

Process & Decision Documentation

Side Quest 5 (Individual Work)

Project/Assignment Decisions

The purpose of Side Quest #5 was to have the player move camera through a storyline creating emotions and unveiling hidden object along the way.

GenAI Documentation

Date Used: February 21st,

Tool Disclosure: Claude Haiku 4.5 (extension in Visual Studios)

Purpose of Use: I used GenAI to help add complex objects to my background along with hidden object unveiled once that respective star was hit.

Summary of Interaction: As I provided the tool with guidelines to ensure the game meets Side Quest requirements, when things became complex various loops were added by GenAI.

Human Decision Point(s): As this week's content with very new to me, I wanted to understand the camera's functionality/constraints up to this point without any edits to the code. I recognized the square moves very effortlessly in all directions creating inspiration for me.

As Super Mario 64 is a very colourful game it kept my attention well, that is why I wanted the player to experience/immerse themselves into a star-filled sky. I thought having the camera going through a night of stars and random swirls would evoke a sense of escapism. The bonus seemed very fun. Therefore, I wanted the camera to unveil items of within the sky (the Moon, Venus, different colour clouds). I manually changed the hidden objects/ star covering them sizes as they weren't hidden enough the first time GenAI displayed them

Integrity & Verification Note: I made sure to understand the added code matched with this week's tutorial/example code I used. Created more prompt-based scenarios to evoke more emotion through design to meet the side quest requirements. As per usual ensured the tool provided comments and I only accepted code I understood and could change when needed.

Scope of GenAI Use: The tool helped bring a further complexity to my game (e.g unveiling objects in an interactive/clean manner) helping with my time management throughout this side quest. The tool also helped me understand the importance of direct and detailed prompting.

Limitations or Misfires: The tool did not understand the emotional aspect of the game the first time around, it created a bland example of a “star-filled night”, therefore this has taught me to create better prompts providing GenAI a scenario that a player would feel (e.g. emotions wise) and play (e.g. unveiling a sky object). Something that I assumed was language indication, GenAI did not have the same definition of “under” and object that I did, it first put the sky object underneath the star itself therefore the objects were already unveiled. Therefore, I learned from this and asked GenAI to put the objects “behind” solving those issues. Lastly, I assumed GenAI would understand the beginning point of the star however, I was required to ask it to unveil the object once the camera hit the middle rather than the side then the objects appeared.

Summary of Process (Human + Tool)

- I refined my prompts with hypothetical scenarios players could experience as they walk through my game
- Went in and manually fixed any code (e.g hidden objects/ star sizing's, the camera colour itself)

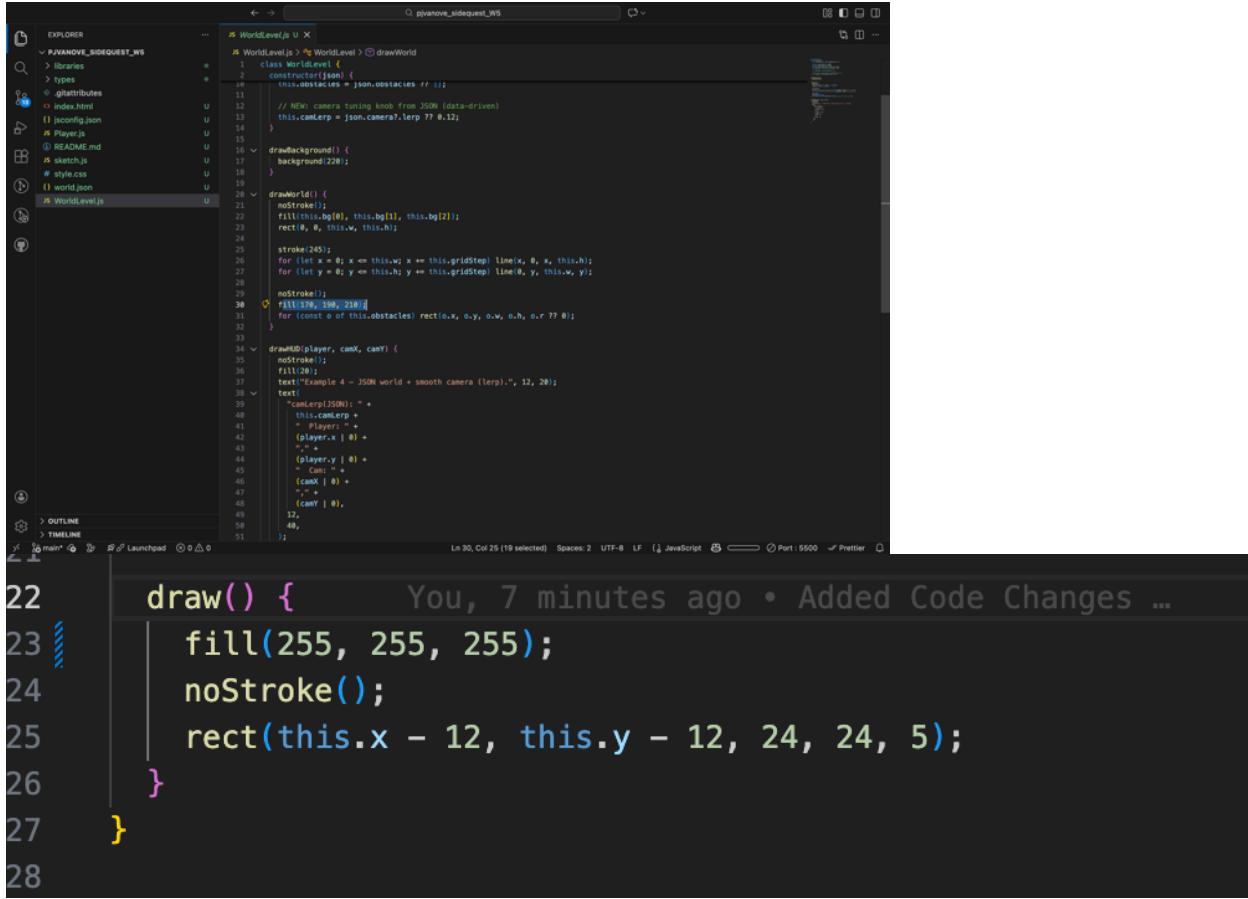
Decision Points & Trade-offs

I accepted the tools addition of the star and hidden objects, however I changed those game elements sizes as they were too large/close for the player to create a prolonged experience. I also realized I had to ask GenAI to make the star with the hidden objects immersive and once hit the object appears, I had the tool add that.

I felt it was important to complete the bonus, for example the game Skull played in class had many hidden objects within the game that kept my interest wanting me to uncover more.

Changing the camera colour to white

Examples original camera colour + my change to white



```

1  <--> pyanove_sidequest_WS
2  > libraries
3  > types
4  > gmattributes
5  > index.html
6  > player.json
7  > Player.js
8  README.md
9  sketch.js
10 # style.css
11 <!-- world.json -->
12 <!-- WorldLevel.js -->
13 <!-- drawWorld -->
14
15 constructor(json) {
16   this.obstacles = json.obstacles || [];
17
18   // NEW! camera tuning code from JSON (data-driven)
19   this.camLerp = json.camera7.lerp ?? 0.12;
20
21   drawBackground();
22   fill(this.bg[0], this.bg[1], this.bg[2]);
23   rect(0, 0, this.w, this.h);
24
25   stroke(20);
26   for (let x = 0; x <= this.w; x += this.gridStep) line(x, 0, x, this.h);
27   for (let y = 0; y <= this.h; y += this.gridStep) line(0, y, this.w, y);
28
29   noStroke();
30   fill(117, 194, 210);
31   for (const o of this.obstacles) rect(o.x, o.y, o.w, o.h, o.r ?? 0);
32
33
34   drawPlayer(camX, camY);
35   noStroke();
36   fill(20);
37   text("Example 4 - JSON world + smooth camera (lerp).", 12, 28);
38   text
39   `camera(${JSON.stringify(
40     this.camLerp,
41     -Player.x +,
42     (player.x | 0) +
43     ",",
44     (player.y | 0) +
45     "-Cam: " +
46     (camX | 0) +
47     ",",
48     (camY | 0),
49     12,
50     48,
51   )})`;

```

22 draw() { You, 7 minutes ago • Added Code Changes ...
23 fill(255, 255, 255);
24 noStroke();
25 rect(this.x - 12, this.y - 12, 24, 24, 5);
26 }
27 }
28

Prompt #2 (added in stars) I wanted the stars to be larger all together I made that change below

GenAI Output

```

19   generateStars(count) {
20     let stars = [];
21     for (let i = 0; i < count; i++) {
22       stars.push({
23         x: random(this.w),
24         y: random(this.h),
25         size: random(1, 3),
26         brightness: random(200, 255),
27         glowSize: random(8, 20),
28         twinkleSpeed: random(0.02, 0.08),
29         phase: random(TWO_PI)
30       });
31     }
32     return stars;
33   }

```

My changes

```
generateStars(count) {
  let stars = [];
  for (let i = 0; i < count; i++) {
    stars.push({
      x: random(this.w),
      y: random(this.h),
      size: random(2, 6),
      brightness: random(200, 255),
      glowSize: random(8, 20),
      twinkleSpeed: random(0.02, 0.08),
      phase: random(TWO_PI)
    });
  }
  return stars;
}
```

Prompt #4/5 (objects added + changing their sizes)

GenAI Output

```
// write (:variations)
positions.push({ x, y });
largeStars.push({
  x: x,
  y: y,
  size: random(12, 18),
  brightness: 255,
  glowSize: random(40, 60),
  twinkleSpeed: random(0.01, 0.04),
  phase: random(TWO_PI)
});
}
return largeStars;
}
```

My changes

```
    positions.push({ x, y });
    largeStars.push({
      x: x,
      y: y,
      size: random(12, 24),
      brightness: 255,
      glowSize: random(40, 60),
      twinkleSpeed: random(0.01, 0.04),
      phase: random(TWO_PI)
    });
  }
  return largeStars;
}
```

Verification & Judgement

Explain how you evaluated whether your decision or change was appropriate:

- Cross-references code structure from this week's tutorial example
- I manually tested in Visual Studio for errors/ debugging needs
- I confirmed misfires and manually changed object sizes for better functionality and visibility through the game
- I also added better scenario prompts versus just adding GenAI's first output given to me (e.g. Prompt #1 to Prompt #2)

Appendix

Transcript

Prompt #1 (removing original background + add new one)

I am creating an immersive experience for the player to go through a star-filled night. Please create a background with multiple stars + remove the large light blue square from canvas please.

```
JS WorldLevel.js X
JS WorldLevel.js > WorldLevel
class WorldLevel {
  constructor(props) {
    ...
    // Generate random stars for the night sky across entire world
    this.stars = this.generateStars(100);
  }

  generateStars(count) {
    let stars = [];
    for (let i = 0; i < count; i++) {
      stars.push({
        x: random(this.w),
        y: random(this.h),
        size: random(8, 5),
        opacity: random(100, 255)
      });
    }
    return stars;
  }

  drawBackground() {
    background(220);
  }

  drawWorld() {
    drawNightSkyBackground();
    noStroke();
    fill(this.bg[0], this.bg[1], this.bg[2]);
    fill(15, 20, 40);
    rect(0, 0, this.w, this.h);

    stroke(245);
    for (let x = 0; x <= this.w; x += this.gridStep) line(x, 0, x, this.h);
    for (let y = 0; y <= this.h; y += this.gridStep) line(0, y, this.w, y);

    noStroke();
    fill(170, 100, 210);
    for (const o of this.obstacles) rect(o.x, o.y, o.w, o.h, o.r ?? 0);

    // Draw stars across entire world
    for (const star of this.stars) {
      fill(255, 255, 255, star.opacity);
      circle(star.x, star.y, star.size);
    }
  }
}

I am creating an immersive experience for the player to go through a star-filled night. Please create a background with multiple stars + remove the large light blue square from canvas please. Ensure it goes through the whole canvas and the camera is still able to move through the canvas.

✓ Read JavaScript and JSON files
Now I'll update the files to create a star-filled night with the camera able to move through the world:
✓ Edited ▾ WorldLevel.js +10 -0
✓ Edited ▾ WorldLevel.js +7 -8
✓ Edited ▾ world.json +1 -1
Perfect! I've created an immersive star-filled night experience:
Changes made:
1. Dark night sky - Changed world background to deep blue-black (RGB: 15, 20, 40)
2. 200 stars - Generated randomly across the entire 2400x1600 world with varying sizes and twinkling opacity
3. Removed the light blue square -
```

Prompt #2 (refined background additions for better emotion and design contents)

Create a night full of stars (shape of stars) that feels like escapism for the player, they are leaving the environment they are currently in to experience a game that keeps them coming. Please ensure the sky is more of an indigo blue, with many bright twinkly lights with a glow around them. Add in swirls of yellow as well. Make sure the camera maintains the same speed/duration going through the canvas.

File: WorldLevel.js

```

1  class WorldLevel {
2    constructor(json) {
3      this.schemaVersion = json.schemaVersion ?? 1;
4
5      this.w = json.world?.w ?? 2400;
6      this.h = json.world?.h ?? 1600;
7      this.bg = json.world?.bg ?? [235, 235, 235];
8      this.gridStep = json.world?.gridStep ?? 100;
9
10     this.obstacles = json.obstacles ?? [];
11
12     // NEW: camera tuning knob from JSON (data-driven)
13     this.camType = json.camera?.type ?? "0.12";
14
15     // Generate random stars for the night sky across entire world
16     this.stars = this.generateStars(300);
17
18     // Generate yellow swirls for atmosphere
19     this.swirls = this.generateSwirls(50);
20+    this.swirls = this.generateSwirls(80);
21
22     // Generate 5 large prominent yellow stars
23     this.largeStars = this.generateLargeStars(5);
24
25     // Generate celestial items below each star
26     this.celestialItems = this.generateCelestialItems();
27
28     // Player reference for visibility checks
29     this.player = null;
30
31     // Camera position for reveal checks
32     this.cameraX = 0;
33     this.cameraY = 0;
34
35     // Track which stars have been revealed
36     this.revealedStars = new Set();
37
38     generateStars(count) {
39       let stars = [];
40       for (let i = 0; i < count; i++) {
41         stars.push({
42           x: random(this.w),
43           y: random(this.h),
44           size: random(2, 6),
45         });
46       }
47
48       return stars;
49     }
50
51     generateLargeStars() {
52       let stars = [];
53       for (let i = 0; i < 5; i++) {
54         stars.push({
55           x: random(this.w),
56           y: random(this.h),
57           size: random(20, 40),
58         });
59       }
60
61       return stars;
62     }
63
64     generateSwirls(count) {
65       let swirls = [];
66       for (let i = 0; i < count; i++) {
67         swirls.push({
68           x: random(this.w),
69           y: random(this.h),
70           radius: random(20, 80),
71           speed: random(0.1, 0.5),
72           tightness: random(0.5, 1.5),
73           opacity: random(0.1, 1.0),
74           rotation: random(-10, 10),
75         });
76       }
77
78       return swirls;
79     }
80
81     generateCelestialItems() {
82       let items = [];
83       for (let i = 0; i < 10; i++) {
84         items.push({
85           x: random(this.w),
86           y: random(this.h),
87           size: random(2, 6),
88           color: "#fff",
89         });
90       }
91
92       return items;
93     }
94
95     drawWorld() {
96       translate(this.w / 2, this.h / 2);
97
98       // Draw regular stars
99       for (const star of this.stars) {
100         // Calculate twinkling effect
101         let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
102         let currentOpacity = star.brightness * twinkle;
103
104         push();
105         translate(star.x, star.y);
106
107         // Draw glow halo
108         fill(180, 150, 255, 30 * twinkle);
109         noStroke();
110         circle(0, 0, star.glowSize);
111
112         // Draw bright star shape
113         fill(255, 255, 255, currentOpacity);
114         this.drawStar(0, 0, star.size * 1.5, star.size * 0.6, 5);
115
116         // Add a subtle inner glow
117         fill(255, 255, 255, 0.1);
118         stroke(0, 0, 0, 0.1);
119         circle(0, 0, star.size * 0.5);
120
121         pop();
122       }
123
124       // Draw regular twinkling stars
125       for (const star of this.stars) {
126         // Calculate twinkling effect
127         let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
128         let currentOpacity = star.brightness * twinkle;
129
130         push();
131         translate(star.x, star.y);
132
133         // Draw yellow swirls for atmosphere and escapism
134         for (const swirl of this.swirls) {
135           push();
136           translate(swirl.x, swirl.y);
137           rotate(frameCount * swirl.speed + swirl.rotation);
138
139           noFill();
140           stroke(255, 220, 100, swirl.opacity);
141           strokeWeight(1.5);
142
143           vertex(x, y);
144
145           endShape();
146
147           pop();
148
149         }
150
151         // Draw regular twinkling stars
152         for (const star of this.stars) {
153           // Calculate twinkling effect
154           let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
155           let currentOpacity = star.brightness * twinkle;
156
157           push();
158           translate(star.x, star.y);
159
160           // Draw glow halo
161           fill(180, 150, 255, 30 * twinkle);
162           noStroke();
163           circle(0, 0, star.glowSize);
164
165           // Draw bright star shape
166           fill(255, 255, 255, currentOpacity);
167           this.drawStar(0, 0, star.size * 1.5, star.size * 0.6, 5);
168
169           // Add a subtle inner glow
170           fill(255, 255, 255, 0.1);
171           stroke(0, 0, 0, 0.1);
172           circle(0, 0, star.size * 0.5);
173
174           pop();
175
176         }
177
178         // Draw regular twinkling stars
179         for (const star of this.stars) {
180           // Calculate twinkling effect
181           let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
182           let currentOpacity = star.brightness * twinkle;
183
184           push();
185           translate(star.x, star.y);
186
187           // Draw yellow swirls for atmosphere and escapism
188           for (const swirl of this.swirls) {
189             push();
190             translate(swirl.x, swirl.y);
191             rotate(frameCount * swirl.speed + swirl.rotation);
192
193             noFill();
194             stroke(255, 220, 100, swirl.opacity);
195             strokeWeight(1.5);
196
197             vertex(x, y);
198
199             endShape();
200             pop();
201
202           }
203
204           // Draw regular twinkling stars
205           for (const star of this.stars) {
206             // Calculate twinkling effect
207             let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
208             let currentOpacity = star.brightness * twinkle;
209
210             push();
211             translate(star.x, star.y);
212
213             // Draw glow halo
214             fill(180, 150, 255, 30 * twinkle);
215             noStroke();
216             circle(0, 0, star.glowSize);
217
218             // Draw bright star shape
219             fill(255, 255, 255, currentOpacity);
220             this.drawStar(0, 0, star.size * 1.5, star.size * 0.6, 5);
221
222             // Add a subtle inner glow
223             fill(255, 255, 255, 0.1);
224             stroke(0, 0, 0, 0.1);
225             circle(0, 0, star.size * 0.5);
226
227             pop();
228
229           }
230
231         }
232
233         // Draw regular twinkling stars
234         for (const star of this.stars) {
235           // Calculate twinkling effect
236           let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
237           let currentOpacity = star.brightness * twinkle;
238
239           push();
240           translate(star.x, star.y);
241
242           // Draw yellow swirls for atmosphere and escapism
243           for (const swirl of this.swirls) {
244             push();
245             translate(swirl.x, swirl.y);
246             rotate(frameCount * swirl.speed + swirl.rotation);
247
248             noFill();
249             stroke(255, 220, 100, swirl.opacity);
250             strokeWeight(1.5);
251
252             vertex(x, y);
253
254             endShape();
255             pop();
256
257           }
258
259           // Draw regular twinkling stars
260           for (const star of this.stars) {
261             // Calculate twinkling effect
262             let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
263             let currentOpacity = star.brightness * twinkle;
264
265             push();
266             translate(star.x, star.y);
267
268             // Draw glow halo
269             fill(180, 150, 255, 30 * twinkle);
270             noStroke();
271             circle(0, 0, star.glowSize);
272
273             // Draw bright star shape
274             fill(255, 255, 255, currentOpacity);
275             this.drawStar(0, 0, star.size * 1.5, star.size * 0.6, 5);
276
277             // Add a subtle inner glow
278             fill(255, 255, 255, 0.1);
279             stroke(0, 0, 0, 0.1);
280             circle(0, 0, star.size * 0.5);
281
282             pop();
283
284           }
285
286         }
287
288         // Draw regular twinkling stars
289         for (const star of this.stars) {
290           // Calculate twinkling effect
291           let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
292           let currentOpacity = star.brightness * twinkle;
293
294           push();
295           translate(star.x, star.y);
296
297           // Draw yellow swirls for atmosphere and escapism
298           for (const swirl of this.swirls) {
299             push();
300             translate(swirl.x, swirl.y);
301             rotate(frameCount * swirl.speed + swirl.rotation);
302
303             noFill();
304             stroke(255, 220, 100, swirl.opacity);
305             strokeWeight(1.5);
306
307             vertex(x, y);
308
309             endShape();
310             pop();
311
312           }
313
314           // Draw regular twinkling stars
315           for (const star of this.stars) {
316             // Calculate twinkling effect
317             let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
318             let currentOpacity = star.brightness * twinkle;
319
320             push();
321             translate(star.x, star.y);
322
323             // Draw glow halo
324             fill(180, 150, 255, 30 * twinkle);
325             noStroke();
326             circle(0, 0, star.glowSize);
327
328             // Draw bright star shape
329             fill(255, 255, 255, currentOpacity);
330             this.drawStar(0, 0, star.size * 1.5, star.size * 0.6, 5);
331
332             // Add a subtle inner glow
333             fill(255, 255, 255, 0.1);
334             stroke(0, 0, 0, 0.1);
335             circle(0, 0, star.size * 0.5);
336
337             pop();
338
339           }
340
341         }
342
343         // Draw regular twinkling stars
344         for (const star of this.stars) {
345           // Calculate twinkling effect
346           let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
347           let currentOpacity = star.brightness * twinkle;
348
349           push();
350           translate(star.x, star.y);
351
352           // Draw yellow swirls for atmosphere and escapism
353           for (const swirl of this.swirls) {
354             push();
355             translate(swirl.x, swirl.y);
356             rotate(frameCount * swirl.speed + swirl.rotation);
357
358             noFill();
359             stroke(255, 220, 100, swirl.opacity);
360             strokeWeight(1.5);
361
362             vertex(x, y);
363
364             endShape();
365             pop();
366
367           }
368
369           // Draw regular twinkling stars
370           for (const star of this.stars) {
371             // Calculate twinkling effect
372             let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
373             let currentOpacity = star.brightness * twinkle;
374
375             push();
376             translate(star.x, star.y);
377
378             // Draw glow halo
379             fill(180, 150, 255, 30 * twinkle);
380             noStroke();
381             circle(0, 0, star.glowSize);
382
383             // Draw bright star shape
384             fill(255, 255, 255, currentOpacity);
385             this.drawStar(0, 0, star.size * 1.5, star.size * 0.6, 5);
386
387             // Add a subtle inner glow
388             fill(255, 255, 255, 0.1);
389             stroke(0, 0, 0, 0.1);
390             circle(0, 0, star.size * 0.5);
391
392             pop();
393
394           }
395
396         }
397
398         // Draw regular twinkling stars
399         for (const star of this.stars) {
400           // Calculate twinkling effect
401           let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
402           let currentOpacity = star.brightness * twinkle;
403
404           push();
405           translate(star.x, star.y);
406
407           // Draw yellow swirls for atmosphere and escapism
408           for (const swirl of this.swirls) {
409             push();
410             translate(swirl.x, swirl.y);
411             rotate(frameCount * swirl.speed + swirl.rotation);
412
413             noFill();
414             stroke(255, 220, 100, swirl.opacity);
415             strokeWeight(1.5);
416
417             vertex(x, y);
418
419             endShape();
420             pop();
421
422           }
423
424           // Draw regular twinkling stars
425           for (const star of this.stars) {
426             // Calculate twinkling effect
427             let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
428             let currentOpacity = star.brightness * twinkle;
429
430             push();
431             translate(star.x, star.y);
432
433             // Draw glow halo
434             fill(180, 150, 255, 30 * twinkle);
435             noStroke();
436             circle(0, 0, star.glowSize);
437
438             // Draw bright star shape
439             fill(255, 255, 255, currentOpacity);
440             this.drawStar(0, 0, star.size * 1.5, star.size * 0.6, 5);
441
442             // Add a subtle inner glow
443             fill(255, 255, 255, 0.1);
444             stroke(0, 0, 0, 0.1);
445             circle(0, 0, star.size * 0.5);
446
447             pop();
448
449           }
450
451         }
452
453         // Draw regular twinkling stars
454         for (const star of this.stars) {
455           // Calculate twinkling effect
456           let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
457           let currentOpacity = star.brightness * twinkle;
458
459           push();
460           translate(star.x, star.y);
461
462           // Draw yellow swirls for atmosphere and escapism
463           for (const swirl of this.swirls) {
464             push();
465             translate(swirl.x, swirl.y);
466             rotate(frameCount * swirl.speed + swirl.rotation);
467
468             noFill();
469             stroke(255, 220, 100, swirl.opacity);
470             strokeWeight(1.5);
471
472             vertex(x, y);
473
474             endShape();
475             pop();
476
477           }
478
479           // Draw regular twinkling stars
480           for (const star of this.stars) {
481             // Calculate twinkling effect
482             let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
483             let currentOpacity = star.brightness * twinkle;
484
485             push();
486             translate(star.x, star.y);
487
488             // Draw glow halo
489             fill(180, 150, 255, 30 * twinkle);
490             noStroke();
491             circle(0, 0, star.glowSize);
492
493             // Draw bright star shape
494             fill(255, 255, 255, currentOpacity);
495             this.drawStar(0, 0, star.size * 1.5, star.size * 0.6, 5);
496
497             // Add a subtle inner glow
498             fill(255, 255, 255, 0.1);
499             stroke(0, 0, 0, 0.1);
500             circle(0, 0, star.size * 0.5);
501
502             pop();
503
504           }
505
506         }
507
508         // Draw regular twinkling stars
509         for (const star of this.stars) {
510           // Calculate twinkling effect
511           let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
512           let currentOpacity = star.brightness * twinkle;
513
514           push();
515           translate(star.x, star.y);
516
517           // Draw yellow swirls for atmosphere and escapism
518           for (const swirl of this.swirls) {
519             push();
520             translate(swirl.x, swirl.y);
521             rotate(frameCount * swirl.speed + swirl.rotation);
522
523             noFill();
524             stroke(255, 220, 100, swirl.opacity);
525             strokeWeight(1.5);
526
527             vertex(x, y);
528
529             endShape();
530             pop();
531
532           }
533
534           // Draw regular twinkling stars
535           for (const star of this.stars) {
536             // Calculate twinkling effect
537             let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
538             let currentOpacity = star.brightness * twinkle;
539
540             push();
541             translate(star.x, star.y);
542
543             // Draw glow halo
544             fill(180, 150, 255, 30 * twinkle);
545             noStroke();
546             circle(0, 0, star.glowSize);
547
548             // Draw bright star shape
549             fill(255, 255, 255, currentOpacity);
550             this.drawStar(0, 0, star.size * 1.5, star.size * 0.6, 5);
551
552             // Add a subtle inner glow
553             fill(255, 255, 255, 0.1);
554             stroke(0, 0, 0, 0.1);
555             circle(0, 0, star.size * 0.5);
556
557             pop();
558
559           }
560
561         }
562
563         // Draw regular twinkling stars
564         for (const star of this.stars) {
565           // Calculate twinkling effect
566           let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
567           let currentOpacity = star.brightness * twinkle;
568
569           push();
570           translate(star.x, star.y);
571
572           // Draw yellow swirls for atmosphere and escapism
573           for (const swirl of this.swirls) {
574             push();
575             translate(swirl.x, swirl.y);
576             rotate(frameCount * swirl.speed + swirl.rotation);
577
578             noFill();
579             stroke(255, 220, 100, swirl.opacity);
580             strokeWeight(1.5);
581
582             vertex(x, y);
583
584             endShape();
585             pop();
586
587           }
588
589           // Draw regular twinkling stars
590           for (const star of this.stars) {
591             // Calculate twinkling effect
592             let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
593             let currentOpacity = star.brightness * twinkle;
594
595             push();
596             translate(star.x, star.y);
597
598             // Draw glow halo
599             fill(180, 150, 255, 30 * twinkle);
600             noStroke();
601             circle(0, 0, star.glowSize);
602
603             // Draw bright star shape
604             fill(255, 255, 255, currentOpacity);
605             this.drawStar(0, 0, star.size * 1.5, star.size * 0.6, 5);
606
607             // Add a subtle inner glow
608             fill(255, 255, 255, 0.1);
609             stroke(0, 0, 0, 0.1);
610             circle(0, 0, star.size * 0.5);
611
612             pop();
613
614           }
615
616         }
617
618         // Draw regular twinkling stars
619         for (const star of this.stars) {
620           // Calculate twinkling effect
621           let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
622           let currentOpacity = star.brightness * twinkle;
623
624           push();
625           translate(star.x, star.y);
626
627           // Draw yellow swirls for atmosphere and escapism
628           for (const swirl of this.swirls) {
629             push();
630             translate(swirl.x, swirl.y);
631             rotate(frameCount * swirl.speed + swirl.rotation);
632
633             noFill();
634             stroke(255, 220, 100, swirl.opacity);
635             strokeWeight(1.5);
636
637             vertex(x, y);
638
639             endShape();
640             pop();
641
642           }
643
644           // Draw regular twinkling stars
645           for (const star of this.stars) {
646             // Calculate twinkling effect
647             let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
648             let currentOpacity = star.brightness * twinkle;
649
650             push();
651             translate(star.x, star.y);
652
653             // Draw glow halo
654             fill(180, 150, 255, 30 * twinkle);
655             noStroke();
656             circle(0, 0, star.glowSize);
657
658             // Draw bright star shape
659             fill(255, 255, 255, currentOpacity);
660             this.drawStar(0, 0, star.size * 1.5, star.size * 0.6, 5);
661
662             // Add a subtle inner glow
663             fill(255, 255, 255, 0.1);
664             stroke(0, 0, 0, 0.1);
665             circle(0, 0, star.size * 0.5);
666
667             pop();
668
669           }
670
671         }
672
673         // Draw regular twinkling stars
674         for (const star of this.stars) {
675           // Calculate twinkling effect
676           let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
677           let currentOpacity = star.brightness * twinkle;
678
679           push();
680           translate(star.x, star.y);
681
682           // Draw yellow swirls for atmosphere and escapism
683           for (const swirl of this.swirls) {
684             push();
685             translate(swirl.x, swirl.y);
686             rotate(frameCount * swirl.speed + swirl.rotation);
687
688             noFill();
689             stroke(255, 220, 100, swirl.opacity);
690             strokeWeight(1.5);
691
692             vertex(x, y);
693
694             endShape();
695             pop();
696
697           }
698
699           // Draw regular twinkling stars
700           for (const star of this.stars) {
701             // Calculate twinkling effect
702             let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
703             let currentOpacity = star.brightness * twinkle;
704
705             push();
706             translate(star.x, star.y);
707
708             // Draw glow halo
709             fill(180, 150, 255, 30 * twinkle);
710             noStroke();
711             circle(0, 0, star.glowSize);
712
713             // Draw bright star shape
714             fill(255, 255, 255, currentOpacity);
715             this.drawStar(0, 0, star.size * 1.5, star.size * 0.6, 5);
716
717             // Add a subtle inner glow
718             fill(255, 255, 255, 0.1);
719             stroke(0, 0, 0, 0.1);
720             circle(0, 0, star.size * 0.5);
721
722             pop();
723
724           }
725
726         }
727
728         // Draw regular twinkling stars
729         for (const star of this.stars) {
730           // Calculate twinkling effect
731           let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
732           let currentOpacity = star.brightness * twinkle;
733
734           push();
735           translate(star.x, star.y);
736
737           // Draw yellow swirls for atmosphere and escapism
738           for (const swirl of this.swirls) {
739             push();
740             translate(swirl.x, swirl.y);
741             rotate(frameCount * swirl.speed + swirl.rotation);
742
743             noFill();
744             stroke(255, 220, 100, swirl.opacity);
745             strokeWeight(1.5);
746
747             vertex(x, y);
748
749             endShape();
750             pop();
751
752           }
753
754           // Draw regular twinkling stars
755           for (const star of this.stars) {
756             // Calculate twinkling effect
757             let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
758             let currentOpacity = star.brightness * twinkle;
759
760             push();
761             translate(star.x, star.y);
762
763             // Draw glow halo
764             fill(180, 150, 255, 30 * twinkle);
765             noStroke();
766             circle(0, 0, star.glowSize);
767
768             // Draw bright star shape
769             fill(255, 255, 255, currentOpacity);
770             this.drawStar(0, 0, star.size * 1.5, star.size * 0.6, 5);
771
772             // Add a subtle inner glow
773             fill(255, 255, 255, 0.1);
774             stroke(0, 0, 0, 0.1);
775             circle(0, 0, star.size * 0.5);
776
777             pop();
778
779           }
780
781         }
782
783         // Draw regular twinkling stars
784         for (const star of this.stars) {
785           // Calculate twinkling effect
786           let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
787           let currentOpacity = star.brightness * twinkle;
788
789           push();
790           translate(star.x, star.y);
791
792           // Draw yellow swirls for atmosphere and escapism
793           for (const swirl of this.swirls) {
794             push();
795             translate(swirl.x, swirl.y);
796             rotate(frameCount * swirl.speed + swirl.rotation);
797
798             noFill();
799             stroke(255, 220, 100, swirl.opacity);
800             strokeWeight(1.5);
801
802             vertex(x, y);
803
804             endShape();
805             pop();
806
807           }
808
809           // Draw regular twinkling stars
810           for (const star of this.stars) {
811             // Calculate twinkling effect
812             let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
813             let currentOpacity = star.brightness * twinkle;
814
815             push();
816             translate(star.x, star.y);
817
818             // Draw glow halo
819             fill(180, 150, 255, 30 * twinkle);
820             noStroke();
821             circle(0, 0, star.glowSize);
822
823             // Draw bright star shape
824             fill(255, 255, 255, currentOpacity);
825             this.drawStar(0, 0, star.size * 1.5, star.size * 0.6, 5);
826
827             // Add a subtle inner glow
828             fill(255, 255, 255, 0.1);
829             stroke(0, 0, 0, 0.1);
830             circle(0, 0, star.size * 0.5);
831
832             pop();
833
834           }
835
836         }
837
838         // Draw regular twinkling stars
839         for (const star of this.stars) {
840           // Calculate twinkling effect
841           let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
842           let currentOpacity = star.brightness * twinkle;
843
844           push();
845           translate(star.x, star.y);
846
847           // Draw yellow swirls for atmosphere and escapism
848           for (const swirl of this.swirls) {
849             push();
850             translate(swirl.x, swirl.y);
851             rotate(frameCount * swirl.speed + swirl.rotation);
852
853             noFill();
854             stroke(255, 220, 100, swirl.opacity);
855             strokeWeight(1.5);
856
857             vertex(x, y);
858
859             endShape();
860             pop();
861
862           }
863
864           // Draw regular twinkling stars
865           for (const star of this.stars) {
866             // Calculate twinkling effect
867             let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
868             let currentOpacity = star.brightness * twinkle;
869
870             push();
871             translate(star.x, star.y);
872
873             // Draw glow halo
874             fill(180, 150, 255, 30 * twinkle);
875             noStroke();
876             circle(0, 0, star.glowSize);
877
878             // Draw bright star shape
879             fill(255, 255, 255, currentOpacity);
880             this.drawStar(0, 0, star.size * 1.5, star.size * 0.6, 5);
881
882             // Add a subtle inner glow
883             fill(255, 255, 255, 0.1);
884             stroke(0, 0, 0, 0.1);
885             circle(0, 0, star.size * 0.5);
886
887             pop();
888
889           }
890
891         }
892
893         // Draw regular twinkling stars
894         for (const star of this.stars) {
895           // Calculate twinkling effect
896           let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
897           let currentOpacity = star.brightness * twinkle;
898
899           push();
900           translate(star.x, star.y);
901
902           // Draw yellow swirls for atmosphere and escapism
903           for (const swirl of this.swirls) {
904             push();
905             translate(swirl.x, swirl.y);
906             rotate(frameCount * swirl.speed + swirl.rotation);
907
908             noFill();
909             stroke(255, 220, 100, swirl.opacity);
910             strokeWeight(1.5);
911
912             vertex(x, y);
913
914             endShape();
915             pop();
916
917           }
918
919           // Draw regular twinkling stars
920           for (const star of this.stars) {
921             // Calculate twinkling effect
922             let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
923             let currentOpacity = star.brightness * twinkle;
924
925             push();
926             translate(star.x, star.y);
927
928             // Draw glow halo
929             fill(180, 150, 255, 30 * twinkle);
930             noStroke();
931             circle(0, 0, star.glowSize);
932
933             // Draw bright star shape
934             fill(255, 255, 255, currentOpacity);
935             this.drawStar(0, 0, star.size * 1.5, star.size * 0.6, 5);
936
937             // Add a subtle inner glow
938             fill(255, 255, 255, 0.1);
939             stroke(0, 0, 0, 0.1);
940             circle(0, 0, star.size * 0.5);
941
942             pop();
943
944           }
945
946         }
947
948         // Draw regular twinkling stars
949         for (const star of this.stars) {
950           // Calculate twinkling effect
951           let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
952           let currentOpacity = star.brightness * twinkle;
953
954           push();
955           translate(star.x, star.y);
956
957           // Draw yellow swirls for atmosphere and escapism
958           for (const swirl of this.swirls) {
959             push();
960             translate(swirl.x, swirl.y);
961             rotate(frameCount * swirl.speed + swirl.rotation);
962
963             noFill();
964             stroke(255, 220, 100, swirl.opacity);
965             strokeWeight(1.5);
966
967             vertex(x, y);
968
969             endShape();
970             pop();
971
972           }
973
974           // Draw regular twinkling stars
975           for (const star of this.stars) {
976             // Calculate twinkling effect
977             let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
978             let currentOpacity = star.brightness * twinkle;
979
980             push();
981             translate(star.x, star.y);
982
983             // Draw glow halo
984             fill(180, 150, 255, 30 * twinkle);
985             noStroke();
986             circle(0, 0, star.glowSize);
987
988             // Draw bright star shape
989             fill(255, 255, 255, currentOpacity);
990             this.drawStar(0, 0, star.size * 1.5, star.size * 0.6, 5);
991
992             // Add a subtle inner glow
993             fill(255, 255, 255, 0.1);
994             stroke(0, 0, 0, 0.1);
995             circle(0, 0, star.size * 0.5);
996
997             pop();
998
999           }
1000
1001         }
1002
1003         // Draw regular twinkling stars
1004         for (const star of this.stars) {
1005           // Calculate twinkling effect
1006           let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
1007           let currentOpacity = star.brightness * twinkle;
1008
1009           push();
1010           translate(star.x, star.y);
1011
1012           // Draw yellow swirls for atmosphere and escapism
1013           for (const swirl of this.swirls) {
1014             push();
1015             translate(swirl.x, swirl.y);
1016             rotate(frameCount * swirl.speed + swirl.rotation);
1017
1018             noFill();
1019             stroke(255, 220, 100, swirl.opacity);
1020             strokeWeight(1.5);
1021
1022             vertex(x, y);
1023
1024             endShape();
1025             pop();
1026
1027           }
1028
1029           // Draw regular twinkling stars
1030           for (const star of this.stars) {
1031             // Calculate twinkling effect
1032             let twinkle = sin(frameCount * star.twinkle) * 0.5 + 0.5;
1033             let currentOpacity = star.brightness * twinkle;
1034
1035             push();
1036             translate(star.x, star.y);
1037
1038             // Draw glow halo
1039             fill(180, 150, 255, 30 * twinkle);
1040             noStroke();
1041             circle(0, 0, star.glowSize);
1042
1043             // Draw bright star shape
1044             fill(255, 255, 255, currentOpacity);
1045             this.drawStar(0, 0, star.size * 1.5, star.size * 0.6, 5);
1046
1047             // Add a subtle inner glow
1048             fill(255, 255, 255, 0.1);
1049             stroke(0, 0, 0, 0.1);
1050             circle(0, 0, star.size * 0.5);
1051
1052             pop();
1053
1054           }
1055
1056         }
1057
1058         // Draw regular twinkling stars
10
```

```

44  class WorldLevel {
45    generateStars(count) {
46      let stars = [];
47      const gSize = random(8, 20);
48      const twinkleSpeed = random(0.02, 0.08),
49            phase = random(TWO_PI);
50    }
51    return stars;
52  }
53
54  generateSwirls(count) {
55    let swirls = [];
56    for (let i = 0; i < count; i++) {
57      swirls.push({
58        x: random(this.w),
59        y: random(this.h),
60        radius: random(20, 60),
61        opacity: random(0.05, 0.15),
62        speed: random(0.005, 0.015),
63        rotation: random(TWO_PI),
64        radius: random(8, 20),
65        opacity: random(40, 100),
66        speed: random(0.01, 0.025),
67        rotation: random(TWO_PI),
68        tightness: random(8, 14)
69      });
70    }
71    return swirls;
72  }
73
74  generateLargeStars(count) {
75    let largeStars = [];
76    let positions = [];
77
78    // Generate vertically spread large stars positioned close together horizontally
79    for (let i = 0; i < count; i++) {
80      let x, y, validPosition;
81
82      do {
83        // Keep X values close together (near the center horizontally)
84        x = random(this.w * 0.4, this.w * 0.6);
85        // Spread Y values across the full canvas vertically
86        y = random(200, this.h - 200);
87        validPosition = true;
88
89        // Check distance from other large stars
90      } while (!validPosition);
91
92      largeStars.push({
93        x,
94        y
95      });
96    }
97
98    return largeStars;
99  }

```

CHAT

LOCATING GAME BACKGROUND CODE...

Create a night full of stars (shape of star) that feels like escapism for the player, they are leaving the environment they are currently in to experience a different one. The stars should be sparse ensure the sky is more of an indigo/blue, with many bright twinkly lights with a glow around them. Add in tight swirls of yellow as well. Make sure the camera maintains the same speed/duration going through the canvases.

WorldLevel.js79

- ✓ Adjusted swirl generation for tighter swirls.
- ✓ Edited [WorldLevel.js](#) v5 → 4
- ✓ Edited [WorldLevel.js](#) v5 → 5
- ✓ Edited [WorldLevel.js](#) v1 → 1

Perfect! I've enhanced the escapist experience with tight yellow swirls:

• Swirl Improvements:

1. Tighter spirals - Reduced radius from 20-60 to 8-20 pixels for compact, intricate patterns
2. More swirls - Increased from 50 to 80 for denser, more immersive canvases

> 1 file changed +41 -0 [Keep](#) [Undo](#)

js WorldLevel.js +

Describe what to build next

Agent Claude Heiku 4.5 v

Prompt #3

Please create five large new stars dispersed throughout the canvas ensure they are a brighter yellow colour and make sure the stars already in the game star as well. Make the stars near each other but still an immersive and somewhat long experience.

```

// Draw stars with glowing halos across entire world
// Draw regular twinkling stars
for (let i = 0; i < count; i++) {
    // calculate twinkling effect
    let twinkle = sin(frameCount * star.twinkleSpeed + star.phase) * 0.5 + 0.5;
    let currentOpacity = star.brightness * twinkle;

    push();
    translate(star.x, star.y);

    // Draw glow halo
    fill(190, 150, 250, 30 * twinkle);
    noStroke();
    circle(0, 0, star.glowSize);

    // Draw bright star shape
    fill(255, 255, 255, currentOpacity);
    this.drawStar(0, 0, star.size * 1.5, star.size * 0.6, 5);

    // Add a subtle inner glow
    fill(200, 220, 255, 100 * twinkle);
    this.drawStar(0, 0, star.size * 0.8, star.size * 0.3, 5);

    pop();
}

// Draw large yellow stars
for (const star of this.largestars) {
    // calculate twinkling effect
    let twinkle = sin(frameCount * star.twinkleSpeed + star.phase) * 0.5 + 0.5;
    let currentOpacity = star.brightness * twinkle;

    push();
    translate(star.x, star.y);

    // Draw golden yellow glow halo
    fill(255, 200, 100, 40 * twinkle);
    noStroke();
    circle(0, 0, star.glowSize);
}

```

PJANNOVE_SIDEQUEST_W6

```

// Draw stars with glowing halos across entire world
// Draw regular twinkling stars
for (let i = 0; i < count; i++) {
    // generatestars(count) {
        generateLargeStars(count) {
            let largestars = [];
            let positions = [];

            // Generate vertically spread large stars positioned close together horizontally
            for (let i = 0; i < count; i++) {
                let x, y, validPosition;
                do {
                    // Keep X values close together (near the center horizontally)
                    x = random(this.w * 0.4, this.w * 0.6);
                    // Randomly rotate the entire full canvas vertically
                    y = random(200, this.h - 200);
                    validPosition = true;
                } while (!validPosition);

                // Check distance from other large stars
                for (let pos of positions) {
                    if (dist(x, y, pos.x, pos.y) < 300) {
                        validPosition = false;
                        break;
                    }
                }
            }
            while (!validPosition);
            positions.push({x, y});
            largestars.push({
                x: x,
                y: y,
                size: random(12, 18),
                brightness: 255,
                glowSize: random(40, 60),
                twinkleSpeed: random(0.01, 0.04),
                phase: random(TWO_PI)
            });
        }
        return largestars;
    }

    drawBackground() {
        background(220);
    }
}

```

Prompt #4

Using the five large stars please ensure one item is under each star in the order: the Moon with an off-white crescent shape, a soft purple cloud in a swirl shape, Venus which is a bright pale-yellow circle, a faint orange hazy cloud, and Neptune which has multiple blues also a circular shape.

File: WorldLevel.js

```

1 1 class WorldLevel {
2 2     constructor(json) {
3 3         this.schemaVersion = json.schemaVersion ?? 1;
4 4         this.w = json.world1.w ?? 2400;
5 5         this.h = json.world1.h ?? 1600;
6 6         this.bg = json.world7.bg ?? [235, 235, 235];
7 7         this.gridStep = json.world7.gridStep ?? 160;
8 8         this.obstacles = json.obstacles ?? [];
9 9
10 10 // NEW: camera tuning knob from JSON (data-driven)
11 11 this.cameraP = json.camera7.terp ?? 0.12;
12 12
13 13 // Generate random stars for the night sky across entire world
14 14 this.stars = this.generateStars(300);
15 15
16 16 // Generate 5 large prominent yellow stars
17 17 this.largeStars = this.generateLargeStars(5);
18 18
19 19 // Generate celestial items below each star
20 20 this.celestialItems = this.generateCelestialItems();
21 21
22 22 generateStars(count) {
23 23     let stars = [];
24 24     for (let i = 0; i < count; i++) {
25 25         stars.push({
26 26             x: random(this.w),
27 27             y: random(this.h),
28 28             size: random(2, 6),
29 29             brightness: random(200, 255),
30 30             glowSize: random(0, 20),
31 31             twinkleSpeed: random(0.02, 0.08),
32 32             phase: random(TWO_PI)
33 33         });
34 34     }
35 35     return stars;
36 36 }
37 37
38 38 generateLargeStars(count) {
39 39     let largeStars = [];
40 40     let positions = [];
41 41
42 42     // Generate vertically spread large stars positioned close together horizontally
43 43
44 44 }
45 45

```

File: WorldLevel.js

CHAT

LOCATING GAME BACKGROUND CODE...
large yellow stars vertically aligned through the center of the canvas.

Using the five large stars please ensure one item is under each star in the order: the Moon with an off-white crescent shape, a soft purple cloud in a swirl shape, Venus which is a bright pale yellow circle, a faint orange hazy cloud, and Neptune which has multiple blues also a circular shape.

- Added celestial items back to WorldLevel.js
- Edited ▲ WorldLevel.js +3 -0
- Edited ▲ WorldLevel.js +12 -1
- Edited ▲ WorldLevel.js +21 -0
- Edited ▲ WorldLevel.js +82 -1

Done! I've added all the celestial items back beneath each of the five large yellow stars:

Celestial Objects:

- 1. Moon - Off-white crescent shape below star 1
- 2. Purple Cloud - Soft swirling purple cloud below star 2

2 files changed +33 - Keep Undo

WorldLevel.js +

Describe what to build next

Agent Claude Heiku 4.5 →

File: WorldLevel.js

```

1 1 class WorldLevel {
2 2     generateLargeStars(count) {
3 3         let largeStars = [];
4 4         let positions = [];
5 5         let largeStars = [];
6 6         let positions = [];
7 7         let largeStars = [];
8 8         let positions = [];
9 9
10 10         positions.push({ x, y });
11 11         largeStars.push(
12 12             {
13 13                 x: x,
14 14                 y: y,
15 15                 size: random(12, 24),
16 16                 brightness: 255,
17 17                 glowSize: random(0, 60),
18 18                 twinkleSpeed: random(0.01, 0.04),
19 19                 phase: random(TWO_PI)
20 20             }
21 21         );
22 22
23 23         return largeStars;
24 24 }
25 25
26 26 generateCelestialItems() {
27 27     const items = [
28 28         { x: this.largeStars[0].x, y: this.largeStars[0].y + 100, type: 'moon' },
29 29         { x: this.largeStars[1].x, y: this.largeStars[1].y + 100, type: 'purple-cloud' },
30 30         { x: this.largeStars[2].x, y: this.largeStars[2].y + 100, type: 'venus' },
31 31         { x: this.largeStars[3].x, y: this.largeStars[3].y + 100, type: 'orange-cloud' },
32 32         { x: this.largeStars[4].x, y: this.largeStars[4].y + 100, type: 'neptune' }
33 33     ];
34 34
35 35     return items;
36 36 }
37 37
38 38 drawBackground() {
39 39     background(220);
40 40 }
41 41
42 42 drawWorld() {
43 43     // Indigo blue night sky background
44 44     noStroke();
45 45     fill(25, 20, 80);
46 46     rect(0, 0, this.w, this.h);
47 47
48 48     // Draw regular twinkling stars
49 49     for (const star of this.stars) {
50 50         // Calculate twinkling effect
51 51         let twinkle = sin(frameCount * star.twinkleSpeed + star.phase) * 0.5 + 0.5;
52 52         let currentOpacity = star.brightness * twinkle;
53 53
54 54         ...
55 55     }
56 56 }
57 57
58 58
59 59
60 60
61 61
62 62
63 63
64 64
65 65
66 66
67 67
68 68
69 69
70 70
71 71
72 72
73 73
74 74
75 75
76 76
77 77
78 78
79 79
80 80
81 81
82 82
83 83
84 84
85 85
86 86
87 87
88 88
89 89
90 90
91 91
92 92
93 93
94 94
95 95
96 96
97 97
98 98
99 99
100 100
101 101
102 102
103 103
104 104
105 105
106 106
107 107
108 108
109 109
110 110
111 111
112 112
113 113
114 114
115 115
116 116
117 117
118 118
119 119
120 120
121 121
122 122
123 123
124 124
125 125
126 126
127 127
128 128
129 129
130 130
131 131
132 132
133 133
134 134
135 135
136 136
137 137
138 138
139 139
140 140
141 141
142 142
143 143
144 144
145 145
146 146
147 147
148 148
149 149
150 150
151 151
152 152
153 153
154 154
155 155
156 156
157 157
158 158
159 159
160 160
161 161
162 162
163 163
164 164
165 165
166 166
167 167
168 168
169 169
170 170
171 171
172 172
173 173
174 174
175 175
176 176
177 177
178 178
179 179
180 180
181 181
182 182
183 183
184 184
185 185
186 186
187 187
188 188
189 189
190 190
191 191
192 192
193 193
194 194
195 195
196 196
197 197
198 198
199 199
200 200
201 201
202 202
203 203
204 204
205 205
206 206
207 207
208 208
209 209
210 210
211 211
212 212
213 213
214 214
215 215
216 216
217 217
218 218
219 219
220 220
221 221
222 222
223 223
224 224
225 225
226 226
227 227
228 228
229 229
230 230
231 231
232 232
233 233
234 234
235 235
236 236
237 237
238 238
239 239
240 240
241 241
242 242
243 243
244 244
245 245
246 246
247 247
248 248
249 249
250 250
251 251
252 252
253 253
254 254
255 255
256 256
257 257
258 258
259 259
260 260
261 261
262 262
263 263
264 264
265 265
266 266
267 267
268 268
269 269
270 270
271 271
272 272
273 273
274 274
275 275
276 276
277 277
278 278
279 279
280 280
281 281
282 282
283 283
284 284
285 285
286 286
287 287
288 288
289 289
290 290
291 291
292 292
293 293
294 294
295 295
296 296
297 297
298 298
299 299
300 300
301 301
302 302
303 303
304 304
305 305
306 306
307 307
308 308
309 309
310 310
311 311
312 312
313 313
314 314
315 315
316 316
317 317
318 318
319 319
320 320
321 321
322 322
323 323
324 324
325 325
326 326
327 327
328 328
329 329
330 330
331 331
332 332
333 333
334 334
335 335
336 336
337 337
338 338
339 339
340 340
341 341
342 342
343 343
344 344
345 345
346 346
347 347
348 348
349 349
350 350
351 351
352 352
353 353
354 354
355 355
356 356
357 357
358 358
359 359
360 360
361 361
362 362
363 363
364 364
365 365
366 366
367 367
368 368
369 369
370 370
371 371
372 372
373 373
374 374
375 375
376 376
377 377
378 378
379 379
380 380
381 381
382 382
383 383
384 384
385 385
386 386
387 387
388 388
389 389
390 390
391 391
392 392
393 393
394 394
395 395
396 396
397 397
398 398
399 399
400 400
401 401
402 402
403 403
404 404
405 405
406 406
407 407
408 408
409 409
410 410
411 411
412 412
413 413
414 414
415 415
416 416
417 417
418 418
419 419
420 420
421 421
422 422
423 423
424 424
425 425
426 426
427 427
428 428
429 429
430 430
431 431
432 432
433 433
434 434
435 435
436 436
437 437
438 438
439 439
440 440
441 441
442 442
443 443
444 444
445 445
446 446
447 447
448 448
449 449
450 450
451 451
452 452
453 453
454 454
455 455
456 456
457 457
458 458
459 459
460 460
461 461
462 462
463 463
464 464
465 465
466 466
467 467
468 468
469 469
470 470
471 471
472 472
473 473
474 474
475 475
476 476
477 477
478 478
479 479
480 480
481 481
482 482
483 483
484 484
485 485
486 486
487 487
488 488
489 489
490 490
491 491
492 492
493 493
494 494
495 495
496 496
497 497
498 498
499 499
500 500
501 501
502 502
503 503
504 504
505 505
506 506
507 507
508 508
509 509
510 510
511 511
512 512
513 513
514 514
515 515
516 516
517 517
518 518
519 519
520 520
521 521
522 522
523 523
524 524
525 525
526 526
527 527
528 528
529 529
530 530
531 531
532 532
533 533
534 534
535 535
536 536
537 537
538 538
539 539
540 540
541 541
542 542
543 543
544 544
545 545
546 546
547 547
548 548
549 549
550 550
551 551
552 552
553 553
554 554
555 555
556 556
557 557
558 558
559 559
560 560
561 561
562 562
563 563
564 564
565 565
566 566
567 567
568 568
569 569
570 570
571 571
572 572
573 573
574 574
575 575
576 576
577 577
578 578
579 579
580 580
581 581
582 582
583 583
584 584
585 585
586 586
587 587
588 588
589 589
590 590
591 591
592 592
593 593
594 594
595 595
596 596
597 597
598 598
599 599
600 600
601 601
602 602
603 603
604 604
605 605
606 606
607 607
608 608
609 609
610 610
611 611
612 612
613 613
614 614
615 615
616 616
617 617
618 618
619 619
620 620
621 621
622 622
623 623
624 624
625 625
626 626
627 627
628 628
629 629
630 630
631 631
632 632
633 633
634 634
635 635
636 636
637 637
638 638
639 639
640 640
641 641
642 642
643 643
644 644
645 645
646 646
647 647
648 648
649 649
650 650
651 651
652 652
653 653
654 654
655 655
656 656
657 657
658 658
659 659
660 660
661 661
662 662
663 663
664 664
665 665
666 666
667 667
668 668
669 669
670 670
671 671
672 672
673 673
674 674
675 675
676 676
677 677
678 678
679 679
680 680
681 681
682 682
683 683
684 684
685 685
686 686
687 687
688 688
689 689
690 690
691 691
692 692
693 693
694 694
695 695
696 696
697 697
698 698
699 699
700 700
701 701
702 702
703 703
704 704
705 705
706 706
707 707
708 708
709 709
710 710
711 711
712 712
713 713
714 714
715 715
716 716
717 717
718 718
719 719
720 720
721 721
722 722
723 723
724 724
725 725
726 726
727 727
728 728
729 729
730 730
731 731
732 732
733 733
734 734
735 735
736 736
737 737
738 738
739 739
740 740
741 741
742 742
743 743
744 744
745 745
746 746
747 747
748 748
749 749
750 750
751 751
752 752
753 753
754 754
755 755
756 756
757 757
758 758
759 759
760 760
761 761
762 762
763 763
764 764
765 765
766 766
767 767
768 768
769 769
770 770
771 771
772 772
773 773
774 774
775 775
776 776
777 777
778 778
779 779
780 780
781 781
782 782
783 783
784 784
785 785
786 786
787 787
788 788
789 789
790 790
791 791
792 792
793 793
794 794
795 795
796 796
797 797
798 798
799 799
800 800
801 801
802 802
803 803
804 804
805 805
806 806
807 807
808 808
809 809
810 810
811 811
812 812
813 813
814 814
815 815
816 816
817 817
818 818
819 819
820 820
821 821
822 822
823 823
824 824
825 825
826 826
827 827
828 828
829 829
830 830
831 831
832 832
833 833
834 834
835 835
836 836
837 837
838 838
839 839
840 840
841 841
842 842
843 843
844 844
845 845
846 846
847 847
848 848
849 849
850 850
851 851
852 852
853 853
854 854
855 855
856 856
857 857
858 858
859 859
860 860
861 861
862 862
863 863
864 864
865 865
866 866
867 867
868 868
869 869
870 870
871 871
872 872
873 873
874 874
875 875
876 876
877 877
878 878
879 879
880 880
881 881
882 882
883 883
884 884
885 885
886 886
887 887
888 888
889 889
890 890
891 891
892 892
893 893
894 894
895 895
896 896
897 897
898 898
899 899
900 900
901 901
902 902
903 903
904 904
905 905
906 906
907 907
908 908
909 909
910 910
911 911
912 912
913 913
914 914
915 915
916 916
917 917
918 918
919 919
920 920
921 921
922 922
923 923
924 924
925 925
926 926
927 927
928 928
929 929
930 930
931 931
932 932
933 933
934 934
935 935
936 936
937 937
938 938
939 939
940 940
941 941
942 942
943 943
944 944
945 945
946 946
947 947
948 948
949 949
950 950
951 951
952 952
953 953
954 954
955 955
956 956
957 957
958 958
959 959
960 960
961 961
962 962
963 963
964 964
965 965
966 966
967 967
968 968
969 969
970 970
971 971
972 972
973 973
974 974
975 975
976 976
977 977
978 978
979 979
980 980
981 981
982 982
983 983
984 984
985 985
986 986
987 987
988 988
989 989
990 990
991 991
992 992
993 993
994 994
995 995
996 996
997 997
998 998
999 999
1000 1000
1001 1001
1002 1002
1003 1003
1004 1004
1005 1005
1006 1006
1007 1007
1008 1008
1009 1009
1010 1010
1011 1011
1012 1012
1013 1013
1014 1014
1015 1015
1016 1016
1017 1017
1018 1018
1019 1019
1020 1020
1021 1021
1022 1022
1023 1023
1024 1024
1025 1025
1026 1026
1027 1027
1028 1028
1029 1029
1030 1030
1031 1031
1032 1032
1033 1033
1034 1034
1035 1035
1036 1036
1037 1037
1038 1038
1039 1039
1040 1040
1041 1041
1042 1042
1043 1043
1044 1044
1045 1045
1046 1046
1047 1047
1048 1048
1049 1049
1050 1050
1051 1051
1052 1052
1053 1053
1054 1054
1055 1055
1056 1056
1057 1057
1058 1058
1059 1059
1060 1060
1061 1061
1062 1062
1063 1063
1064 1064
1065 1065
1066 1066
1067 1067
1068 1068
1069 1069
1070 1070
1071 1071
1072 1072
1073 1073
1074 1074
1075 1075
1076 1076
1077 1077
1078 1078
1079 1079
1080 1080
1081 1081
1082 1082
1083 1083
1084 1084
1085 1085
1086 1086
1087 1087
1088 1088
1089 1089
1090 1090
1091 1091
1092 1092
1093 1093
1094 1094
1095 1095
1096 1096
1097 1097
1098 1098
1099 1099
1100 1100
1101 1101
1102 1102
1103 1103
1104 1104
1105 1105
1106 1106
1107 1107
1108 1108
1109 1109
1110 1110
1111 1111
1112 1112
1113 1113
1114 1114
1115 1115
1116 1116
1117 1117
1118 1118
1119 1119
1120 1120
1121 1121
1122 1122
1123 1123
1124 1124
1125 1125
1126 1126
1127 1127
1128 1128
1129 1129
1130 1130
1131 1131
1132 1132
1133 1133
1134 1134
1135 1135
1136 1136
1137 1137
1138 1138
1139 1139
1140 1140
1141 1141
1142 1142
1143 1143
1144 1144
1145 1145
1146 1146
1147 1147
1148 1148
1149 1149
1150 1150
1151 1151
1152 1152
1153 1153
1154 1154
1155 1155
1156 1156
1157 1157
1158 1158
1159 1159
1160 1160
1161 1161
1162 1162
1163 1163
1164 1164
1165 1165
1166 1166
1167 1167
1168 1168
1169 1169
1170 1170
1171 1171
1172 1172
1173 1173
1174 1174
1175 1175
1176 1176
1177 1177
1178 1178
1179 1179
1180 1180
1181 1181
1182 1182
1183 1183
1184 1184
1185 1185
1186 1186
1187 1187
1188 1188
1189 1189
1190 1190
1191 1191
1192 1192
1193 1193
1194 1194
1195 1195
1196 1196
1197 1197
1198 1198
1199 1199
1200 1200
1201 1201
1202 1202
1203 1203
1204 1204
1205 1205
1206 1206
1207 1207
1208 1208
1209 1209
1210 1210
1211 1211
1212 1212
1213 1213
1214 1214
1215 1215
1216 1216
1217 1217
1218 1218
1219 1219
1220 1220
1221 1221
1222 1222
1223 1223
1224 1224
1225 1225
1
```

pjvanove_sidequest_W5

```

EXPLORER JS WorldLevel.js ⇄ WorldLevel.js
pjvanove_sidequest_W5
> libraries > types <LOCATING GAME BACKGROUND CODE...>
> .gitattributes <large yellow stars vertically aligned through
> index.html <the center of the canvas.
> jsconfig.json
> README.md
> sketch.js
# style.css
> world.json
JS WorldLevel.js
136 3 class WorldLevel {
137 93 drawWorld() {
138 138 // Draw bright yellow star shape
139 139 fill(255, 240, 100, currentOpacity);
140 140 this.drawStar(0, 0, star.size * 1.5, star.size * 0.6, 5);
141 141 // Add a warm inner glow
142 142 fill(255, 255, 200, 120 + twinkle);
143 143 this.drawStar(0, 0, star.size * 0.8, star.size * 0.3, 5);
144 144 pop();
145 145
146 146
147 147
148+ // Draw celestial items
149+ for (const item of this.celestialItems) {
150+ switch(item.type) {
151+ case 'moon':
152+ this.drawMoon(item.x, item.y);
153+ break;
154+ case 'purple-cloud':
155+ this.drawPurpleCloud(item.x, item.y);
156+ break;
157+ case 'venus':
158+ this.drawVenus(item.x, item.y);
159+ break;
160+ case 'orange-cloud':
161+ this.drawOrangeCloud(item.x, item.y);
162+ break;
163+ case 'neptune':
164+ this.drawNeptune(item.x, item.y);
165+ break;
166+ }
167+
168+ }
169+
170 drawStar(x, y, radius1, radius2, points) {
171 beginShape();
172 for (let i = 0; i < points * 2; i++) {
173 let radius = i % 2 === 0 ? radius1 : radius2;
174 let angle = TWO_PI / (points * 2) * i - PI / 2;
175 let sx = x + cos(angle) * radius;
176 let sy = y + sin(angle) * radius;
177 vertex(sx, sy);
178 }
179 endShape(CLOSE);
180
181
182 drawMoon(x, y) {
183 push();
184 noStroke();
185 fill(240, 235, 220);
186 circle(x, y, 40);
187
188 // Draw crescent by covering part with background color
189 fill(25, 20, 80);
190 circle(x + 10, y, 40);
191
192 pop();
193
194
195 drawPurpleCloud(x, y) {
196 push();
197 noStroke();
198 fill(180, 150, 220, 200);
199
200 // Draw swirling cloud shape
201 circle(x - 20, y - 15, 25);
202 circle(x + 5, y - 25, 30);
203 circle(x + 25, y - 15, 25);
204 circle(x + 10, y + 5, 20);
205 circle(x - 5, y + 10, 20);
206
207 pop();
208
209
210 drawVenus(x, y) {
211 push();
212 noStroke();
213
214 // Bright glow
215 fill(255, 250, 150, 80);
216 circle(x, y, 50);
217
218 // Bright pale yellow circle
219 fill(255, 250, 100);
220 circle(x, y, 30);
221
222 pop();
223
224 }

```

CHAT

Using the five large stars please ensure one item is under each star in the order: the Moon with an off-white crescent shape, a soft purple cloud in a swirl shape, Venus which is a bright pale yellow circle, a faint orange hazy cloud, and Neptune which has multiple blues also a circular shape.

✓ Added celestial items back to WorldLevel.js

✓ Edited □ WorldLevel.js +3 -0

✓ Edited □ WorldLevel.js +12 -1

✓ Edited □ WorldLevel.js +21 -0

✓ Edited □ WorldLevel.js +82 -1

Done! I've added all the celestial items back beneath each of the five large yellow stars:

• Celestial Objects:

1. Moon - Off-white crescent shape below star 1
2. Purple Cloud - Soft swirling purple cloud below star 2

> 2 files changed +33 - Keep Undo

pjvanove_sidequest_W5

Describe what to build next

Agent Claude Haiku 4.5

Port: 5500 Prettier

pjvanove_sidequest_W5

```

EXPLORER JS WorldLevel.js ⇄ WorldLevel.js
pjvanove_sidequest_W5
> libraries > types <LOCATING GAME BACKGROUND CODE...>
> .gitattributes <large yellow stars vertically aligned through
> index.html <the center of the canvas.
> jsconfig.json
> README.md
> sketch.js
# style.css
> world.json
JS WorldLevel.js
136 3 class WorldLevel {
137 93 drawWorld() {
138 138 // Draw bright yellow star shape
139 139 fill(255, 240, 100, currentOpacity);
140 140 this.drawStar(0, 0, star.size * 1.5, star.size * 0.6, 5);
141 141 // Add a warm inner glow
142 142 fill(255, 255, 200, 120 + twinkle);
143 143 this.drawStar(0, 0, star.size * 0.8, star.size * 0.3, 5);
144 144 pop();
145 145
146 146
147 147
148+ // Draw celestial items
149+ for (const item of this.celestialItems) {
150+ switch(item.type) {
151+ case 'moon':
152+ this.drawMoon(item.x, item.y);
153+ break;
154+ case 'purple-cloud':
155+ this.drawPurpleCloud(item.x, item.y);
156+ break;
157+ case 'venus':
158+ this.drawVenus(item.x, item.y);
159+ break;
160+ case 'orange-cloud':
161+ this.drawOrangeCloud(item.x, item.y);
162+ break;
163+ case 'neptune':
164+ this.drawNeptune(item.x, item.y);
165+ break;
166+ }
167+
168+ }
169+
170 drawStar(x, y, radius1, radius2, points) {
171 beginShape();
172 for (let i = 0; i < points * 2; i++) {
173 let radius = i % 2 === 0 ? radius1 : radius2;
174 let angle = TWO_PI / (points * 2) * i - PI / 2;
175 let sx = x + cos(angle) * radius;
176 let sy = y + sin(angle) * radius;
177 vertex(sx, sy);
178 }
179 endShape(CLOSE);
180
181
182 drawMoon(x, y) {
183 push();
184 noStroke();
185 fill(240, 235, 220);
186 circle(x, y, 40);
187
188 // Draw crescent by covering part with background color
189 fill(25, 20, 80);
190 circle(x + 10, y, 40);
191
192 pop();
193
194
195 drawPurpleCloud(x, y) {
196 push();
197 noStroke();
198 fill(180, 150, 220, 200);
199
200 // Draw swirling cloud shape
201 circle(x - 20, y - 15, 25);
202 circle(x + 5, y - 25, 30);
203 circle(x + 25, y - 15, 25);
204 circle(x + 10, y + 5, 20);
205 circle(x - 5, y + 10, 20);
206
207 pop();
208
209
210 drawVenus(x, y) {
211 push();
212 noStroke();
213
214 // Bright glow
215 fill(255, 250, 150, 80);
216 circle(x, y, 50);
217
218 // Bright pale yellow circle
219 fill(255, 250, 100);
220 circle(x, y, 30);
221
222 pop();
223
224 }

```

CHAT

Using the five large stars please ensure one item is under each star in the order: the Moon with an off-white crescent shape, a soft purple cloud in a swirl shape, Venus which is a bright pale yellow circle, a faint orange hazy cloud, and Neptune which has multiple blues also a circular shape.

✓ Added celestial items back to WorldLevel.js

✓ Edited □ WorldLevel.js +3 -0

✓ Edited □ WorldLevel.js +12 -1

✓ Edited □ WorldLevel.js +21 -0

✓ Edited □ WorldLevel.js +82 -1

Done! I've added all the celestial items back beneath each of the five large yellow stars:

• Celestial Objects:

1. Moon - Off-white crescent shape below star 1
2. Purple Cloud - Soft swirling purple cloud below star 2

> 2 files changed +33 - Keep Undo

pjvanove_sidequest_W5

Describe what to build next

Agent Claude Haiku 4.5

Port: 5500 Prettier

Prompt #5

Allow me to clarify, please have the objects (e.g. the Moon, a soft purple cloud, Venus, a faint orange cloud, and Neptune) BEHIND the large stars and once the camera hits a star they appear. Keep all the stars and objects that correspond with the stars behind the star and have the be reviled once the star is hit. Have the star disappear to show the object.

```

    5 1 class WorldLevel {
    6 2   constructor() {
    7 3     this.bg = json.world7.bg ?? [228, 235, 235];
    8 4     this.gridStep = json.world7.gridStep ?? 160;
    9 5     this.obstacles = json.obstacles ?? [];
   10 6     // NEW: camera tuning knob from JSON (data-driven)
   11 7     this.camLerp = json.camera?.lerp ?? 0.12;
   12 8     // Generate random stars for the night sky across entire world
   13 9     this.stars = this.generateStars(300);
   14 10    // Generate 5 large prominent yellow stars
   15 11    this.largeStars = this.generateLargeStars(5);
   16 12    // Generate celestial items below each star
   17 13    this.celestialItems = this.generateCelestialItems();
   18 14    // Player reference for visibility checks
   19 15    this.player = null;
   20 16    // Track which stars have been revealed
   21 17    this.revealedStars = new Set();
   22 18  }
   23 19  generateStars(count) {
   24 20    let stars = [];
   25 21    for (let i = 0; i < count; i++) {
   26 22      stars.push({
   27 23        x: random(this.w),
   28 24        y: random(this.h),
   29 25        size: random(2, 6),
   30 26        brightness: random(200, 255),
   31 27        glowSize: random(8, 20),
   32 28        twinkleSpeed: random(0.02, 0.08),
   33 29        phase: random(TWO_PI)
   34 30      });
   35 31    }
   36 32    return stars;
   37 33  }
   38 34  generateLargeStars(count) {
   39 35    let largeStars = [];
   40 36    let positions = [];
   41 37    for (let i = 0; i < count; i++) {
   42 38      const position = randomPoint(this.w, this.h);
   43 39      const star = {
   44 40        x: position.x,
   45 41        y: position.y,
   46 42        size: random(10, 20),
   47 43        brightness: random(200, 255),
   48 44        glowSize: random(10, 20),
   49 45        twinkleSpeed: random(0.02, 0.08),
   50 46        phase: random(TWO_PI)
   51 47      };
   52 48      largeStars.push(star);
   53 49      positions.push(position);
   54 50    }
   55 51    return { stars: largeStars, positions: positions };
   56 52  }
   57 53  generateCelestialItems() {
   58 54    const items = [
   59 55      { name: 'moon', x: 100, y: 100 },
   60 56      { name: 'purple-cloud', x: 200, y: 200 },
   61 57      { name: 'orange-cloud', x: 300, y: 300 },
   62 58      { name: 'venus', x: 400, y: 400 },
   63 59      { name: 'neptune', x: 500, y: 500 }
   64 60    ];
   65 61    return items;
   66 62  }
   67 63  drawWorld() {
   68 64    noStroke();
   69 65    circle(0, 0, star.glowSize);
   70 66    // Draw bright star shape
   71 67    fill(255, 255, 255, currentOpacity);
   72 68    this.drawStar(0, 0, star.size * 1.5, star.size * 0.6, 5);
   73 69    // Add a subtle inner glow
   74 70    fill(200, 220, 255, 100 * twinkle);
   75 71    this.drawStar(0, 0, star.size * 0.8, star.size * 0.3, 5);
   76 72    pop();
   77 73  }
   78 74  // Draw celestial items BEHIND the large stars
   79 75  if (this.player) {
   80 76    for (let i = 0; i < this.celestialItems.length; i++) {
   81 77      const item = this.celestialItems[i];
   82 78      const star = this.largeStars[i];
   83 79      // Only draw if player is within 300 pixels of the star
   84 80      if (dist(this.player.x, this.player.y, star.x, star.y) < 300) {
   85 81        switch (item.type) {
   86 82          case 'moon':
   87 83            this.drawMoon(item.x, item.y);
   88 84            break;
   89 85          case 'purple-cloud':
   90 86            this.drawPurpleCloud(item.x, item.y);
   91 87            break;
   92 88          case 'venus':
   93 89            this.drawVenus(item.x, item.y);
   94 90            break;
   95 91          case 'orange-cloud':
   96 92            this.drawOrangeCloud(item.x, item.y);
   97 93            break;
   98 94          case 'neptune':
   99 95            this.drawNeptune(item.x, item.y);
   100 96            break;
   101 97        }
   102 98      }
   103 99    }
   104 100  }
   105 101  }
   106 102  
```

```

    114 1 class WorldLevel {
    115 2   drawWorld() {
    116 3     noStroke();
    117 4     circle(0, 0, star.glowSize);
    118 5     // Draw bright star shape
    119 6     fill(255, 255, 255, currentOpacity);
    120 7     this.drawStar(0, 0, star.size * 1.5, star.size * 0.6, 5);
    121 8     // Add a subtle inner glow
    122 9     fill(200, 220, 255, 100 * twinkle);
    123 10    this.drawStar(0, 0, star.size * 0.8, star.size * 0.3, 5);
    124 11    pop();
    125 12  }
    126 13  // Draw celestial items BEHIND the large stars
    127 14  if (this.player) {
    128 15    for (let i = 0; i < this.celestialItems.length; i++) {
    129 16      const item = this.celestialItems[i];
    130 17      const star = this.largeStars[i];
    131 18      // Only draw if player is within 300 pixels of the star
    132 19      if (dist(this.player.x, this.player.y, star.x, star.y) < 300) {
    133 20        switch (item.type) {
    134 21          case 'moon':
    135 22            this.drawMoon(item.x, item.y);
    136 23            break;
    137 24          case 'purple-cloud':
    138 25            this.drawPurpleCloud(item.x, item.y);
    139 26            break;
    140 27          case 'venus':
    141 28            this.drawVenus(item.x, item.y);
    142 29            break;
    143 30          case 'orange-cloud':
    144 31            this.drawOrangeCloud(item.x, item.y);
    145 32            break;
    146 33          case 'neptune':
    147 34            this.drawNeptune(item.x, item.y);
    148 35            break;
    149 36        }
    150 37      }
    151 38    }
    152 39  }
    153 40  }
    154 41  }
    155 42  }
    156 43  }
    157 44  }
    158 45  
```

Prompt #6

Please ensure the camera is in the centre of the large star, once that occurs the object behind can appear. Do not reveal under the square hits the star in the middle.

```

class WorldLevel {
    drawWorld() {
        // Calculate camera center in world coordinates
        const cameraWorldX = this.cameraX + width / 2;
        const cameraWorldY = this.cameraY + height / 2;
        const distanceToCamera = dist(cameraWorldX, cameraWorldY, star.x, star.y);

        // Mark as revealed if camera center is within 150 pixels of star
        if (distanceToCamera < 150) {
            this.revealedStars.add(i);
        }

        // Only draw if not revealed
        if (!this.revealedStars.has(i)) {
            // Calculate twinkling effect
            let twinkle = sin(frameCount * star.twinkleSpeed + star.phase) * 0.5 + 0.5;
            let currentOpacity = star.brightness * twinkle;

            push();
            translate(star.x, star.y);

            // Draw golden yellow glow halo
            fill(255, 200, 100, 40 * twinkle);
            noStroke();
            circle(0, 0, star.glowSize);

            // Draw bright yellow star shape
            fill(255, 240, 100, currentOpacity);
            this.drawStar(0, 0, star.size * 1.5, star.size * 0.6, 5);

            // Add a warm inner glow
            fill(255, 255, 200, 100 * twinkle);
            this.drawStar(0, 0, star.size * 0.8, star.size * 0.3, 5);
            pop();
        }
    } else {
        // If no player yet, draw all stars
        for (const star of this.largeStars) {
            // Calculate twinkling effect
            let twinkle = sin(frameCount * star.twinkleSpeed + star.phase) * 0.5 + 0.5;
            let currentOpacity = star.brightness * twinkle;

            push();
            translate(star.x, star.y);

            // Draw bright star shape
            fill(255, 255, 255, currentOpacity);
            this.drawStar(0, 0, star.size * 1.5, star.size * 0.6, 5);

            // Add a subtle inner glow
            fill(200, 220, 255, 100 * twinkle);
            this.drawStar(0, 0, star.size * 0.8, star.size * 0.3, 5);
            pop();
        }
    }
}

// Draw celestial items BEHIND the large stars
if (this.player) {
    for (let i = 0; i < this.celestialItems.length; i++) {
        const item = this.celestialItems[i];
        const star = this.largeStars[i];

        // Draw if camera center is near the star (within 150 pixels)
        const cameraWorldX = this.cameraX + width / 2;
        const cameraWorldY = this.cameraY + height / 2;
        if (dist(cameraWorldX, cameraWorldY, star.x, star.y) < 150) {
            // Only draw if camera is centered on the star (within 150 pixels)
            if (dist(this.player.x, this.player.y, star.x, star.y) < 150) {
                switch(item.type) {
                    case 'moon':
                        this.drawMoon(item.x, item.y);
                        break;
                    case 'purple-cloud':
                        this.drawPurpleCloud(item.x, item.y);
                        break;
                    case 'venus':
                        this.drawVenus(item.x, item.y);
                        break;
                    case 'orange-cloud':
                        this.drawOrangeCloud(item.x, item.y);
                        break;
                    case 'neptune':
                        this.drawNeptune(item.x, item.y);
                        break;
                }
            }
        }
    }
}

```

```

class WorldLevel {
    drawWorld() {
        // Calculate camera center in world coordinates
        const cameraWorldX = this.cameraX + width / 2;
        const cameraWorldY = this.cameraY + height / 2;
        const distanceToCamera = dist(cameraWorldX, cameraWorldY, star.x, star.y);

        // Mark as revealed if camera center is within 150 pixels of star
        if (distanceToCamera < 150) {
            this.revealedStars.add(i);
        }

        // Only draw if not revealed
        if (!this.revealedStars.has(i)) {
            // Calculate twinkling effect
            let twinkle = sin(frameCount * star.twinkleSpeed + star.phase) * 0.5 + 0.5;
            let currentOpacity = star.brightness * twinkle;

            push();
            translate(star.x, star.y);

            // Draw golden yellow glow halo
            fill(255, 200, 100, 40 * twinkle);
            noStroke();
            circle(0, 0, star.glowSize);

            // Draw bright yellow star shape
            fill(255, 240, 100, currentOpacity);
            this.drawStar(0, 0, star.size * 1.5, star.size * 0.6, 5);

            // Add a warm inner glow
            fill(255, 255, 200, 100 * twinkle);
            this.drawStar(0, 0, star.size * 0.8, star.size * 0.3, 5);
            pop();
        }
    } else {
        // If no player yet, draw all stars
        for (const star of this.largeStars) {
            // Calculate twinkling effect
            let twinkle = sin(frameCount * star.twinkleSpeed + star.phase) * 0.5 + 0.5;
            let currentOpacity = star.brightness * twinkle;

            push();
            translate(star.x, star.y);

            // Draw bright star shape
            fill(255, 255, 255, currentOpacity);
            this.drawStar(0, 0, star.size * 1.5, star.size * 0.6, 5);

            // Add a subtle inner glow
            fill(200, 220, 255, 100 * twinkle);
            this.drawStar(0, 0, star.size * 0.8, star.size * 0.3, 5);
            pop();
        }
    }
}

// Draw celestial items BEHIND the large stars
if (this.player) {
    for (let i = 0; i < this.celestialItems.length; i++) {
        const item = this.celestialItems[i];
        const star = this.largeStars[i];

        // Draw if camera center is near the star (within 150 pixels)
        const cameraWorldX = this.cameraX + width / 2;
        const cameraWorldY = this.cameraY + height / 2;
        if (dist(cameraWorldX, cameraWorldY, star.x, star.y) < 150) {
            // Only draw if camera is centered on the star (within 150 pixels)
            if (dist(this.player.x, this.player.y, star.x, star.y) < 150) {
                switch(item.type) {
                    case 'moon':
                        this.drawMoon(item.x, item.y);
                        break;
                    case 'purple-cloud':
                        this.drawPurpleCloud(item.x, item.y);
                        break;
                    case 'venus':
                        this.drawVenus(item.x, item.y);
                        break;
                    case 'orange-cloud':
                        this.drawOrangeCloud(item.x, item.y);
                        break;
                    case 'neptune':
                        this.drawNeptune(item.x, item.y);
                        break;
                }
            }
        }
    }
}

```