

Process & Decision Documentation

Name: Ayushi Lad

Role(s): Editor

Primary responsibility for this work: Edit/add to the example 3 code

Goal of Work Session

Edit the Example 3 code so the blob expresses a feeling.

Tools: VsCode, GitHub, ChatGPT 5.2

GenAI Documentation

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

Date Used: January 27, 2026

Tool Disclosure: ChatGPT 5.2

Purpose of Use: Help with coding

Summary of Interaction: ChatGPT helped me find pieces of code to add so I could make the blob bounce

Human Decision Point(s): I modified the bounce variables to my liking → it bounced higher

Integrity & Verification Note: I made sure the code was in p5.js and was nothing I did not understand

Scope of GenAI Use: GenAI did not contribute to the README comments, the blob colour, speed, gravity, appearance and environment colour and friction. It only helped with the bounce mechanics.

Limitations or Misfires: The tool did not know how much I wanted the bounce to be.

Summary of Process (Human + Tool)

I used the code and edited it to make sure it was to what I wanted.

Decision Points & Trade-offs

I changed the bounce height because I wanted it to be higher.

Verification & Judgement

I mainly tested out the snippet it gave me and in VScode I played around with the values.

Limitations, Dead Ends, or Open Questions

It made it so the blob would bounce only once but I think it would've been interesting if ChatGPT interpreted my initial prompt so the blob would bounce gradually until it stopped.

Appendix

based on this code, how could i make it so the blob bounces once it touches the ground after jumping:

```
// 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: 30, // Number of points used to draw the blob
```

```
    wobble: 0, // Edge deformation amount
```

```
    wobbleFreq: 0,
```

```
    // 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: 5.0, // Maximum horizontal speed
gravity: 0.25, // 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: 0.95, // 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(133, 194, 230);

// --- Draw all platforms ---

fill(10, 143, 53);

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(254, 241, 31);

    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;
    }
}
```

/* In-class tweaks for experimentation:

- Add a new platform:

```
platforms.push({ x: 220, y: floorY3 - 150, w: 80, h: 12 });
```

- “Ice” feel → frictionGround = 0.95

- “Sand” feel → frictionGround = 0.80

```
*/
```

Ooo yes, a little squashy bounce will make the blob feel way more alive 😊 Right now you **kill all vertical velocity** when landing:

```
blob3.vy = 0;  
blob3.onGround = true;
```

Instead, we’ll **reverse** some of that downward speed so it bounces back up — but only once, not forever like a rubber ball.

⭐ Step 1: Add bounce settings to the blob

Put these inside blob3:

```
bounce: 0.5, // how strong the bounce is (0–1)  
minBounce: 1.5, // prevents tiny micro-bounces  
hasBounced: false // tracks if we've already bounced this jump
```

⭐ Step 2: Reset bounce when a jump starts

In keyPressed() add this line when jumping:

```
blob3.vy = blob3.jumpV;  
blob3.onGround = false;  
blob3.hasBounced = false; // allow a new bounce
```

⭐ Step 3: Change landing logic to bounce

Find this part in **STEP 2 vertical collisions**:

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

Replace it with:

```
if (blob3.vy > 0) {  
    box.y = s.y - box.h;  
  
    // If we haven't bounced yet and we hit with enough speed → bounce!  
    if (!blob3.hasBounced && blob3.vy > blob3.minBounce) {  
        blob3.vy *= -blob3.bounce; // reverse + reduce speed
```

```
blob3.hasBounced = true;  
blob3.onGround = false;  
}  
else {  
    // Otherwise just land normally  
    blob3.vy = 0;  
    blob3.onGround = true;  
}  
}
```

🎯 What this does

Situation	Result
First hit after a jump	Blob bounces upward
Second time touching ground	Blob stays on ground
Small drops	No weird jitter bouncing

🔧 Tweak the feel

- **Higher bounce** → bounce: 0.7 (more cartoony)
 - **Softer blob** → bounce: 0.3
 - **Only bounce from big falls** → increase minBounce
-

If you want next-level polish, we can make the blob **squash on impact and stretch on takeoff** to sell the bounce visually 