Process & Decision Documentation

Project/Assignment Decisions

The only change I made was to keep the colour change of the blob from pink to yellow more simple than the initial output from ChatGPT.

Role-Based Process Evidence

Name: Amanda Mihalaros

Role(s): Completing Side Quest 2

Primary responsibility for this work: Redesign the blob's movement and the environment in the code to express a specific emotion, in this case joy.

Goal of Work Session

Use ChatGPT to make adjustments to the initial code in order for the blob's movement and the environment to express joy to the user.

Tools, Resources, or Inputs Used:

- ChatGPT
- Github
- Visual Studio Code

GenAI Documentation

Date Used: Jan 26 2026

Tool Disclosure: ChatGPT

Purpose of Use: Adjusting and reworking previous code

Summary of Interaction: Directed GenAI to make minor changes to the code to express a specific emotion in the output. Did not explicity state what changes to make. Kept it open ended.

Human Decision Point(s): Initially GenAI layed out the changes that should be made, I then prompted it to put it all into one code that can be copy and pasted in full.

Integrity & Verification Note: Ensured it functioned without errors.

Scope of GenAI Use: GenAI did not contribute to the README document or any of the initial code that was given to us.

Limitations or Misfires: Makes simple changes if prompts are not clear or direct enough.

Summary of Process (Human + Tool)

- Tested the code for functionality

Appendix

Prompt:

Using this code, Redesign the blob's movement and environment to express a specific emotion (e.g., joy, frustration, panic). in this case make it express joy. // Y-position of the floor (ground level) let floorY3; // Player character (soft, animated blob) let blob3 = { // Position (centre of the blob) x: 80, y: 0, // Visual properties r: 26, // Base radius points: 48, // Number of points used to draw the blob wobble: 7, // Edge deformation amount wobbleFreq: 0.9, // Time values for breathing animation t: 0, tSpeed: 0.01, // Physics: velocity vx: 0, // Horizontal velocity vy: 0, // Vertical velocity // Movement tuning accel: 0.55, // Horizontal acceleration maxRun: 4.0, // Maximum horizontal speed gravity: 0.65, // Downward force jumpV: -11.0, // Initial jump impulse // State onGround: false, // True when standing on a platform // Friction frictionAir: 0.995, // Light friction in air frictionGround: 0.88, // Stronger friction on ground }; // List of solid platforms the blob can stand on // Each platform is an axis-aligned rectangle (AABB) let platforms = []; function setup() { createCanvas(640, 360); // Define the floor height floorY3 = height - 36; noStroke(); textFont("sans-serif"); textSize(14); // Create platforms (floor + steps) platforms = [{ x: 0, y: floorY3, w: width, h: height - floorY3 }, // floor { x: 120, y: floorY3 - 70, w: 120, h: 12 }, // low step { x: 300, y: floorY3 - 120, w: 90, h: 12 }, // mid step { x: 440, y: floorY3 - 180, w: 130, h: 12 }, // high step { x: 520, y: floorY3 - 70, w: 90, h: 12 }, // return ramp]; // Start the blob resting on the floor blob3.y = floorY3 - blob3.r - 1; } function draw() { background(240); // --- Draw all platforms --- fill(200); for (const p of platforms) { rect(p.x, p.y, p.w, p.h); } // --- Input: left/right movement --- let move = 0; if (keyIsDown(65) || keyIsDown(LEFT_ARROW)) move -= 1; // A or ← if (keyIsDown(68) || keyIsDown(RIGHT_ARROW)) move += 1; // D or → blob3.vx += blob3.accel * move; // --- Apply friction and clamp speed --- blob3.vx *= blob3.onGround ? blob3.frictionGround : blob3.frictionAir; blob3.vx = constrain(blob3.vx, -blob3.maxRun, blob3.maxRun); // --- Apply gravity --- blob3.vy += blob3.gravity; // --- Collision representation --- // We collide using a rectangle (AABB), // even though the blob is drawn as a circle let box = { x: blob3.x - blob3.r, y: blob3.y - blob3.r, w: blob3.r * 2, h: blob3.r * 2, }; // --- STEP 1: Move horizontally, then resolve X collisions --- box.x += blob3.vx; for (const s of platforms) { if (overlap(box, s)) { if (blob3.vx > 0) { // Moving right → hit the left side of a platform box.x = s.x - box.w; } else if (blob3.vx < 0) { // Moving left → hit the right side of a platform box.x = s.x + s.w; } blob3.vx = 0; } } // --- STEP 2: Move vertically, then resolve Y collisions --- box.y += blob3.vy; blob3.onGround =

false; for (const s of platforms) { if (overlap(box, s)) { if (blob3.vy > 0) { // Falling → land on top of a platform box.y = s.y - box.h; blob3.vy = 0; blob3.onGround = true; } else if (blob3.vy < 0) { // Rising → hit the underside of a platform box.y = s.y + s.h; blob3.vy = 0; } } } // --- Convert collision box back to blob centre --- blob3.x = box.x + box.w / 2; blob3.y = box.y + box.h / 2; // Keep blob inside the canvas horizontally blob3.x = constrain(blob3.x, blob3.r, width - blob3.r); // --- Draw the animated blob --- blob3.t += blob3.tSpeed; drawBlobCircle(blob3); // --- HUD --- fill(0); text("Move: A/D or ←/→ • Jump: Space/W/↑ • Land on platforms", 10, 18); } // Axis-Aligned Bounding Box (AABB) overlap test // Returns true if rectangles a and b intersect function overlap(a, b) { return (a.x < b.x + b.w && a.x + a.w > b.x && a.y < b.y + b.h && a.y + a.h > b.y); } // Draws the blob using Perlin noise for a soft, breathing effect function drawBlobCircle(b) { fill(20, 120, 255); beginShape(); for (let i = 0; i < b.points; i++) { const a = (i / b.points) * TAU; // Noise-based radius offset const n = noise(cos(a) * b.wobbleFreq + 100, sin(a) * b.wobbleFreq + 100, b.t,); const r = b.r + map(n, 0, 1, -b.wobble, b.wobble); vertex(b.x + cos(a) * r, b.y + sin(a) * r); } endShape(CLOSE); } // Jump input (only allowed when grounded) function keyPressed() { if ((key === " " || key === "W" || key === "w" || keyCode === UP_ARROW) && blob3.onGround) { blob3.vy = blob3.jumpV; blob3.onGround = false; } }

Response:

Ooo love this prompt 🥹 Let's turn your blob into a **bouncy, happy little bean full of joy**.

Joy =
✨ light, floaty movement
🌈 bright playful world
🎈 extra bounce and squash
💖 energetic wiggly animation

Below are **targeted changes** you can paste in — I'll explain the "why" after.

🌤️ 1. Make the WORLD feel joyful

🌈 Happy sky gradient + floating bubbles

Replace your background(240); line in draw() with:

drawJoySky();

Add this new function anywhere below:

function drawJoySky() {

```
  // Soft vertical gradient sky
  for (let y = 0; y < height; y++) {
    let inter = map(y, 0, height, 0, 1);
    let c = lerpColor(color(255, 230, 255), color(200, 240, 255), inter);
    stroke(c);
    line(0, y, width, y);
  }
  noStroke();

  // Floating happy bubbles
  for (let i = 0; i < 6; i++) {
    let bx = (frameCount * 0.3 + i * 120) % width;
    let by = 60 + sin(frameCount * 0.02 + i) * 20;
    fill(255, 255, 255, 120);
    ellipse(bx, by, 30, 30);
  }
}
```

☁️ 2. Make platforms feel soft and playful

In draw(), change platform color:

```
fill(180, 220, 255); // soft blue platforms
```

Optional: make them look like clouds:

```
for (const p of platforms) {
  rect(p.x, p.y, p.w, p.h, 10); // rounded corners
}
```

🫧 3. Make the blob move with JOY (floaty + bouncy)

Adjust blob physics values

Inside blob3 change:

```
wobble: 10,        // more squishy
wobbleFreq: 1.2,   // livelier surface
tSpeed: 0.02,      // faster happy wiggle

accel: 0.7,        // more energetic movement
maxRun: 5.2,       // quicker zoomies
gravity: 0.45,     // lighter, floatier
jumpV: -13.5,      // higher jump = excitement!
frictionGround: 0.92, // slides a bit = playful
```

🎈 4. Add a HAPPY BOUNCE when landing

Joyful characters don't just stop — they **boing**.

Find this part in vertical collision:

```
if (blob3.vy > 0) {
  box.y = s.y - box.h;
  blob3.vy = 0;
  blob3.onGround = true;
}
```

Replace with:

```
if (blob3.vy > 0) {
  box.y = s.y - box.h;

  // Tiny bounce if landing fast (joyful springiness)
  if (blob3.vy > 2) {
    blob3.vy *= -0.25;
```

```
  } else {
    blob3.vy = 0;
    blob3.onGround = true;
  }
}
```

😊 5. Give the blob a smiling face

Replace drawBlobCircle(blob3); with:

```
drawHappyBlob(blob3);
```

Add this new function:

```
function drawHappyBlob(b) {
  push();

  // Slight squash & stretch based on vertical speed
  let stretch = map(abs(b.vy), 0, 10, 1, 1.25);
  let squash = 1 / stretch;

  translate(b.x, b.y);
  scale(stretch, squash);

  // Body
  fill(255, 180, 220); // cheerful pink
  beginShape();
  for (let i = 0; i < b.points; i++) {
    const a = (i / b.points) * TAU;
    const n = noise(
      cos(a) * b.wobbleFreq + 100,
```

```
      sin(a) * b.wobbleFreq + 100,
      b.t
    );
    const r = b.r + map(n, 0, 1, -b.wobble, b.wobble);
    vertex(cos(a) * r, sin(a) * r);
  }
  endShape(CLOSE);

  // Face (does not squash)
  scale(1 / stretch, 1 / squash);

  fill(0);
  ellipse(-8, -5, 4, 6); // left eye
  ellipse(8, -5, 4, 6);  // right eye

  noFill();
  stroke(0);
  strokeWeight(2);
  arc(0, 4, 16, 10, 0, PI); // smile
  noStroke();

  pop();
}
```

✨ 6. Add little sparkles when jumping

Inside keyPressed() after setting blob3.vy:

```
spawnJoyParticles(blob3.x, blob3.y);
```

Add this system:

```
let joyParticles = [];

function spawnJoyParticles(x, y) {
  for (let i = 0; i < 8; i++) {
    joyParticles.push({
      x: x,
      y: y,
      vx: random(-2, 2),
      vy: random(-3, -1),
      life: 40
    });
  }
}

function updateJoyParticles() {
  for (let p of joyParticles) {
    p.x += p.vx;
    p.y += p.vy;
    p.vy += 0.1;
    p.life--;
    fill(255, 220, 120, p.life * 6);
    ellipse(p.x, p.y, 6, 6);
  }
  joyParticles = joyParticles.filter(p => p.life > 0);
}
```

Then in draw() (near the end) add:

updateJoyParticles();

🎉 **Resulting Emotion: JOY**

Your blob now:

- Moves **light and floaty**
- **Bounces** instead of thuds
- **Wiggles energetically**
- Has a **smiling face**
- Jumps with **sparkly excitement**
- Lives in a **bright dreamy sky**

If you want, next we can make:
😄 Joy = double jump
🥳 Joy = rainbow trail when running
🐓 Joy = little dance when idle

Prompt:

Write all the code in one so I can just paste it. ONly make the blob change colour to yellow when it jumps.

Response:

Yesss joyful blob coming right up ☀️💛
Here's the **full working sketch** with:

- Floaty, happy movement
- Soft sky background
- Bouncy landings
- Squishy animation
- Sparkle particles on jump
- **Blob turns YELLOW only while in the air after jumping**

Just paste this into p5.js 👇

```
// ================== FLOOR ==================
let floorY3;

// ================== BLOB ==================
let blob3 = {
  x: 80,
  y: 0,

  r: 26,
  points: 48,
  wobble: 10,
  wobbleFreq: 1.2,

  t: 0,
  tSpeed: 0.02,

  vx: 0,
  vy: 0,

  accel: 0.7,
  maxRun: 5.2,
  gravity: 0.45,
  jumpV: -13.5,

  onGround: false,
```

```
  frictionAir: 0.995,
  frictionGround: 0.92,
};

// ================== PLATFORMS ==================
let platforms = [];

// ================== PARTICLES ==================
let joyParticles = [];

function setup() {
  createCanvas(640, 360);
  floorY3 = height - 36;

  noStroke();
  textFont("sans-serif");
  textSize(14);

  platforms = [
    { x: 0, y: floorY3, w: width, h: height - floorY3 },
    { x: 120, y: floorY3 - 70, w: 120, h: 12 },
    { x: 300, y: floorY3 - 120, w: 90, h: 12 },
    { x: 440, y: floorY3 - 180, w: 130, h: 12 },
    { x: 520, y: floorY3 - 70, w: 90, h: 12 },
  ];

  blob3.y = floorY3 - blob3.r - 1;
```

```
}

function draw() {
  drawJoySky();

  // Platforms (soft & playful)
  fill(180, 220, 255);
  for (const p of platforms) {
    rect(p.x, p.y, p.w, p.h, 10);
  }

  // ================= MOVEMENT INPUT =================
  let move = 0;
  if (keyIsDown(65) || keyIsDown(LEFT_ARROW)) move -= 1;
  if (keyIsDown(68) || keyIsDown(RIGHT_ARROW)) move += 1;
  blob3.vx += blob3.accel * move;

  blob3.vx *= blob3.onGround ? blob3.frictionGround : blob3.frictionAir;
  blob3.vx = constrain(blob3.vx, -blob3.maxRun, blob3.maxRun);

  blob3.vy += blob3.gravity;

  let box = {
    x: blob3.x - blob3.r,
    y: blob3.y - blob3.r,
    w: blob3.r * 2,
    h: blob3.r * 2,
```

```
};

// Horizontal collisions
box.x += blob3.vx;
for (const s of platforms) {
  if (overlap(box, s)) {
    if (blob3.vx > 0) box.x = s.x - box.w;
    else if (blob3.vx < 0) box.x = s.x + s.w;
    blob3.vx = 0;
  }
}

// Vertical collisions
box.y += blob3.vy;
blob3.onGround = false;

for (const s of platforms) {
  if (overlap(box, s)) {
    if (blob3.vy > 0) {
      box.y = s.y - box.h;

      // Joyful bounce
      if (blob3.vy > 2) {
        blob3.vy *= -0.25;
      } else {
        blob3.vy = 0;
        blob3.onGround = true;
```

```
      }
    } else if (blob3.vy < 0) {
      box.y = s.y + s.h;
      blob3.vy = 0;
    }
  }
}

blob3.x = box.x + box.w / 2;
blob3.y = box.y + box.h / 2;
blob3.x = constrain(blob3.x, blob3.r, width - blob3.r);

blob3.t += blob3.tSpeed;

drawHappyBlob(blob3);
updateJoyParticles();

fill(0);
text("Move: A/D or ←/→   Jump: Space/W/↑", 10, 18);
}

// ================= SKY =================
function drawJoySky() {
  for (let y = 0; y < height; y++) {
    let inter = map(y, 0, height, 0, 1);
    let c = lerpColor(color(255, 230, 255), color(200, 240, 255), inter);
    stroke(c);
```

```
    line(0, y, width, y);
  }
  noStroke();

  for (let i = 0; i < 6; i++) {
    let bx = (frameCount * 0.3 + i * 120) % width;
    let by = 60 + sin(frameCount * 0.02 + i) * 20;
    fill(255, 255, 255, 120);
    ellipse(bx, by, 30, 30);
  }
}

// ================= BLOB DRAW =================
function drawHappyBlob(b) {
  push();

  let stretch = map(abs(b.vy), 0, 10, 1, 1.25);
  let squash = 1 / stretch;

  translate(b.x, b.y);
  scale(stretch, squash);

  // 🎨 COLOR CHANGE WHEN JUMPING
  if (!b.onGround) fill(255, 230, 80); // yellow in air
  else fill(255, 180, 220); // pink on ground

  beginShape();
```

```
  for (let i = 0; i < b.points; i++) {
    const a = (i / b.points) * TAU;
    const n = noise(
      cos(a) * b.wobbleFreq + 100,
      sin(a) * b.wobbleFreq + 100,
      b.t
    );
    const r = b.r + map(n, 0, 1, -b.wobble, b.wobble);
    vertex(cos(a) * r, sin(a) * r);
  }
  endShape(CLOSE);

  // Face
  scale(1 / stretch, 1 / squash);
  fill(0);
  ellipse(-8, -5, 4, 6);
  ellipse(8, -5, 4, 6);
  noFill();
  stroke(0);
  strokeWeight(2);
  arc(0, 4, 16, 10, 0, PI);
  noStroke();

  pop();
}

// ================= PARTICLES =================
```

```
function spawnJoyParticles(x, y) {
  for (let i = 0; i < 8; i++) {
    joyParticles.push({
      x: x,
      y: y,
      vx: random(-2, 2),
      vy: random(-3, -1),
      life: 40
    });
  }
}

function updateJoyParticles() {
  for (let p of joyParticles) {
    p.x += p.vx;
    p.y += p.vy;
    p.vy += 0.1;
    p.life--;
    fill(255, 220, 120, p.life * 6);
    ellipse(p.x, p.y, 6, 6);
  }
  joyParticles = joyParticles.filter(p => p.life > 0);
}

// ================= COLLISION =================
function overlap(a, b) {
  return (
```

```
    a.x < b.x + b.w &&
    a.x + a.w > b.x &&
    a.y < b.y + b.h &&
    a.y + a.h > b.y
  );
}

// ================== JUMP ==================
function keyPressed() {
  if (
    (key === " " || key === "W" || key === "w" || keyCode === UP_ARROW) &&
    blob3.onGround
  ) {
    blob3.vy = blob3.jumpV;
    blob3.onGround = false;
    spawnJoyParticles(blob3.x, blob3.y);
  }
}
```