

Artist Name: Tess Baker

Title: *Rings of Resonance, Flowers of Resonance*

Medium: Generative Drawing and Sound Visualization

Size: 36 in. x 108 in. x 1/8 in.

Artist Statement:

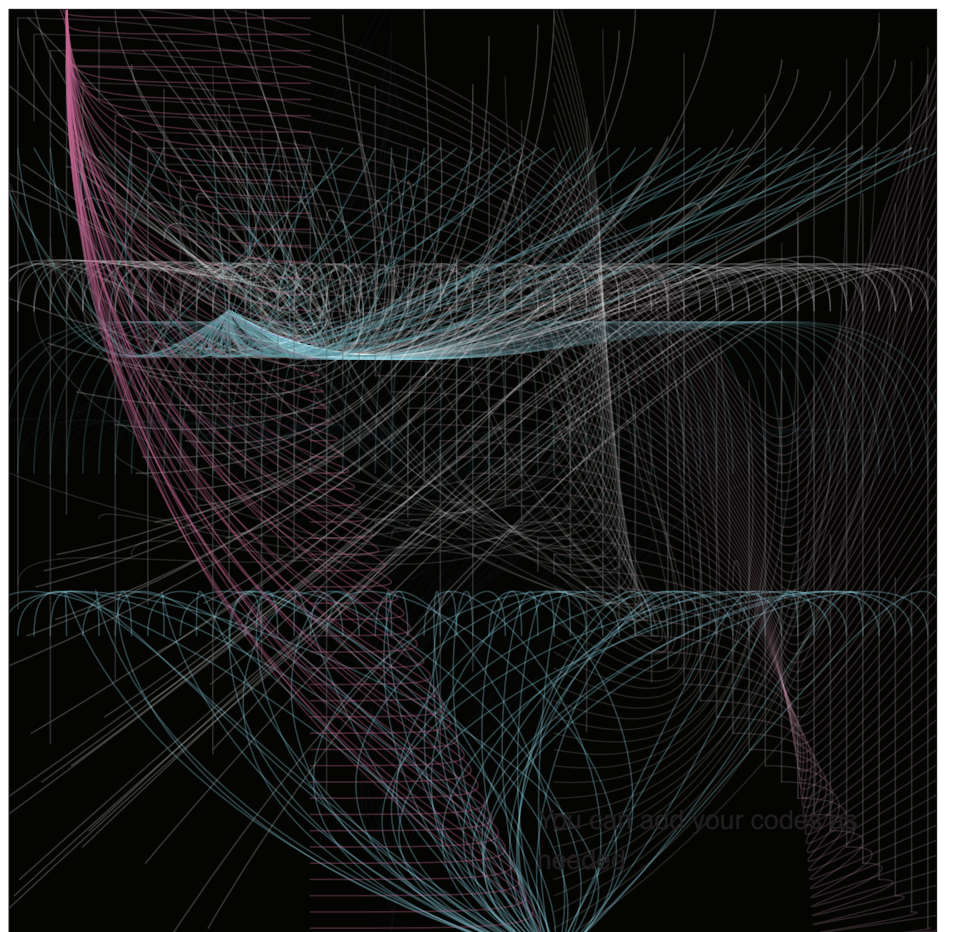
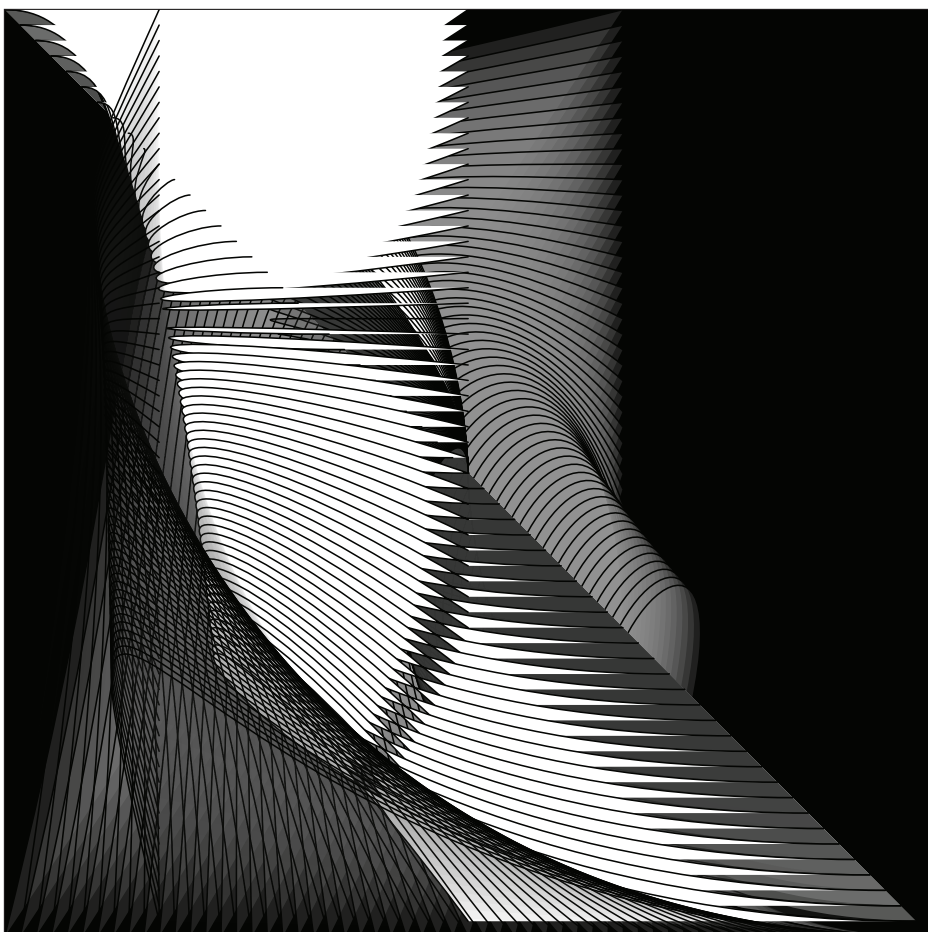
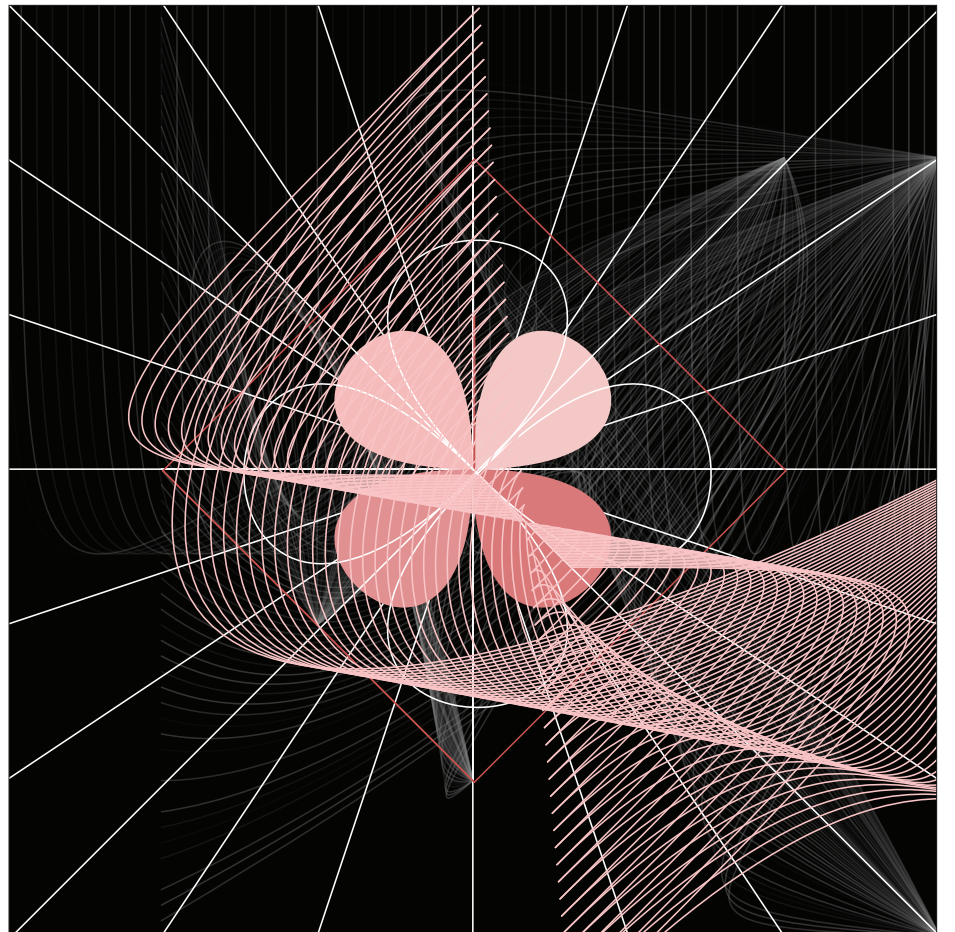
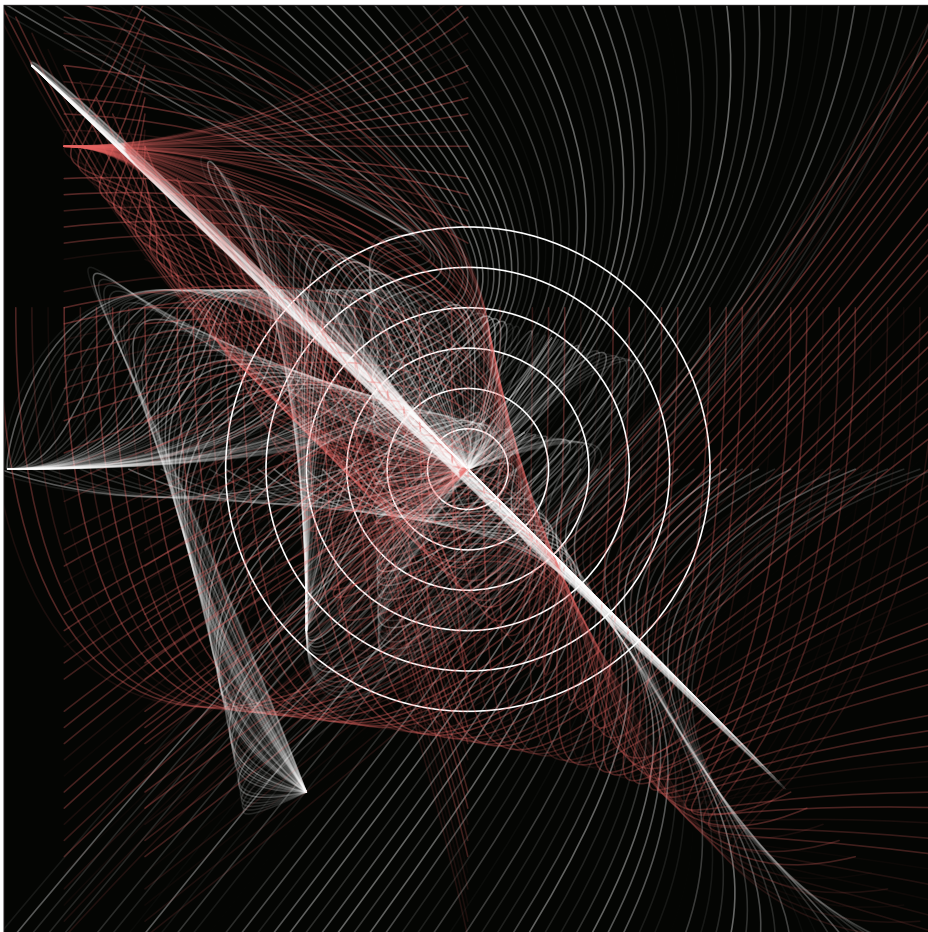
In *Rings of Resonance* and *Flower of Resonance*, the two together make a diptych that reacts to sound created by the viewer. The works explore computational drawing with the implementation of sound visualization through real time audio input by audience. The drawing utilizes an abstract style with both geometric and organic forms through the use of bezier curves and other shapes such as ellipses and line within a composition. When the audience creates sound interaction, the rings and flower create a visual for resonance – the louder the sound the larger the shape created, or movement initiated. It is intended to utilize visual movement and physical sound in order to allow the audience to interact themselves with the work – the audience can choose whether to be quiet or loud to initiate how the piece returns the interaction.

Technical Statement:

The interaction utilizes a basic programming language that was initiated by Ben Fry and Casey Reas, a software called Processing. To implement the interaction of sound, it uses the Processing Foundation's Audio Input library by using a real time microphone input.

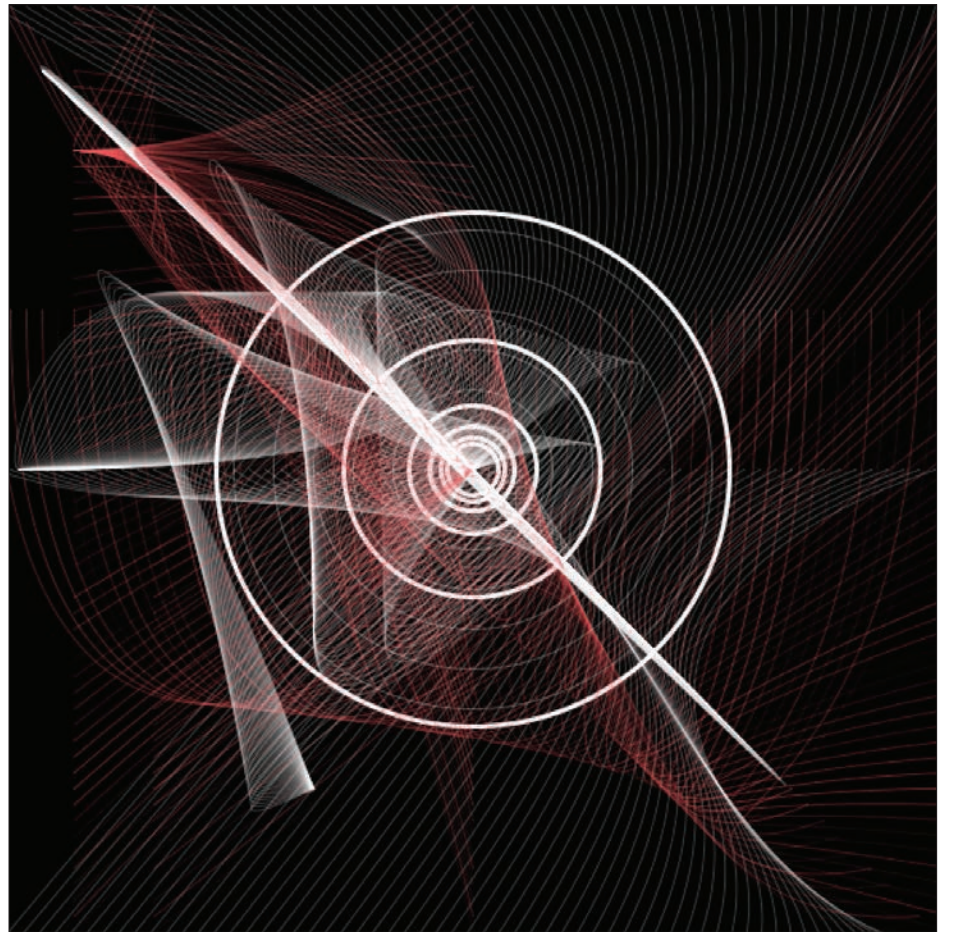
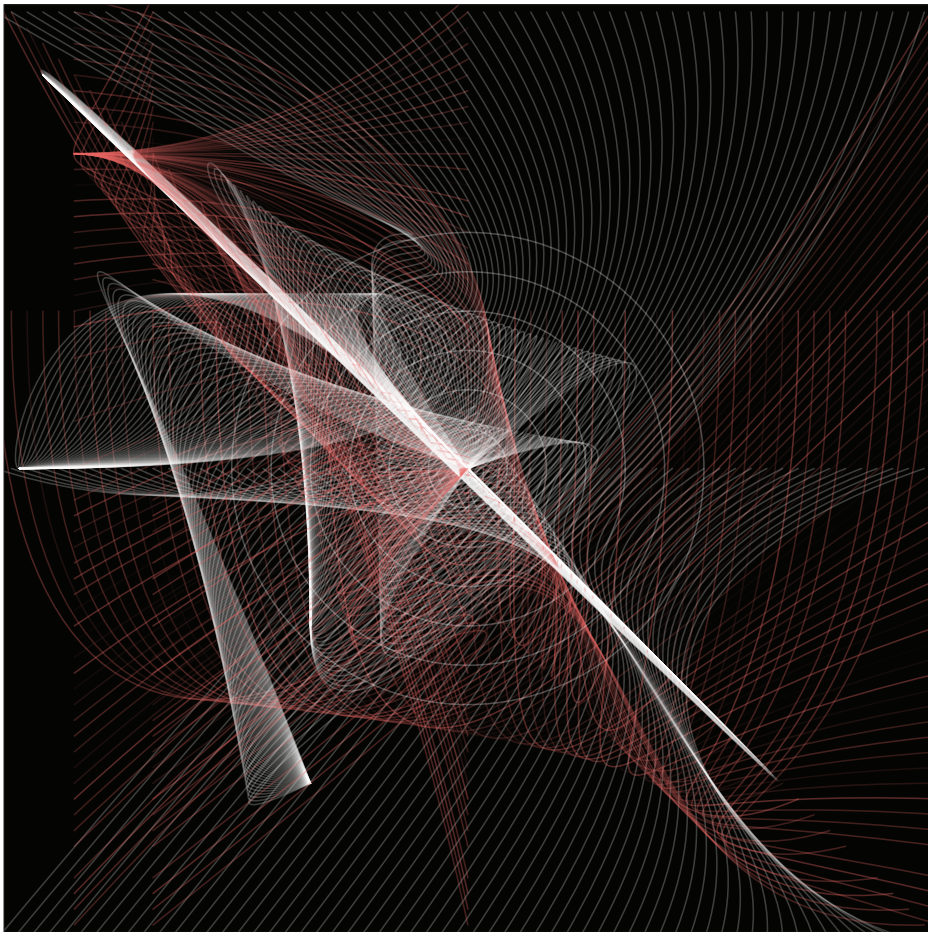
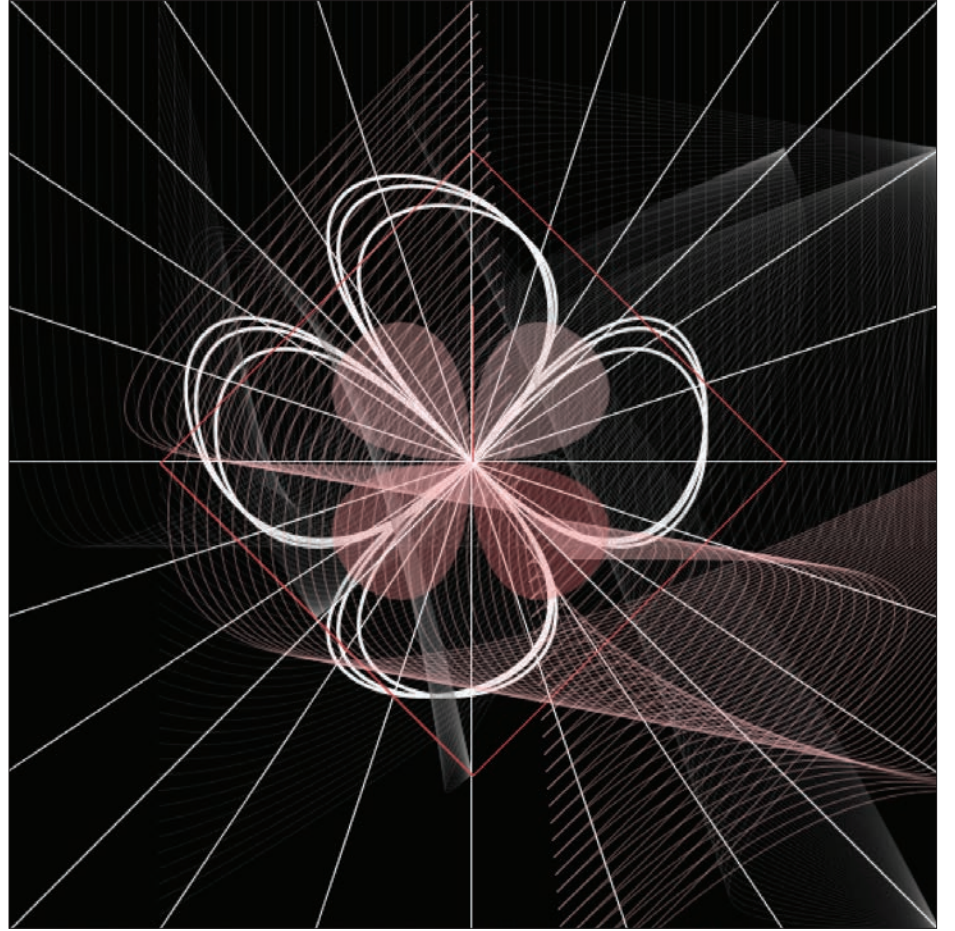
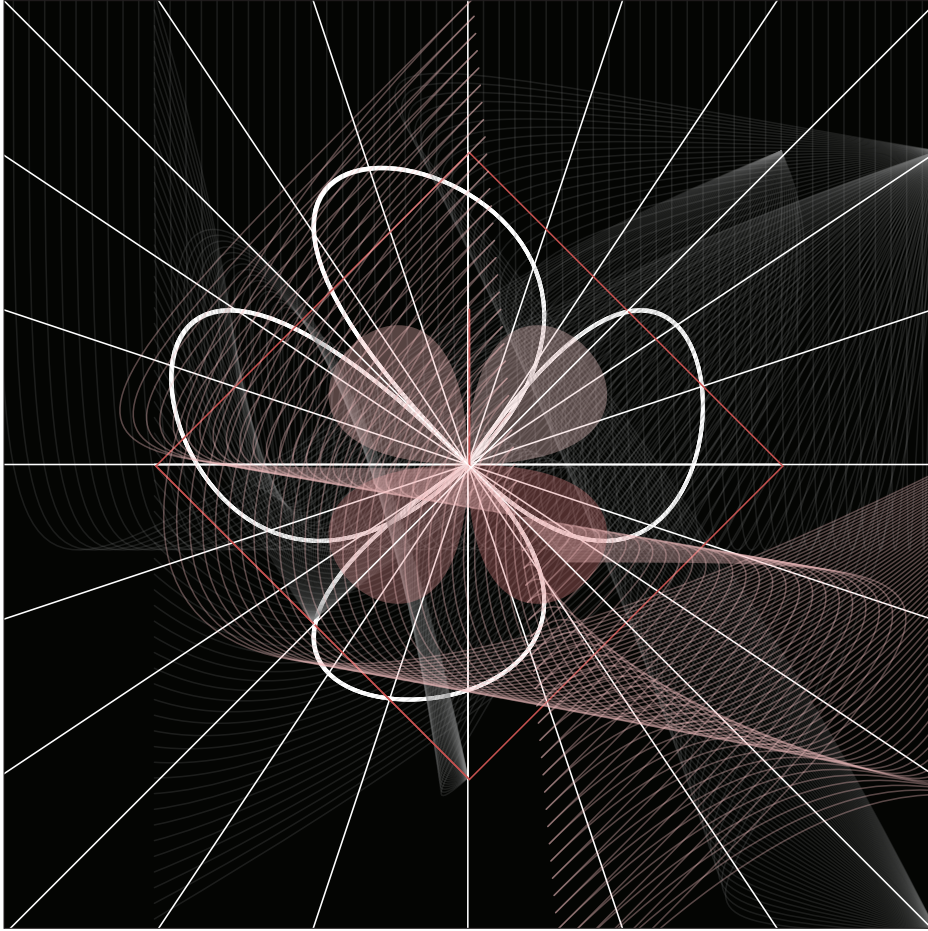
1. Computational Drawings

Initial Drawings and Sketches using computer code from Project 1.



2. Sound Visualization

Expansion of rings when sound is created.



```

AudioIn input;
Amplitude loudness;

void setup() {
  size(600, 600);
  background(0);
  beginRecord(PDF, "Project2Pt1.PDF");
  smooth();

  input = new AudioIn(this, 0);

  // Begin capturing the audio input
  input.start();
  // start() activates audio capture so that you can use it as
  // the input to live sound analysis, but it does NOT cause the
  // captured audio to be played back to you. If you also want the
  // microphone input to be played back to you, call
  // input.play();
  // instead (be careful with your speaker volume, you might produce
  // painful audio feedback: best to first try it out wearing headphones!)

  // Create a new Amplitude analyzer
  loudness = new Amplitude(this);

  // Patch the input to the volume analyzer
  loudness.input(input);
}

void draw() {
  // Adjust the volume of the audio input based on mouse position
  float inputLevel = map(mouseY, 0, height, 1.0, 0.0);
  input.amp(inputLevel);

  // loudness.analyze() return a value between 0 and 1. To adjust
  // the scaling and mapping of an ellipse we scale from 0 to 0.5
  float volume = loudness.analyze();
  int size = int(map(volume, 0, 0.5, 1, 350));

  background(0);
  strokeWeight(3);
  stroke(255);
  noFill();
  // We draw a circle whose size is coupled to the audio analysis
  ellipse(width/2, height/2, size, size);
  ellipse(width/2, height/2, size/2, size/2);
  ellipse(width/2, height/2, size/4, size/4);
  ellipse(width/2, height/2, size/6, size/6);
  ellipse(width/2, height/2, size/8, size/8);
  ellipse(width/2, height/2, size/10, size/10);

  for(int i=0; i<600; i+=10)
  {
    noFill();
    strokeWeight(1);
    stroke(255,255,255,60);
    bezier(300,300,50,50,i,300,15,300);
    bezier(300,300,i,200,10,i,200,500);
    bezier(300,300,100,i,500,300,i,10);
    bezier(300,300,i,50,200,i,200,400);
    bezier(300,300,600,600,i,i,30,50);
    bezier(i,600,500,i,300,400,i,300);

    stroke(230,102,100,random(100));
    bezier(300,300,40,i,i,200,100,i);
    bezier(300,i,300,i,200,100,50,100);
    bezier(i,i,300,600,500,100,50,i);
    bezier(600,i,300,500,i,600,i,200);
  }

  noFill();
  strokeWeight(1);
  stroke(255,255,255,70);
  ellipse(300,300,50,50);
  ellipse(300,300,100,100);
  ellipse(300,300,150,150);
  ellipse(300,300,200,200);
  ellipse(300,300,250,250);
  ellipse(300,300,300,300);

  endRecord();
}

```

```

ellipse(width/2, height/2, size/10, size/10);

for(int i=0; i<600; i+=10)
{
  noFill();
  strokeWeight(1);
  stroke(255,255,255,60);
  bezier(300,300,50,50,i,300,15,300);
  bezier(300,300,i,200,10,i,200,500);
  bezier(300,300,100,i,500,300,i,10);
  bezier(300,300,i,50,200,i,200,400);
  bezier(300,300,600,600,i,i,30,50);
  bezier(i,600,500,i,300,400,i,300);

  stroke(230,102,100,random(100));
  bezier(300,300,40,i,i,200,100,i);
  bezier(300,i,300,i,200,100,50,100);
  bezier(i,i,300,600,500,100,50,i);
  bezier(600,i,300,500,i,600,i,200);
}

noFill();
strokeWeight(1);
stroke(255,255,255,70);
ellipse(300,300,50,50);
ellipse(300,300,100,100);
ellipse(300,300,150,150);
ellipse(300,300,200,200);
ellipse(300,300,250,250);
ellipse(300,300,300,300);

endRecord();
}

```

```

import processing.sound.*;

AudioIn input;
Amplitude loudness;

void setup() {
  size(600, 600);
  background(0);
  beginRecord(PDF, "Project2Pt2.PDF");

  // Create an Audio input and grab the 1st channel
  input = new AudioIn(this, 0);

  // Begin capturing the audio input
  input.start();
  // start() activates audio capture so that you can use it as
  // the input to live sound analysis, but it does NOT cause the
  // captured audio to be played back to you. If you also want the
  // microphone input to be played back to you, call
  // input.play();
  // instead (be careful with your speaker volume, you might produce
  // painful audio feedback: best to first try it out wearing headphones!)

  // Create a new Amplitude analyzer
  loudness = new Amplitude(this);

  // Patch the input to the volume analyzer
  loudness.input(input);
}

void draw() {
  // Adjust the volume of the audio input based on mouse position
  float inputLevel = map(mouseY, 0, height, 1.0, 0.0);
  input.amp(inputLevel);

  // loudness.analyze() return a value between 0 and 1. To adjust
  // the scaling and mapping of an ellipse we scale from 0 to 0.5
  float volume = loudness.analyze();
  int size = int(map(volume, 0, 0.5, 1, 350));
  for(int i=0; i<600; i+=10)
  {
    background(0);
    stroke(255);
    strokeWeight(2.5);
    noFill();
    // We draw a circle whose size is coupled to the audio analysis
    bezier(width/2, height/2, size, size, 500,100,300,300);
    bezier(width/2, height/2, size/2, size/2, 100,500,300,300);
    bezier(width/2, height/2, size/4, size/4, 100,500,300,300);

    bezier(width/2, height/2, size, size, 500,500,300,300);
    bezier(width/2, height/2, size/2, size/2, 500,500,300,300);
    bezier(width/2, height/2, size/4, size/4, 500,500,300,300);

    bezier(width/2, height/2, size, size, 500,500,300,300);
    bezier(width/2, height/2, size/2, size/2, 500,500,300,300);
    bezier(width/2, height/2, size/4, size/4, 500,500,300,300);
  }
}

```

```

for(int i=0; i<600; i+=10)
{
  noFill();
  stroke(130,130,130,50);
  strokeWeight(1);
  bezier(600,100,1,300,1,600,1,0);
  bezier(600,100,1,1,300,1,600,600);
  bezier(200,400,1,100,200,1,300,500);
  bezier(100,1,250,500,1,300,500,100);
}

stroke(255,255,255);
line(300,300,100,300);
line(300,300,200,0);
line(300,300,300,0);
line(300,300,400,0);
line(300,300,500,0);

line(300,300,0,100);
line(300,300,0,200);
line(300,300,0,300);
line(300,300,0,400);
line(300,300,0,500);

line(300,300,600,100);

stroke(196,80,80);
translate(1,1);
line(300,500,100,300);
line(300,200,300,300);
line(300,500,100,300);
line(300,200,300,300);
line(500,300,300,100);
line(100,300,300,100);
line(500,300,300,500);

stroke(255,255,255);
translate(1,1);
line(300,300,600,600);
line(300,300,0,0);
line(300,300,600,0);
line(300,300,0,600);

for(int i=0; i<600; i+=10)
{
  translate(1,1);
  noFill();
  stroke(253,197,197,90);
  smooth();
  bezier(300,300,1,300,1,300,300,1);
  bezier(600,300,1,600,1,300,300,1);
}

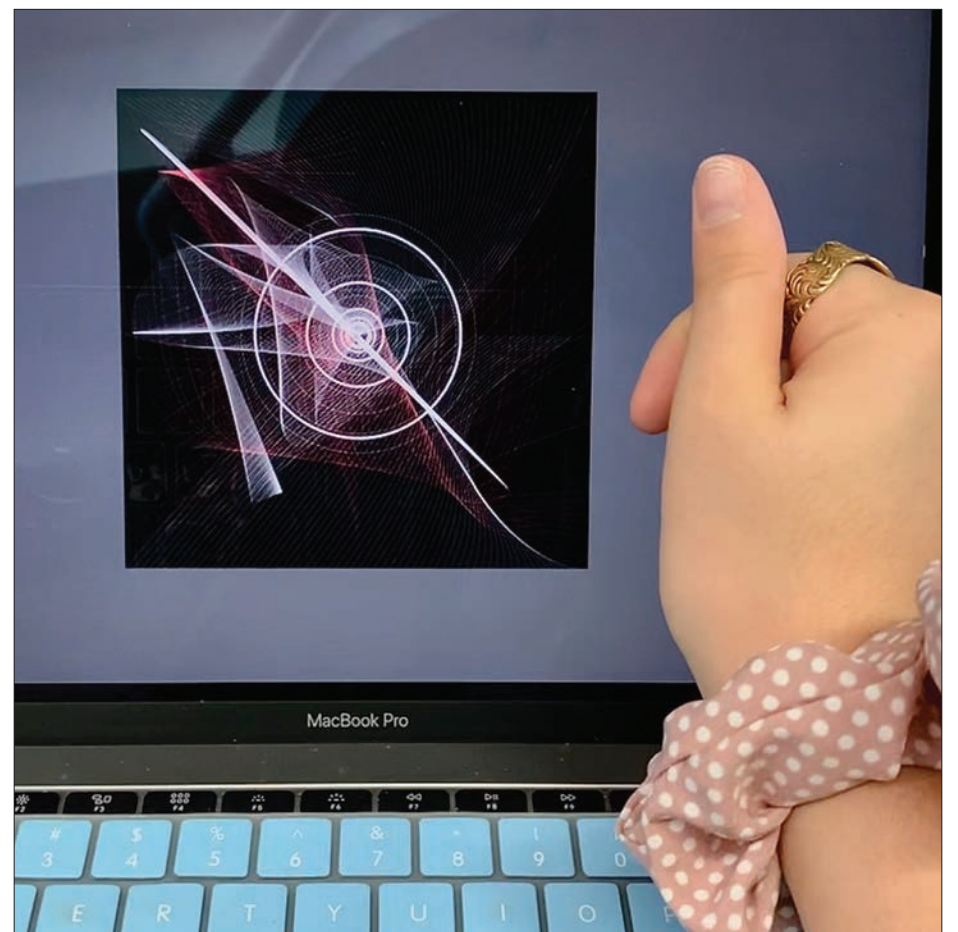
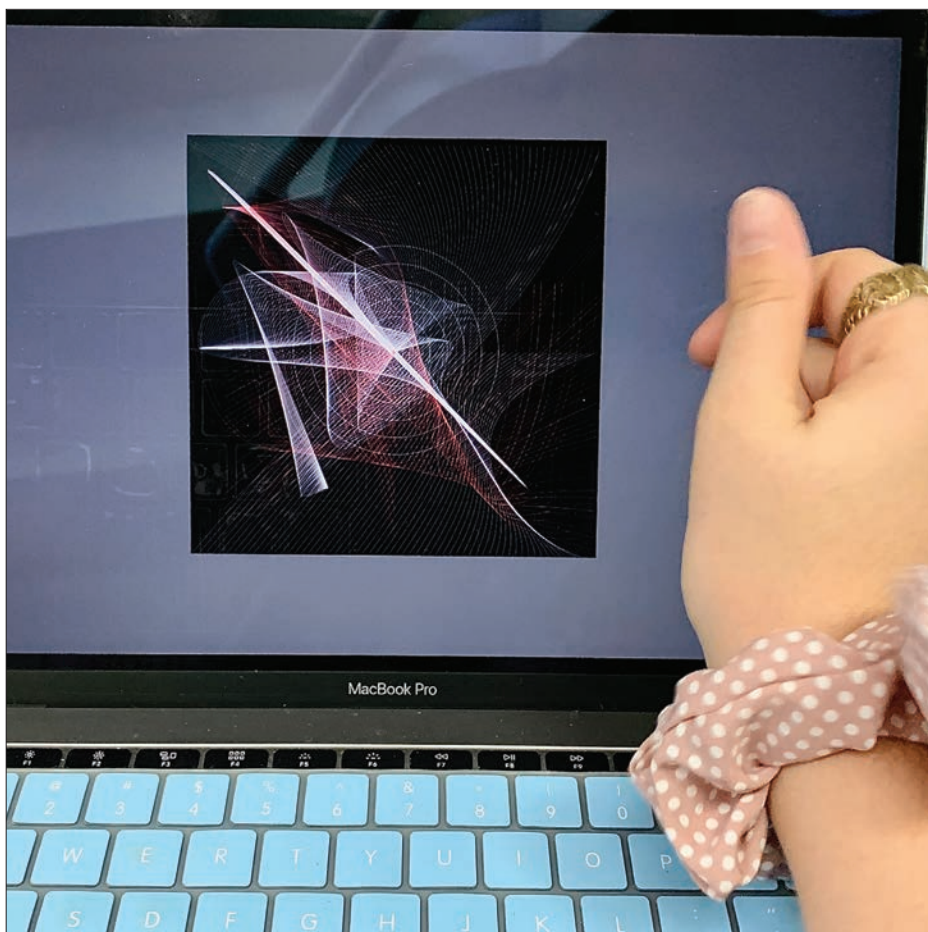
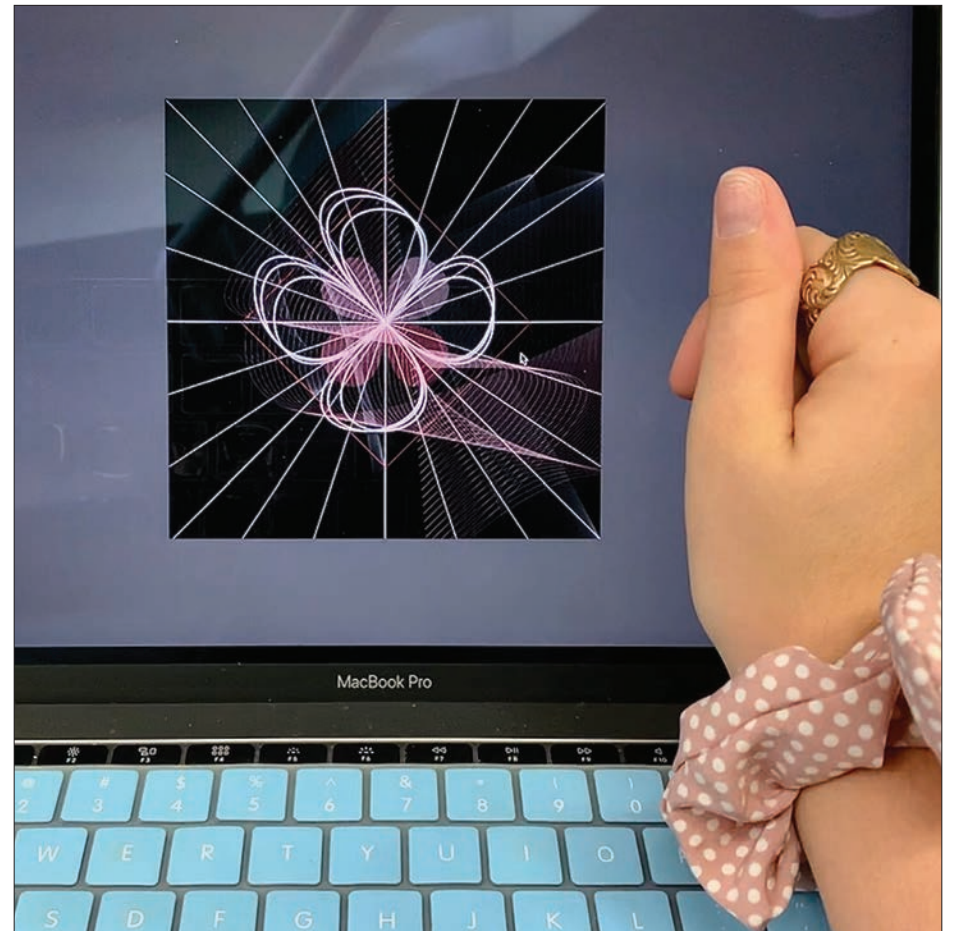
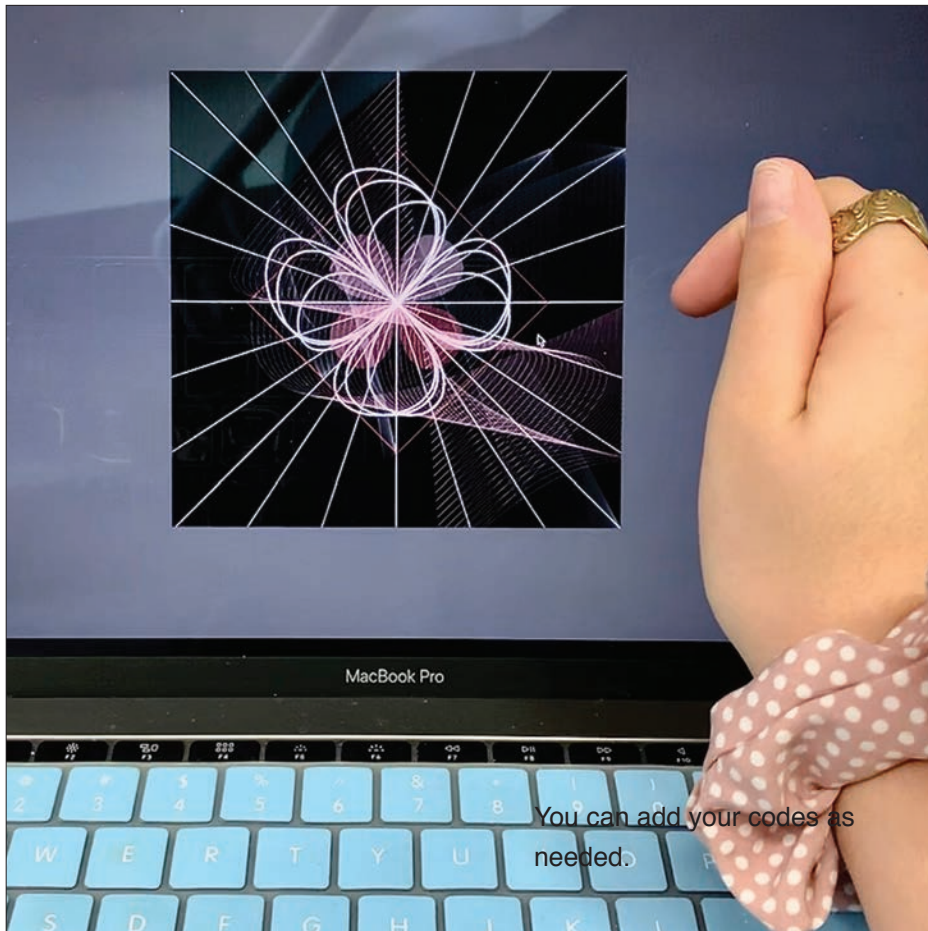
endRecord();
}

```

Sequence of computer code that creates the interactions and visuals presented above.

3. Physical Interaction

To create sound, I utilized a snap of my fingers in order to interact with the visuals. Any noise of any kind will create the visuals.



4. Gallery Plan

The two pieces will be projected onto the back wall in a dark room. Here, the viewers can create sound in order to create the visuals and interact with the work. The louder the noise, the larger the rings and more movement that is created.

