

## Process & Decision Document

**Name:** Jasmine Henri

**Role(s):** Coder & Reviewer of AI changes

**Primary responsibility for this work:** Work with AI and independently to come up with modifications for the code that will successfully demonstrate specific emotions

**Goal of Work Session:** During this work session, I focused on making changes to the blobs movements and environment to express specific emotions.

**GenAI tools (if used):**

- Chatbot on Visual Studio Code 6

### GenAI Documentation

**Date Used:** January 25th, 2026

**Tool Disclosure:** Chatbot on Visual Studio Code 6

**Purpose of Use:** Add new code lines to the forked code to bring my suggestions and ideas to light.

**Summary of Interaction:** I explained to the chatbot what my ideas were to modify the code to express emotion and it would take my instructions and make edits to the codes, then I was able to see review the added code to determine if I either accept the change or wish to decline it.

**Human Decision Point(s): I was happy with the specific changes that were made to the code and had to initiate minimal changes.**

- Changed the colour of the blob when on platforms to random(255) to make it more fun instead of one colour change (however it caused a flickering which would have triggered light sensitive viewers so I turned to the chatbot to help me fix the flickering)
- Modified the background myself so the colour would be grey when the blob is on the ground/ in the air and light blue when the blob is on a platform

### Integrity & Verification Note:

All decisions were verified using:

- My prior knowledge of coding from CS105 and CS106
- Knowledge of coding that was reviewed in Class 2 lecture by David

### **Scope of GenAI Use:**

Specifically, GenAI did *not*:

- Create the initial background colour change code (however it did modify the location in draw and slightly edited the coding so that the background wasn't drawn over the ledges)
- Randomize the blobs colour

### **Limitations or Misfires:**

The model demonstrated no limitations and strongly aided in bringing my design changes to life.

### **Summary of Process (Human + Tool)**

Four iterations with GenAI:

- 1: The Chatbot added code to make the platforms turn yellow when the blob is actively standing on it
- 2: The Chatbot added code to make the blob black if it's on the ground but blue if it's on a platform
- 3: The Chatbot corrected the flickering after I had added the random colour code
- 4: The Chatbot corrected and re-organized the code when I added a conditional statement to control the background colour that had interfered with the blob changing colours

### **Decision Points & Trade-offs**

I made individual design decisions to change the blob colour when on a ledge and the background colour change, however this ended up requiring the chatbots help because my changes caused unintentional changes to other aspects of the code.

### **Verification & Judgement**

I evaluated the appropriateness of my changes by:

- Reviewing notes from CS105 and CS106

- Reviewing Class 2 lecture by David
- Reflecting on how flickering colours can negatively impact light sensitive viewers

### Limitations, Dead Ends, or Open Questions

- All issues were resolved successfully by the chatbot

### Appendix - Visual Studio Code 6 Chatbot

#1 Could you make edits to the code to make the platforms turn yellow when the blob is actively standing on them

Now I'll modify the code to track which platform the blob is standing on and draw that platform in yellow. I need to:

1. Add a property to track which platform the blob is on
2. Check which platform the blob is standing on during collision detection
3. Draw platforms with different colors based on whether the blob is on them

```
Edited  
  
sketch.js+1-0  
  
Edited  
  
sketch.js+6-0  
  
Edited  
  
sketch.js+2-0
```

Done! I've updated the code to make platforms turn yellow when the blob is actively standing on them.

Here's what I changed:

1. **Added a tracking property** (`standingOnPlatform`) to the blob to remember which platform it's currently standing on
2. **Updated the collision detection** to set `blob3.standingOnPlatform` when the blob lands on a platform
3. **Modified the platform drawing** to check if each platform is the one the blob is standing on, and if so, draw it in yellow instead of gray

Now when you run the sketch, any platform the blob is standing on will turn bright yellow, and it will return to gray when the blob jumps off.

File: sketch.js

```

43 function setup() {
52 // Create platforms (floor + steps)
53 platforms = [
54   {
55     x: 0, y: floorY3, w: width, h: height - floorY3 }, // floor
56     { x: 120, y: floorY3 - 70, w: 120, h: 12 }, // low step
57     { x: 300, y: floorY3 - 120, w: 90, h: 12 }, // mid step
58     { x: 440, y: floorY3 - 180, w: 130, h: 12 }, // high step
59     { x: 520, y: floorY3 - 70, w: 90, h: 12 } // return ramp
60   ];
61 }
62 // Start the blob resting on the floor
63 blob3.y = floorY3 - blob3.r - 1;
64 }

65 function draw() {
66   background(240);
67   // --- Draw all platforms ---
68   fill(200);
69   for (const p of platforms) {
70     // Change the colour of the platform to yellow if the blob is standing on it, if the ledge is not
71     if (p === blob3.standingOnPlatform) {
72       fill(255, 255, 0); // Yellow
73     } else {
74       fill(200); // Grey
75     }
76     rect(p.x, p.y, p.w, p.h);
77   }
78   // Change the blob's color based on its position
79   if (blob3.onGround) {
80     fill(0); // Black when on the ground
81   } else {
82     fill(random(255), random(255), random(255)); // Random color when on a platform
83   }
84 }

85 // --- Input: left/right movement ---
86 let move = 0;
87 if (keyIsDown(65) || keyIsDown(LEFT_ARROW)) move -= 1; // A or ←
88 if (keyIsDown(68) || keyIsDown(RIGHT_ARROW)) move += 1; // D or →
89 blob3.vx += blob3.accel * move;
90

91 // --- Apply friction and clamp speed ---
92 blob3.vx *= blob3.onGround ? blob3.frictionGround : blob3.frictionAir;
93

94 // --- Apply gravity ---
95 blob3.vy += blob3.gravity;
96

97 // --- Collision representation ---
98 // We collide using a rectangle (AABB),
99 // even though the blob is drawn as a circle
100 let box = [
101   { x: 0, y: 0, w: 10, h: 10 }
102 ];
103
104 // Check collision with platforms
105 for (const p of platforms) {
106   const box = [
107     { x: p.x, y: p.y, w: p.w, h: p.h }
108   ];
109   if (collide(box, box)) {
110     blob3.standingOnPlatform = true;
111   } else {
112     blob3.standingOnPlatform = false;
113   }
114 }
115
116 // Update blob's position
117 blob3.x += move * blob3.accel;
118 blob3.y += blob3.vy;
119
120 // Clamp blob's position to the floor
121 if (blob3.y <= floorY3) {
122   blob3.y = floorY3;
123 }
124
125 // Draw the blob
126 blob3.draw();
127
128 // Draw the floor
129 fill(200);
130 rect(0, floorY3, width, height - floorY3);
131
132 // Draw the platforms
133 for (const p of platforms) {
134   rect(p.x, p.y, p.w, p.h);
135 }
136
137 // Draw the blob's shadow
138 fill(100);
139 rect(blob3.x, blob3.y + 10, blob3.r, 10);
140
141 // Draw the floor's shadow
142 fill(100);
143 rect(0, floorY3 + 10, width, 10);
144
145 // Draw the platforms' shadows
146 for (const p of platforms) {
147   rect(p.x, p.y + 10, p.w, 10);
148 }
149
150 // Draw the blob's reflection
151 fill(100);
152 rect(blob3.x, height - blob3.r, blob3.r, 10);
153
154 // Draw the floor's reflection
155 fill(100);
156 rect(0, height - 10, width, 10);
157
158 // Draw the platforms' reflections
159 for (const p of platforms) {
160   rect(p.x, height - 10, p.w, 10);
161 }
162
163 // Draw the blob's highlights
164 fill(255);
165 rect(blob3.x, blob3.y, blob3.r, 10);
166
167 // Draw the floor's highlights
168 fill(255);
169 rect(0, floorY3, width, 10);
170
171 // Draw the platforms' highlights
172 for (const p of platforms) {
173   rect(p.x, floorY3, p.w, 10);
174 }
175
176 // Draw the blob's shadow highlights
177 fill(255);
178 rect(blob3.x, blob3.y + 10, blob3.r, 10);
179
180 // Draw the floor's shadow highlights
181 fill(255);
182 rect(0, floorY3 + 10, width, 10);
183
184 // Draw the platforms' shadow highlights
185 for (const p of platforms) {
186   rect(p.x, floorY3 + 10, p.w, 10);
187 }
188
189 // Draw the blob's reflection highlights
190 fill(255);
191 rect(blob3.x, height - blob3.r, blob3.r, 10);
192
193 // Draw the floor's reflection highlights
194 fill(255);
195 rect(0, height - 10, width, 10);
196
197 // Draw the platforms' reflection highlights
198 for (const p of platforms) {
199   rect(p.x, height - 10, p.w, 10);
200 }
201
202 // Draw the blob's highlights
203 fill(255);
204 rect(blob3.x, blob3.y, blob3.r, 10);
205
206 // Draw the floor's highlights
207 fill(255);
208 rect(0, floorY3, width, 10);
209
210 // Draw the platforms' highlights
211 for (const p of platforms) {
212   rect(p.x, floorY3, p.w, 10);
213 }
214
215 // Draw the blob's shadow highlights
216 fill(255);
217 rect(blob3.x, blob3.y + 10, blob3.r, 10);
218
219 // Draw the floor's shadow highlights
220 fill(255);
221 rect(0, floorY3 + 10, width, 10);
222
223 // Draw the platforms' shadow highlights
224 for (const p of platforms) {
225   rect(p.x, floorY3 + 10, p.w, 10);
226 }
227
228 // Draw the blob's reflection highlights
229 fill(255);
230 rect(blob3.x, height - blob3.r, blob3.r, 10);
231
232 // Draw the floor's reflection highlights
233 fill(255);
234 rect(0, height - 10, width, 10);
235
236 // Draw the platforms' reflection highlights
237 for (const p of platforms) {
238   rect(p.x, height - 10, p.w, 10);
239 }
240
241 // Draw the blob's highlights
242 fill(255);
243 rect(blob3.x, blob3.y, blob3.r, 10);
244
245 // Draw the floor's highlights
246 fill(255);
247 rect(0, floorY3, width, 10);
248
249 // Draw the platforms' highlights
250 for (const p of platforms) {
251   rect(p.x, floorY3, p.w, 10);
252 }
253
254 // Draw the blob's shadow highlights
255 fill(255);
256 rect(blob3.x, blob3.y + 10, blob3.r, 10);
257
258 // Draw the floor's shadow highlights
259 fill(255);
260 rect(0, floorY3 + 10, width, 10);
261
262 // Draw the platforms' shadow highlights
263 for (const p of platforms) {
264   rect(p.x, floorY3 + 10, p.w, 10);
265 }
266
267 // Draw the blob's reflection highlights
268 fill(255);
269 rect(blob3.x, height - blob3.r, blob3.r, 10);
270
271 // Draw the floor's reflection highlights
272 fill(255);
273 rect(0, height - 10, width, 10);
274
275 // Draw the platforms' reflection highlights
276 for (const p of platforms) {
277   rect(p.x, height - 10, p.w, 10);
278 }
279
280 // Draw the blob's highlights
281 fill(255);
282 rect(blob3.x, blob3.y, blob3.r, 10);
283
284 // Draw the floor's highlights
285 fill(255);
286 rect(0, floorY3, width, 10);
287
288 // Draw the platforms' highlights
289 for (const p of platforms) {
290   rect(p.x, floorY3, p.w, 10);
291 }
292
293 // Draw the blob's shadow highlights
294 fill(255);
295 rect(blob3.x, blob3.y + 10, blob3.r, 10);
296
297 // Draw the floor's shadow highlights
298 fill(255);
299 rect(0, floorY3 + 10, width, 10);
300
301 // Draw the platforms' shadow highlights
302 for (const p of platforms) {
303   rect(p.x, floorY3 + 10, p.w, 10);
304 }
305
306 // Draw the blob's reflection highlights
307 fill(255);
308 rect(blob3.x, height - blob3.r, blob3.r, 10);
309
310 // Draw the floor's reflection highlights
311 fill(255);
312 rect(0, height - 10, width, 10);
313
314 // Draw the platforms' reflection highlights
315 for (const p of platforms) {
316   rect(p.x, height - 10, p.w, 10);
317 }
318
319 // Draw the blob's highlights
320 fill(255);
321 rect(blob3.x, blob3.y, blob3.r, 10);
322
323 // Draw the floor's highlights
324 fill(255);
325 rect(0, floorY3, width, 10);
326
327 // Draw the platforms' highlights
328 for (const p of platforms) {
329   rect(p.x, floorY3, p.w, 10);
330 }
331
332 // Draw the blob's shadow highlights
333 fill(255);
334 rect(blob3.x, blob3.y + 10, blob3.r, 10);
335
336 // Draw the floor's shadow highlights
337 fill(255);
338 rect(0, floorY3 + 10, width, 10);
339
340 // Draw the platforms' shadow highlights
341 for (const p of platforms) {
342   rect(p.x, floorY3 + 10, p.w, 10);
343 }
344
345 // Draw the blob's reflection highlights
346 fill(255);
347 rect(blob3.x, height - blob3.r, blob3.r, 10);
348
349 // Draw the floor's reflection highlights
350 fill(255);
351 rect(0, height - 10, width, 10);
352
353 // Draw the platforms' reflection highlights
354 for (const p of platforms) {
355   rect(p.x, height - 10, p.w, 10);
356 }
357
358 // Draw the blob's highlights
359 fill(255);
360 rect(blob3.x, blob3.y, blob3.r, 10);
361
362 // Draw the floor's highlights
363 fill(255);
364 rect(0, floorY3, width, 10);
365
366 // Draw the platforms' highlights
367 for (const p of platforms) {
368   rect(p.x, floorY3, p.w, 10);
369 }
370
371 // Draw the blob's shadow highlights
372 fill(255);
373 rect(blob3.x, blob3.y + 10, blob3.r, 10);
374
375 // Draw the floor's shadow highlights
376 fill(255);
377 rect(0, floorY3 + 10, width, 10);
378
379 // Draw the platforms' shadow highlights
380 for (const p of platforms) {
381   rect(p.x, floorY3 + 10, p.w, 10);
382 }
383
384 // Draw the blob's reflection highlights
385 fill(255);
386 rect(blob3.x, height - blob3.r, blob3.r, 10);
387
388 // Draw the floor's reflection highlights
389 fill(255);
390 rect(0, height - 10, width, 10);
391
392 // Draw the platforms' reflection highlights
393 for (const p of platforms) {
394   rect(p.x, height - 10, p.w, 10);
395 }
396
397 // Draw the blob's highlights
398 fill(255);
399 rect(blob3.x, blob3.y, blob3.r, 10);
400
401 // Draw the floor's highlights
402 fill(255);
403 rect(0, floorY3, width, 10);
404
405 // Draw the platforms' highlights
406 for (const p of platforms) {
407   rect(p.x, floorY3, p.w, 10);
408 }
409
410 // Draw the blob's shadow highlights
411 fill(255);
412 rect(blob3.x, blob3.y + 10, blob3.r, 10);
413
414 // Draw the floor's shadow highlights
415 fill(255);
416 rect(0, floorY3 + 10, width, 10);
417
418 // Draw the platforms' shadow highlights
419 for (const p of platforms) {
420   rect(p.x, floorY3 + 10, p.w, 10);
421 }
422
423 // Draw the blob's reflection highlights
424 fill(255);
425 rect(blob3.x, height - blob3.r, blob3.r, 10);
426
427 // Draw the floor's reflection highlights
428 fill(255);
429 rect(0, height - 10, width, 10);
430
431 // Draw the platforms' reflection highlights
432 for (const p of platforms) {
433   rect(p.x, height - 10, p.w, 10);
434 }
435
436 // Draw the blob's highlights
437 fill(255);
438 rect(blob3.x, blob3.y, blob3.r, 10);
439
440 // Draw the floor's highlights
441 fill(255);
442 rect(0, floorY3, width, 10);
443
444 // Draw the platforms' highlights
445 for (const p of platforms) {
446   rect(p.x, floorY3, p.w, 10);
447 }
448
449 // Draw the blob's shadow highlights
450 fill(255);
451 rect(blob3.x, blob3.y + 10, blob3.r, 10);
452
453 // Draw the floor's shadow highlights
454 fill(255);
455 rect(0, floorY3 + 10, width, 10);
456
457 // Draw the platforms' shadow highlights
458 for (const p of platforms) {
459   rect(p.x, floorY3 + 10, p.w, 10);
460 }
461
462 // Draw the blob's reflection highlights
463 fill(255);
464 rect(blob3.x, height - blob3.r, blob3.r, 10);
465
466 // Draw the floor's reflection highlights
467 fill(255);
468 rect(0, height - 10, width, 10);
469
470 // Draw the platforms' reflection highlights
471 for (const p of platforms) {
472   rect(p.x, height - 10, p.w, 10);
473 }
474
475 // Draw the blob's highlights
476 fill(255);
477 rect(blob3.x, blob3.y, blob3.r, 10);
478
479 // Draw the floor's highlights
480 fill(255);
481 rect(0, floorY3, width, 10);
482
483 // Draw the platforms' highlights
484 for (const p of platforms) {
485   rect(p.x, floorY3, p.w, 10);
486 }
487
488 // Draw the blob's shadow highlights
489 fill(255);
490 rect(blob3.x, blob3.y + 10, blob3.r, 10);
491
492 // Draw the floor's shadow highlights
493 fill(255);
494 rect(0, floorY3 + 10, width, 10);
495
496 // Draw the platforms' shadow highlights
497 for (const p of platforms) {
498   rect(p.x, floorY3 + 10, p.w, 10);
499 }
500
501 // Draw the blob's reflection highlights
502 fill(255);
503 rect(blob3.x, height - blob3.r, blob3.r, 10);
504
505 // Draw the floor's reflection highlights
506 fill(255);
507 rect(0, height - 10, width, 10);
508
509 // Draw the platforms' reflection highlights
510 for (const p of platforms) {
511   rect(p.x, height - 10, p.w, 10);
512 }
513
514 // Draw the blob's highlights
515 fill(255);
516 rect(blob3.x, blob3.y, blob3.r, 10);
517
518 // Draw the floor's highlights
519 fill(255);
520 rect(0, floorY3, width, 10);
521
522 // Draw the platforms' highlights
523 for (const p of platforms) {
524   rect(p.x, floorY3, p.w, 10);
525 }
526
527 // Draw the blob's shadow highlights
528 fill(255);
529 rect(blob3.x, blob3.y + 10, blob3.r, 10);
530
531 // Draw the floor's shadow highlights
532 fill(255);
533 rect(0, floorY3 + 10, width, 10);
534
535 // Draw the platforms' shadow highlights
536 for (const p of platforms) {
537   rect(p.x, floorY3 + 10, p.w, 10);
538 }
539
540 // Draw the blob's reflection highlights
541 fill(255);
542 rect(blob3.x, height - blob3.r, blob3.r, 10);
543
544 // Draw the floor's reflection highlights
545 fill(255);
546 rect(0, height - 10, width, 10);
547
548 // Draw the platforms' reflection highlights
549 for (const p of platforms) {
550   rect(p.x, height - 10, p.w, 10);
551 }
552
553 // Draw the blob's highlights
554 fill(255);
555 rect(blob3.x, blob3.y, blob3.r, 10);
556
557 // Draw the floor's highlights
558 fill(255);
559 rect(0, floorY3, width, 10);
560
561 // Draw the platforms' highlights
562 for (const p of platforms) {
563   rect(p.x, floorY3, p.w, 10);
564 }
565
566 // Draw the blob's shadow highlights
567 fill(255);
568 rect(blob3.x, blob3.y + 10, blob3.r, 10);
569
570 // Draw the floor's shadow highlights
571 fill(255);
572 rect(0, floorY3 + 10, width, 10);
573
574 // Draw the platforms' shadow highlights
575 for (const p of platforms) {
576   rect(p.x, floorY3 + 10, p.w, 10);
577 }
578
579 // Draw the blob's reflection highlights
580 fill(255);
581 rect(blob3.x, height - blob3.r, blob3.r, 10);
582
583 // Draw the floor's reflection highlights
584 fill(255);
585 rect(0, height - 10, width, 10);
586
587 // Draw the platforms' reflection highlights
588 for (const p of platforms) {
589   rect(p.x, height - 10, p.w, 10);
590 }
591
592 // Draw the blob's highlights
593 fill(255);
594 rect(blob3.x, blob3.y, blob3.r, 10);
595
596 // Draw the floor's highlights
597 fill(255);
598 rect(0, floorY3, width, 10);
599
600 // Draw the platforms' highlights
601 for (const p of platforms) {
602   rect(p.x, floorY3, p.w, 10);
603 }
604
605 // Draw the blob's shadow highlights
606 fill(255);
607 rect(blob3.x, blob3.y + 10, blob3.r, 10);
608
609 // Draw the floor's shadow highlights
610 fill(255);
611 rect(0, floorY3 + 10, width, 10);
612
613 // Draw the platforms' shadow highlights
614 for (const p of platforms) {
615   rect(p.x, floorY3 + 10, p.w, 10);
616 }
617
618 // Draw the blob's reflection highlights
619 fill(255);
620 rect(blob3.x, height - blob3.r, blob3.r, 10);
621
622 // Draw the floor's reflection highlights
623 fill(255);
624 rect(0, height - 10, width, 10);
625
626 // Draw the platforms' reflection highlights
627 for (const p of platforms) {
628   rect(p.x, height - 10, p.w, 10);
629 }
630
631 // Draw the blob's highlights
632 fill(255);
633 rect(blob3.x, blob3.y, blob3.r, 10);
634
635 // Draw the floor's highlights
636 fill(255);
637 rect(0, floorY3, width, 10);
638
639 // Draw the platforms' highlights
640 for (const p of platforms) {
641   rect(p.x, floorY3, p.w, 10);
642 }
643
644 // Draw the blob's shadow highlights
645 fill(255);
646 rect(blob3.x, blob3.y + 10, blob3.r, 10);
647
648 // Draw the floor's shadow highlights
649 fill(255);
650 rect(0, floorY3 + 10, width, 10);
651
652 // Draw the platforms' shadow highlights
653 for (const p of platforms) {
654   rect(p.x, floorY3 + 10, p.w, 10);
655 }
656
657 // Draw the blob's reflection highlights
658 fill(255);
659 rect(blob3.x, height - blob3.r, blob3.r, 10);
660
661 // Draw the floor's reflection highlights
662 fill(255);
663 rect(0, height - 10, width, 10);
664
665 // Draw the platforms' reflection highlights
666 for (const p of platforms) {
667   rect(p.x, height - 10, p.w, 10);
668 }
669
670 // Draw the blob's highlights
671 fill(255);
672 rect(blob3.x, blob3.y, blob3.r, 10);
673
674 // Draw the floor's highlights
675 fill(255);
676 rect(0, floorY3, width, 10);
677
678 // Draw the platforms' highlights
679 for (const p of platforms) {
680   rect(p.x, floorY3, p.w, 10);
681 }
682
683 // Draw the blob's shadow highlights
684 fill(255);
685 rect(blob3.x, blob3.y + 10, blob3.r, 10);
686
687 // Draw the floor's shadow highlights
688 fill(255);
689 rect(0, floorY3 + 10, width, 10);
690
691 // Draw the platforms' shadow highlights
692 for (const p of platforms) {
693   rect(p.x, floorY3 + 10, p.w, 10);
694 }
695
696 // Draw the blob's reflection highlights
697 fill(255);
698 rect(blob3.x, height - blob3.r, blob3.r, 10);
699
700 // Draw the floor's reflection highlights
701 fill(255);
702 rect(0, height - 10, width, 10);
703
704 // Draw the platforms' reflection highlights
705 for (const p of platforms) {
706   rect(p.x, height - 10, p.w, 10);
707 }
708
709 // Draw the blob's highlights
710 fill(255);
711 rect(blob3.x, blob3.y, blob3.r, 10);
712
713 // Draw the floor's highlights
714 fill(255);
715 rect(0, floorY3, width, 10);
716
717 // Draw the platforms' highlights
718 for (const p of platforms) {
719   rect(p.x, floorY3, p.w, 10);
720 }
721
722 // Draw the blob's shadow highlights
723 fill(255);
724 rect(blob3.x, blob3.y + 10, blob3.r, 10);
725
726 // Draw the floor's shadow highlights
727 fill(255);
728 rect(0, floorY3 + 10, width, 10);
729
730 // Draw the platforms' shadow highlights
731 for (const p of platforms) {
732   rect(p.x, floorY3 + 10, p.w, 10);
733 }
734
735 // Draw the blob's reflection highlights
736 fill(255);
737 rect(blob3.x, height - blob3.r, blob3.r, 10);
738
739 // Draw the floor's reflection highlights
740 fill(255);
741 rect(0, height - 10, width, 10);
742
743 // Draw the platforms' reflection highlights
744 for (const p of platforms) {
745   rect(p.x, height - 10, p.w, 10);
746 }
747
748 // Draw the blob's highlights
749 fill(255);
750 rect(blob3.x, blob3.y, blob3.r, 10);
751
752 // Draw the floor's highlights
753 fill(255);
754 rect(0, floorY3, width, 10);
755
756 // Draw the platforms' highlights
757 for (const p of platforms) {
758   rect(p.x, floorY3, p.w, 10);
759 }
760
761 // Draw the blob's shadow highlights
762 fill(255);
763 rect(blob3.x, blob3.y + 10, blob3.r, 10);
764
765 // Draw the floor's shadow highlights
766 fill(255);
767 rect(0, floorY3 + 10, width, 10);
768
769 // Draw the platforms' shadow highlights
770 for (const p of platforms) {
771   rect(p.x, floorY3 + 10, p.w, 10);
772 }
773
774 // Draw the blob's reflection highlights
775 fill(255);
776 rect(blob3.x, height - blob3.r, blob3.r, 10);
777
778 // Draw the floor's reflection highlights
779 fill(255);
780 rect(0, height - 10, width, 10);
781
782 // Draw the platforms' reflection highlights
783 for (const p of platforms) {
784   rect(p.x, height - 10, p.w, 10);
785 }
786
787 // Draw the blob's highlights
788 fill(255);
789 rect(blob3.x, blob3.y, blob3.r, 10);
790
791 // Draw the floor's highlights
792 fill(255);
793 rect(0, floorY3, width, 10);
794
795 // Draw the platforms' highlights
796 for (const p of platforms) {
797   rect(p.x, floorY3, p.w, 10);
798 }
799
800 // Draw the blob's shadow highlights
801 fill(255);
802 rect(blob3.x, blob3.y + 10, blob3.r, 10);
803
804 // Draw the floor's shadow highlights
805 fill(255);
806 rect(0, floorY3 + 10, width, 10);
807
808 // Draw the platforms' shadow highlights
809 for (const p of platforms) {
810   rect(p.x, floorY3 + 10, p.w, 10);
811 }
812
813 // Draw the blob's reflection highlights
814 fill(255);
815 rect(blob3.x, height - blob3.r, blob3.r, 10);
816
817 // Draw the floor's reflection highlights
818 fill(255);
819 rect(0, height - 10, width, 10);
820
821 // Draw the platforms' reflection highlights
822 for (const p of platforms) {
823   rect(p.x, height - 10, p.w, 10);
824 }
825
826 // Draw the blob's highlights
827 fill(255);
828 rect(blob3.x, blob3.y, blob3.r, 10);
829
830 // Draw the floor's highlights
831 fill(255);
832 rect(0, floorY3, width, 10);
833
834 // Draw the platforms' highlights
835 for (const p of platforms) {
836   rect(p.x, floorY3, p.w, 10);
837 }
838
839 // Draw the blob's shadow highlights
840 fill(255);
841 rect(blob3.x, blob3.y + 10, blob3.r, 10);
842
843 // Draw the floor's shadow highlights
844 fill(255);
845 rect(0, floorY3 + 10, width, 10);
846
847 // Draw the platforms' shadow highlights
848 for (const p of platforms) {
849   rect(p.x, floorY3 + 10, p.w, 10);
850 }
851
852 // Draw the blob's reflection highlights
853 fill(255);
854 rect(blob3.x, height - blob3.r, blob3.r, 10);
855
856 // Draw the floor's reflection highlights
857 fill(255);
858 rect(0, height - 10, width, 10);
859
860 // Draw the platforms' reflection highlights
861 for (const p of platforms) {
862   rect(p.x, height - 10, p.w, 10);
863 }
864
865 // Draw the blob's highlights
866 fill(255);
867 rect(blob3.x, blob3.y, blob3.r, 10);
868
869 // Draw the floor's highlights
870 fill(255);
871 rect(0, floorY3, width, 10);
872
873 // Draw the platforms' highlights
874 for (const p of platforms) {
875   rect(p.x, floorY3, p.w, 10);
876 }
877
878 // Draw the blob's shadow highlights
879 fill(255);
880 rect(blob3.x, blob3.y + 10, blob3.r, 10);
881
882 // Draw the floor's shadow highlights
883 fill(255);
884 rect(0, floorY3 + 10, width, 10);
885
886 // Draw the platforms' shadow highlights
887 for (const p of platforms) {
888   rect(p.x, floorY3 + 10, p.w, 10);
889 }
890
891 // Draw the blob's reflection highlights
892 fill(255);
893 rect(blob3.x, height - blob3.r, blob3.r, 10);
894
895 // Draw the floor's reflection highlights
896 fill(255);
897 rect(0, height - 10, width, 10);
898
899 // Draw the platforms' reflection highlights
900 for (const p of platforms) {
901   rect(p.x, height - 10, p.w, 10);
902 }
903
904 // Draw the blob's highlights
905 fill(255);
906 rect(blob3.x, blob3.y, blob3.r, 10);
907
908 // Draw the floor's highlights
909 fill(255);
910 rect(0, floorY3, width, 10);
911
912 // Draw the platforms' highlights
913 for (const p of platforms) {
914   rect(p.x, floorY3, p.w, 10);
915 }
916
917 // Draw the blob's shadow highlights
918 fill(255);
919 rect(blob3.x, blob3.y + 10, blob3.r, 10);
920
921 // Draw the floor's shadow highlights
922 fill(255);
923 rect(0, floorY3 + 10, width, 10);
924
925 // Draw the platforms' shadow highlights
926 for (const p of platforms) {
927   rect(p.x, floorY3 + 10, p.w, 10);
928 }
929
930 // Draw the blob's reflection highlights
931 fill(255);
932 rect(blob3.x, height - blob3.r, blob3.r, 10);
933
934 // Draw the floor's reflection highlights
935 fill(255);
936 rect(0, height - 10, width, 10);
937
938 // Draw the platforms' reflection highlights
939 for (const p of platforms) {
940   rect(p.x, height - 10, p.w, 10);
941 }
942
943 // Draw the blob's highlights
944 fill(255);
945 rect(blob3.x, blob3.y, blob3.r, 10);
946
947 // Draw the floor's highlights
948 fill(255);
949 rect(0, floorY3, width, 10);
950
951 // Draw the platforms' highlights
952 for (const p of platforms) {
953   rect(p.x, floorY3, p.w, 10);
954 }
955
956 // Draw the blob's shadow highlights
957 fill(255);
958 rect(blob3.x, blob3.y + 10, blob3.r, 10);
959
960 // Draw the floor's shadow highlights
961 fill(255);
962 rect(0, floorY3 + 10, width, 10);
963
964 // Draw the platforms' shadow highlights
965 for (const p of platforms) {
966   rect(p.x, floorY3 + 10, p.w, 10);
967 }
968
969 // Draw the blob's reflection highlights
970 fill(255);
971 rect(blob3.x, height - blob3.r, blob3.r, 10);
972
973 // Draw the floor's reflection highlights
974 fill(255);
975 rect(0, height - 10, width, 10);
976
977 // Draw the platforms' reflection highlights
978 for (const p of platforms) {
979   rect(p.x, height - 10, p.w, 10);
980 }
981
982 // Draw the blob's highlights
983 fill(255);
984 rect(blob3.x, blob3.y, blob3.r, 10);
985
986 // Draw the floor's highlights
987 fill(255);
988 rect(0, floorY3, width, 10);
989
990 // Draw the platforms' highlights
991 for (const p of platforms) {
992   rect(p.x, floorY3, p.w, 10);
993 }
994
995 // Draw the blob's shadow highlights
996 fill(255);
997 rect(blob3.x, blob3.y + 10, blob3.r, 10);
998
999 // Draw the floor's shadow highlights
1000 fill(255);
1001 rect(0, floorY3 + 10, width, 10);
1002
1003 // Draw the platforms' shadow highlights
1004 for (const p of platforms) {
1005   rect(p.x, floorY3 + 10, p.w, 10);
1006 }
1007
1008 // Draw the blob's reflection highlights
1009 fill(255);
1010 rect(blob3.x, height - blob3.r, blob3.r, 10);
1011
1012 // Draw the floor's reflection highlights
1013 fill(255);
1014 rect(0, height - 10, width, 10);
1015
1016 // Draw the platforms' reflection highlights
1017 for (const p of platforms) {
1018   rect(p.x, height - 10, p.w, 10);
1019 }
1020
1021 // Draw the blob's highlights
1022 fill(255);
1023 rect(blob3.x, blob3.y, blob3.r, 10);
1024
1025 // Draw the floor's highlights
1026 fill(255);
1027 rect(0, floorY3, width, 10);
1028
1029 // Draw the platforms' highlights
1030 for (const p of platforms) {
1031   rect(p.x, floorY3, p.w, 10);
1032 }
1033
1034 // Draw the blob's shadow highlights
1035 fill(255);
1036 rect(blob3.x, blob3.y + 10, blob3.r, 10);
1037
1038 // Draw the floor's shadow highlights
1039 fill(255);
1040 rect(0, floorY3 + 10, width, 10);
1041
1042 // Draw the platforms' shadow highlights
1043 for (const p of platforms) {
1044   rect(p.x, floorY3 + 10, p.w, 10);
1045 }
1046
1047 // Draw the blob's reflection highlights
1048 fill(255);
1049 rect(blob3.x, height - blob3.r, blob3.r, 10);
1050
1051 // Draw the floor's reflection highlights
1052 fill(255);
1053 rect(0, height - 10, width, 10);
1054
1055 // Draw the platforms' reflection highlights
1056 for (const p of platforms) {
1057   rect(p.x, height - 10, p.w, 10);
1058 }
1059
1060 // Draw the blob's highlights
1061 fill(255);
1062 rect(blob3.x, blob3.y, blob3.r, 10);
1063
1064 // Draw the floor's highlights
1065 fill(255);
1066 rect(0, floorY3, width, 10);
1067
1068 // Draw the platforms' highlights
1069 for (const p of platforms) {
1070   rect(p.x, floorY3, p.w, 10);
1071 }
1072
1073 // Draw the blob's shadow highlights
1074 fill(255);
1075 rect(blob3.x, blob3.y + 10, blob3.r, 10);
1076
1077 // Draw the floor's shadow highlights
1078 fill(255);
1079 rect(0, floorY3 + 10, width, 10);
1080
1081 // Draw the platforms' shadow highlights
1082 for (const p of platforms) {
1083   rect(p.x, floorY3 + 10, p.w, 10);
1084 }
1085
1086 // Draw the blob's reflection highlights
1087 fill(255);
1088 rect(blob3.x, height - blob3.r, blob3.r, 10);
1089
1090 // Draw the floor's reflection highlights
1091 fill(255);
1092 rect(0, height - 10, width, 10);
1093
1094 // Draw the platforms' reflection highlights
1095 for (const p of
```

The screenshot shows a code editor interface with the following details:

- Explorer:** Shows files in the project: sketch.js, README.md, JHENRI\_SIDEQUEST\_W2, libraries, .gitattributes, index.html, and jsonconfig.json.
- Editor:** The main area displays the `sketch.js` file with the following code snippet highlighted:
 

```

13 5 let blob3 = {
14 14   wobbleFreq: 0.9,
15 15   // Time values for breathing animation
16 16   t: 0,
17 17   tSpeed: 0.01,
18 18   // Physics: velocity
19 19   vx: 0, // Horizontal velocity
20 20   vy: 0, // Vertical velocity
21 21   // Movement tuning
22 22   accel: 0.55, // Horizontal acceleration
23 23   maxRun: 4.0, // Maximum horizontal speed
24 24   gravity: 0.65, // Downward force
25 25   jumpV: -11.0, // Initial jump impulse
26 26   // State
27 27   onGround: false, // True when standing on a platform
28 28   standingOnPlatform: null, // Reference to the platform currently being stood on
29 29   // Friction
30 30   frictionAir: 0.995, // Light friction in air
31 31   frictionGround: 0.88, // Stronger friction on ground
32 32 },
33 33   // List of solid platforms the blob can stand on
34 34   // Each platform is an axis-aligned rectangle (AABB)
35 35   platforms: []
36 36 };
37 37 }
38 38 // Define the floor height
39 39 floorY3 = height - 36;
40 40 // Create platforms (floor + steps)
41 41 let platforms = [
42 42   // Create platforms (floor + steps)
43 43   platforms = [
44 44     { x: 0, y: floorY3, w: width, h: height - floorY3 }, // floor
45 45     { x: 120, y: floorY3 - 70, w: 120, h: 12 }, // low step
46 46     { x: 300, y: floorY3 - 120, w: 90, h: 12 } // mid step
47 47 ];
48 48 noStroke();
49 49 textFont("sans-serif");
50 50 textSize(14);
51 51
52 52 // Create platforms (floor + steps)
53 53 platforms = [
54 54     { x: 0, y: floorY3, w: width, h: height - floorY3 }, // floor
55 55     { x: 120, y: floorY3 - 70, w: 120, h: 12 }, // low step
56 56     { x: 300, y: floorY3 - 120, w: 90, h: 12 } // mid step
57 57 ];
      
```
- Sidebar:** Shows a list of changes made to the code:
  1. Add a property to track which platform the blob is on
  2. Check which platform the blob is standing on during collision detection
  3. Draw platforms with different colors based on whether the blob is on them

Edited sketch.js +1 -0  
Edited sketch.js +6 -0  
Edited sketch.js +12 -3

Done! I've updated the code to make platforms turn yellow when the blob is actively standing on them. Here's what I changed:

  1. Added a tracking property (`blob3.standingOnPlatform`) to the blob to remember which platform it's currently standing on
  2. Updated the collision detection to set `blob3.standingOnPlatform` when the blob lands on a platform
  3. Modified the platform drawing to check if each platform is the one the blob is standing on, and if so, draw it in yellow instead of gray

Now when you run the sketch, any platform

#2: Can you make edits to the code to make the blob black if on the ground but if the blob is on a platform the blob colour is changed to blue

Now I'll add the blob color logic. I need to set the blob color before drawing it. I'll add code after the platforms are drawn to set the color based on whether the blob is on the floor or a platform, then modify the `drawBlobCircle` function to use the pre-set color.

Let me fix the JSON structure:

Edited

sketch.js+9-2

Done! I've updated the code so the blob will now:

- Turn **black** when standing on the floor (the main ground platform)
- Turn **blue** when standing on any of the elevated platforms

The color is set in the `draw()` function based on which platform the blob is currently standing on, and the `drawBlobCircle()` function uses that pre-set color.

```

66 function draw() {
67   background(240);
68
69   // --- Draw all platforms ---
70   fill(200);
71   for (const p of platforms) {
72     // Change the colour of the platform to yellow if the blob is standing on it, if the ledge is not being
73     if (p === blob3.standingOnPlatform) {
74       fill(255, 255, 0); // Yellow
75     } else {
76       fill(200); // Grey
77     }
78     rect(p.x, p.y, p.w, p.h);
79   }
80
81+ // Set blob color based on which platform it's standing on
82+ if (blob3.standingOnPlatform === platforms[0]) {
83+   // Standing on the floor (first platform)
84+   fill(0); // Black
85+ } else if (blob3.standingOnPlatform !== null) {
86+   // Standing on any other platform
87+   fill(random(255), random(255), random(255));
88+ }
89
90   // --- Input: left/right movement ---
91   let move = 0;
92
93   function draw() {
94     // --- Convert collision box back to blob centre ---
95     blob3.x = box.x + box.w / 2;
96     blob3.y = box.y + box.h / 2;
97
98     // Keep blob inside the canvas horizontally
99     blob3.x = constrain(blob3.x, blob3.r, width - blob3.r);
100
101    // --- Draw the animated blob ---
102    blob3.t += blob3.tSpeed;
103    drawBlobCircle(blob3);
104
105
106    // --- HUD ---
107    fill(0);
108    text("Move: A/D or ←→ • Jump: Space/W/r • Land on platforms", 10, 18);
109  }
110
111  // Axis-Aligned Bounding Box (AABB) overlap test
112  // Returns true if rectangles a and b intersect
113  function overlap(a, b) {
114    return (
115      a.x < b.x + b.w && a.x + a.w > b.x && a.y < b.y + b.h && a.y + a.h > b.y
116    );
117  }
118
119  // Draws the blob using Perlin noise for a soft, breathing effect
120  function drawBlobCircle(b) {
121    fill(20, 120, 255);
122    // Color is already set in the draw() function before calling this
123    beginShape();
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176

```

← CHANGING PLATFORM COLOR ... □  
✓ Read lines 1 to 200 of sketch.js

Now I'll add the blob color logic. I need to set the blob color before drawing it. I'll add code after the platforms are drawn to set the color based on whether the blob is on the floor or a platform, then modify the drawBlobCircle function to use the pre-set color.

Let me fix the JSON structure:

✓ Edited [sketch.js +10 -4](#)

Done! I've updated the code so the blob will now:

- Turn **black** when standing on the floor (the main ground platform)
- Turn **blue** when standing on any of the elevated platforms

The color is set in the `draw()` function based on which platform the blob is currently standing on, and the

← CHANGING PLATFORM COLOR ... □  
✓ Read lines 1 to 200 of sketch.js

Now I'll add the blob color logic. I need to set the blob color before drawing it. I'll add code after the platforms are drawn to set the color based on whether the blob is on the floor or a platform, then modify the drawBlobCircle function to use the pre-set color.

Let me fix the JSON structure:

✓ Edited [sketch.js +10 -4](#)

Done! I've updated the code so the blob will now:

- Turn **black** when standing on the floor (the main ground platform)
- Turn **blue** when standing on any of the elevated platforms

The color is set in the `draw()` function based on which platform the blob is currently standing on, and the `drawBlobCircle()` function uses that pre-set color.

#3 I've set the else/ if statement to be set to a random colour when the blob is on a ledge, however it is causing colour flickering. Could you edit the code so that when the blob lands on a platform it sticks to one random colour but if it jumps to another it switches to a different singular random colour

I can see the issue - random() is being called every frame in the draw loop. I need to:

1. Add a property to store the platform's color when the blob lands on it
2. Only generate a new random color when the blob lands on a *different* platform
3. Use that stored color instead of regenerating it each frame

Edited

sketch.js+7-2

Done! I've fixed the flickering by:

1. Added a **platformColor** property to the blob that stores the color for the current platform
2. Generate a new random color only once when the blob first lands on a platform (when **platformColor** is null)
3. Reuse that color every frame while on the same platform
4. Reset the color when the blob lands back on the floor

Now when the blob jumps to a different platform, it will get a new random color, but it will stay that color consistently while on the same platform without flickering.

The screenshot shows a code editor with a dark theme. On the left is the code, and on the right is a sidebar with notes and a preview area.

```
5 let blob3 = {
0   // Physics: velocity
1   vx: 0, // Horizontal velocity
2   vy: 0, // Vertical velocity
3
4   // Movement tuning
5   accel: 0.55, // Horizontal acceleration
6   maxRun: 4.0, // Maximum horizontal speed
7   gravity: 0.65, // Downward force
8   jumpV: -11.0, // Initial jump impulse
9
0   // State
1   onGround: false, // True when standing on a platform
2   standingOnPlatform: null, // Reference to the platform currently being stood on
3+   platformColor: null, // Stores the color for the current platform
4
5   // Friction
6   frictionAir: 0.995, // Light friction in air
7   frictionGround: 0.88, // Stronger friction on ground
8 };
9
0 // List of solid platforms the blob can stand on
1 // Each platform is an axis-aligned rectangle (AABB)
2 let platforms = [];
```

**← CHANGING PLATFORM**  
being called every frame loop. I need to:

1. Add a property to store the platform's color when the blob lands on it
2. Only generate a new random color when the blob lands on a **different** platform
3. Use that stored color, instead of regenerating it each frame

✓ Edited [sketch.js](#) +

Done! I've fixed the flickering.

1. Added a **platformColor** property to the blob that stores the color for the current platform
2. Generate a new random color only once when the blob lands on a platform (when **platformColor** is null)

```

66
67+ // Set blob color based on which platform it's standing on
68+ if (blob3.standingOnPlatform === platforms[0]) {
69+   // Standing on the floor (first platform)
70+   background(240); // Grey background
71+   fill(0); // Black
72+   blob3.platformColor = null;
73+ } else if (blob3.standingOnPlatform !== null) {
74+   // Generate a new random color only when landing on a new platform
75+   if (blob3.platformColor === null) {
76+     blob3.platformColor = color(random(255), random(255), random(255));
77+   }
78+   background(220, 240, 255); // Light blue background
79+   fill(blob3.platformColor);
80+
81+ }
82+
83+ }
84
85 // ---- Draw all platforms ----
86 fill(200);
87 for (const p of platforms) {
88   // Change the colour of the platform to yellow if the blob is standing on it, if the ledge is not being
89   if (p === blob3.standingOnPlatform) {
90     fill(255, 255, 0); // Yellow
91   } else {
92     fill(200); // Grey
93   }
94   rect(p.x, p.y, p.w, p.h);
95 }

-- // Set blob color based on which platform it's standing on
- if (blob3.standingOnPlatform === platforms[0]) {
-   // Standing on the floor (first platform)
-   fill(0); // Black
- } else if (blob3.standingOnPlatform !== null) {
-   // Standing on any other platform
-   fill(random(255), random(255), random(255));
- }

```

← CHANGING PLATFORM COLOR ...  
Being called every frame in the draw loop. I need to:

1. Add a property to store the platform's color when the blob lands on it
2. Only generate a new random color when the blob lands on a different platform
3. Use that stored color instead of regenerating it each frame

✓ Edited sketch.js +16 -9

Done! I've fixed the flickering by:

1. Added a `platformColor` property to the blob that stores the color for the current platform
2. Generate a new random color only once when the blob first lands on a platform (when `platformColor` is null)
3. Reuse that color every frame while on the same platform
4. Reset the color when the blob lands back on the floor

Now when the blob jumps to a different platform, it will get a new random color, but it will stay that color consistently while on the same.

sketch.js

Describe what to build next

## #4 Now that i've added some other changes the blob colour change isn't happening

I see the issue - the color logic is at the start of `draw()` before `blob3.standingOnPlatform` has been set by collision detection. I need to move it to after the collision detection. Let me read more to see where collision detection ends:

Read sketch.js, lines 140 to 175

Now I'll move the color-setting logic from the beginning of `draw()` to right before `drawBlobCircle()` is called, after collision detection:

Edited

sketch.js +20-15

Done! I've moved the blob color logic to after collision detection. The issue was that the code was checking `blob3.standingOnPlatform` at the beginning of the draw loop before it was actually set by the collision detection. Now it will:

- Turn **black** when standing on the floor
- Turn to a **random color** when landing on a platform
- Maintain that color while jumping

- Return to black when landing back on the floor

```
4 function setup() {
5
6 // Create platforms (floor + steps)
7 platforms = [
8   { x: 0, y: floorY3, w: width, h: height - floorY3 }, // floor
9   { x: 120, y: floorY3 - 70, w: 120, h: 12 }, // low step
10  { x: 300, y: floorY3 - 120, w: 90, h: 12 }, // mid step
11  { x: 440, y: floorY3 - 180, w: 130, h: 12 }, // high step
12  { x: 520, y: floorY3 - 70, w: 90, h: 12 }, // return ramp
13 ];
14
15 // Start the blob resting on the floor
16 blob3.y = floorY3 - blob3.r - 1;
17 }
18
19 function draw() {
20 | background(240);
21
22 // Set blob color based on which platform it's standing on
23 if (blob3.standingOnPlatform === platforms[0]) {
24   // Standing on the floor (first platform)
25   background(240); // Grey background
26   fill(0); // Black
27   blob3.platformColor = null;
28 } else if (blob3.standingOnPlatform !== null) {
29   // Generate a new random color only when landing on a new platform
30   if (blob3.platformColor === null) {
31     blob3.platformColor = color(random(255), random(255), random(255));
32   }
33   background(220, 240, 255); // Light blue background
34   fill(blob3.platformColor);
35 }
36
37 function draw() {
38 | }
39 }
40
41 // --- Convert collision box back to blob centre ---
42 blob3.x = box.x + box.w / 2;
43 blob3.y = box.y + box.h / 2;
44
45 // Keep blob inside the canvas horizontally
46 blob3.x = constrain(blob3.x, blob3.r, width - blob3.r);
47+
48+ // Set blob color based on which platform it's standing on (after collision detection)
49+ if (blob3.standingOnPlatform === platforms[0]) {
50+   // Standing on the floor (first platform)
51+   fill(0); // Black
52+   blob3.platformColor = null;
53+ } else if (blob3.standingOnPlatform !== null) {
54+   // Generate a new random color only when landing on a new platform
55+   if (blob3.platformColor === null) {
56+     blob3.platformColor = color(random(255), random(255), random(255));
57+   }
58+   fill(blob3.platformColor);
59+ } else {
60+   // In the air - use the last platform color or default
61+   if (blob3.platformColor !== null) {
62+     fill(blob3.platformColor);
63+   } else {
64+     fill(20, 120, 255); // Default blue
65+   }
66+ }
```

← CHANGING PLATFORM  
DETECTION. ELECTRIC ROAD  
where collision detection

✓ Read [sketch.js](#) ,  
Now I'll move the color-  
from the beginning of c  
before [drawBlobCir](#)  
after collision detection

✓ Edited `sketch.js`

Done! I've moved the blob3 object to after collision detection. The reason was that the code was checking if the blob3 object was standing on the ground at the beginning of the draw loop. However, the collision detection was actually set by the previous frame's draw loop. Now it will:

- Turn **black** when standing on the floor
  - Turn to a **random color** when landing on a platform
  - Maintain that color
  - Return to black when standing on the floor

Like

← CHANGING PLATFORM COLOR ...

- ✓ Read lines 1 to 100 of sketch.js

I see the issue - the color logic is at the start of draw() before `blob3.standingOnPlatform` has been set by collision detection. I need to move it to after the collision detection. Let me read more to see where collision detection ends:

✓ Read `sketch.js`, lines 140 to 175  
Now I'll move the color-setting logic from the beginning of `draw()` to right before `drawBlobCircle()` is called after collision detection:

✓ Edited [sketch.js +20 -15](#)

Done! I've moved the blob color logic to after collision detection. The issue was that the code was checking [blob3.standingOnPlatform](#) at the beginning of the draw loop before it was actually set by the collision detection. Now it will: