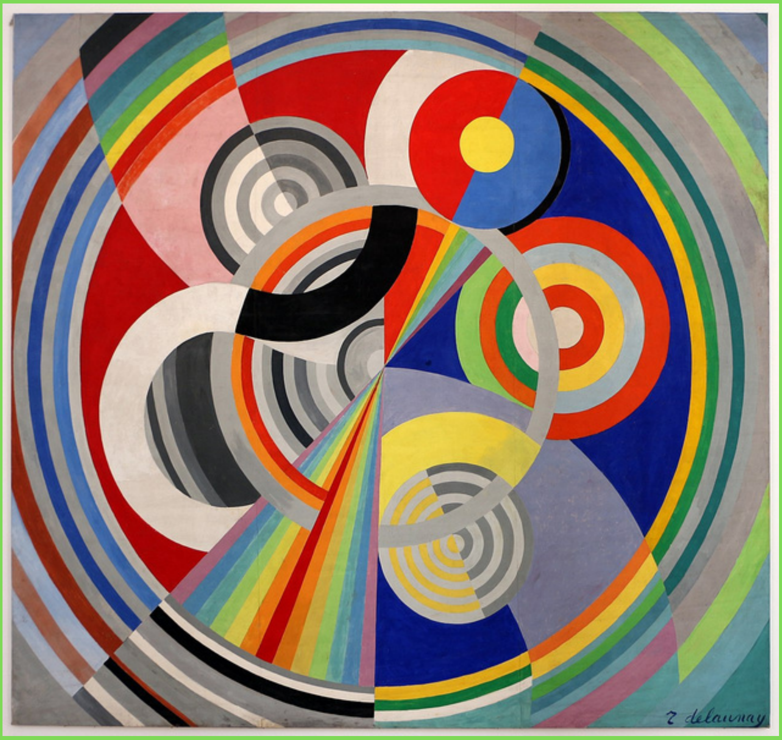
SOUND AND IMAGE

JETHA KAUR

GIT-HUB

Delaunay

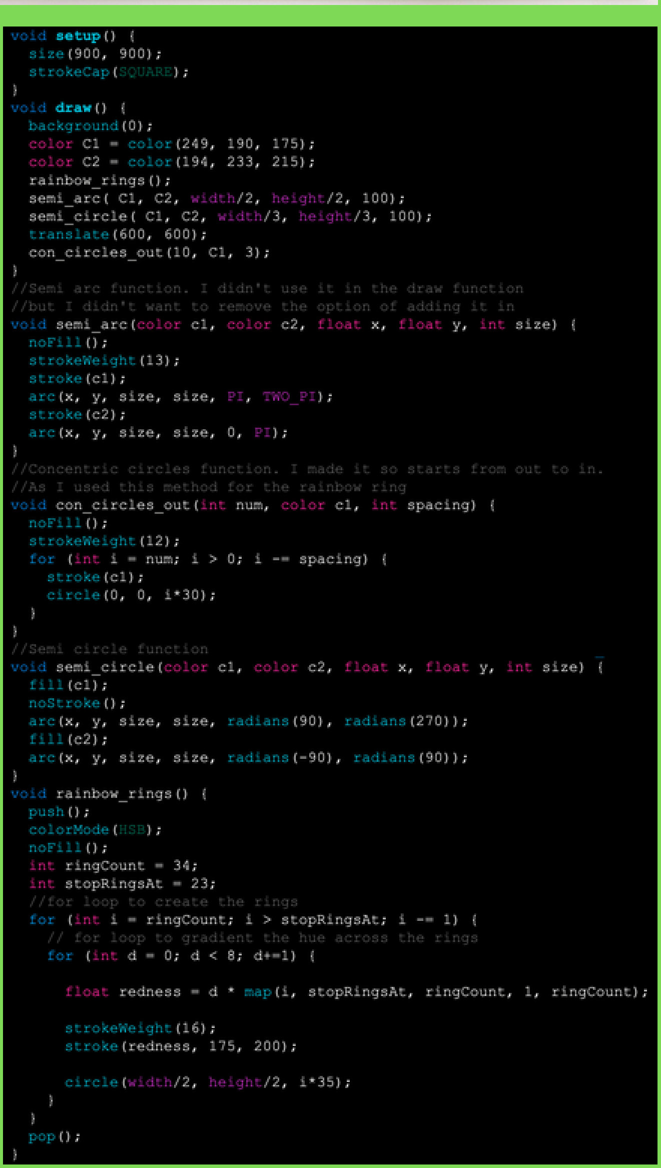
I picked Robert Delaunay for this project as I liked his use of colours, shapes and composition. Delaunay used contrasting complementary colours, Discordant colours, and a mixture of curves, circles and straight lines



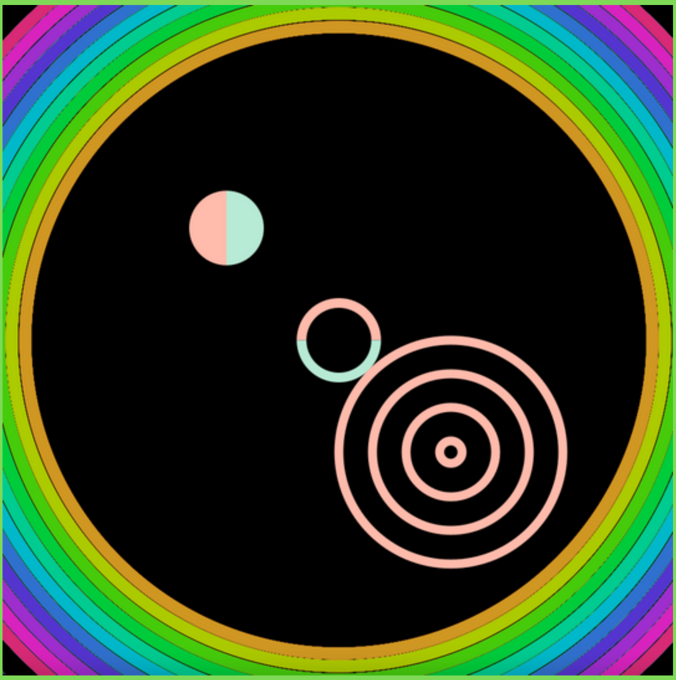
I started by breaking down the idea into layers:

- Make Shapes
- Add colours
- Make generative
  - position
  - colours

I made functions for the different shapes. They were all simple to code except the rainbow rings which I brought to office hours and got help with looping the hue over the rings. I used Processing reference for the syntax of some parts like the array for the colour palette. I also used ChatGPT in instances where I couldn't locate the error in my code.



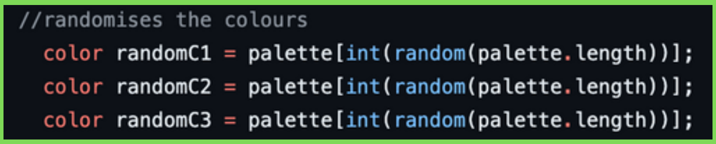
```
void setup() {
  size(900, 900);
  strokeCap(SQUARE);
}
void draw() {
  background(0);
  color C1 = color(249, 190, 175);
  color C2 = color(194, 233, 215);
  rainbow_rings();
  semi_arc( C1, C2, width/2, height/2, 100);
  semi_circle( C1, C2, width/3, height/3, 100);
  translate(600, 600);
  con_circles_out(10, C1, 3);
}
//Semi arc function. I didn't use it in the draw function
//but I didn't want to remove the option of adding it in
void semi_arc(color c1, color c2, float x, float y, int size) {
  noFill();
  strokeWeight(13);
  stroke(c1);
  arc(x, y, size, size, PI, TWO_PI);
  stroke(c2);
  arc(x, y, size, size, 0, PI);
}
//Concentric circles function. I made it so starts from out to in.
//As I used this method for the rainbow ring
void con_circles_out(int num, color c1, int spacing) {
  noFill();
  strokeWeight(12);
  for (int i = num; i > 0; i -= spacing) {
    stroke(c1);
    circle(0, 0, i*30);
  }
}
//Semi circle function
void semi_circle(color c1, color c2, float x, float y, int size) {
  fill(c1);
  noStroke();
  arc(x, y, size, size, radians(90), radians(270));
  fill(c2);
  arc(x, y, size, size, radians(-90), radians(90));
}
void rainbow_rings() {
  push();
  colorMode(HSB);
  noFill();
  int ringCount = 34;
  int stopRingsAt = 23;
  //for loop to create the rings
  for (int i = ringCount; i > stopRingsAt; i -= 1) {
    // for loop to gradient the hue across the rings
    for (int d = 0; d < 8; d+=1) {

      float redness = d * map(i, stopRingsAt, ringCount, 1, ringCount);

      strokeWeight(16);
      stroke(redness, 175, 200);

      circle(width/2, height/2, i*35);
    }
  }
  pop();
}
```



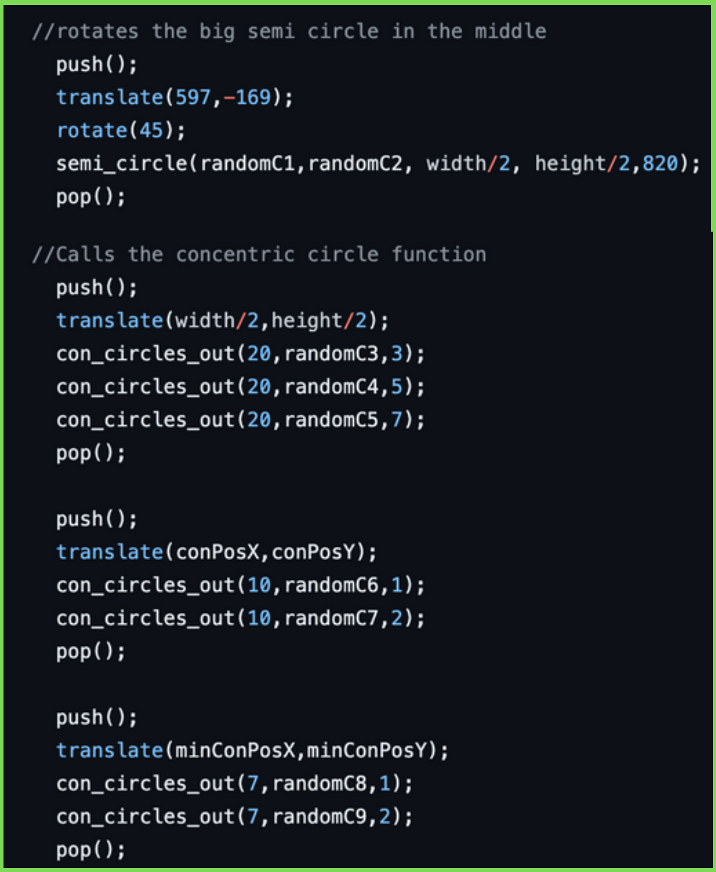
For the generative part, I randomised the position and colour using random. I added 13 colours to make the probability of picking the same colour low.

I also added translate for the positioning of the concentric circles.

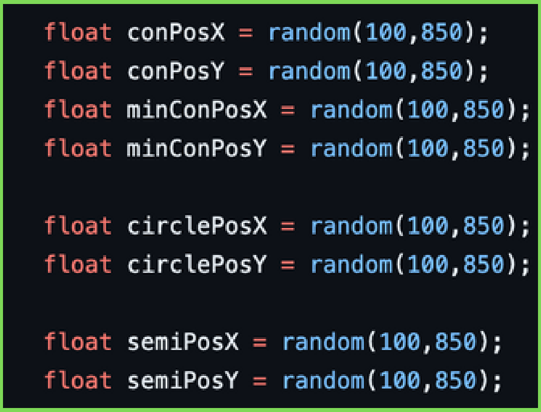
I think next time I would modify the colour selection so there was no chance for the same colour to be picked. I would also modify the positioning so that shapes don't overlap



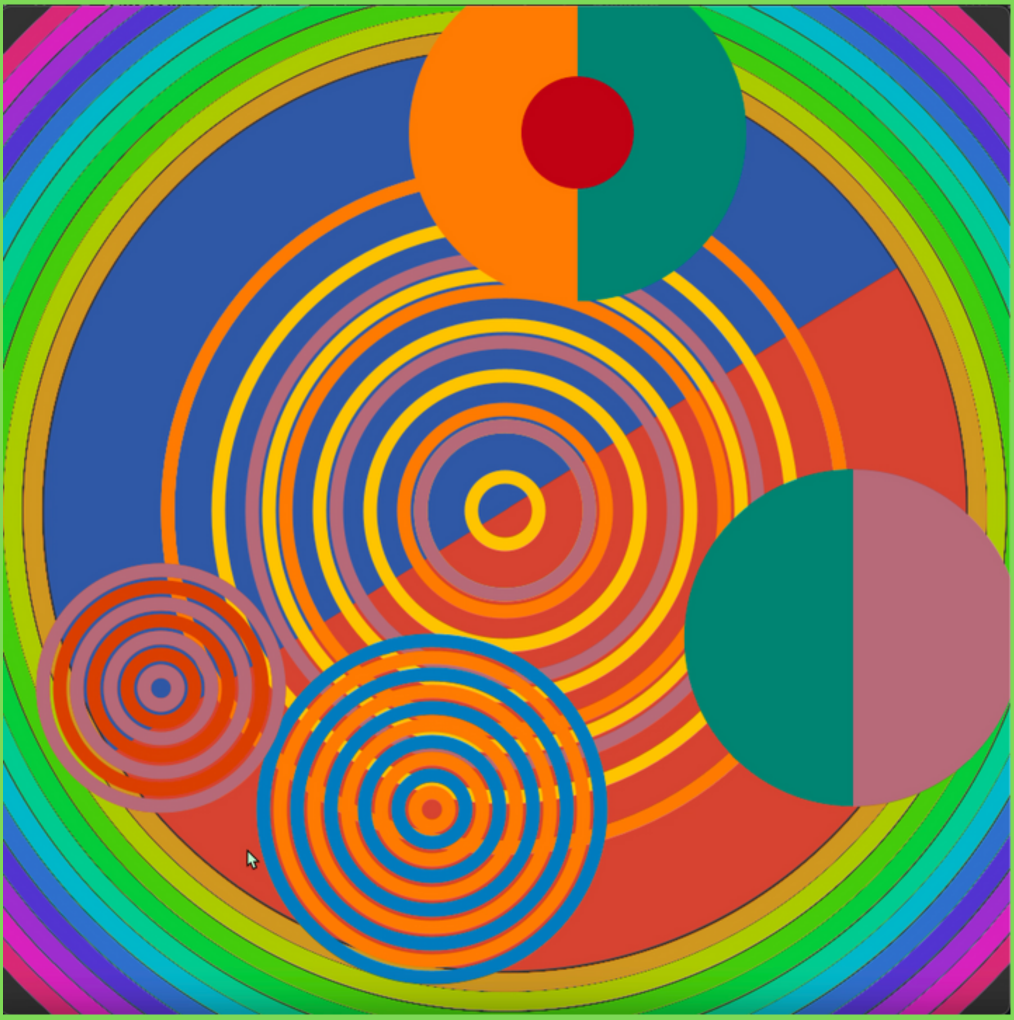
```
//randomises the colours
  color randomC1 = palette[int(random(palette.length))];
  color randomC2 = palette[int(random(palette.length))];
  color randomC3 = palette[int(random(palette.length))];
```



```
//rotates the big semi circle in the middle
  push();
  translate(597,-169);
  rotate(45);
  semi_circle(randomC1,randomC2, width/2, height/2,820);
  pop();

//Calls the concentric circle function
  push();
  translate(width/2,height/2);
  con_circles_out(20,randomC3,3);
  con_circles_out(20,randomC4,5);
  con_circles_out(20,randomC5,7);
  pop();

  push();
  translate(conPosX,conPosY);
  con_circles_out(10,randomC6,1);
  con_circles_out(10,randomC7,2);
  pop();

  push();
  translate(minConPosX,minConPosY);
  con_circles_out(7,randomC8,1);
  con_circles_out(7,randomC9,2);
  pop();
```



```
  float conPosX = random(100,850);
  float conPosY = random(100,850);
  float minConPosX = random(100,850);
  float minConPosY = random(100,850);

  float circlePosX = random(100,850);
  float circlePosY = random(100,850);

  float semiPosX = random(100,850);
  float semiPosY = random(100,850);
```



<u>Final piece</u>

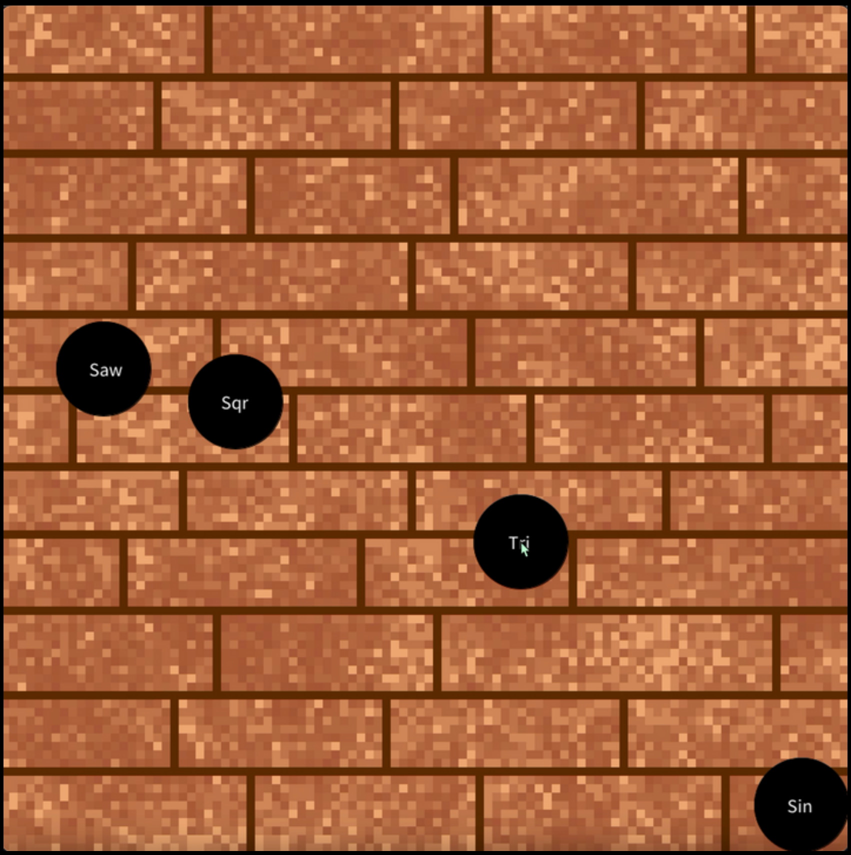
SOUND

For this project, I wasn't sure what to do. I knew I wanted to use the oscillators. So, I started by attaching the mouse position to the frequency and amplitude. Also assigning each oscillator to a key.

I then attached each oscillator to a circle. With the frequency and amplitude mouse position, I realised I could use mouse click to drag the circles into different positions. Which allows the user to pick the frequency and amplitude for each oscillator.

I thought it would also be fun to have an animated part which is why I made a party mode. Which uses speed to change the positions of the circles and I used an if statement to bounce the circles off the edges.

I think for next time I would make it more stylised making the oscillators into characters. It would also be nice to add filters as an option.

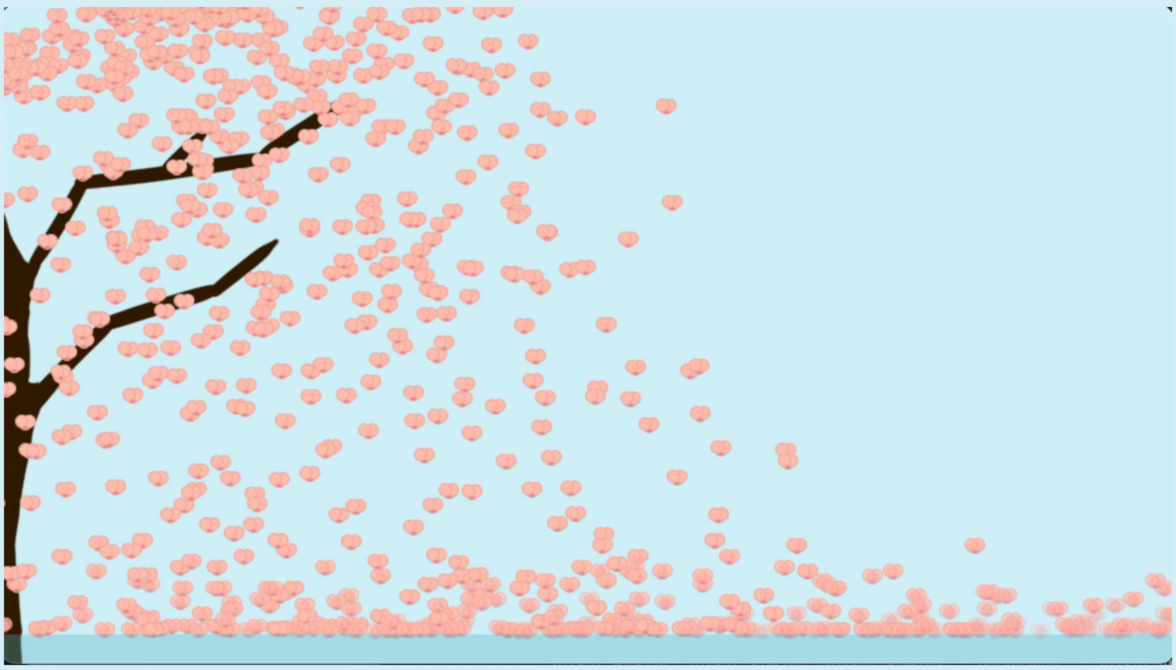


Final piece

Cherry Blossom

For this project, I wanted to do something with water. I tried to use sin and cos to create a circular motion in the current from a wave. However, I was not successful with it and ended up using velocity to create a straightforward water current motion. I also added a wind force when "W" is clicked.

I think next time I would try to figure out the wave force with the circular motion. Also make the particle systems produce less particles.



Final piece