Process & Decision Document

Name: Jasmine Henri

Role(s): Coder & Reviewer of AI changes

Primary responsibility for this work: Worked independently and with the Visual Studio Chatbot to come up with modifications for the code that will add another level and include a door that the blob has reach to move onto level 2.

Goal of Work Session: During this work session, I focused on making changes to the blob game so that loops and arrays were utulised to place obstacles and a end to the level through a door.

GenAI tools (if used):

- Chatbot on Visual Studio Code 6

GenAI Documentation

Date Used: February 9th, 2026

Tool Disclosure: Chatbot on Visual Studio Code 6

Purpose of Use: To help refine the code I added when it had caused errors and to help add code that was too complex for my knowing

Summary of Interaction:

- I explained to the chatbot that I had added a new door function and asked it to help me troubleshoot a code error I created
- Help place the spikes randomly and restart the game if the spikes are touched by the blob

Human Decision Point(s):

- I did not have to overrule any AI decisions, all outputs were helpful

Integrity & Verification Note:

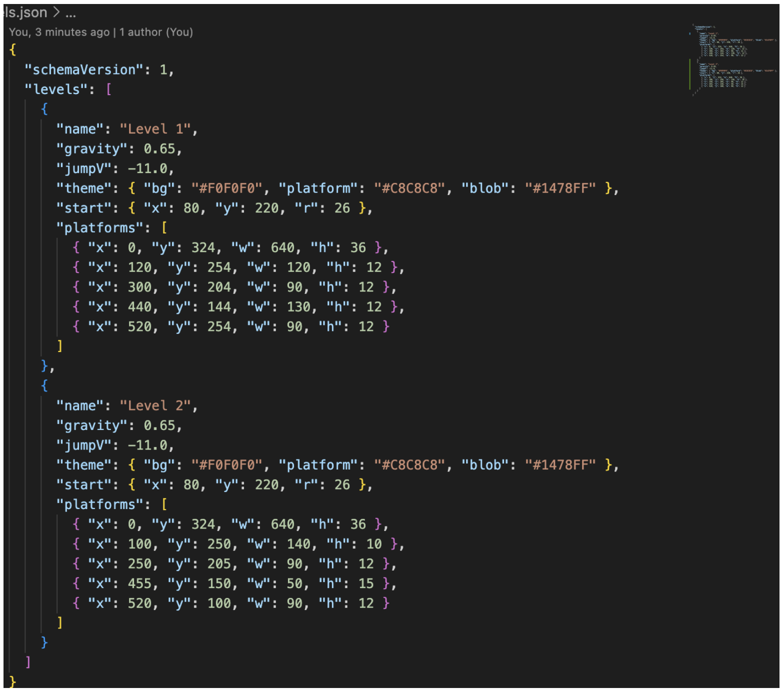
All decisions were verified using:

- My prior knowledge of coding from CS105 and CS106
- Knowledge of coding that was reviewed in Class 4 lecture by David

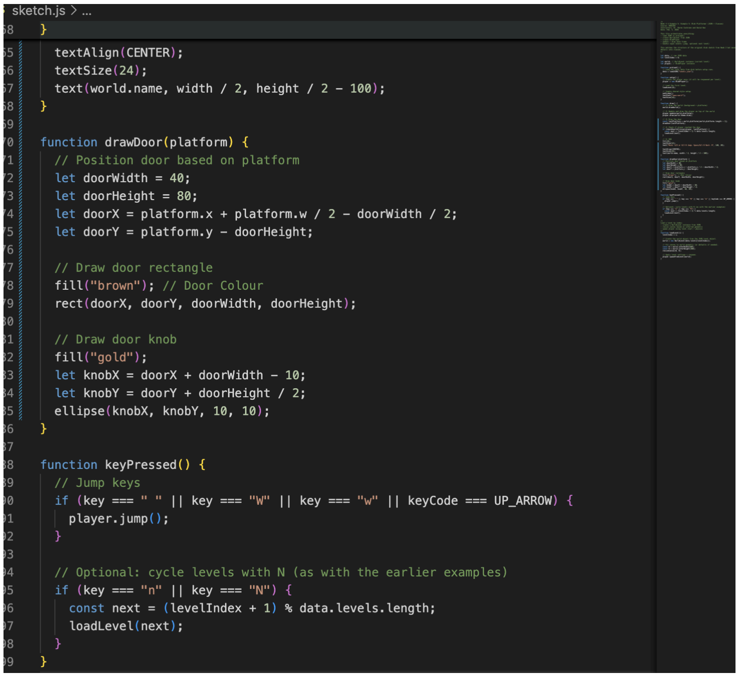
Scope of GenAI Use:

Specifically, GenAI did *not*:

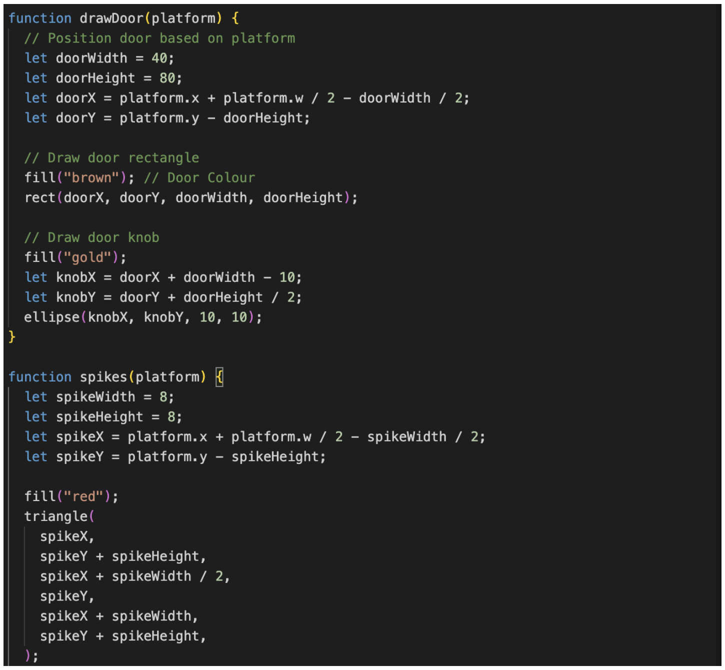
- Added the second level in levels.json
- Created the drawDoor function and filled it with code
- Created the spikes function and filled it with code



```json
{
  "schemaVersion": 1,
  "levels": [
    {
      "name": "Level 1",
      "gravity": 0.65,
      "jumpV": -11.0,
      "theme": { "bg": "#F0F0F0", "platform": "#C8C8C8", "blob": "#1478FF" },
      "start": { "x": 80, "y": 220, "r": 26 },
      "platforms": [
        { "x": 0, "y": 324, "w": 640, "h": 36 },
        { "x": 120, "y": 254, "w": 120, "h": 12 },
        { "x": 300, "y": 204, "w": 90, "h": 12 },
        { "x": 440, "y": 144, "w": 130, "h": 12 },
        { "x": 520, "y": 254, "w": 90, "h": 12 }
      ]
    },
    {
      "name": "Level 2",
      "gravity": 0.65,
      "jumpV": -11.0,
      "theme": { "bg": "#F0F0F0", "platform": "#C8C8C8", "blob": "#1478FF" },
      "start": { "x": 80, "y": 220, "r": 26 },
      "platforms": [
        { "x": 0, "y": 324, "w": 640, "h": 36 },
        { "x": 100, "y": 250, "w": 140, "h": 10 },
        { "x": 250, "y": 205, "w": 90, "h": 12 },
        { "x": 455, "y": 150, "w": 50, "h": 15 },
        { "x": 520, "y": 100, "w": 90, "h": 12 }
      ]
    }
  ]
}
```



```javascript
}
  textAlign(CENTER);
  textSize(24);
  text(world.name, width / 2, height / 2 - 100);
}

function drawDoor(platform) {
  // Position door based on platform
  let doorWidth = 40;
  let doorHeight = 80;
  let doorX = platform.x + platform.w / 2 - doorWidth / 2;
  let doorY = platform.y - doorHeight;

  // Draw door rectangle
  fill("brown"); // Door Colour
  rect(doorX, doorY, doorWidth, doorHeight);

  // Draw door knob
  fill("gold");
  let knobX = doorX + doorWidth - 10;
  let knobY = doorY + doorHeight / 2;
  ellipse(knobX, knobY, 10, 10);
}

function keyPressed() {
  // Jump keys
  if (key === " " || key === "W" || key === "w" || keyCode === UP_ARROW) {
    player.jump();
  }

  // Optional: cycle levels with N (as with the earlier examples)
  if (key === "n" || key === "N") {
    const next = (levelIndex + 1) % data.levels.length;
    loadLevel(next);
  }
}
```



```
function drawDoor(platform) {
  // Position door based on platform
  let doorWidth = 40;
  let doorHeight = 80;
  let doorX = platform.x + platform.w / 2 - doorWidth / 2;
  let doorY = platform.y - doorHeight;

  // Draw door rectangle
  fill("brown"); // Door Colour
  rect(doorX, doorY, doorWidth, doorHeight);

  // Draw door knob
  fill("gold");
  let knobX = doorX + doorWidth - 10;
  let knobY = doorY + doorHeight / 2;
  ellipse(knobX, knobY, 10, 10);
}

function spikes(platform) {
  let spikeWidth = 8;
  let spikeHeight = 8;
  let spikeX = platform.x + platform.w / 2 - spikeWidth / 2;
  let spikeY = platform.y - spikeHeight;

  fill("red");
  triangle(
    spikeX,
    spikeY + spikeHeight,
    spikeX + spikeWidth / 2,
    spikeY,
    spikeX + spikeWidth,
    spikeY + spikeHeight,
  );
```

Limitations or Misfires:

There were no misfires with AI’s addition to my codes and it resolved my code errors.

Summary of Process (Human + Tool)

Three iterations with GenAI:

- Added code for the second level in levels.json (code screenshots above)
- Created the drawDoor function and filled it with code (code screenshots above)
  - First iteration happened after this where I asked the chatbot to fix the level and blob not loading and add a conditional if statement that determines if you reach level 2 if you have reached the door
- Created the spikes function and filled it with code (code screenshots above)
  - Second and final iteration where I asked the chatbot to choose random platforms for each level to put the spikes on and make a conditional statement that returns the blob to the start of the level if it touches the spike

Decision Points & Trade-offs

I considered allowing the chatbot to make a fun detailed door and scary spikes but I wanted to practice and put my coding knowledge to use so I did it myself instead. That did cause some errors because I didn’t encorporate the door in properly and the spikes needed more coding as well. Overall, I was happy with my changes and how the chatbot helped me refine some of the changes.

Verification & Judgement

I evaluated the appropriateness of my changes by:

- Reviewing notes from CS105 and CS106
- Reviewing Class 4 lecture by David

Limitations, Dead Ends, or Open Questions

- All issues were resolved successfully by the chatbot

Appendix - Visual Studio Code 6 Chatbot

#1 I added a door to the final platform of each level but it'd not showing up and now the blob won't move, could you help me fix that please. Could you also add a conditional if statement that determines if you reach level 2 if you have reached the door

The MCP server `io.github.github/github\-mcp\-server` *may have new tools and requires interaction to start.* *Start it now?*

I'll open the key game files (sketch.js, blobPlayer.js, worldLevel.js, platform.js, levels.json) to find the door, blob movement, and level transition logic.

Now I'll search the project for any existing checkDoorCollision references.

Next, I'll add a missing checkDoorCollision function and adjust the level-advance conditional in sketch.js. I'll patch the file now.

Edited

sketch.js+31-2

Now I'll mark the todo steps as completed.

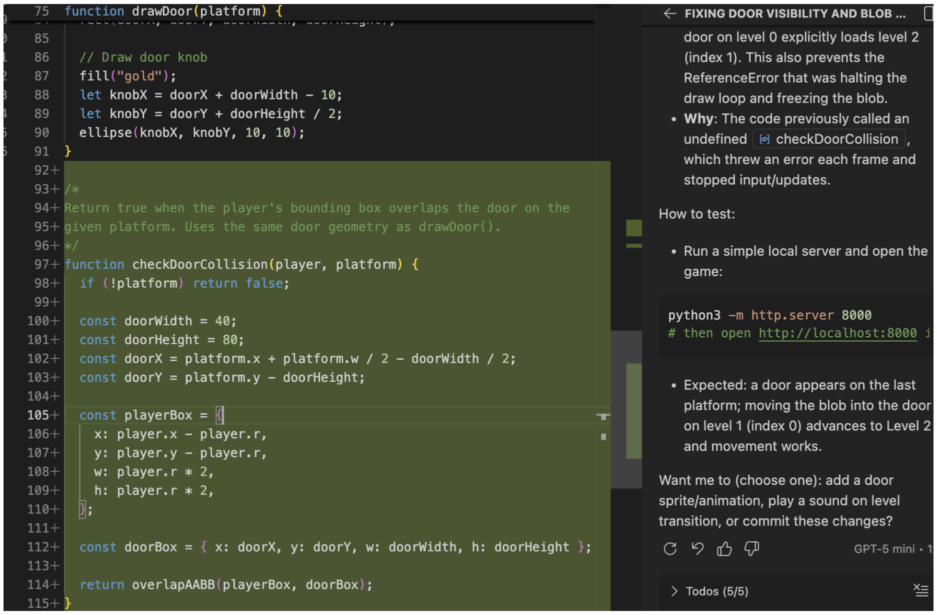
I'll open index.html to confirm script load order.

Done — I fixed the missing door collision and level-transition logic so the blob moves again.

- **File**: sketch.js — added checkDoorCollision and changed the level-advance `if` so reaching the door on level 0 explicitly loads level 2 (index 1). This also prevents the ReferenceError that was halting the draw loop and freezing the blob.
- **Why**: The code previously called an undefined checkDoorCollision, which threw an error each frame and stopped input/updates.

How to test:

- Run a simple local server and open the game:
- Expected: a door appears on the last platform; moving the blob into the door on level 1 (index 0) advances to Level 2 and movement works.



```
function drawDoor(platform) {

  // Draw door knob
  fill("gold");
  let knobX = doorX + doorWidth - 10;
  let knobY = doorY + doorHeight / 2;
  ellipse(knobX, knobY, 10, 10);
}

/*
Return true when the player's bounding box overlaps the door on the
given platform. Uses the same door geometry as drawDoor().
*/
function checkDoorCollision(player, platform) {
  if (!platform) return false;

  const doorWidth = 40;
  const doorHeight = 80;
  const doorX = platform.x + platform.w / 2 - doorWidth / 2;
  const doorY = platform.y - doorHeight;

  const playerBox = {
    x: player.x - player.r,
    y: player.y - player.r,
    w: player.r * 2,
    h: player.r * 2,
  };

  const doorBox = { x: doorX, y: doorY, w: doorWidth, h: doorHeight };

  return overlapAABB(playerBox, doorBox);
}
```

door on level 0 explicitly loads level 2 (index 1). This also prevents the ReferenceError that was halting the draw loop and freezing the blob.

- **Why**: The code previously called an undefined checkDoorCollision, which threw an error each frame and stopped input/updates.

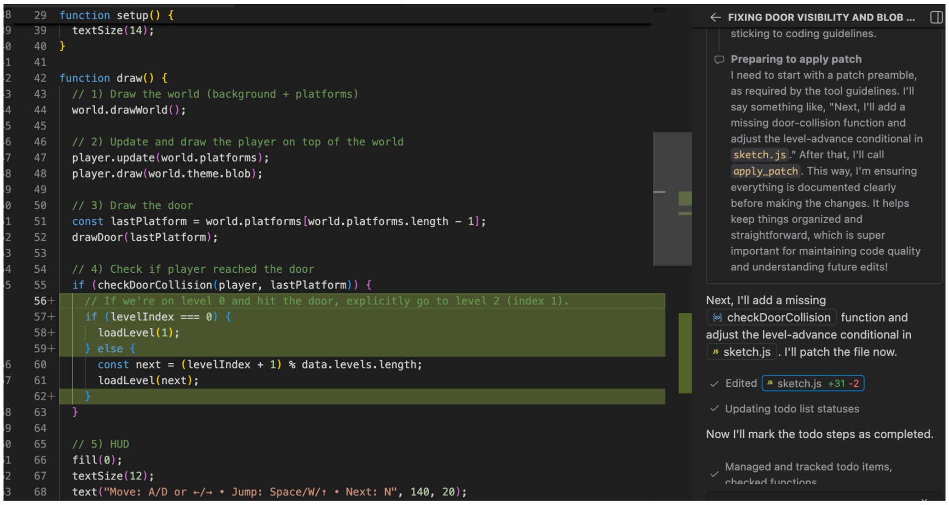
How to test:

- Run a simple local server and open the game:

```
python3 -m http.server 8000
# then open http://localhost:8000 i
```

- Expected: a door appears on the last platform; moving the blob into the door on level 1 (index 0) advances to Level 2 and movement works.

Want me to (choose one): add a door sprite/animation, play a sound on level transition, or commit these changes?



```
function setup() {
  textSize(14);
}

function draw() {
  // 1) Draw the world (background + platforms)
  world.drawWorld();

  // 2) Update and draw the player on top of the world
  player.update(world.platforms);
  player.draw(world.theme.blob);

  // 3) Draw the door
  const lastPlatform = world.platforms[world.platforms.length - 1];
  drawDoor(lastPlatform);

  // 4) Check if player reached the door
  if (checkDoorCollision(player, lastPlatform)) {
    // If we're on level 0 and hit the door, explicitly go to level 2 (index 1).
    if (levelIndex === 0) {
      loadLevel(1);
    } else {
      const next = (levelIndex + 1) % data.levels.length;
      loadLevel(next);
    }
  }

  // 5) HUD
  fill(0);
  textSize(12);
  text("Move: A/D or ←/→ • Jump: Space/W/↑ • Next: N", 140, 20);
```

sticking to coding guidelines.

Preparing to apply patch
I need to start with a patch preamble, as required by the tool guidelines. I'll say something like, "Next, I'll add a missing door-collision function and adjust the level-advance conditional in `sketch.js`." After that, I'll call `apply_patch`. This way, I'm ensuring everything is documented clearly before making the changes. It helps keep things organized and straightforward, which is super important for maintaining code quality and understanding future edits!

Next, I'll add a missing checkDoorCollision function and adjust the level-advance conditional in sketch.js. I'll patch the file now.

✓ Edited sketch.js +31 -2

✓ Updating todo list statuses

Now I'll mark the todo steps as completed.

✓ Managed and tracked todo items, checked functions

#2 I've created a function draw spikes on some of the platforms, can you choose random platforms for each level to put these spikes on and can you make a conditional statement that returns the blob to the start of the level if it touches the spike

I'll search the project for any spike-related code and where to integrate spike placement.

I'll update the todo list, then patch worldLevel.js to mark a random platform with spikes and call spikes() during drawing, and patch sketch.js to respawn the player when they hit a spike.

Created 5 todos

Edited

worldLevel.js+13-0

Edited

sketch.js+25-0

Starting: *Run quick in-repo checks* (5/5)

Progress update: I added random spikes and spike-collision respawn.

What I changed

- worldLevel.js: choose one random non-ground/non-final platform per level and set hasSpike; draw spikes in drawWorld().
- sketch.js: check spike AABB collision each frame and call player.spawnFromLevel(world) to respawn when hit.

How to test

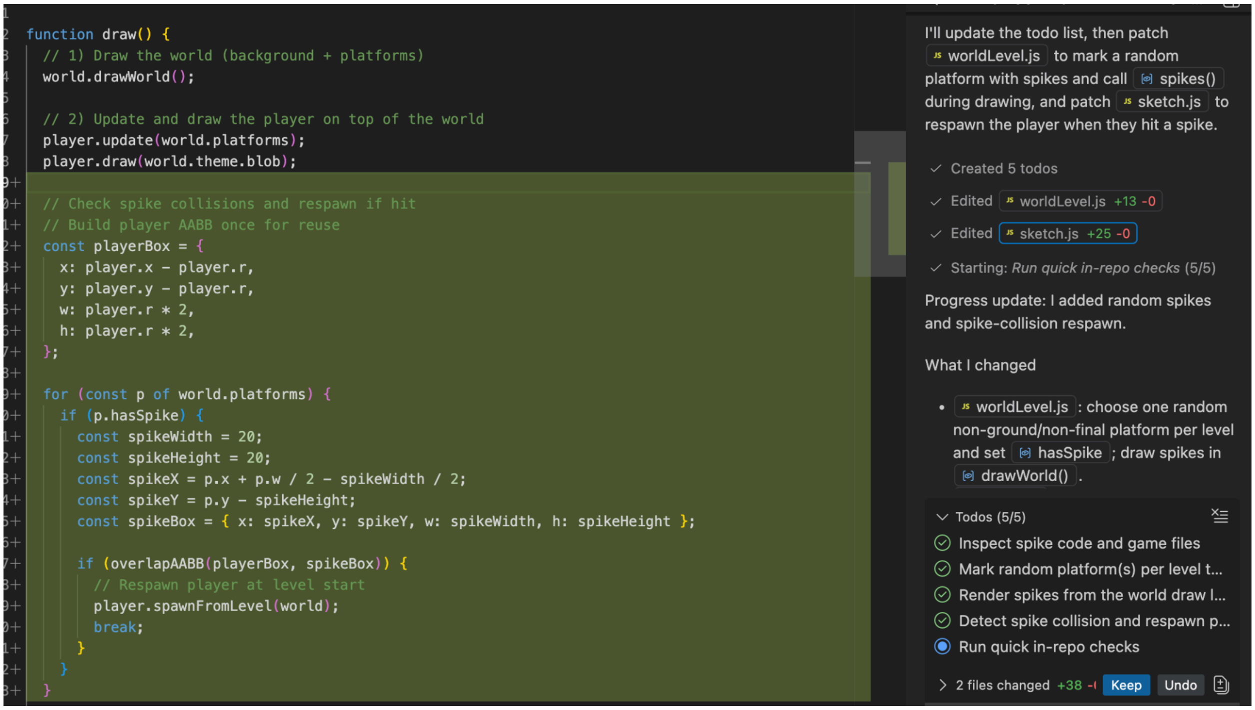
1. Start a local server from the project root:
2. Open http://localhost:8000 and play. A red spike should appear on a random mid-level platform; touching it respawns the blob to the level start.

Next steps (optional)

- Place more than one spike per level.
- Exclude specific platforms when choosing spikes.
- Add damage animation/sound on hit.

Want me to implement any of those next?

GPT-5 mini • 1x



```
function draw() {
  // 1) Draw the world (background + platforms)
  world.drawWorld();

  // 2) Update and draw the player on top of the world
  player.update(world.platforms);
  player.draw(world.theme.blob);

  // Check spike collisions and respawn if hit
  // Build player AABB once for reuse
  const playerBox = {
    x: player.x - player.r,
    y: player.y - player.r,
    w: player.r * 2,
    h: player.r * 2,
  };

  for (const p of world.platforms) {
    if (p.hasSpike) {
      const spikeWidth = 20;
      const spikeHeight = 20;
      const spikeX = p.x + p.w / 2 - spikeWidth / 2;
      const spikeY = p.y - spikeHeight;
      const spikeBox = { x: spikeX, y: spikeY, w: spikeWidth, h: spikeHeight };

      if (overlapAABB(playerBox, spikeBox)) {
        // Respawn player at level start
        player.spawnFromLevel(world);
        break;
      }
    }
  }
```

I'll update the todo list, then patch `worldLevel.js` to mark a random platform with spikes and call `spikes()` during drawing, and patch `sketch.js` to respawn the player when they hit a spike.

✓ Created 5 todos

✓ Edited `worldLevel.js` +13 -0

✓ Edited `sketch.js` +25 -0

✓ Starting: *Run quick in-repo checks* (5/5)

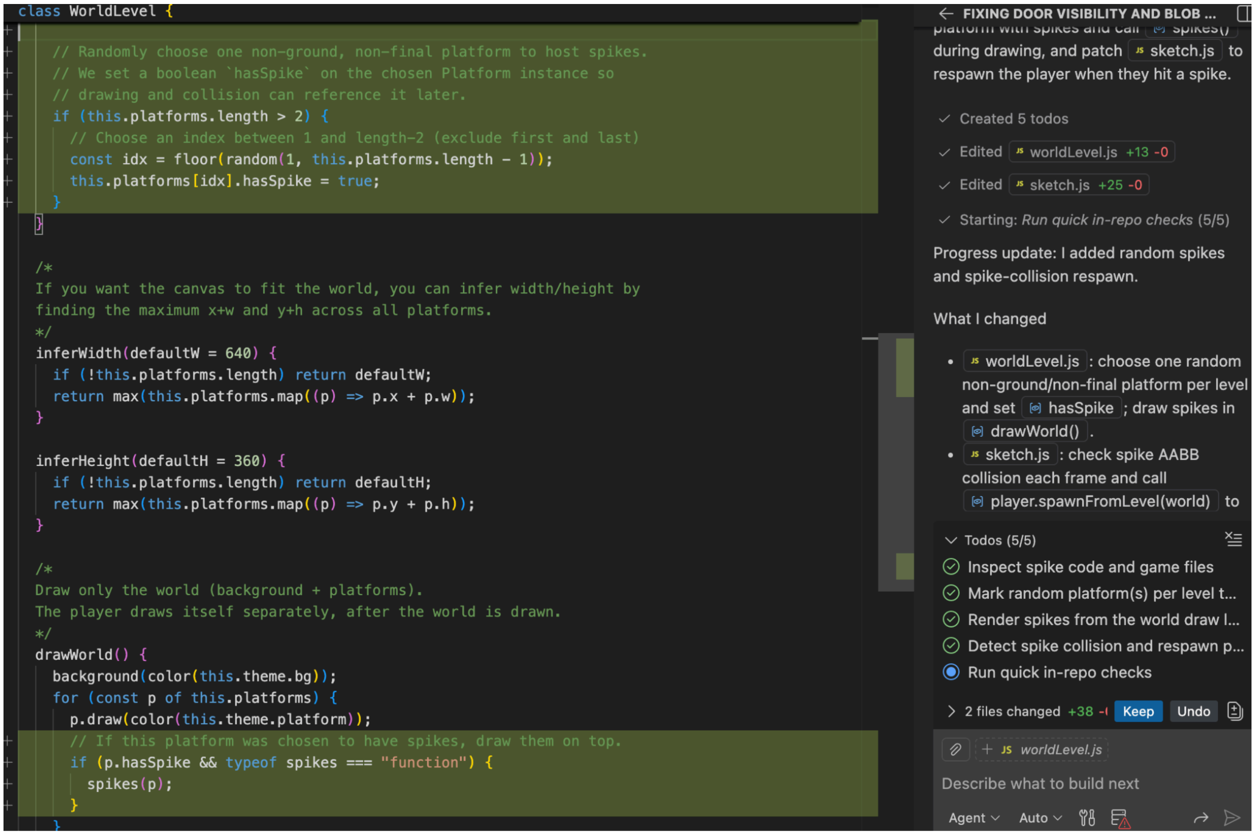
Progress update: I added random spikes and spike-collision respawn.

What I changed

- `worldLevel.js` : choose one random non-ground/non-final platform per level and set `hasSpike` ; draw spikes in `drawWorld()` .

Todos (5/5)

- ✓ Inspect spike code and game files
- ✓ Mark random platform(s) per level t...
- ✓ Render spikes from the world draw l...
- ✓ Detect spike collision and respawn p...
- ● Run quick in-repo checks



```
class WorldLevel {

    // Randomly choose one non-ground, non-final platform to host spikes.
    // We set a boolean `hasSpike` on the chosen Platform instance so
    // drawing and collision can reference it later.
    if (this.platforms.length > 2) {
      // Choose an index between 1 and length-2 (exclude first and last)
      const idx = floor(random(1, this.platforms.length - 1));
      this.platforms[idx].hasSpike = true;
    }
  }

  /*
  If you want the canvas to fit the world, you can infer width/height by
  finding the maximum x+w and y+h across all platforms.
  */
  inferWidth(defaultW = 640) {
    if (!this.platforms.length) return defaultW;
    return max(this.platforms.map((p) => p.x + p.w));
  }

  inferHeight(defaultH = 360) {
    if (!this.platforms.length) return defaultH;
    return max(this.platforms.map((p) => p.y + p.h));
  }

  /*
  Draw only the world (background + platforms).
  The player draws itself separately, after the world is drawn.
  */
  drawWorld() {
    background(color(this.theme.bg));
    for (const p of this.platforms) {
      p.draw(color(this.theme.platform));
      // If this platform was chosen to have spikes, draw them on top.
      if (p.hasSpike && typeof spikes === "function") {
        spikes(p);
      }
    }
```

during drawing, and patch `sketch.js` to respawn the player when they hit a spike.

✓ Created 5 todos

✓ Edited `worldLevel.js` +13 -0

✓ Edited `sketch.js` +25 -0

✓ Starting: *Run quick in-repo checks* (5/5)

Progress update: I added random spikes and spike-collision respawn.

What I changed

- `worldLevel.js` : choose one random non-ground/non-final platform per level and set `hasSpike` ; draw spikes in `drawWorld()` .
- `sketch.js` : check spike AABB collision each frame and call `player.spawnFromLevel(world)` to

Todos (5/5)

- ✓ Inspect spike code and game files
- ✓ Mark random platform(s) per level t...
- ✓ Render spikes from the world draw l...
- ✓ Detect spike collision and respawn p...
- ● Run quick in-repo checks