

Entry Header

Name: Alexandra Dillon

Goal of Work Session

Brainstorming ideas for SideQuest 3, coding, and debugging.

GenAI Documentation

Everyone must complete this section. If not GenAI was used, write, "No GenAI used for this task."

When GenAI is not used, process evidence should

still demonstrate iteration, revision, or development over time.

Because GenAI can closely mimic human-created work, instructors or TAs may occasionally request additional process evidence to confirm non-use.

This may include original working files (e.g., an illustrator file), intermediate drafts, or a brief check-in with a TA to walk through your process.

These requests are not an assumption of misconduct. They are part of ensuring academic integrity in an environment where distinguishing between human-created and AI-generated work is increasingly difficult.

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

Date Used

: Tue Feb 10nd

Tool Disclosure

: ChatGPT 5.2

Purpose of Use

: Brainstorming, Clarification/questions, and Debugging

Summary of Interaction

: GenAI gave me guidance on handling different platform types (solid, ghost, lava) as well as helped me debug the collision and spawn issues in my code from when the blob wasn't appearing.

Human Decision Point(s)

: I chose how to implement respawn logic and collision handling, though I mostly followed debugging instructions from GenAI, but adjusted them based on my own understanding and testing.

Integrity & Verification Note

: I verified that the new levels and platform mechanics followed the assignment instructions. I confirmed debugging worked by running the game and checking that the blob could move, jump, and respond to ghost/lava platforms correctly.

Scope of GenAI Use

: Helped with debugging collision and respawn issues and verifying how to implement platform types. The actual coding, platform placement, and level decisions were done by me. (other than the previous example code)

Limitations or Misfires

: GenAI needed me to provide all my code before it could successfully debug the blob display and collisions. It couldn't fully resolve the problem until it had the complete context.

Summary of Process (Human + Tool)

I came up with the idea to add ghost (transparent) and lava platforms for extra challenge and began coding in VS Code, adding the new levels, updating blobPlayer.js with respawn logic, and implementing collision handling. When the blob did not appear and collisions weren't working, I used ChatGPT to help me get started on fixing the update and draw methods, handling the ghost/lava platforms, and integrating the spawn logic. After applying the fixes, I made minor adjustments like correcting spelling errors and adjusting the colours for clarity. I then ran multiple tests to ensure the blob could move, jump, collide, and respawn correctly, and that both new levels were playable.

Appendix

Me: my blob isn't showing up and collisions aren't working. Here's my blobPlayer.js. (...)

Absolutely! Let's go step by step to get your blob showing up and collisions working properly. From your earlier `blobPlayer.js`, there are a few common issues that can cause this:

1. Make sure your `update()` loops use the right variable names

In your earlier code, you sometimes had:

```
for (const s of platforms) {  
  if (s.type === "lava" && overlapAABB(box, p)) { // <- 'p' is  
    undefined  
    this.respawn();  
    return;  
  }  
}
```

Here, `p` should be `s`. Using the wrong variable name breaks collision detection. Always reference the current platform in the loop (`s` in this case).

2. Apply collisions in the right order

- **X-axis first**, then **Y-axis**.
- Check for ghost platforms first so the player passes through them.
- Then check lava platforms to respawn.
- Then resolve solid collisions.

For example:

```
// Move horizontally  
box.x += this.vx;  
for (const p of platforms) {  
  if (p.type === "ghost") continue;  
  if (overlapAABB(box, p)) {  
    if (this.vx > 0) box.x = p.x - box.w;  
    else if (this.vx < 0) box.x = p.x + p.w;  
    this.vx = 0;  
  }  
}
```

```

    }
}

// Move vertically
box.y += this.vy;
this.onGround = false;
for (const p of platforms) {
    if (p.type === "ghost") continue;
    if (p.type === "lava" && overlapAABB(box, p)) {
        this.respawn();
        return;
    }
    if (overlapAABB(box, p)) {
        if (this.vy > 0) {
            box.y = p.y - box.h;
            this.vy = 0;
            this.onGround = true;
        } else if (this.vy < 0) {
            box.y = p.y + p.h;
            this.vy = 0;
        }
    }
}
}

```

3. Ensure `spawnFromLevel()` stores spawn coordinates

```

spawnFromLevel(level) {
    this.startX = level.start.x;
    this.startY = level.start.y;

    this.gravity = level.gravity;
    this.jumpV = level.jumpV;

    this.x = this.startX;
    this.y = this.startY;
    this.r = level.start.r;

    this.vx = 0;
    this.vy = 0;
}

```

```
    this.onGround = false;
}
```

Without `startX` and `startY`, your `respawn()` function won't know where to reset the blob.

4. Update your `draw()`

Make sure you're passing the correct color:

```
player.draw(world.theme.blob);
```

And in `draw()`:

```
draw(colourHex) {
  fill(color(colourHex));
  beginShape();
  for (let i = 0; i < this.points; i++) {
    const a = (i / this.points) * TAU;
    const n = noise(
      cos(a) * this.wobbleFreq + 100,
      sin(a) * this.wobbleFreq + 100,
      this.t
    );
    const rr = this.r + map(n, 0, 1, -this.wobble, this.wobble);
    vertex(this.x + cos(a) * rr, this.y + sin(a) * rr);
  }
  endShape(CLOSE);
}
```

5. Test one thing at a time

1. Make sure the blob appears on the first level.
2. Test solid platforms collisions.
3. Add ghost platforms and verify they are ignored.

4. Add lava platforms and verify respawn works.

Me: I want lava platforms that make the blob respawn when touched. Where does that code go?

Perfect — the lava platform logic needs to go **inside your `update()` method** in `BlobPlayer.js`, because that's where you move the blob and check for collisions each frame.

Here's **where it fits**:

1. Build your blob's bounding box

```
let playerBox = {  
  x: this.x - this.r,  
  y: this.y - this.r,  
  w: this.r * 2,  
  h: this.r * 2  
};
```

2. Check horizontal movement and collisions (X-axis)

You usually skip ghost platforms here. Lava can also be checked, but it's usually easier to handle it after vertical movement.

3. Move vertically and handle collisions (Y-axis)

This is where the **lava check** should go:

```
// Move vertically  
playerBox.y += this.vy;  
this.onGround = false;  
  
for (const p of platforms) {  
  // Lava kills the player  
  if (p.type === "lava" && overlapAABB(playerBox, p)) {  
    this.respawn(); // send blob back to spawn  
  }  
}
```

```

    return; // stop further collision checks this frame
}

// Ghost platforms: ignore collisions
if (p.type === "ghost") continue;

// Solid platforms
if (overlapAABB(playerBox, p)) {
    if (this.vy > 0) {
        // Falling: land on top
        playerBox.y = p.y - playerBox.h;
        this.vy = 0;
        this.onGround = true;
    } else if (this.vy < 0) {
        // Jumping: hit bottom of platform
        playerBox.y = p.y + p.h;
        this.vy = 0;
    }
}
}
}

```

4. Write back the blob's center

```

this.x = playerBox.x + playerBox.w / 2;
this.y = playerBox.y + playerBox.h / 2;

```

✓ Key points:

- `respawn()` should reset the blob's `x` and `y` to its starting coordinates.
- The lava check must happen **before checking solid collisions**, so touching lava doesn't let the blob "stick" to a platform.
- Place it in the **Y-axis collision loop** in `update()`. You can also duplicate it for X-axis if you want side-touching lava to work, but usually vertical collision is enough.

Me: it still isnt working heres code from my other files. Tell me whats wrong: (...)

Ah! I can see the problem clearly now. The **main reason the blob doesn't appear in level 3** (or any new level) is in your **lava/ghost collision code inside `update()`**. There are a few mistakes:

❶ Issue with your lava check

```
for (const s of platforms) {  
  if (s.type === "lava" && overlapAABB(box, p)) {  
    this.respawn();  
    return;  
  }  
}
```

Notice `overlapAABB(box, p)` — you're looping over `s` but using `p`. That will throw a JS error and **stop the `update()`**, so the blob is never drawn. This could make it "invisible" or just a gray screen.

✅ Fix: use `s` consistently:

```
if (s.type === "lava" && overlapAABB(box, s)) {  
  this.respawn();  
  return;  
}
```

❷ Ghost logic is fine

Ghost platforms are meant to be ignored, and that part is okay:

```
if (s.type === "ghost") continue;
```

❸ Respawn should reset velocities and onGround

Your `respawn()` is okay, but you might also want to reset `onGround`:

```
respawn() {
```



```
this.x = this.startX;
this.y = this.startY;
this.vx = 0;
this.vy = 0;
this.onGround = false;
}
```

4 Extra tip: avoid multiple **returns** in update

Right now, if lava triggers, you **return** immediately. That's fine, but make sure you don't accidentally skip drawing or other logic.

✓ Corrected **update()** snippet

Here's the horizontal loop fixed:

```
box.x += this.vx;

for (const s of platforms) {
  // Lava kills you
  if (s.type === "lava" && overlapAABB(box, s)) {
    this.respawn();
    return;
  }

  // Ghost platforms: ignore physics
  if (s.type === "ghost") continue;

  // Normal solid collision
  if (overlapAABB(box, s)) {
    if (this.vx > 0) box.x = s.x - box.w;
    else if (this.vx < 0) box.x = s.x + s.w;
    this.vx = 0;
  }
}
```

And vertical loop:

```
box.y += this.vy;
this.onGround = false;

for (const s of platforms) {
  if (s.type === "lava" && overlapAABB(box, s)) {
    this.respawn();
    return;
  }
  if (s.type === "ghost") continue;

  if (overlapAABB(box, s)) {
    if (this.vy > 0) {
      box.y = s.y - box.h;
      this.vy = 0;
      this.onGround = true;
    } else if (this.vy < 0) {
      box.y = s.y + s.h;
      this.vy = 0;
    }
  }
}
```

5 Extra checks

- Make sure **level 3 JSON** has **start** coordinates.
- Make sure **theme.blob** exists.
- Log the player's spawn point after loading level 3:

```
console.log("Player spawn:", this.x, this.y, this.r);
```

If you want, I can rewrite your **entire BlobPlayer.js** with **lava, ghost, respawn, and safe defaults for all levels**, so level 3 always works.

Do you want me to do that?