

Process & Decision Documentation

Project/Assignment Decisions

In side quest #2, I learned to redesign the blob's movement and environment to express a specific emotion, in this case, the emotion of joy. I also did the bonus where I added a "mischief" mechanic, where my blob bumps into mini balls around the map.

I started with the starter code, where the blob was a normal blue circle that could jump on platforms. I then put the code into ChatGPT and asked it to enhance the blob's movement and environment to make it more joyful. At first, it suggested making the blob move very quickly, but there was an error where it kept disappearing off the map, so I didn't accept that change.

Next, I changed the color of the blob and background myself to yellow and blue to better express happiness. I then asked ChatGPT to add movement and animation, which I kept. After that, I requested balls that the blob could interact with, and ChatGPT generated multi-colored balls and removed the platforms, which I was happy with. Finally, I asked it to add a smiley face to the blob, which I also kept.

GenAI Documentation

If GenAI was used (keep each response as brief as possible):

Date Used: January 26th, 2026

Tool Disclosure: ChatGPT -5.2

Purpose of Use: Help to create a more joyful environment.

Summary of Interaction: The tool provided ideas to make the environment feel more joyful. It also helped me add stuff such as balls and a smiley face.

Integrity & Verification Note: I checked to see if the code it gave me worked and if it enhanced/defeated the purpose of the side quest.

Appendix

My Prompts	GenAI Response
<p>Using the starter code, change the blob's movement and environment to make it express joy.</p>	<p>Absolutely! I'll change the starter code to make the blob's movement and environment express more joy. Here's the updated version with comments:</p> <pre> // --- Extra: balls you can play with --- let balls = []; const numBalls = 5; function setup() { createCanvas(640, 360); floorY3 = height - 36; noStroke(); textFont("sans-serif"); textSize(14); // Create platforms 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; // Initialize balls for (let i = 0; i < numBalls; i++) { balls.push({ x: random(50, width - 50), y: random(50, floorY3 - 100), r: random(10, 18), vx: random(-2, 2), vy: random(-2, 2), color: color(random(100, 255), random(100, 255), random(100, 255)) }) } </pre>

```

    });
  }
}

function draw() {
  background(220, 245, 255);

  // Draw platforms
  fill(255, 210, 120);
  for (const p of platforms) rect(p.x, p.y, p.w,
p.h);

  // Player 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;

  // Blob collision box
  let box = { x: blob3.x - blob3.r, y: blob3.y -
blob3.r, w: blob3.r * 2, h: blob3.r * 2 };

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

  // Vertical collision
  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;
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);

    // Draw blob
    blob3.t += blob3.tSpeed;
    drawBlobCircle(blob3);
    drawSmileyFace(blob3); // <-- new

    // Draw balls
    for (let b of balls) {
        b.x += b.vx;
        b.y += b.vy;
        b.vy += 0.3; // gravity

        // Bounce off floor
        if (b.y + b.r > floorY3) { b.y = floorY3 - b.r;
b.vy *= -0.7; }
        // Bounce off walls
        if (b.x - b.r < 0 || b.x + b.r > width) { b.vx
*= -1; }

        fill(b.color);
        ellipse(b.x, b.y, b.r * 2);
    }

    // HUD
    fill(0);
    text("Move: A/D or ←/→ • Jump: Space/W/↑ •
Land on platforms • Bounce balls!", 10, 18);
}

// Draw a smiley face on the blob
function drawSmileyFace(b) {

```

```
fill(0);
// eyes
ellipse(b.x - b.r / 3, b.y - b.r / 5, 4, 4);
ellipse(b.x + b.r / 3, b.y - b.r / 5, 4, 4);
// smile
noFill();
stroke(0);
strokeWeight(2);
arc(b.x, b.y + b.r / 8, b.r / 1.2, b.r / 1.2, 0,
PI);
noStroke();
}
```

Can you add a happy movement to the blob?

Yesss 🥳 let's give your blob some *happy energy* ✨
We'll add **bouncy bobbing**, **squash & stretch**, and a tiny **wiggle** when it moves — nothing too complicated, very “cute platformer” vibes.

Below are **small add-ons**, not a full rewrite 🙌

1 Add these properties to **blob3**

Put them **inside your **blob3** object**:

```
// Happy movement extras
bobT: 0,           // time for bobbing
bobSpeed: 0.12,    // how fast it bobs
bobAmount: 4,      // how strong the bounce
```

2 Update blob animation timing

Inside **draw()** **before** **drawBlobCircle(blob3)**; add:

```
// Happy idle + movement bob
blob3.bobT += blob3.bobSpeed + abs(blob3.vx) *
0.02;
```

3 Replace `drawBlobCircle()` with this happy version 😊

This adds:

- vertical bounce
- squash when landing
- stretch when jumping
- subtle wiggle when running

```
function drawBlobCircle(b) {
  // Bobbing offset
  let bob = sin(b.bobT) * b.bobAmount;

  // Squash & stretch
  let stretchY = map(abs(b.vy), 0, 12, 1, 1.25);
  let stretchX = map(abs(b.vy), 0, 12, 1, 0.85);

  // Extra squash on ground
  if (b.onGround) {
    stretchY = 0.9;
    stretchX = 1.1;
  }

  fill(255, 210, 120);
  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);

    let x = cos(a) * r * stretchX;
    let y = sin(a) * r * stretchY;

    vertex(b.x + x, b.y + y + bob);
}

endShape(CLOSE);
}
```



What this does visually

Your blob now:

- 🦊 **bounces happily** while idle
- 🐡 **squashes** when landing
- 🚀 **stretches** when jumping
- 🐸 **wiggles more** when moving faster

It feels *alive* instead of stiff.

If you want next-level cute, I can:

- make it **do a little hop** when changing direction
- add **excited bounce** when touching balls
- or give it a **tail** or **blush cheeks** 😊

	Just tell me the vibe 🧡
Can you add balls around the map that the blob can touch and move around?	<p>Yup — let's do pushable balls 🟠🟢🟣</p> <p>These will sit around the map, fall with gravity, collide with the floor/platforms, and move when the blob touches them.</p> <hr/> <p>1 Add this ball system near the top of your file</p> <p>Put it under your <code>platforms</code> variable:</p> <pre>// --- Pushable balls --- let balls = [];</pre> <hr/> <p>2 Create the balls in <code>setup()</code></p> <p>Add this near the end of <code>setup()</code>:</p> <pre>// Create balls around the map for (let i = 0; i < 6; i++) { balls.push({ x: random(60, width - 60), y: random(40, floorY3 - 120), r: random(10, 16), vx: 0, vy: 0, }); }</pre> <hr/> <p>3 Add ball physics + collisions in <code>draw()</code></p> <p>Put this AFTER drawing the blob (after <code>drawBlobCircle(blob3);</code>):</p>


```

// --- Update & draw balls ---
for (let ball of balls) {
  // Gravity
  ball.vy += 0.4;

  // Move
  ball.x += ball.vx;
  ball.y += ball.vy;

  // Friction
  ball.vx *= 0.98;

  // Platform collisions
  for (const p of platforms) {
    if (
      ball.x + ball.r > p.x &&
      ball.x - ball.r < p.x + p.w &&
      ball.y + ball.r > p.y &&
      ball.y - ball.r < p.y + p.h
    ) {
      // Land on top
      if (ball.vy > 0) {
        ball.y = p.y - ball.r;
        ball.vy = 0;
      }
    }
  }

  // Wall bounds
  if (ball.x - ball.r < 0 || ball.x + ball.r >
width) {
    ball.vx *= -0.8;
  }

  // Blob → ball interaction
  let dx = ball.x - blob3.x;
  let dy = ball.y - blob3.y;
  let dist = sqrt(dx * dx + dy * dy);
  let minDist = ball.r + blob3.r;

```

	<pre> if (dist < minDist) { let angle = atan2(dy, dx); let push = 1.2; ball.vx += cos(angle) * push; ball.vy += sin(angle) * push; // Separate them so they don't stick let overlap = minDist - dist; ball.x += cos(angle) * overlap; ball.y += sin(angle) * overlap; } // Draw ball fill(180, 120, 255); ellipse(ball.x, ball.y, ball.r * 2); } </pre>
<p>Can you add a smiley face to the blob?</p>	<p>Yes!! Super easy — we'll just draw a face on top of the blob so it follows all the happy movement you already added 😊</p> <p>No physics changes, just visuals.</p> <hr/> <p>1) Add this function anywhere below your other functions</p> <p>This draws eyes + a smile relative to the blob's position:</p> <pre> function drawSmileyFace(b) { // Face color fill(60); // Eyes let eyeOffsetX = b.r * 0.35; let eyeOffsetY = b.r * 0.2; let eyeSize = 4; ellipse(b.x - eyeOffsetX, b.y - eyeOffsetY, </pre>

```
eyeSize, eyeSize);
  ellipse(b.x + eyeOffsetX, b.y - eyeOffsetY,
eyeSize, eyeSize);

  // Smile
  noFill();
  stroke(60);
  strokeWeight(2);
  arc(
    b.x,
    b.y + b.r * 0.15,
    b.r * 0.9,
    b.r * 0.6,
    0,
    PI
  );
  noStroke();
}
```

2 Call it in `draw()`

Right **after** this line:



```
drawBlobCircle(blob3);
```

Add:

```
drawSmileyFace(blob3);
```

What you'll see

Your blob now has:

-  cute little eyes
-  a happy smile

- | | |
|--|---|
| | <ul style="list-style-type: none">• 🦊 the face naturally bobs, squashes, and stretches with the body |
|--|---|

Because it's drawn *after* the blob, it stays perfectly on top.