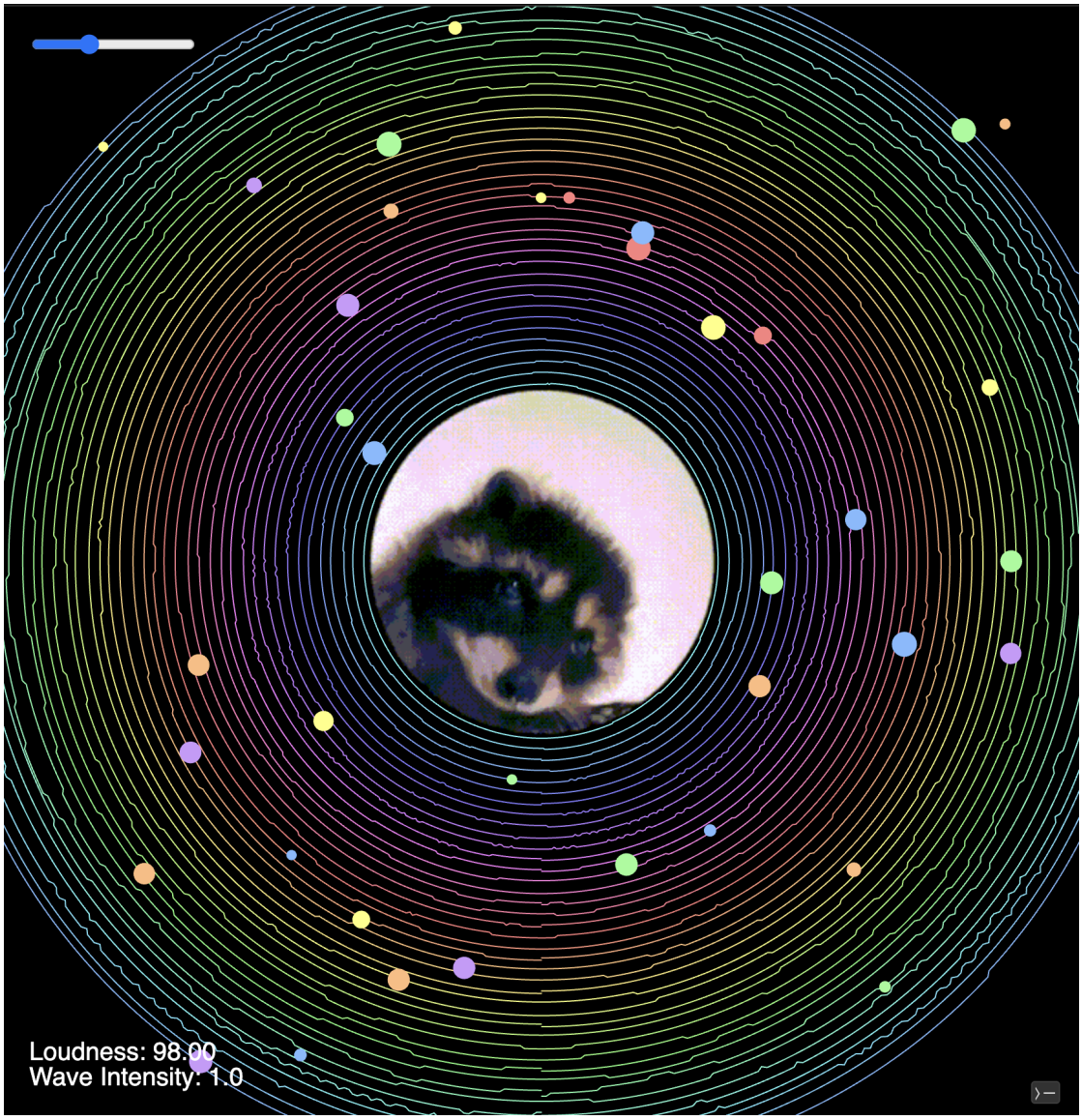
Documentation

Racoon Rave Audio Visualizer

By Nicole Lee



This audio visualizer responds to microphone input, adjusting wave oscillation speed and width based on loudness. A spinning raccoon GIF sits at the center, with particles appearing when the loudness exceeds 100. A slider in the top left allows users to zoom into the oscillating waves for better visibility (wave intensity).

Inspiration

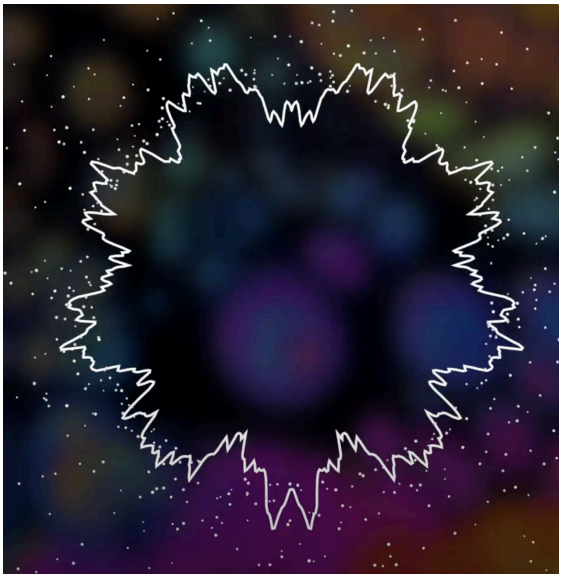
I was inspired by the 2000s CD audio visualizers that I used to watch as a kid—those colorful, hypnotic patterns that pulsed and warped in sync with the music. Back then, visualizers transformed music into something alive, turning simple sounds into mesmerizing, ever-changing animations. I wanted to recreate that feeling of nostalgia and wonder but in a way that felt more interactive and playful.

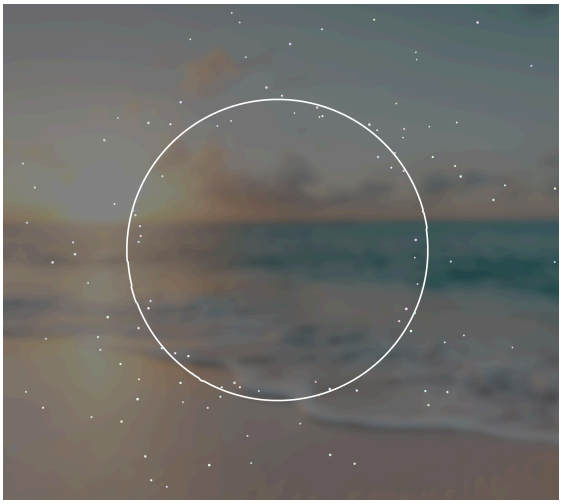
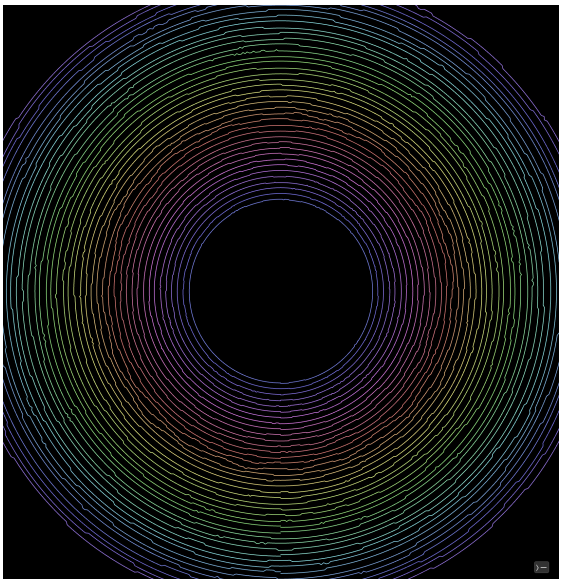
Unlike traditional visualizers that simply react to music passively, my project encourages direct engagement. The idea is to have the visual elements respond not just to music, but also to the user's voice—creating a personalized, dynamic experience. Every word or sound generates ripples of color, echoing outward like waves, evolving over time. This interaction makes the visualizer feel like a living organism, responding uniquely to each session.

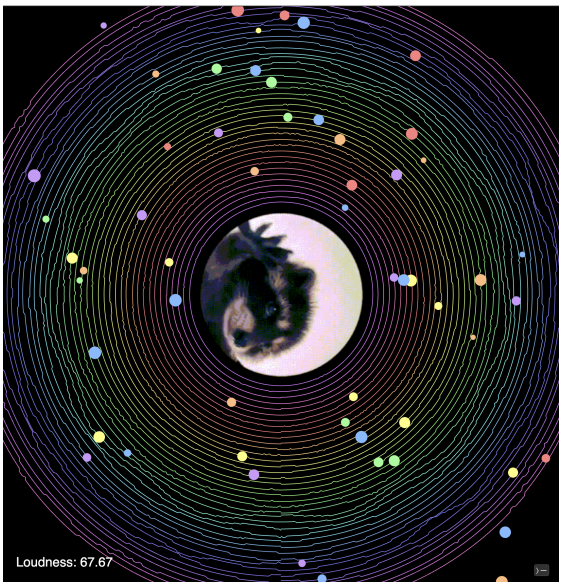
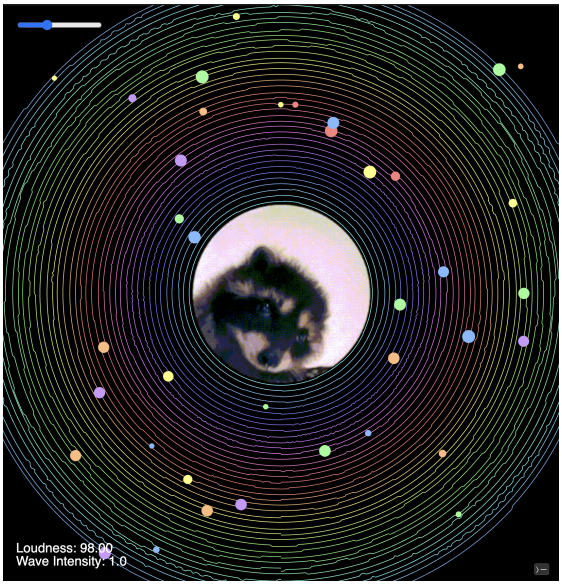
I also wanted to add a touch of playfulness and charm, which is why I incorporated a whimsical dancing raccoon GIF at the center. It brings a sense of humor and lightheartedness to the experience, breaking away from the sleek, abstract designs of old-school visualizers. The pastel-colored circles and fluid particle effects give the piece an almost dreamlike quality, reminiscent of those classic music visualizers but with a modern, interactive twist.

Ultimately, this project is a mix of nostalgia, interactivity, and personal expression—a way to visually connect with sound in a fun, engaging, and ever-changing way.

Process

Version #	Description	Photo
1	https://www.youtube.com/watch?v=uk96O7N1Yo0 This was a YouTube video that inspired my project. In this video, they upload a song (in the form of mp3) as audio, and as the beat drops in the song, the circle moves and spits out particles.	

2	https://openprocessing.org/sketch/2580745 I then coded my own version tweaking what it looks like. A very simple white circle with white particles. But this version uses your computer mic rather than plugging in your own audio.	
	https://openprocessing.org/sketch/2582176 I decided to duplicate the circles to see a cool wave/echo effect and make it a rainbow. I took out the particles and changed the background to black.	

	https://openprocessing.org/sketch/2580868 To make it fun, I added a raccoon that spins in the middle. Now when sound is generated, the sound wave is echoed outward and the particles move outward with it.	
Final	https://openprocessing.org/sketch/2581071 In this final version, the particles move a little faster. I also added a slider displaying wave intensity which zooms in or out of the circle so that you can see how much the waves oscillate/fluctuate.	
	This project began as a simple audio visualizer that reacted to sound waves using basic circular waveforms. Initially inspired by existing music visualizers, I started with a minimal design—a white circle with particle effects responding to microphone input. As I experimented, I added multiple wave layers to create a dynamic echo effect and introduced color variation to enhance visual depth. I incorporated a spinning raccoon GIF at the center to make the piece more engaging, transforming it from a standard visualizer into a playful, interactive experience. Further refinements included adjusting particle movement, optimizing wave oscillations, and adding a slider to control wave intensity, allowing users to customize the visual effect. Through this iterative process, the	

	project evolved from a technical audio analysis tool into a vibrant, interactive artwork that blends generative design with elements of fun and creativity. Problems and Issues: Throughout the iterations, I **attempted** some experimental things: ● Instead of circle particles made them stars ● Tried to gamify it ○ Move a character to collect the particles and then shoot them like agar.io ● Change the picture in the middle All of these didn't work. This visualizer was already very intense. My computer could not load it consistently since there was so much movement. I tried making the circles stars but it couldn't load and the shape was too complicated as I had to manually draw a star first as code (which had more than 1 component for each star) and then spawn each of them individually which was too intensive. Then I tried gamifying it, the character worked but the particles could not register as an "object" meaning that it didn't register as having a hitbox. That means the whole concept fell apart because you could not "collect" anything and I was left with just a moving image. This was really discouraging to figure out as I took over an hour trying to fix it but couldn't Lastly, I tried changing the picture in the middle. But it was hard to find a similar GIF that was a circle that didn't mess with the rest of the circular waves. But after a lot of messing around,d I feel that it was already enough with the wave visualizer. Making it too busy didn't look to good so I decided to scratch it off.

Conclusion / Reflection

What is the goal of this piece? Did that goal change over the course of your project? In what ways do you feel successful about achieving that goal – and what did you learn for the future?

The goal of this piece was to create an interactive audio visualizer that responds to sound in a visually engaging way. Initially, I envisioned a simple waveform visualizer, but as the project evolved, I incorporated additional elements like particle effects, wave history, and a spinning raccoon GIF to make it more playful and immersive. I feel successful in achieving the goal because the final piece is both aesthetically engaging and functionally responsive to sound. Moving forward, I've learned the importance of iterative design, experimenting with new ideas, and refining them based on what works best visually and interactively.

How do you feel about the way it turned out? Does it remind you of other artworks, media, stories, or people? How have you changed through the process of creating this piece? How do you hope pieces like this might impact others?

I'm really happy with how it turned out. It has a unique blend of structured waveforms and playful randomness with the particles and GIF. It reminds me of old-school music visualizers like those in Winamp or iTunes, but with a modern, fun twist. Through this project, I've grown in my ability to experiment with generative art, pushing myself to think beyond basic visual representations. I hope that this piece brings joy and curiosity to viewers, encouraging them to see sound in a new way and inspiring them to explore creative coding.

What's the plan for this piece?

Right now, I see this as a completed experimental piece, but I'd love to refine it further or incorporate it into a larger interactive project. One idea is to allow users to upload their own music or add more interactivity, such as enabling user-controlled visual modifications.

What will you take into future projects? Any acknowledgments for folks who helped along the way?

For future projects, I'll take away the importance of iteration—starting with a basic concept and letting it evolve naturally based on testing and feedback. I also learned a lot about balancing aesthetics with real-time responsiveness in p5.js. I'd like to acknowledge this youtube video/channel for inspiring this project https://www.youtube.com/watch?v=uk96O7N1Yo0

Links and Sources

Code	Google Drive:In open processing: : https://openprocessing.org/sketch/2581071
Audio Visualizer starting code	https://www.youtube.com/watch?v=uk96O7N1Yo0
Open Processing	Final sketch: https://openprocessing.org/sketch/2581071
CHATGPT	https://chatgpt.com/?model=text-davinci-002-render
Final video	Google Drive: Nicole Generative Art Project.mp4

FINAL CODE

```
// LED1
const int A_LEDPin = 13;
const int B_LEDPin = 12;
const int C_LEDPin = 11;

// LED2
const int D_LEDPin = 10;
const int E_LEDPin = 9;
const int F_LEDPin = 8;

// LED3
const int G_LEDPin = 7;
const int H_LEDPin = 6;
const int I_LEDPin = 5;

// LED4
const int J_LEDPin = 4;
const int K_LEDPin = 3;
const int L_LEDPin = 2;

const int buttonPin = 1; // Button pin

// RGB values for 6 categories with 4 colors each
const int categories[6][4][3] = {
  { // Reds
    {255, 20, 60},  // Crimson
    {255, 34, 34},  // Firebrick
    {255, 99, 71},  // Tomato
    {225, 92, 92}   // Indian Red
  },
  { // Oranges
    {255, 128, 0},   // Orange
    {255, 153, 51},  // Dark Orange
    {255, 178, 102}, // Red Orange
    {255, 204, 153}  // Gold
  },
  { // Yellows
    {255, 255, 0},    // Yellow
    {255, 255, 51},   // Light Yellow
    {255, 255, 102},  // Khaki
    {255, 255, 153}   // Papaya Whip
  },
  { // Greens
```

```
    {0, 255, 0},      // Green
    {51, 255, 51},    // Forest Green
    {102, 255, 102},  // Light Green
    {153, 255, 153}   // Lime Green
  },
  { // Blues
    {0, 0, 255},      // Blue
    {51, 51, 255},    // Dodger Blue
    {102, 102, 255},  // Steel Blue
    {153, 153, 255}   // Sky Blue
  },
  { // Purples
    {127, 0, 255},    // Purple
    {153, 51, 255},   // Blue Violet
    {178, 102, 255},  // Dark Violet
    {204, 153, 255}   // Medium Orchid
  }
};

int currentCategory = 0; // Tracks the current category
bool buttonPressed = false;

void setup() {
  // Initialize all LED pins as outputs
  pinMode(A_LEDPin, OUTPUT);
  pinMode(B_LEDPin, OUTPUT);
  pinMode(C_LEDPin, OUTPUT);

  pinMode(D_LEDPin, OUTPUT);
  pinMode(E_LEDPin, OUTPUT);
  pinMode(F_LEDPin, OUTPUT);

  pinMode(G_LEDPin, OUTPUT);
  pinMode(H_LEDPin, OUTPUT);
  pinMode(I_LEDPin, OUTPUT);

  pinMode(J_LEDPin, OUTPUT);
  pinMode(K_LEDPin, OUTPUT);
  pinMode(L_LEDPin, OUTPUT);

  // Initialize button pin as input with pullup resistor
  pinMode(buttonPin, INPUT_PULLUP);
```

```
  // Turn off all LEDs initially
  setAllLEDs(0, 0, 0);
}

void loop() {
  // Check if the button is pressed
  if (digitalRead(buttonPin) == LOW && !buttonPressed) { // Button pressed
    delay(200); // Debounce delay
    buttonPressed = true; // Prevent multiple triggers
    displayCategory(currentCategory); // Display the current category
    currentCategory = (currentCategory + 1) % 6; // Move to the next category
    while (digitalRead(buttonPin) == LOW); // Wait until button is released
    buttonPressed = false;
  }
}

// Function to display the current category
void displayCategory(int category) {
  setLEDColor(A_LEDPin, B_LEDPin, C_LEDPin, categories[category][0][0],
categories[category][0][1], categories[category][0][2]); // LED1
  setLEDColor(D_LEDPin, E_LEDPin, F_LEDPin, categories[category][1][0],
categories[category][1][1], categories[category][1][2]); // LED2
  setLEDColor(G_LEDPin, H_LEDPin, I_LEDPin, categories[category][2][0],
categories[category][2][1], categories[category][2][2]); // LED3
  setLEDColor(J_LEDPin, K_LEDPin, L_LEDPin, categories[category][3][0],
categories[category][3][1], categories[category][3][2]); // LED4
}

// Function to set the color of an LED
void setLEDColor(int redPin, int greenPin, int bluePin, int redValue, int greenValue, int blueValue)
{
  analogWrite(redPin, redValue);   // Set red pin
  analogWrite(greenPin, greenValue); // Set green pin
  analogWrite(bluePin, blueValue); // Set blue pin
}

// Function to set all LEDs to the same color
void setAllLEDs(int red, int green, int blue) {
  analogWrite(A_LEDPin, red);
  analogWrite(B_LEDPin, green);
  analogWrite(C_LEDPin, blue);

  analogWrite(D_LEDPin, red);
```

```
  analogWrite(E_LEDPin, green);
  analogWrite(F_LEDPin, blue);

  analogWrite(G_LEDPin, red);
  analogWrite(H_LEDPin, green);
  analogWrite(I_LEDPin, blue);

  analogWrite(J_LEDPin, red);
  analogWrite(K_LEDPin, green);
  analogWrite(L_LEDPin, blue);
}
```

Other Versions

VERSION 1→ hooking up the lights

```
//LED1
const int A_LEDPin = 13;
const int B_LEDPin = 12;
const int C_LEDPin = 11;

//LED2
const int D_LEDPin = 10;
const int E_LEDPin = 9;
const int F_LEDPin = 8;

//LED3
const int G_LEDPin = 7;
const int H_LEDPin = 6;
const int I_LEDPin = 5;

//LED4
const int J_LEDPin = 4;
const int K_LEDPin = 3;
const int L_LEDPin = 2;

// Set your desired color here
int redValue = 0;   // Set to 255 for red intensity, or 0 if you don't want red
int greenValue = 0; // Set to 255 for green intensity, or 0 if you don't want green
int blueValue = 255; // Set to 255 for blue intensity, or 0 if you don't want blue

void setup() {
  // Initialize all LED pins as outputs
  pinMode(A_LEDPin, OUTPUT);
  pinMode(B_LEDPin, OUTPUT);
  pinMode(C_LEDPin, OUTPUT);

  pinMode(D_LEDPin, OUTPUT);
  pinMode(E_LEDPin, OUTPUT);
  pinMode(F_LEDPin, OUTPUT);

  pinMode(G_LEDPin, OUTPUT);
```

```
  pinMode(H_LEDPin, OUTPUT);
  pinMode(I_LEDPin, OUTPUT);

  pinMode(J_LEDPin, OUTPUT);
  pinMode(K_LEDPin, OUTPUT);
  pinMode(L_LEDPin, OUTPUT);

  // Turn all LEDs to the same color
  setAllLEDs(redValue, greenValue, blueValue);
}

void loop() {
  // No changes in loop since we want all LEDs to stay on in one color
}

// Function to set all LEDs to the same color
void setAllLEDs(int red, int green, int blue) {
  // Set all red pins
  analogWrite(A_LEDPin, red);
  analogWrite(D_LEDPin, red);
  analogWrite(G_LEDPin, red);
  analogWrite(J_LEDPin, red);

  // Set all green pins
  analogWrite(B_LEDPin, green);
  analogWrite(E_LEDPin, green);
  analogWrite(H_LEDPin, green);
  analogWrite(K_LEDPin, green);

  // Set all blue pins
  analogWrite(C_LEDPin, blue);
  analogWrite(F_LEDPin, blue);
  analogWrite(I_LEDPin, blue);
  analogWrite(L_LEDPin, blue);
}
```

VERSION 2 → turning all the lights on as blue

Assignment title Student name - 5

```
const int A_LEDPin = 13;
const int B_LEDPin = 12;
const int C_LEDPin = 11;

const int D_LEDPin = 10;
const int E_LEDPin = 9;
const int F_LEDPin = 8;

const int G_LEDPin = 7;
const int H_LEDPin = 6;
const int I_LEDPin = 5;

const int J_LEDPin = 4;
const int K_LEDPin = 3;
const int L_LEDPin = 2;

void setup() {
  // Initialize all LED pins as outputs
  pinMode(A_LEDPin, OUTPUT);
  pinMode(B_LEDPin, OUTPUT);
  pinMode(C_LEDPin, OUTPUT);

  pinMode(D_LEDPin, OUTPUT);
  pinMode(E_LEDPin, OUTPUT);
  pinMode(F_LEDPin, OUTPUT);

  pinMode(G_LEDPin, OUTPUT);
  pinMode(H_LEDPin, OUTPUT);
  pinMode(I_LEDPin, OUTPUT);

  pinMode(J_LEDPin, OUTPUT);
  pinMode(K_LEDPin, OUTPUT);
  pinMode(L_LEDPin, OUTPUT);
}

void loop() {
  // Fade each group of LEDs into blue one at a time
  fadeGroupToBlue(A_LEDPin, B_LEDPin, C_LEDPin); // First group
```

```
  delay(1000); // Wait 1 second

  fadeGroupToBlue(D_LEDPin, E_LEDPin, F_LEDPin); // Second group
  delay(1000); // Wait 1 second

  fadeGroupToBlue(G_LEDPin, H_LEDPin, I_LEDPin); // Third group
  delay(1000); // Wait 1 second

  fadeGroupToBlue(J_LEDPin, K_LEDPin, L_LEDPin); // Fourth group
  delay(1000); // Wait 1 second
}

// Function to fade a group of LEDs into blue
void fadeGroupToBlue(int redPin, int greenPin, int bluePin) {
  // Gradually turn off red and green while turning on blue
  for (int brightness = 0; brightness <= 255; brightness += 5) {
    analogWrite(redPin, 255 - brightness);  // Fade out red
    analogWrite(greenPin, 255 - brightness); // Fade out green
    analogWrite(bluePin, brightness);       // Fade in blue
    delay(30); // Small delay for smooth fading
  }

  // Gradually turn off blue (fade out)
  for (int brightness = 255; brightness >= 0; brightness -= 5) {
    analogWrite(redPin, 0);       // Keep red off
    analogWrite(greenPin, 0);     // Keep green off
    analogWrite(bluePin, brightness); // Fade out blue
    delay(30); // Small delay for smooth fading
  }
}
```

VERSION 3 → turn all the LEDs blue from button

```
//LED1
const int A_LEDPin = 13;
const int B_LEDPin = 12;
const int C_LEDPin = 11;
```

```
//LED2
const int D_LEDPin = 10;
const int E_LEDPin = 9;
const int F_LEDPin = 8;

//LED3
const int G_LEDPin = 7;
const int H_LEDPin = 6;
const int I_LEDPin = 5;

//LED4
const int J_LEDPin = 4;
const int K_LEDPin = 3;
const int L_LEDPin = 2;

const int buttonPin = 1; // Button pin
bool ledsOn = true;      // Tracks whether LEDs are on or off

void setup() {
  // Initialize all LED pins as outputs
  pinMode(A_LEDPin, OUTPUT);
  pinMode(B_LEDPin, OUTPUT);
  pinMode(C_LEDPin, OUTPUT);

  pinMode(D_LEDPin, OUTPUT);
  pinMode(E_LEDPin, OUTPUT);
  pinMode(F_LEDPin, OUTPUT);

  pinMode(G_LEDPin, OUTPUT);
  pinMode(H_LEDPin, OUTPUT);
  pinMode(I_LEDPin, OUTPUT);

  pinMode(J_LEDPin, OUTPUT);
  pinMode(K_LEDPin, OUTPUT);
  pinMode(L_LEDPin, OUTPUT);

  // Initialize button pin as input with pullup resistor
  pinMode(buttonPin, INPUT_PULLUP);
```

```
  // Set all LEDs initially to "on" (blue in this case)
  setAllLEDs(0, 0, 255);
}

void loop() {
  // Check if the button is pressed
  if (digitalRead(buttonPin) == LOW) { // Button pressed
    delay(200); // Debounce delay
    toggleLEDs(); // Toggle LEDs on/off
    while (digitalRead(buttonPin) == LOW); // Wait until button is released
  }
}

// Function to toggle LEDs on or off
void toggleLEDs() {
  if (ledsOn) {
    setAllLEDs(0, 0, 0); // Turn all LEDs off
  } else {
    setAllLEDs(0, 0, 255); // Turn all LEDs back to blue
  }
  ledsOn = !ledsOn; // Flip the state
}

// Function to set all LEDs to the same color
void setAllLEDs(int red, int green, int blue) {
  // Set all red pins
  analogWrite(A_LEDPin, red);
  analogWrite(D_LEDPin, red);
  analogWrite(G_LEDPin, red);
  analogWrite(J_LEDPin, red);

  // Set all green pins
  analogWrite(B_LEDPin, green);
  analogWrite(E_LEDPin, green);
  analogWrite(H_LEDPin, green);
  analogWrite(K_LEDPin, green);

  // Set all blue pins
  analogWrite(C_LEDPin, blue);
```

```
  analogWrite(F_LEDPin, blue);
  analogWrite(I_LEDPin, blue);
  analogWrite(L_LEDPin, blue);
}
```

Version 4 → old color categories

```
//LED1
const int A_LEDPin = 13;
const int B_LEDPin = 12;
const int C_LEDPin = 11;

//LED2
const int D_LEDPin = 10;
const int E_LEDPin = 9;
const int F_LEDPin = 8;

//LED3
const int G_LEDPin = 7;
const int H_LEDPin = 6;
const int I_LEDPin = 5;

//LED4
const int J_LEDPin = 4;
const int K_LEDPin = 3;
const int L_LEDPin = 2;

const int buttonPin = 1; // Button pin

// RGB values for 6 categories with 4 colors each
const int categories[6][4][3] = {
   { // Reds
      {220, 20, 60},  // Crimson
      {178, 34, 34},  // Firebrick
      {255, 99, 71},  // Tomato
      {205, 92, 92}   // Indian Red
   },
   { // Oranges
      {255, 165, 0},  // Orange
      {255, 140, 0},  // Dark Orange
      {255, 69, 0},   // Red Orange
```

```
      {255, 215, 0}   // Gold
    },
    { // Yellows
      {255, 255, 0},  // Yellow
      {255, 255, 102}, // Light Yellow
      {240, 230, 140}, // Khaki
      {255, 239, 213}  // Papaya Whip
    },
    { // Greens
      {0, 128, 0},    // Green
      {34, 139, 34},  // Forest Green
      {144, 238, 144}, // Light Green
      {50, 205, 50}   // Lime Green
    },
    { // Blues
      {0, 0, 255},    // Blue
      {30, 144, 255}, // Dodger Blue
      {70, 130, 180}, // Steel Blue
      {135, 206, 250} // Sky Blue
    },
    { // Purples
      {128, 0, 128},   // Purple
      {138, 43, 226},  // Blue Violet
      {148, 0, 211},   // Dark Violet
      {186, 85, 211}   // Medium Orchid
    }
};

int currentCategory = 0; // Tracks the current category
bool buttonPressed = false;

void setup() {
  // Initialize all LED pins as outputs
  pinMode(A_LEDPin, OUTPUT);
  pinMode(B_LEDPin, OUTPUT);
  pinMode(C_LEDPin, OUTPUT);

  pinMode(D_LEDPin, OUTPUT);
  pinMode(E_LEDPin, OUTPUT);
  pinMode(F_LEDPin, OUTPUT);

  pinMode(G_LEDPin, OUTPUT);
  pinMode(H_LEDPin, OUTPUT);
```

```
  pinMode(I_LEDPin, OUTPUT);

  pinMode(J_LEDPin, OUTPUT);
  pinMode(K_LEDPin, OUTPUT);
  pinMode(L_LEDPin, OUTPUT);

  // Initialize button pin as input with pullup resistor
  pinMode(buttonPin, INPUT_PULLUP);

  // Turn off all LEDs initially
  setAllLEDs(0, 0, 0);
}

void loop() {
  // Check if the button is pressed
  if (digitalRead(buttonPin) == LOW && !buttonPressed) { // Button pressed
    delay(200); // Debounce delay
    buttonPressed = true; // Prevent multiple triggers
    displayCategory(currentCategory); // Display the current category
    currentCategory = (currentCategory + 1) % 6; // Move to the next category
    while (digitalRead(buttonPin) == LOW); // Wait until button is released
    buttonPressed = false;
  }
}

// Function to display the current category
void displayCategory(int category) {
  fadeLED(A_LEDPin, B_LEDPin, C_LEDPin, categories[category][0][0], categories[category][0][1],
categories[category][0][2]); // LED1
  fadeLED(D_LEDPin, E_LEDPin, F_LEDPin, categories[category][1][0], categories[category][1][1],
categories[category][1][2]); // LED2
  fadeLED(G_LEDPin, H_LEDPin, I_LEDPin, categories[category][2][0], categories[category][2][1],
categories[category][2][2]); // LED3
  fadeLED(J_LEDPin, K_LEDPin, L_LEDPin, categories[category][3][0], categories[category][3][1],
categories[category][3][2]); // LED4
}

// Function to fade an LED to a specific RGB color
void fadeLED(int redPin, int greenPin, int bluePin, int redValue, int greenValue, int blueValue) {
  for (int brightness = 0; brightness <= 255; brightness += 5) {
    analogWrite(redPin, (redValue * brightness) / 255);   // Red pin
    analogWrite(greenPin, (greenValue * brightness) / 255); // Green pin
    analogWrite(bluePin, (blueValue * brightness) / 255); // Blue pin
```

```
    delay(30); // Small delay for smooth fading
  }
}

// Function to set all LEDs to the same color
void setAllLEDs(int red, int green, int blue) {
  analogWrite(A_LEDPin, red);
  analogWrite(B_LEDPin, green);
  analogWrite(C_LEDPin, blue);

  analogWrite(D_LEDPin, red);
  analogWrite(E_LEDPin, green);
  analogWrite(F_LEDPin, blue);

  analogWrite(G_LEDPin, red);
  analogWrite(H_LEDPin, green);
  analogWrite(I_LEDPin, blue);

  analogWrite(J_LEDPin, red);
  analogWrite(K_LEDPin, green);
  analogWrite(L_LEDPin, blue);
}
```

Version 6 → user input colors!

```
// LED1
const int A_LEDPin = 13;
const int B_LEDPin = 12;
const int C_LEDPin = 11;

// LED2
const int D_LEDPin = 10;
const int E_LEDPin = 9;
const int F_LEDPin = 8;

// LED3
const int G_LEDPin = 7;
const int H_LEDPin = 6;
const int I_LEDPin = 5;
```

```
// LED4
const int J_LEDPin = 4;
const int K_LEDPin = 3;
const int L_LEDPin = 2;

// RGB values for 6 categories with 4 colors each
const int categories[6][4][3] = {
  { // Reds
    {255, 20, 60},  // Crimson
    {255, 34, 34},  // Firebrick
    {255, 99, 71},  // Tomato
    {225, 92, 92}   // Indian Red
  },
  { // Oranges
    {255, 128, 0},   // Orange
    {255, 153, 51},  // Dark Orange
    {255, 178, 102}, // Red Orange
    {255, 204, 153}  // Gold
  },
  { // Yellows
    {255, 255, 0},    // Yellow
    {255, 255, 51},   // Light Yellow
    {255, 255, 102},  // Khaki
    {255, 255, 153}   // Papaya Whip
  },
  { // Greens
    {0, 255, 0},      // Green
    {51, 255, 51},    // Forest Green
    {102, 255, 102},  // Light Green
    {153, 255, 153}   // Lime Green
  },
  { // Blues
    {0, 0, 255},      // Blue
    {51, 51, 255},    // Dodger Blue
    {102, 102, 255},  // Steel Blue
    {153, 153, 255}   // Sky Blue
  },
  { // Purples
    {127, 0, 255},    // Purple
    {153, 51, 255},   // Blue Violet
    {178, 102, 255},  // Dark Violet
    {204, 153, 255}   // Medium Orchid
  }
```

```
};

int currentCategory = 0; // Tracks the current category

void setup() {
  // Initialize all LED pins as outputs
  pinMode(A_LEDPin, OUTPUT);
  pinMode(B_LEDPin, OUTPUT);
  pinMode(C_LEDPin, OUTPUT);

  pinMode(D_LEDPin, OUTPUT);
  pinMode(E_LEDPin, OUTPUT);
  pinMode(F_LEDPin, OUTPUT);

  pinMode(G_LEDPin, OUTPUT);
  pinMode(H_LEDPin, OUTPUT);
  pinMode(I_LEDPin, OUTPUT);

  pinMode(J_LEDPin, OUTPUT);
  pinMode(K_LEDPin, OUTPUT);
  pinMode(L_LEDPin, OUTPUT);

  // Start serial communication
  Serial.begin(9600);
  Serial.println("What color group do you want the LEDs to show? (red, orange, yellow, green,
blue, purple)");

  // Turn off all LEDs initially
  setAllLEDs(0, 0, 0);
}

void loop() {
  // Check if there is user input
  if (Serial.available() > 0) {
    String input = Serial.readStringUntil('\n');
    input.trim(); // Remove any whitespace or newline characters
    int category = getCategory(input);

    if (category != -1) {
      displayCategory(category);
      Serial.println("LEDs updated to: " + input);
    } else {
      Serial.println("Invalid input. Please enter red, orange, yellow, green, blue, or purple.");
```

```
    }
  }
}

// Function to get the category index from user input
int getCategory(String input) {
  if (input.equalsIgnoreCase("red")) return 0;
  if (input.equalsIgnoreCase("orange")) return 1;
  if (input.equalsIgnoreCase("yellow")) return 2;
  if (input.equalsIgnoreCase("green")) return 3;
  if (input.equalsIgnoreCase("blue")) return 4;
  if (input.equalsIgnoreCase("purple")) return 5;
  return -1; // Invalid input
}

// Function to display the current category
void displayCategory(int category) {
  setLEDColor(A_LEDPin, B_LEDPin, C_LEDPin, categories[category][0][0],
categories[category][0][1], categories[category][0][2]); // LED1
  setLEDColor(D_LEDPin, E_LEDPin, F_LEDPin, categories[category][1][0],
categories[category][1][1], categories[category][1][2]); // LED2
  setLEDColor(G_LEDPin, H_LEDPin, I_LEDPin, categories[category][2][0],
categories[category][2][1], categories[category][2][2]); // LED3
  setLEDColor(J_LEDPin, K_LEDPin, L_LEDPin, categories[category][3][0],
categories[category][3][1], categories[category][3][2]); // LED4
}

// Function to set the color of an LED
void setLEDColor(int redPin, int greenPin, int bluePin, int redValue, int greenValue, int blueValue)
{
  analogWrite(redPin, redValue);   // Set red pin
  analogWrite(greenPin, greenValue); // Set green pin
  analogWrite(bluePin, blueValue); // Set blue pin
}

// Function to set all LEDs to the same color
void setAllLEDs(int red, int green, int blue) {
  analogWrite(A_LEDPin, red);
  analogWrite(B_LEDPin, green);
  analogWrite(C_LEDPin, blue);

  analogWrite(D_LEDPin, red);
  analogWrite(E_LEDPin, green);
```

```
  analogWrite(F_LEDPin, blue);

  analogWrite(G_LEDPin, red);
  analogWrite(H_LEDPin, green);
  analogWrite(I_LEDPin, blue);

  analogWrite(J_LEDPin, red);
  analogWrite(K_LEDPin, green);
  analogWrite(L_LEDPin, blue);
}
```

Version 7: → different color on button press

```
// LED1
const int A_LEDPin = 13;
const int B_LEDPin = 12;
const int C_LEDPin = 11;

// LED2
const int D_LEDPin = 10;
const int E_LEDPin = 9;
const int F_LEDPin = 8;

// LED3
const int G_LEDPin = 7;
const int H_LEDPin = 6;
const int I_LEDPin = 5;

// LED4
const int J_LEDPin = 4;
const int K_LEDPin = 3;
const int L_LEDPin = 2;

const int buttonPin = 1; // Button pin

// RGB values for 6 categories with 4 colors each
const int categories[6][4][3] = {
   { // Reds
      {255, 0, 0},  // Crimson
      {200, 0, 0},  // Firebrick
      {140, 0, 0},  // Tomato
```

```
      {140, 0, 0}   // Indian Red
    },
    { // Oranges
      {255, 128, 0},   // Orange
      {255, 153, 51},  // Dark Orange
      {255, 178, 102}, // Red Orange
      {255, 204, 153}  // Gold
    },
    { // Yellows
      {255, 255, 0},    // Yellow
      {255, 255, 51},   // Light Yellow
      {255, 255, 102},  // Khaki
      {255, 255, 153}   // Papaya Whip
    },
    { // Greens
      {0, 255, 0},      // Green
      {51, 255, 51},    // Forest Green
      {102, 255, 102},  // Light Green
      {153, 255, 153}   // Lime Green
    },
    { // Blues
      {0, 0, 255},      // Blue
      {51, 51, 255},    // Dodger Blue
      {102, 102, 255},  // Steel Blue
      {153, 153, 255}   // Sky Blue
    },
    { // Purples
      {127, 0, 255},    // Purple
      {153, 51, 255},   // Blue Violet
      {178, 102, 255},  // Dark Violet
      {204, 153, 255}   // Medium Orchid
    }
};

int currentCategory = 0; // Tracks the current category
bool buttonPressed = false;

void setup() {
  // Initialize all LED pins as outputs
  pinMode(A_LEDPin, OUTPUT);
  pinMode(B_LEDPin, OUTPUT);
  pinMode(C_LEDPin, OUTPUT);
```

```
  pinMode(D_LEDPin, OUTPUT);
  pinMode(E_LEDPin, OUTPUT);
  pinMode(F_LEDPin, OUTPUT);

  pinMode(G_LEDPin, OUTPUT);
  pinMode(H_LEDPin, OUTPUT);
  pinMode(I_LEDPin, OUTPUT);

  pinMode(J_LEDPin, OUTPUT);
  pinMode(K_LEDPin, OUTPUT);
  pinMode(L_LEDPin, OUTPUT);

  // Initialize button pin as input with pullup resistor
  pinMode(buttonPin, INPUT_PULLUP);

  // Turn off all LEDs initially
  setAllLEDs(0, 0, 0);
}

void loop() {
  // Check if the button is pressed
  if (digitalRead(buttonPin) == LOW && !buttonPressed) { // Button pressed
    delay(200); // Debounce delay
    buttonPressed = true; // Prevent multiple triggers
    displayCategory(currentCategory); // Display the current category
    currentCategory = (currentCategory + 1) % 6; // Move to the next category
    while (digitalRead(buttonPin) == LOW); // Wait until button is released
    buttonPressed = false;
  }
}

// Function to display the current category
void displayCategory(int category) {
  setLEDColor(A_LEDPin, B_LEDPin, C_LEDPin, categories[category][0][0],
categories[category][0][1], categories[category][0][2]); // LED1
  setLEDColor(D_LEDPin, E_LEDPin, F_LEDPin, categories[category][1][0],
categories[category][1][1], categories[category][1][2]); // LED2
  setLEDColor(G_LEDPin, H_LEDPin, I_LEDPin, categories[category][2][0],
categories[category][2][1], categories[category][2][2]); // LED3
  setLEDColor(J_LEDPin, K_LEDPin, L_LEDPin, categories[category][3][0],
categories[category][3][1], categories[category][3][2]); // LED4
}
```

```
// Function to set the color of an LED
void setLEDColor(int redPin, int greenPin, int bluePin, int redValue, int greenValue, int blueValue)
{
  analogWrite(redPin, redValue);   // Set red pin
  analogWrite(greenPin, greenValue); // Set green pin
  analogWrite(bluePin, blueValue); // Set blue pin
}

// Function to set all LEDs to the same color
void setAllLEDs(int red, int green, int blue) {
  analogWrite(A_LEDPin, red);
  analogWrite(B_LEDPin, green);
  analogWrite(C_LEDPin, blue);

  analogWrite(D_LEDPin, red);
  analogWrite(E_LEDPin, green);
  analogWrite(F_LEDPin, blue);

  analogWrite(G_LEDPin, red);
  analogWrite(H_LEDPin, green);
  analogWrite(I_LEDPin, blue);

  analogWrite(J_LEDPin, red);
  analogWrite(K_LEDPin, green);
  analogWrite(L_LEDPin, blue);
}
```