University of California, Santa Cruz

Experiment 4:

Images, Video, and Sound Art

Dominic Berardi & Nathan Prieto

CMPM 169: Creative Coding

Modes

Due Date 2/7/2023

Galaxy Rave

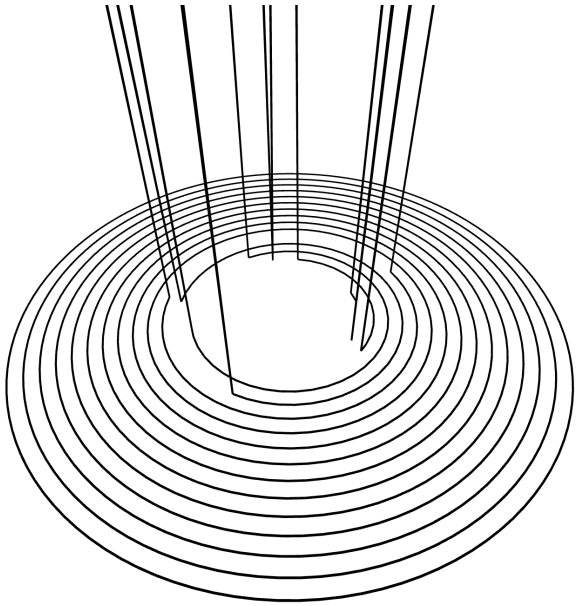
A sound art piece that combines FFT analysis with WebGL interaction to explore a “rave galaxy”. Also integrates some image code, using GIFs.

Playable link:
https://ncprieto.github.io/Experiment-4-CMPM-169-Rave-Galaxy-Nathan-Prieto-Dominic-Berardi/
Sketch.js code: https://github.com/domberardi10/cmpm169/blob/master/experiment4/js/sketch.js

Step 1: Imitate

For this assignment, there were two different examples that stuck out to us: using the webcam to create really awesome art, and using sound analysis to create reactionary art pieces. In discussing which we wanted to go with, we realized that neither of us have working webcams, and thus video was out of the question. The sound art that visually stuck out to us was this piece, “Concentric Sound”, which utilized Fast Fourier Transform (FFT) to analyze the sound coming through the viewer’s microphone. https://openprocessing.org/sketch/1782562

We really just wanted to play around with the piece at first, to learn exactly how it worked in the first place. This meant changing any and all values we could see could be changed. For example, the below left image depicts editing the “layers” constant value to be larger (more circles), changing the color to be monochrome, and then upping the intensity factor of the Z-axis additions (the “jumps” or “spikes” seen above), which is dictated by the spectrum array– in other words, the values of the analyzed sounds coming into the program (below right image).

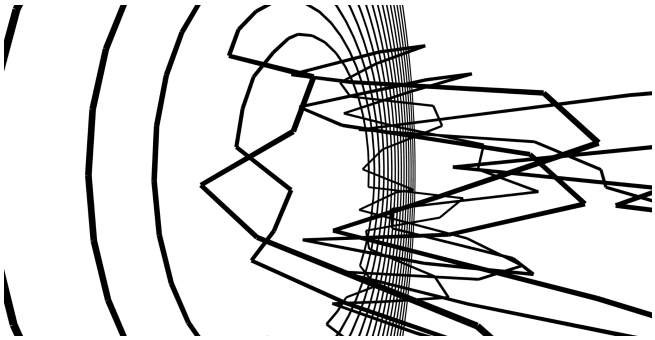


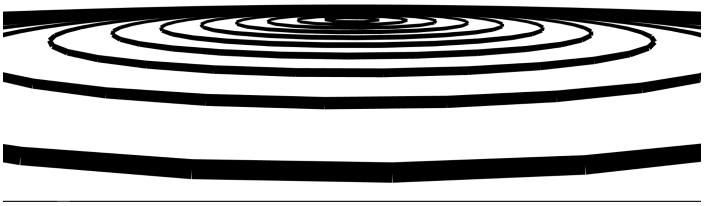
```
if (i < spectrum.length){
    z += spectrum[i++]*1000;
    smooth();
    stroke(0, 0, 0);
}
```

At this point, we have a good understanding of how the code worked. After taking in the user's audio and analyzing it using FFT, a spectrum array is given. The whole shape is rotated like a record using rotateZ(), and the orientation of the user's view is changed using rotateX(). The spectrum array is shifted a few times for variance's sake, and then the number of layers is iterated through– this tells us how many circles are drawn. The nested for loop is what calculates the angles for each circle segment, and then draws those segments using vertex(). This is also where the formerly dubbed "jumps" occur.

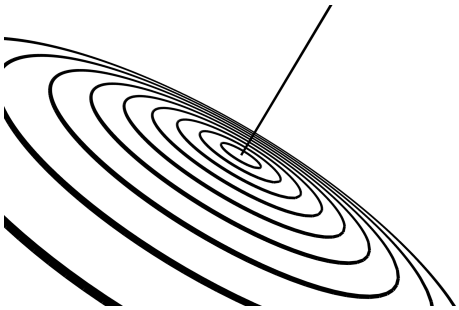
```
function draw(){
    background(255);
    stroke(0);
    strokeWeight(4)
    noFill();
    rotateX(0.25*PI);
    rotateZ(frameCount/80);
    let d = 30;
    let x,y,z;
    let spectrum = fft.analyze();
    spectrum.shift();
    spectrum.shift();
    spectrum.shift();
    smooth();
    let i = 0;
    for (let k = 0; k < layers; k++){
      beginShape();
      for (let a = 0; a < 2*PI; a+= 1/k * 1){
          x = d*k*cos(a);
          y = d*k*sin(a);
          z = k*cos(frameCount/1);
          //z *= noise(x/30,y/30);
          if (i < spectrum.length){
              z += spectrum[i++]*1000;
              smooth();
              stroke(0, 0, 0);
          }
          vertex(x,y,z);
```

We continued to play with values to find something that caught our eye. These next images involve changing line weights, rotation angles, and circular angles.



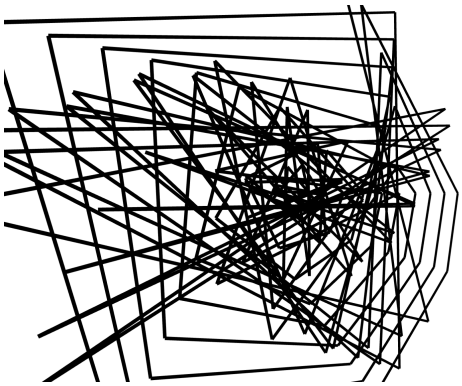


This last image reminded us of a sort of massive galaxy, viewed from an almost flat angle. Changing the angle was, again, just using the rotateX and rotateY functions (though they had to be converted to degrees to easily work with them). Could we make these sound circles… sound galaxies?

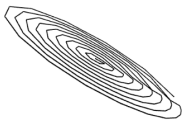


```
angleMode(DEGREES);
rotateX(75);
rotateY(30);
rotateZ(frameCount/10);
```

Galaxies follow a sort of spiral pattern, but we currently still have circles. However, the next drawn angle of each circle segment could be modified; instead of drawing a complete circle using the correct radians, we can add some value to the radians so that it never actually completes the circle. Though, we found that the added radians have to be quite large, or your spiral looks like… this.

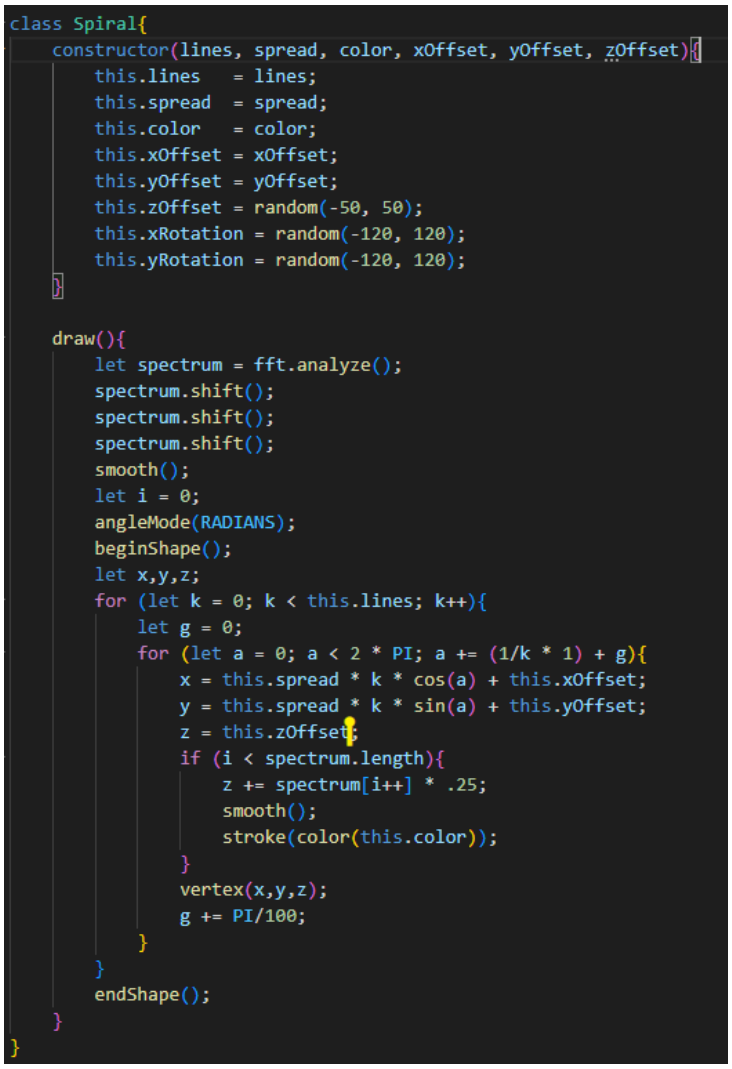


Making the added radians (the g variable, added to the a variable as part of the inner for loop) a much larger value such as PI/100 create a smoother spiral (though, not perfect). The spirals also had to be a bit smaller to look like distant galaxies, to which making the amount of layers (k) and the distance between each line segment (d) smaller also made the entire spiral smaller. We had a small, singular spiral sound galaxy!



Step 2: Integrate

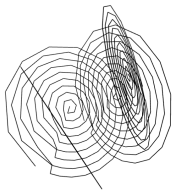
So, we only have one lone, singular galaxy, which by itself takes quite a block of code to create. We want more. What's the best way to easily create more of an existing thing? OOP Conversion Time!



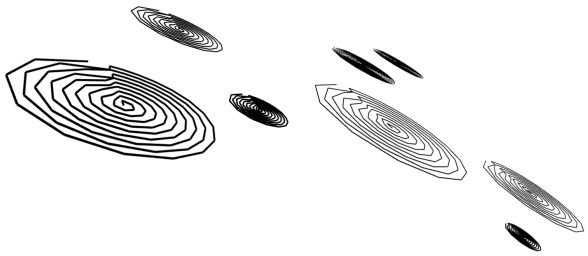
```
class Spiral{
    constructor(lines, spread, color, xOffset, yOffset, zOffset){
        this.lines   = lines;
        this.spread  = spread;
        this.color   = color;
        this.xOffset = xOffset;
        this.yOffset = yOffset;
        this.zOffset = random(-50, 50);
        this.xRotation = random(-120, 120);
        this.yRotation = random(-120, 120);
    }

    draw(){
        let spectrum = fft.analyze();
        spectrum.shift();
        spectrum.shift();
        spectrum.shift();
        smooth();
        let i = 0;
        angleMode(RADIANS);
        beginShape();
        let x,y,z;
        for (let k = 0; k < this.lines; k++){
            let g = 0;
            for (let a = 0; a < 2 * PI; a += (1/k * 1) + g){
                x = this.spread * k * cos(a) + this.xOffset;
                y = this.spread * k * sin(a) + this.yOffset;
                z = this.zOffset;
                if (i < spectrum.length){
                    z += spectrum[i++] * .25;
                    smooth();
                    stroke(color(this.color));
                }
                vertex(x,y,z);
                g += PI/100;
            }
        }
        endShape();
    }
}
```

The Spiral class was created to contain all of the information that a galaxy holds within. Lines and spread are essentially layers and distance, just renamed. Color is obvious, though it hasn't been changed at all yet. The x, y and z offsets dictate how far from the center the spiral is created at. These values are added to the x, y and z vertexes during the drawing process. As a whole, the draw function can be a part of the Spiral class itself, as nothing else is being drawn in regards to an individual Spiral.

The OOP conversion worked fine, but there was a glaring issue in what our plan was. We wanted to randomly place Spirals everywhere on the canvas, each with their own rotations. But given how the spiral drawing process was set up and how rotations work within p5 in a 3D space, this was causing rotations to be applied multiplicatively– as in, laters Spirals were rotating around rotating Spirals, which were rotating around rotating Spirals, and so on– it was a mess. A three dimensional mess.

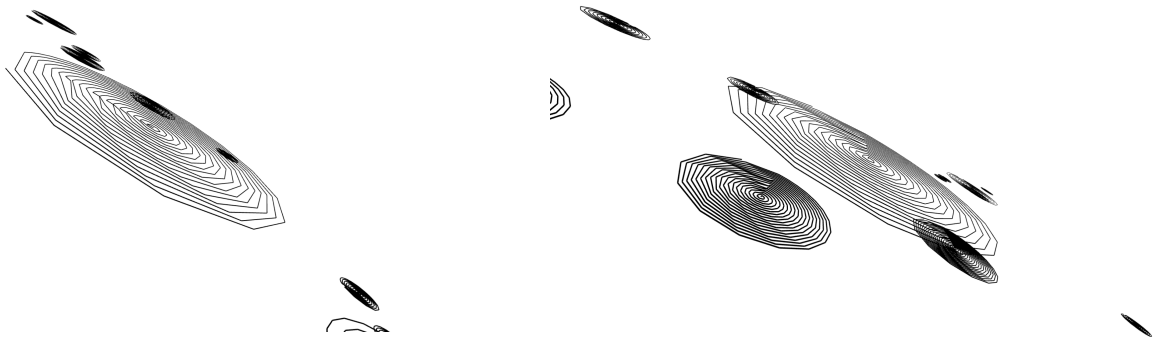


After trying to fix this by saving the canvas using push() and pop() but to no avail, we found a compromise. Instead of having each galaxy rotate independently, we could create a "galaxy of galaxies" where they all rotate around a single rotating one. At first, we hard coded in galaxy placement values to ensure the idea had any semblance of working and looking good. And it did!



Randomizing the offsets and placements of the Spirals was a trickier task, as they had to stay within certain bounds and orientations. Thus we created the GetOffsets() function, which assists in choosing semi-random values for valid offsets.

```
GetOffsets(flag){
    if(!flag){
        this.xOffset = 1;
        this.yOffset = 1;
        return;
    }
    else{
        let a = Math.floor(random(0, 2)) == 0;
        let b = Math.floor(random(0, 2)) == 0;
        if(!a && !b){
            this.xOffset = random(-500, -1000);
            this.yOffset = random(-200, -1000);
        }
        else if(!a && b){
            this.xOffset = random(-500, 1000);
            this.yOffset = random(-500, 1000);
        }
        else if(a && !b){
            this.xOffset = random(500, -1000);
            this.yOffset = random(500, -1000);
        }
        else{
            this.xOffset = random(500, 1000);
            this.yOffset = random(500, 1000);
        }
    }
```



Step 3: Innovate

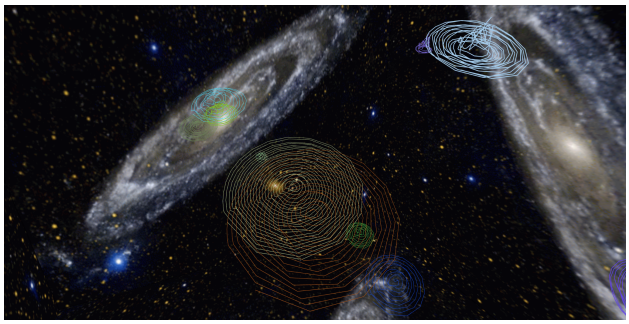
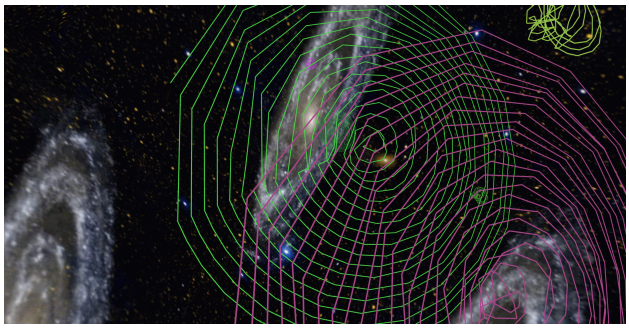
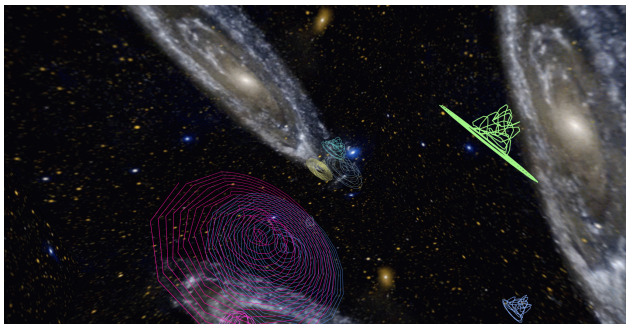
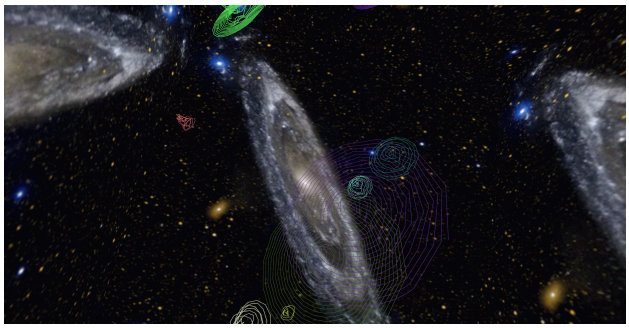
For the final step, our plan was to add some sort of image or video of a galaxy that would act as the background. We found a gif of a galaxy, but placing it as an image meant that the z-axis could not be changed– our WebGL shapes would clip through (below is a demonstration of what would happen, not ours however).

We found that this could be solved by placing the gif as a texture of a cube, and making the cube so large that the far side acts as the background. In researching how to do this, we found an awesome WebGL function called orbitControl() that allows the user to move around the origin point. This ended up making our galaxies be “explorable” but of course made the background cube gif more of a skybox. We liked how it worked, though, and decided: “Screw it, let’s just make a galaxy rave thing.”

Randomize the colors every frame of each Spiral, get some electronic music (https://freesound.org/people/Creeper_Ciller78/sounds/346895/) and make the spirals react. Add the preload() function…

```
function preload(){
    gif = loadImage("galaxy.gif");
    createGif = createImg("galaxy.gif");
    rave = loadSound('rave.wav');
    rave.setVolume(0.5);
    rave.loop();
}

function setup(){
    rave.play();
    createCanvas(windowWidth,windowHeight,WEBGL);
    fft = new p5.FFT();
    userStartAudio();
    fft.setInput(rave);
    galaxies.push(new Spiral(20, 25, "#ffffff", false));
    galaxies.push(new Spiral(10, 20, "#ffffff", true));
    galaxies.push(new Spiral(20, 7, "#ffffff", true));
    galaxies.push(new Spiral(10, 10, "#ffffff", true));
    galaxies.push(new Spiral(10, 5, "#ffffff", true));
    galaxies.push(new Spiral(20, 7, "#ffffff", true));
    galaxies.push(new Spiral(5, 10, "#ffffff", true));
    galaxies.push(new Spiral(10, 20, "#ffffff", true));
    galaxies.push(new Spiral(25, 20, "#ffffff", true));
    galaxies.push(new Spiral(20, 7, "#ffffff", true));
    galaxies.push(new Spiral(10, 10, "#ffffff", true));
    galaxies.push(new Spiral(10, 5, "#ffffff", true));
}
```

Boom. Cool, crazy, randomly inspired thing that reacts to music.

Reflect

Dominic:

In this experiment, I honestly did not do too much coding, Nate handled most of it. However, I did try to be the passenger as often as I could. I mainly wrote up the doc while we made progress on the piece. I did ensure I understood all of the code being written either way. We hit some barriers in the middle of making this, stemming from being unsure exactly how to calculate spiral patterns while staying within how the code worked. A high point was getting the electronic music in and reactive– though that part was not hard, it was super satisfying to see the end result and cool music. The lowest point was that I am super hungry writing this right now.

Nathan:

In this experiment Dominic and I tackled a reference project in 3D that uses WebGL for rendering as we were both inspired by the original and decided to expand upon it. We spent a good amount of time understanding the base code to get a sense of how the original project created the placed vertices to create lines as well as how they were animated. To me this task was somewhat daunting as I thought that adding a third dimension was going to be complicated to

understand, but ultimately turn out to be very easy to grasp and didn't create to much extra confusion. We did run into a bit of trouble when attempting to individually tilt and rotate each spiral. We decided creating a singular central spiral with many surrounding it to be the best option due to canvas limitations and our own lack of understanding the difficulties of WebGL. This was a great project to work on and seeing the end result was a great first attempt at creating something in 3D using P5.

Self-Evaluation

Dominic

Self Evaluation Rubric						
Did you complete the assignment and did you complete it on time?	**Submitted on time**	**Up to 1 day late**	**Up to 2 days late**	**Up to 3 days late**	**4 days late or more**	**Do you need to clarify?** **No**
	X	☐	☐	☐	☐	
Did you collaborate with a partner?	**Worked with partner**			**Worked alone**		**Do you need to clarify?** **No**
	X			☐		
Did you put in earnest effort and provide an articulate summary of your experience?	**Excellent**	**Pretty good**	**About average**	**Could be improved**	**Not this time**	**What supports this?** **I tried to explain my process as thoroughly as possible. Used many pictures.**
	X	☐	☐	☐	☐	
Was the assignment complete, with minimal errors, correct output, and good style?	**Excellent**	**Pretty good**	**About average**	**Could be improved**	**Not this time**	**What supports this?** **Had two people look over each part of the assignment.**
	X		☐	☐	☐	
How much EXTRA effort did you put into the assignment?	**A lot of extra effort**		**Some extra effort**		**Not this time**	**What supports this?** **Just wanted it to be done after a while.**
	☐		X		☐	

Reflection in section above.

Nathan

Self Evaluation Rubric						
Did you complete the assignment and did you complete it on time?	**Submitted on time**	**Up to 1 day late**	**Up to 2 days late**	**Up to 3 days late**	**4 days late or more**	**Do you need to clarify?** **No**
	X	☐	☐	☐	☐	
Did you collaborate with a partner?	**Worked with partner**			**Worked alone**		**Do you need to clarify?** **No**
	X					
Did you put in earnest effort and provide an articulate summary of your experience?	**Excellent**	**Pretty good**	**About average**	**Could be improved**	**Not this time**	**What supports this?** **Our process is well documented with adequate pictures that show every step of the project.**
	X	☐	☐	☐	☐	
Was the assignment complete, with minimal errors, correct output, and good style?	**Excellent**	**Pretty good**	**About average**	**Could be improved**	**Not this time**	**What supports this?** **The writeup is sufficient enough for what the teaching team expects and our code is neatly organized.**
	X		☐	☐	☐	
How much EXTRA effort did you put into the assignment?	**A lot of extra effort**		**Some extra effort**		**Not this time**	**What supports this?** **We really put in the honest effort into creating code that adheres to modern programming practices. Our project is certainly unique but definitely has room for extra features.**
	☐		**X**		☐	

Reflection in section above.