

I decided to display the emotion of panic, Some of the major changes include changes to the blob to be more irregular and the wobble frequency to make it look more chaotic, and more.

As for the mischief mechanic i added the small blocks on the map so that the blob could push them around.

ChatGPT prompt

Lightly edit this code to express the emotion of panic, redesign things like the blob's movement and environment. Also add a "mischief" mechanic, so the blob bumps objects on the map.-

```
// 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 --- fill(0); text("Blob breathing via noise(). Try wobble and tSpeed.", 10, 18); }
```

ChatGPT output

Perfect! We can lightly edit your existing code to make the blob look panicked and add a mischief mechanic with minimal changes, keeping it simple and readable.

Here's a copy-pastable version:

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

```
let floorY3;
```

```

// Player character (soft, animated blob)
let blob3 = {
  x: 80,
  y: 0,
  // Visual properties
  r: 26,
  points: 48,
  wobble: 16,
  wobbleFreq: 1.6,
  // Animation
  t: 0,
  tSpeed: 0.06,
  // Physics
  vx: 0,
  vy: 0,
  accel: 1.0,
  maxRun: 7.0,
  gravity: 1.0,
  jumpV: -14.5,
  onGround: false,
  frictionAir: 0.99,
  frictionGround: 0.78,
};

// Platforms
let platforms = [];
// Mischief objects (small boxes the blob can bump)
let objects = [];

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 },
  ]
}

```

```

];
// Small movable objects
objects = [
  { x: 200, y: floorY3 - 20, w: 20, h: 20, vx: 0 },
  { x: 350, y: floorY3 - 20, w: 20, h: 20, vx: 0 },
  { x: 500, y: floorY3 - 20, w: 20, h: 20, vx: 0 },
];
blob3.y = floorY3 - blob3.r - 1;
}
function draw() {
  background(240);
  // Draw platforms
  fill(200);
  for (const p of platforms) {
    rect(p.x, p.y, p.w, p.h);
  }
  // Draw and update mischief objects
  fill(120);
  for (const o of objects) {
    o.x += o.vx;
    o.vx *= 0.9; // friction
    o.x = constrain(o.x, 0, width - o.w);
    rect(o.x, o.y, o.w, o.h);
  }
  // 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;
  // Collision box
  let box = {
    x: blob3.x - blob3.r,
    y: blob3.y - blob3.r,
    w: blob3.r * 2,
    h: blob3.r * 2,
  };
  // Horizontal movement

```

```

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 movement
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);

// Mischief: bump objects
for (const o of objects) {
  if (
    blob3.x + blob3.r > o.x &&
    blob3.x - blob3.r < o.x + o.w &&
    blob3.y + blob3.r > o.y &&
    blob3.y - blob3.r < o.y + o.h
  ) {
    o.vx += blob3.vx * 0.8; // transfer energy
  }
}

```

```

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

fill(0);
text("Emotion: Panic + Mischief — bump objects to cause chaos", 10, 18);
}

// AABB overlap
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 blob
function drawBlobCircle(b) {
  fill(200, 40, 40);
  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) +
      random(-1.5, 1.5);
    vertex(b.x + cos(a) * r, b.y + sin(a) * r);
  }

  endShape(CLOSE);
}

```

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