Process & Decision Documentation

Project/Assignment Decisions

I ended up making my blob angry. To showcase this I made it red, because anger is generally associated with red. I also wanted to make it seeme erratic, and the use of noise helped this. The biggest part of how i made it seem angry was its pulsing, the blob occasionally pulses, which is meant to represent an angry outburst. This is represented with sudden jagged spikes as well as a change in colour and size.

GenAI Documentation

Date Used: February 01, 2026

Tool Disclosure: ChatGPT 5.2

Purpose of Use: I used Gen AI to get the foundation of my changes to the blob code, based on the creative decisions I inputted to it.

Summary of Interaction: The AI tool provided me with a solid foundation and most functionality for my pulsing blob. I outlined what I wanted the ai to do, such as change the blob to give it a pulsing functionality, as well as its colours and characteristics. I then edited the AI code to better portray my vision.

Human Decision Point(s): The code I received did not represent my intended portrayal of the angry emotion, so I edited variables, changed coordinates, changed loop timing and conditions, and adjusted the blob's characteristics. The outputted text for the button as well as instruction were changed to better provide an example of how to use the sketch.

Integrity & Verification Note: The Gen AI's code was functional upon its first output, but had to be altered. I went over each line of teh code to make sure I was able to understand what it did. I then changed certain variables to adjust the behaviour of the blob. The AI added functionality, but I edited the functionality and verified it worked properly.

Scope of GenAI Use: The input I gave the AI consisted of the raw code from the example exercise, and its output reflected a changed version of the same code. I altered the AI output to better fit my vision, and portray the emotion of the blob. THe AI's output influenced my final product, but no code was used directly from the chat without editing.

Gen AI Transcript

```
// Object representing a soft animated blob
let blob = {
  // Position of the blob (centre of the shape)
  x: 240,
  y: 160, // centre of the canvas

  // Base size and shape resolution
  r: 28, // Base radius of the blob
  points: 48, // Number of vertices around the circle (higher = smoother)

  // Shape deformation settings
  wobble: 8, // Maximum amount the edge can move in or out
  wobbleFreq: 0.8, // Controls how lumpy or smooth the blob looks

  // Time values for animation
  t: 0, // Time input for noise()
  tSpeed: 0.01, // How fast the blob "breathes"
};

function setup() {
  createCanvas(480, 320);
  noStroke();

  // Text settings for on-screen instructions
  textFont("sans-serif");
  textSize(14);
}

function draw() {
  background(240);

  // --- Animate over time ---
  // Increment time so noise() changes smoothly every frame
  blob.t += blob.tSpeed;

  // --- Draw the blob ---
  // We draw a circle made of many points,
  // then push each point in or out using Perlin noise
  fill(20, 120, 255);
  beginShape();

  // Loop once around the circle
  for (let i = 0; i < blob.points; i++) {
    // Angle around the circle (0 → TAU)
    const a = (i / blob.points) * TAU;
    // Sample Perlin noise using:
    // - direction (cos/sin of angle)
    // - time (blob.t) for animation
    const n = noise(
      cos(a) * blob.wobbleFreq + 100,
      sin(a) * blob.wobbleFreq + 100,
      blob.t,
    );

    // Convert noise value (0–1) into a radius offset
    const r = blob.r + map(n, 0, 1, -blob.wobble, blob.wobble);

    // Convert polar coordinates (angle + radius)
    // into screen coordinates (x, y)
    vertex(blob.x + cos(a) * r, blob.y + sin(a) * r);
  }

  // Close the shape to form a solid blob
  endShape(CLOSE);

  // --- On-screen tip for experimentation ---change
  fill(0);
  text("Blob breathing via noise(). Try wobble and tSpeed.", 10, 18);
}
```

change this simple blob code to make it red. Make it pulse very few seconds, the pulse should drastically reduce its vertices briefly as well as make the vertices jut out more, the pulse shouls also change it do a brighter red. Optionally add a button that controls when the pulse happens

```
let blob = {
  x: 240,
  y: 160,
  r: 28,
  points: 48,
  wobble: 8,
  wobbleFreq: 0.8,
  t: 0,
  tSpeed: 0.01,
  pulsing: false,
  pulseT: 0,
  pulseDur: 0.4,
  pulseGap: 180,
  frameLast: 0,
};

let pulseBtn;

function setup() {
  createCanvas(480, 320);
  noStroke();
  textFont("sans-serif");
  textSize(14);

  pulseBtn = createButton("Pulse");
  pulseBtn.position(10, height + 10);
  pulseBtn.mousePressed(triggerPulse);
}

function triggerPulse() {
  blob.pulsing = true;
  blob.pulseT = 0;
}

function draw() {
  background(240);
  blob.t += blob.tSpeed;

  if (!blob.pulsing && frameCount - blob.frameLast > blob.pulseGap) {
    triggerPulse();
  }

  let p = 0;
  if (blob.pulsing) {
    blob.pulseT += deltaTime / 1000;
    p = constrain(blob.pulseT / blob.pulseDur, 0, 1);
    if (p >= 1) {
      blob.pulsing = false;
      blob.frameLast = frameCount;
    }
  }

  let pulseAmt = sin(p * PI);
  let pts = floor(lerp(blob.points, 10, pulseAmt));
  let wob = lerp(blob.wobble, 26, pulseAmt);
  let col = lerpColor(color(180, 0, 0), color(255, 60, 60), pulseAmt);

  fill(col);
  beginShape();
  for (let i = 0; i < pts; i++) {
    const a = (i / pts) * TAU;
    const n = noise(
      cos(a) * blob.wobbleFreq + 100,
      sin(a) * blob.wobbleFreq + 100,
      blob.t
    );
    const r = blob.r + map(n, 0, 1, -wob, wob);
    vertex(blob.x + cos(a) * r, blob.y + sin(a) * r);
  }
  endShape(CLOSE);

  fill(0);
  text("Red blob pulses every few seconds (or press button).", 10, 18);
}
```

add comments to explain teh code

```
// --- Blob data object ---
// Stores all parameters controlling the blob’s shape, animation, and pulse
let blob = {
  x: 240,
  y: 160,
  r: 28,
  points: 48,
  wobble: 8,
  wobbleFreq: 0.8,
  t: 0,
  tSpeed: 0.01,

  // Pulse-related state
  pulsing: false,
  pulseT: 0,
  pulseDur: 0.4,
  pulseGap: 180,
  frameLast: 0,
};

let pulseBtn;

function setup() {
  createCanvas(480, 320);
  noStroke();
  textFont("sans-serif");
  textSize(14);

  // Button to manually trigger a pulse
  pulseBtn = createButton("Pulse");
  pulseBtn.position(10, height + 10);
  pulseBtn.mousePressed(triggerPulse);
}

// Starts a pulse animation
function triggerPulse() {
  blob.pulsing = true;
  blob.pulseT = 0;
}

function draw() {
  background(240);

  // Advance noise time so the blob “breathes”
  blob.t += blob.tSpeed;

  // Automatically trigger a pulse every few seconds
  if (!blob.pulsing && frameCount - blob.frameLast > blob.pulseGap) {
    triggerPulse();
  }

  // Normalized pulse progress (0 → 1)
  let p = 0;

  // If pulsing, advance pulse timer
  if (blob.pulsing) {
    blob.pulseT += deltaTime / 1000;
    p = constrain(blob.pulseT / blob.pulseDur, 0, 1);

    // End pulse
    if (p >= 1) {
      blob.pulsing = false;
      blob.frameLast = frameCount;
    }
  }

  // Sinusoidal pulse curve (smooth in/out)
  let pulseAmt = sin(p * PI);
```

```
// Reduce vertex count sharply during pulse
let pts = floor(lerp(blob.points, 10, pulseAmt));

// Increase wobble so vertices jut out more
let wob = lerp(blob.wobble, 26, pulseAmt);

// Shift color to brighter red during pulse
let col = lerpColor(
  color(180, 0, 0),
  color(255, 60, 60),
  pulseAmt
);

// Draw the blob shape
fill(col);
beginShape();

// Loop around the circle and distort radius using noise
for (let i = 0; i < pts; i++) {
  const a = (i / pts) * TAU;

  // Sample 3D Perlin noise (angle + time)
  const n = noise(
    cos(a) * blob.wobbleFreq + 100,
    sin(a) * blob.wobbleFreq + 100,
    blob.t
  );

  // Convert noise to radial displacement
  const r = blob.r + map(n, 0, 1, -wob, wob);

  // Polar → Cartesian
  vertex(
    blob.x + cos(a) * r,
    blob.y + sin(a) * r
  );
}

endShape(CLOSE);

// UI text
fill(0);
text("Red blob pulses every few seconds (or press button).", 10, 18);
```

```
}
```

make the blob change slowly from red to a slightly darker red and back and forth while not pulsing, also make the vertices more spread out while pulsing, kind of like a jagged star

```
// --- Blob data object ---
// Stores all parameters controlling the blob’s shape, animation, color, and pulse
let blob = {
  x: 240,
  y: 160,

  r: 28,
  points: 48,

  wobble: 8,
  wobbleFreq: 0.8,

  t: 0,
  tSpeed: 0.01,

  // Pulse state
  pulsing: false,
  pulseT: 0,
  pulseDur: 0.4,
  pulseGap: 180,
  frameLast: 0,

  // Slow color breathing when NOT pulsing
  colorT: 0,
  colorSpeed: 0.005,
};

let pulseBtn;

function setup() {
  createCanvas(480, 320);
  noStroke();
  textFont("sans-serif");
  textSize(14);

  // Button to manually trigger pulse
  pulseBtn = createButton("Pulse");
  pulseBtn.position(10, height + 10);
  pulseBtn.mousePressed(triggerPulse);
}

// Starts a pulse animation
function triggerPulse() {
  blob.pulsing = true;
  blob.pulseT = 0;
}

function draw() {
  background(240);

  // Advance time for noise-based motion
  blob.t += blob.tSpeed;

  // Automatically trigger pulse every few seconds
  if (!blob.pulsing && frameCount - blob.frameLast > blob.pulseGap) {
    triggerPulse();
  }

  // --- Pulse timing ---
  let p = 0;
  if (blob.pulsing) {
    blob.pulseT += deltaTime / 1000;
    p = constrain(blob.pulseT / blob.pulseDur, 0, 1);

    if (p >= 1) {
      blob.pulsing = false;
      blob.frameLast = frameCount;
    }
  }

  // Smooth pulse curve
  let pulseAmt = sin(p * PI);

  // --- Geometry changes ---
  // Fewer vertices + extreme wobble = jagged star look
  let pts = floor(lerp(blob.points, 9, pulseAmt));
  let wob = lerp(blob.wobble, 34, pulseAmt);

  // Increase frequency so spikes are sharper while pulsing
  let freq = lerp(blob.wobbleFreq, 2.4, pulseAmt);
```

```
// --- Color breathing when idle ---
blob.colorT += blob.colorSpeed;
let idleColAmt = (sin(blob.colorT) + 1) * 0.5;

let idleCol = lerpColor(
  color(150, 0, 0),
  color(200, 0, 0),
  idleColAmt
);

// Brighter red during pulse
let pulseCol = color(255, 60, 60);

// Blend idle color → pulse color
let col = lerpColor(idleCol, pulseCol, pulseAmt);

// --- Draw blob ---
fill(col);
beginShape();

for (let i = 0; i < pts; i++) {
  const a = (i / pts) * TAU;

  // Noise sample based on angle + time
  const n = noise(
    cos(a) * freq + 100,
    sin(a) * freq + 100,
    blob.t
  );

  // Radial distortion
  const r = blob.r + map(n, 0, 1, -wob, wob);

  // Polar → Cartesian
  vertex(
    blob.x + cos(a) * r,
    blob.y + sin(a) * r
  );
}

endShape(CLOSE);

// UI text
```

```
  fill(0);
  text(
    "Idle: slow dark/light red breathing · Pulse: jagged star burst",
    10,
    18
  );
}
```

make the blobs pulse bigger/more spread out

```
// --- Blob data object ---
// Controls shape, animation, color, and pulsing behaviour
let blob = {
  x: 240,
  y: 160,

  r: 28,          // Base radius
  points: 48,     // Smooth circle resolution

  wobble: 8,
  wobbleFreq: 0.8,

  t: 0,
  tSpeed: 0.01,

  // Pulse state
  pulsing: false,
  pulseT: 0,
  pulseDur: 0.45,
  pulseGap: 180,
  frameLast: 0,

  // Idle color breathing
  colorT: 0,
  colorSpeed: 0.005,
};

let pulseBtn;

function setup() {
  createCanvas(480, 320);
  noStroke();
  textFont("sans-serif");
  textSize(14);

  pulseBtn = createButton("Pulse");
  pulseBtn.position(10, height + 10);
  pulseBtn.mousePressed(triggerPulse);
}

// Manually start a pulse
function triggerPulse() {
  blob.pulsing = true;
  blob.pulseT = 0;
}

function draw() {
  background(240);

  // Advance noise time
  blob.t += blob.tSpeed;

  // Auto pulse every few seconds
  if (!blob.pulsing && frameCount - blob.frameLast > blob.pulseGap) {
    triggerPulse();
  }

  // --- Pulse timing ---
  let p = 0;
  if (blob.pulsing) {
    blob.pulseT += deltaTime / 1000;
    p = constrain(blob.pulseT / blob.pulseDur, 0, 1);

    if (p >= 1) {
      blob.pulsing = false;
      blob.frameLast = frameCount;
    }
  }

  // Smooth pulse curve (0 → 1 → 0)
  let pulseAmt = sin(p * PI);

  // --- Shape exaggeration during pulse ---
  // Much fewer points = star-like silhouette
  let pts = floor(lerp(blob.points, 8, pulseAmt));
```

```
// Strong outward spikes
let wob = lerp(blob.wobble, 46, pulseAmt);

// Higher frequency sharpens spikes
let freq = lerp(blob.wobbleFreq, 3.0, pulseAmt);

// Overall size expansion during pulse
let baseR = lerp(blob.r, 52, pulseAmt);

// --- Idle color breathing ---
blob.colorT += blob.colorSpeed;
let idleAmt = (sin(blob.colorT) + 1) * 0.5;

let idleCol = lerpColor(
  color(140, 0, 0),
  color(190, 0, 0),
  idleAmt
);

// Bright flash color for pulse
let pulseCol = color(255, 70, 70);

// Blend idle → pulse color
let col = lerpColor(idleCol, pulseCol, pulseAmt);

// --- Draw blob ---
fill(col);
beginShape();

for (let i = 0; i < pts; i++) {
  const a = (i / pts) * TAU;

  // Angle-based Perlin noise sample
  const n = noise(
    cos(a) * freq + 100,
    sin(a) * freq + 100,
    blob.t
  );

  // Large radial displacement for explosive pulse
  const r = baseR + map(n, 0, 1, -wob, wob);

  vertex(
    blob.x + cos(a) * r,
      blob.y + sin(a) * r
    );
  }

  endShape(CLOSE);

  fill(0);
  text(
    "Idle: slow red breathing · Pulse: larger jagged star burst",
    10,
    18
  );
}
```