use of each particular background is determined by random(). The 3 coloured pictures with lines you have just seen make up my final program.

The code that picks the background is an if loop. It uses a random variable \( k \) that is a number no higher than 2. If \( k \) is less than 1 then the multi coloured striped background is selected. Once the striped background is selected there is another if loop that determines whether it will have the thin white stripes on top or not. Alternatively if \( k \) is greater than 1 then the gradient background is used.