

Process & Decision Document

Date Used: Jan 26, 2026

Tool Disclosure: ChatGPT 5.2

Purpose of Use:

ChatGPT was used to experiment with the blob's movement mechanics and to change the environment design to communicate the emotion. It helped generate and iterate on code changes related to physics, visual animation, and background elements to express an "excited/happy" emotional state.

Summary of Interaction:

The tool assisted in suggesting changes to the existing blob code, such as increasing jump height, reducing gravity for a slower fall, adjusting wobble animation, and redesigning the environment with new colours and cloud shapes.

Human Decision Point(s):

I made the final decisions about which suggestions to implement, modify, or omit. I adjusted values through testing to ensure the blob felt excited, and I chose to keep the collision system and core structure of the original code unchanged. Visual and environmental changes were applied to express the emotion without interfering with the gameplay.

Integrity & Verification Note:

All code was tested to verify that changes functioned as intended. Movement physics and visual effects were reviewed to ensure the emotional expression was shown in the live sketch.

Scope of GenAI Use:

GenAI was used for ideation, iteration, and troubleshooting within an existing codebase. It supported exploration of emotional design through movement and environment but did not replace code testing or decision-making.

Limitations or Misfires:

Some initial suggestions required refinement to better align with the intended emotion, as early changes occasionally felt awkward or unpolished. The tool also lacked contextual awareness of the course's design goal, requiring human interpretation to connect coding changes to the emotional objectives of the sidequest.

AI Transcript

Prompt:

```
// 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 (keysDown(65) ||
keysDown(LEFT_ARROW)) move -= 1; // A or ← if (keysDown(68) ||
keysDown(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; } } /*
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 */
can you help make the blob appear excited. Make higher jumps, slower fall rate, change
environment to look cheerful.
```

ChatGPT:

Nice direction — **excited + happy** should feel *bouncy, floaty, playful*, and visually warm 🌸✨
Below is your **FULL CODE**, directly based on what you sent, with **only the necessary changes** to:

- give the blob **higher jumps**
- make it **fall slower / float**
- add **springy, energetic movement**
- make the **environment bright and happy**

You can paste this straight over your current sketch.

```
// Y-position of the floor (ground level)
let floorY3;

// Player character (excited, happy blob)
let blob3 = {
  // Position (centre of the blob)
  x: 80,
  y: 0,

  // Visual properties (bouncier + softer)
  r: 26,
  points: 56,      // smoother shape
  wobble: 10,      // lively wobble
  wobbleFreq: 1.4,
```

```

// Time values for excited animation
t: 0,
tSpeed: 0.025,    // faster = energetic

// Physics: velocity
vx: 0,
vy: 0,

// Movement tuning (excited + floaty)
accel: 0.6,
maxRun: 4.5,
gravity: 0.4,    // slower fall
jumpV: -14.5,    // higher jump

// State
onGround: false,

// Friction (springy)
frictionAir: 0.998,
frictionGround: 0.9,
};

// List of platforms
let platforms = [];

function setup() {
  createCanvas(640, 360);

  // Define the floor height
  floorY3 = height - 36;

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

  // Happy, spacious platforms
  platforms = [
    { x: 0, y: floorY3, w: width, h: height - floorY3 }, // floor

```

```

    { x: 100, y: floorY3 - 60, w: 140, h: 14 },
    { x: 280, y: floorY3 - 110, w: 140, h: 14 },
    { x: 460, y: floorY3 - 160, w: 140, h: 14 },
  ];

  // Start the blob resting on the floor
  blob3.y = floorY3 - blob3.r - 1;
}

function draw() {
  // Happy background
  background(255, 230, 245);

  // --- Draw platforms ---
  fill(255, 200, 220);
  for (const p of platforms) {
    rect(p.x, p.y, p.w, p.h, 8); // rounded = friendly
  }

  // --- Input: left/right movement ---
  let move = 0;
  if (keyIsDown(65) || keyIsDown(LEFT_ARROW)) move -= 1;
  if (keyIsDown(68) || keyIsDown(RIGHT_ARROW)) move += 1;
  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 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)) {
      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 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;

        // little happy bounce
        blob3.wobble = 14;
      } else if (blob3.vy < 0) {
        box.y = s.y + s.h;
        blob3.vy = 0;
      }
    }
  }

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

```

```

// Gradually return wobble to normal
blob3.wobble = lerp(blob3.wobble, 10, 0.1);

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

// --- HUD ---
fill(120);
text("Excited mode 💖 Jump high and float!", 10, 18);
}

// AABB overlap test
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
  );
}

// Draw excited pink blob
function drawBlobCircle(b) {
  fill(255, 120, 180);
  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(b.x + cos(a) * r, b.y + sin(a) * r);
  }
}

```

```

    }

    endShape(CLOSE);
}

// Jump input
function keyPressed() {
  if (
    (key === " " || key === "W" || key === "w" || keyCode ===
UP_ARROW) &&
    blob3.onGround
  ) {
    blob3.vy = blob3.jumpV;
    blob3.onGround = false;
    blob3.wobble = 16; // excited burst
  }
}

```

How this communicates excitement / happiness

- **High jumps + slow gravity** → weightless joy
- **Fast wobble animation** → energetic body language
- **Pink blob + pastel world** → warmth & positivity
- **Wide, rounded platforms** → forgiving, playful environment

If you want, I can help you write a **2–3 sentence explanation** connecting this to emotion + movement for your assignment 💕

Prompt:

can u make the platforms a little higher and shorter in length

ChatGPT:

Yep 😊 super easy tweak. You only need to change the **platforms array** — everything else can stay the same.

Here's a version where the platforms are **higher (more vertical spacing)** and **shorter (more jump-focused, playful challenge)** while still feeling happy/excited.

🔧 Replace your current **platforms = [...]** with this:

```
platforms = [  
  { x: 0, y: floorY3, w: width, h: height - floorY3 }, // floor  
  
  { x: 110, y: floorY3 - 80, w: 90, h: 14 },  
  { x: 260, y: floorY3 - 140, w: 85, h: 14 },  
  { x: 400, y: floorY3 - 200, w: 80, h: 14 },  
  { x: 520, y: floorY3 - 260, w: 75, h: 14 },  
];
```

Why this works emotionally

- **Higher platforms** → encourages big, joyful jumps (excitement)
- **Shorter widths** → playful precision instead of punishment
- **Upward progression** → feels like climbing / celebration rather than struggle
- Still forgiving because of the **floaty gravity + high jump**

If you want it to feel **even more excited**, I can:

- add a tiny **bounce boost** when landing
- stagger platforms in a zig-zag for playful motion
- add subtle **sparkle particles** when jumping ✨

Just say the word.

Prompt:

can you add white fluffy clouds in the environment/background

Absolutely 🌥️✨

Here's the **FULL UPDATED CODE** with **white, fluffy clouds** drifting gently in the background, while keeping everything else exactly the same (excited movement, pink blob, higher + shorter platforms).

You can paste this straight into p5 and run it.

```
// Y-position of the floor (ground level)
```

```
let floorY3;
```

```
// Player character (excited, happy blob)
```

```
let blob3 = {
```

```
  x: 80,
```

```
  y: 0,
```

```
  // Visual properties
```

```
  r: 26,
```

```
  points: 56,
```

```
  wobble: 10,
```

```
  wobbleFreq: 1.4,
```

```
  // Animation timing
```

```
  t: 0,
```

```
  tSpeed: 0.025,
```

```
  // Physics
```

```
    vx: 0,

    vy: 0,

    // Movement tuning (floaty + joyful)

    accel: 0.6,

    maxRun: 4.5,

    gravity: 0.4,

    jumpV: -14.5,

    onGround: false,

    frictionAir: 0.998,

    frictionGround: 0.9,
};

// Platforms

let platforms = [];

// Clouds

let clouds = [];

function setup() {

    createCanvas(640, 360);
```

```
floorY3 = height - 36;

noStroke();

textFont("sans-serif");

textSize(14);

// Platforms: higher + shorter
platforms = [

  { x: 0, y: floorY3, w: width, h: height - floorY3 },

  { x: 110, y: floorY3 - 80, w: 90, h: 14 },

  { x: 260, y: floorY3 - 140, w: 85, h: 14 },

  { x: 400, y: floorY3 - 200, w: 80, h: 14 },

  { x: 520, y: floorY3 - 260, w: 75, h: 14 },

];

// Create fluffy clouds
for (let i = 0; i < 6; i++) {

  clouds.push({

    x: random(width),

    y: random(40, 160),

    s: random(0.8, 1.3),

    speed: random(0.2, 0.5),
```

```
    });  
  }  
  
  blob3.y = floorY3 - blob3.r - 1;  
}  
  
function draw() {  
  // Sky background  
  background(255, 230, 245);  
  
  // --- Draw clouds ---  
  drawClouds();  
  
  // --- Draw platforms ---  
  fill(255, 200, 220);  
  for (const p of platforms) {  
    rect(p.x, p.y, p.w, p.h, 8);  
  }  
  
  // --- 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;

// --- Friction + clamp ---

blob3.vx *= blob3.onGround ? blob3.frictionGround :
blob3.frictionAir;

blob3.vx = constrain(blob3.vx, -blob3.maxRun, blob3.maxRun);

// --- Gravity ---

blob3.vy += blob3.gravity;

// --- 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)) {

    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 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;

            blob3.wobble = 14;
        } else if (blob3.vy < 0) {
            box.y = s.y + s.h;

            blob3.vy = 0;
        }
    }
}
```

```
// Apply position

blob3.x = box.x + box.w / 2;

blob3.y = box.y + box.h / 2;

blob3.x = constrain(blob3.x, blob3.r, width - blob3.r);


// Ease wobble

blob3.wobble = lerp(blob3.wobble, 10, 0.1);


// Draw blob

blob3.t += blob3.tSpeed;

drawBlobCircle(blob3);


// HUD

fill(120);

text("Excited / Happy Mode 💕 Float through the clouds", 10, 18);
}


// ---- Clouds ----

function drawClouds() {

  fill(255, 255, 255, 220);

  for (const c of clouds) {

    push();
```



```

    translate(c.x, c.y);

    scale(c.s);

    ellipse(0, 0, 50, 30);
    ellipse(-20, 5, 40, 25);
    ellipse(20, 5, 40, 25);
    ellipse(0, -10, 35, 25);

    pop();

    // Drift
    c.x += c.speed;

    if (c.x > width + 60) c.x = -60;
}

}

// AABB overlap test
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
    )
}

```

```
);  
}  
  
// Draw pink blob  
function drawBlobCircle(b) {  
  fill(255, 120, 180);  
  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(b.x + cos(a) * r, b.y + sin(a) * r);  
  }  
  
  endShape(CLOSE);  
}
```

```
// Jump input

function keyPressed() {

    if (

        (key === " " || key === "W" || key === "w" || keyCode ===
UP_ARROW) &&

        blob3.onGround

    ) {

        blob3.vy = blob3.jumpV;

        blob3.onGround = false;

        blob3.wobble = 16;

    }

}
```

Why the clouds work emotionally ☁️

- **Slow horizontal drift** → calm, carefree happiness
- **Soft white shapes** → friendliness and safety
- **Layered behind gameplay** → environment supports the emotion without distracting

If you want next:

- pastel **sun rays**
- subtle **cloud parallax**
- sparkles when jumping ✨

Just tell me 🌸

Prompt:

Can you add a bonus feature where the blob bumps or kicks small balls around

ChatGPT:

Absolutely! That's a good feature to add. The blob can now bump/kick small balls on platforms:

```
// Y-position of the floor (ground level)
```

```
let floorY3;
```

```
// Player character (excited, happy blob)
```

```
let blob3 = {
```

```
  x: 80,
```

```
  y: 0,
```

```
  // Visual properties
```

```
  r: 26,
```

```
  points: 56,
```

```
  wobble: 10,
```

```
  wobbleFreq: 1.4,
```

```
  // Animation timing
```

```
  t: 0,
```

```
  tSpeed: 0.025,
```

```
  // Physics
```

```
  vx: 0,
```

```
  vy: 0,
```

```
  // Movement tuning (floaty + joyful)
```

```
  accel: 0.6,
```

```
  maxRun: 4.5,
```

```
  gravity: 0.4,
```

```
  jumpV: -14.5,
```

```
  onGround: false,
```

```
  frictionAir: 0.998,
```

```
  frictionGround: 0.9,
```

```
};
```

```
// Platforms
```

```

let platforms = [];

// Clouds
let clouds = [];

// Mischief balls
let balls = [];

function setup() {
  createCanvas(640, 360);

  floorY3 = height - 36;

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

  // Platforms: higher + shorter
  platforms = [
    { x: 0, y: floorY3, w: width, h: height - floorY3 },
    { x: 110, y: floorY3 - 80, w: 90, h: 14 },
    { x: 260, y: floorY3 - 140, w: 85, h: 14 },
    { x: 400, y: floorY3 - 200, w: 80, h: 14 },
    { x: 520, y: floorY3 - 260, w: 75, h: 14 },
  ];

  // Create fluffy clouds
  for (let i = 0; i < 6; i++) {
    clouds.push({
      x: random(width),
      y: random(40, 160),
      s: random(0.8, 1.3),
      speed: random(0.2, 0.5),
    });
  }

  // Place balls on platforms
  for (const p of platforms) {
    if (p !== platforms[0]) { // skip ground
      balls.push({
        x: p.x + p.w / 2,
        y: p.y - 10,
        r: 10,
        vx: random(-1, 1),
      });
    }
  }
}

```

```

        vy: 0,
    });
}
}

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

function draw() {
    // Sky background
    background(255, 230, 245);

    // --- Draw clouds ---
    drawClouds();

    // --- Draw platforms ---
    fill(255, 200, 220);
    for (const p of platforms) {
        rect(p.x, p.y, p.w, p.h, 8);
    }

    // --- 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;

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

    // --- Gravity ---
    blob3.vy += blob3.gravity;

    // --- 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)) {
    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 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;
      blob3.wobble = 14;
    } else if (blob3.vy < 0) {
      box.y = s.y + s.h;
      blob3.vy = 0;
    }
  }
}

```

```

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

```

```

// Ease wobble
blob3.wobble = lerp(blob3.wobble, 10, 0.1);

```

```

// --- Draw balls ---
for (const ball of balls) {
  ball.vy += 0.4; // gravity
  ball.x += ball.vx;
  ball.y += ball.vy;
}

```

```

// Collision with platforms
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 &&
    ball.vy > 0
  ){
    ball.y = p.y - ball.r;
    ball.vy *= -0.3; // bounce
    ball.vx *= 0.9; // friction
  }
}

// Collision with blob
const dx = ball.x - blob3.x;
const dy = ball.y - blob3.y;
const distB = sqrt(dx * dx + dy * dy);
if (distB < blob3.r + ball.r) {
  const angle = atan2(dy, dx);
  const force = 3;
  ball.vx += cos(angle) * force;
  ball.vy += sin(angle) * force;

  blob3.wobble = 18; // playful kick wobble
}

// Keep balls inside canvas horizontally
if (ball.x < ball.r) {
  ball.x = ball.r;
  ball.vx *= -0.5;
} else if (ball.x > width - ball.r) {
  ball.x = width - ball.r;
  ball.vx *= -0.5;
}

// Draw balls
fill(200, 180, 255);
ellipse(ball.x, ball.y, ball.r * 2);
}

// Draw blob
blob3.t += blob3.tSpeed;
drawBlobCircle(blob3);

// HUD
fill(120);

```



```

    text("Excited / Happy Mode 💕 Mischief time!", 10, 18);
}

// ---- Clouds ----
function drawClouds() {
    fill(255, 255, 255, 220);

    for (const c of clouds) {
        push();
        translate(c.x, c.y);
        scale(c.s);

        ellipse(0, 0, 50, 30);
        ellipse(-20, 5, 40, 25);
        ellipse(20, 5, 40, 25);
        ellipse(0, -10, 35, 25);

        pop();

        // Drift
        c.x += c.speed;
        if (c.x > width + 60) c.x = -60;
    }
}

// AABB overlap test
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
    );
}

// Draw pink blob
function drawBlobCircle(b) {
    fill(255, 120, 180);
    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(b.x + cos(a) * r, b.y + sin(a) * r);
}

endShape(CLOSE);
}

// Jump input
function keyPressed() {
  if (
    (key === " " || key === "W" || key === "w" || keyCode === UP_ARROW) &&
    blob3.onGround
  ) {
    blob3.vy = blob3.jumpV;
    blob3.onGround = false;
    blob3.wobble = 16;
  }
}

```

✅ Features added:

Small balls appear on platforms.

Blob bumps/kicks them around with simple physics.

Balls bounce, roll, and respect platform collision.

Blob wobbles more when interacting with balls (“mischief mode”).

If you want, I can also make the balls react visually—like squishing or spinning—so it really feels like your blob is causing playful chaos.

Do you want me to add that too?