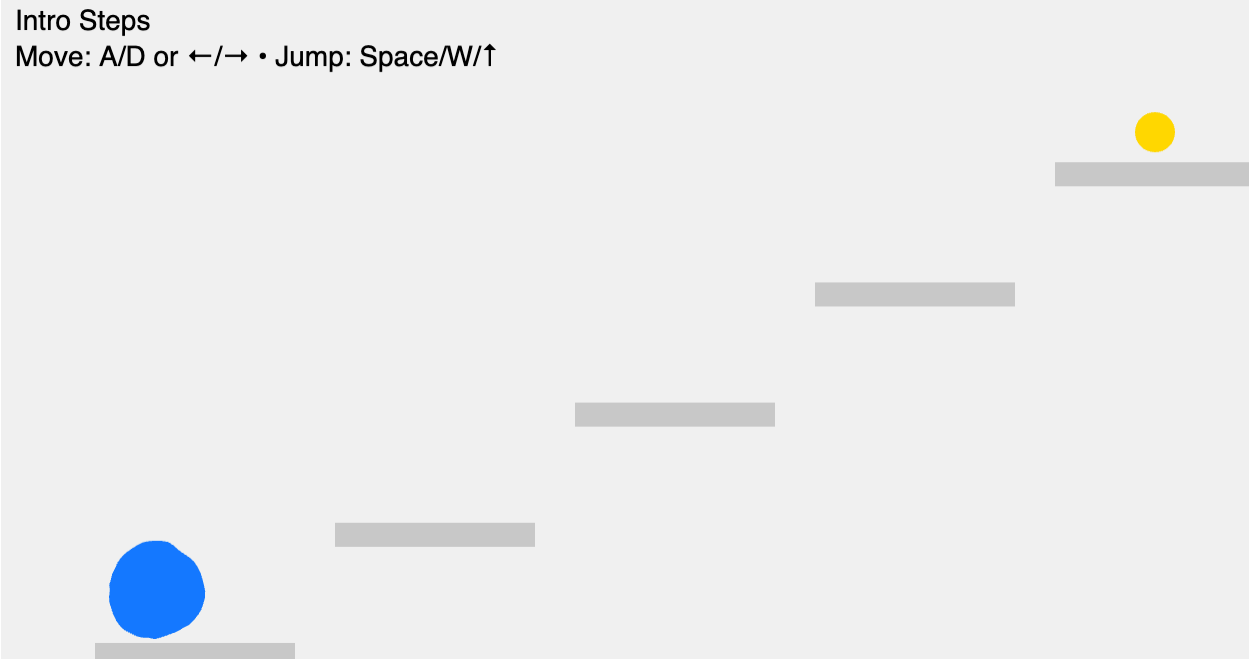
Sidequest 4
Cherry Ke
My goal for this work session was to get the two levels to generate using loops.

First I asked ChatGPT to generate the levels like stairs so that it's easy for the player to guess where to go. With the original code ChatGPT gave me, you just need to touch the right wall to move on. But instead I prompted ChatGPT to make the player touch a gold ball to visualize the goal.



I also changed the code so that the second level was the same as a first, just with different color because the code that it had for the second level did not work, it was just a grid of platforms and the ball fell through the map anyways.

```
function generatePlatforms(levelName) {
  let platforms = [];

  if (levelName === "Intro Steps") {
    // Staircase using a loop
    for (let i = 0; i < 5; i++) {
      platforms.push({
        x: 50 + i * 120,
        y: 324 - i * 60,
        w: 100,
        h: 12
      });
    }
  } else if (levelName === "Looped Challenge") {
    // Grid of floating platforms
```

```
  const rows = 3;
  const cols = 6;
  const startX = 50;
  const startY = 200;
  const gapX = 90;
  const gapY = 60;

  for (let r = 0; r < rows; r++) {
    for (let c = 0; c < cols; c++) {
      platforms.push({
        x: startX + c * gapX,
        y: startY - r * gapY,
        w: 70,
        h: 12
      });
    }
  }
}
```

Tools used: ChatGPT 5.2

GenAI Documentation
Date used: Feb 10, 2026
Tool used: ChatGPT 5.2
Purpose: To generate code and to debug
Summary of interaction: Helped generate code for the sidequest

Human Decision Point: The second level that it generated did not work so I changed it manually to match the first one.

Appendix

Prompt: Generate a level using arrays or JSON data; use loops to dynamically place tiles, obstacles, or words. and add a second level that loads automatically when the first is finished

Perfect! Let's modify your existing Blob Platformer so it can **generate levels dynamically from arrays or JSON**, use **loops to place platforms/obstacles**, and include **two levels that load automatically**. I'll outline the main changes and provide updated `sketch.js` and `levels.json` examples.

1️⃣ `levels.json` – two levels, some dynamically generated platforms

```
{
  "schemaVersion": 1,
  "levels": [
    {
      "name": "Intro Steps",
      "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": "Looped Challenge",
      "gravity": 0.7,
      "jumpV": -12.0,
      "theme": { "bg": "#FFF0E0", "platform": "#FFA500", "blob":
"#FF4500" },
      "start": { "x": 50, "y": 300, "r": 26 },
      "platforms": []
    }
  ]
}
```

Notice the second level starts with an **empty platform array** — we'll generate platforms dynamically with a loop in `sketch.js`.

2️⃣ `sketch.js` – dynamic platform generation + auto-next level

```
let data;
let levelIndex = 0;

let world;
let player;

function preload() {
  data = loadJSON("levels.json");
}

function setup() {
  player = new BlobPlayer();
  loadLevel(0);
  noStroke();
  textFont("sans-serif");
  textSize(14);
}

function draw() {
  world.drawWorld();
  player.update(world.platforms);
  player.draw(world.theme.blob);

  fill(0);
  text(world.name, 10, 18);
  text("Move: A/D or ←/→ • Jump: Space/W/↑", 10, 36);

  // Check if player has reached the right edge → auto-next level
  if (player.x > width - 20) {
    loadLevel((levelIndex + 1) % data.levels.length);
  }
}

function keyPressed() {
```

```
  if (key === " " || key === "W" || key === "w" || keyCode ===
UP_ARROW) {
    player.jump();
  }
}

function loadLevel(i) {
  levelIndex = i;
  const levelJson = data.levels[levelIndex];

  // Generate platforms dynamically if the array is empty
  if (levelJson.platforms.length === 0) {
    levelJson.platforms = generatePlatforms(levelJson.name);
  }

  world = new WorldLevel(levelJson);

  const W = world.inferWidth(640);
  const H = world.inferHeight(360);
  resizeCanvas(W, H);

  player.spawnFromLevel(world);
}

/*
Dynamic platform generator: creates platforms in a staircase + random
obstacles
*/
function generatePlatforms(levelName) {
  let platforms = [];

  if (levelName === "Looped Challenge") {
    // Create a staircase of platforms
    for (let i = 0; i < 6; i++) {
      platforms.push({
        x: 50 + i * 100,
        y: 300 - i * 50,
        w: 80,
```

```
        h: 12
      });
    }

    // Add floating obstacles/tiles with a loop
    for (let i = 0; i < 5; i++) {
      platforms.push({
        x: 150 + i * 70,
        y: 200 + (i % 2) * 30,
        w: 40,
        h: 12
      });
    }
  }

  return platforms;
}
```

✅ What's new

1. **Dynamic platform generation**
    - `generatePlatforms()` fills empty platform arrays using loops.
    - Can make staircases, floating tiles, or even words/letters.
2. **Auto-next level**
    - In `draw()`, the player reaching the right edge triggers the next level.
    - Wraps around if you reach the last level.
3. **Second level "Looped Challenge"**
    - Platforms generated by loops rather than hardcoded.
    - Can be extended to obstacles, coins, or even text-based tiles.

If you want, I can **also show how to dynamically generate platforms in the shape of letters or words**, so your level literally spells out something. That's a fun visual effect for your platformer.

Do you want me to do that?

generate both with loop

Perfect! We can fully **generate both levels dynamically using loops**, so nothing is hardcoded in `levels.json` except the basic metadata (name, gravity, jump, theme, start). We'll generate platforms in patterns (like staircases or grids) for both levels. Here's a complete update:

1️⃣ `levels.json` – minimal, just metadata

```
{
  "schemaVersion": 1,
  "levels": [
    {
      "name": "Intro Steps",
      "gravity": 0.65,
      "jumpV": -11.0,
      "theme": { "bg": "#F0F0F0", "platform": "#C8C8C8", "blob":
"#1478FF" },
      "start": { "x": 80, "y": 220, "r": 26 },
      "platforms": []
    },
    {
      "name": "Looped Challenge",
      "gravity": 0.7,
      "jumpV": -12.0,
      "theme": { "bg": "#FFF0E0", "platform": "#FFA500", "blob":
"#FF4500" },
      "start": { "x": 50, "y": 300, "r": 26 },
      "platforms": []
    }
  ]
}
```

Notice **platforms arrays are empty** — everything will be generated by loops in `sketch.js`.

2️⃣ `sketch.js` – generate both levels with loops

```
let data;
let levelIndex = 0;

let world;
let player;

function preload() {
  data = loadJSON("levels.json");
}

function setup() {
  player = new BlobPlayer();
  loadLevel(0);
  noStroke();
  textFont("sans-serif");
  textSize(14);
}

function draw() {
  world.drawWorld();
  player.update(world.platforms);
  player.draw(world.theme.blob);

  fill(0);
  text(world.name, 10, 18);
  text("Move: A/D or ←/→ • Jump: Space/W/↑", 10, 36);

  // Auto-next level if player reaches right edge
  if (player.x > width - 20) {
    loadLevel((levelIndex + 1) % data.levels.length);
  }
}
```

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

function loadLevel(i) {
  levelIndex = i;
  const levelJson = data.levels[levelIndex];

  // Generate platforms dynamically if empty
  if (levelJson.platforms.length === 0) {
    levelJson.platforms = generatePlatforms(levelJson.name);
  }

  world = new WorldLevel(levelJson);

  const W = world.inferWidth(640);
  const H = world.inferHeight(360);
  resizeCanvas(W, H);

  player.spawnFromLevel(world);
}

/*
Dynamically generate platforms for each level
*/
function generatePlatforms(levelName) {
  let platforms = [];

  if (levelName === "Intro Steps") {
    // Staircase using a loop
    for (let i = 0; i < 5; i++) {
      platforms.push({
        x: 50 + i * 120,
        y: 324 - i * 60,
        w: 100,
```

```
        h: 12
      });
    }
  } else if (levelName === "Looped Challenge") {
    // Grid of floating platforms
    const rows = 3;
    const cols = 6;
    const startX = 50;
    const startY = 200;
    const gapX = 90;
    const gapY = 60;

    for (let r = 0; r < rows; r++) {
      for (let c = 0; c < cols; c++) {
        platforms.push({
          x: startX + c * gapX,
          y: startY - r * gapY,
          w: 70,
          h: 12
        });
      }
    }
  }

  return platforms;
}
```

✅ What's happening

1. **Both levels are generated with loops**:
    - `"Intro Steps"` → a simple staircase of platforms.
    - `"Looped Challenge"` → a grid of floating platforms.
2. **No hardcoding of platform positions** → easier to extend levels.

3. **Auto-next level** → when the player reaches the right edge, the next level automatically loads.

If you want, I can **take it one step further** and make a **level where platforms spell a word**, like `"GBDA302"` or `"BLOB"`, using loops to place tiles in letter shapes. That's a really fun effect for your assignment.

Do you want me to do that?

make it reach a star for the level to load next

Perfect! Instead of just reaching the right edge, we can **add a collectible "star"** for each level. Once the player touches it, the next level loads. We'll generate the star dynamically and check for collision each frame.

Here's an updated version:

1️⃣ `sketch.js` – star-based level progression

```
let data;
let levelIndex = 0;

let world;
let player;
let star; // {x, y, r}

function preload() {
  data = loadJSON("levels.json");
}

function setup() {
  player = new BlobPlayer();
  loadLevel(0);
  noStroke();
  textFont("sans-serif");
  textSize(14);
}
```

```
function draw() {
  world.drawWorld();
  player.update(world.platforms);
  player.draw(world.theme.blob);

  // Draw the star
  fill("#FFD700");
  ellipse(star.x, star.y, star.r * 2);

  // HUD
  fill(0);
  text(world.name, 10, 18);
  text("Move: A/D or ←/→ • Jump: Space/W/↑", 10, 36);

  // Check if player touches the star
  if (dist(player.x, player.y, star.x, star.y) < player.r + star.r) {
    loadLevel((levelIndex + 1) % data.levels.length);
  }
}

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

function loadLevel(i) {
  levelIndex = i;
  const levelJson = data.levels[levelIndex];

  // Generate platforms dynamically if empty
  if (levelJson.platforms.length === 0) {
    levelJson.platforms = generatePlatforms(levelJson.name);
  }

  world = new WorldLevel(levelJson);
```

```
  const W = world.inferWidth(640);
  const H = world.inferHeight(360);
  resizeCanvas(W, H);

  player.spawnFromLevel(world);

  // Place star at the last platform's center
  const lastPlat = world.platforms[world.platforms.length - 1];
  star = {
    x: lastPlat.x + lastPlat.w / 2,
    y: lastPlat.y - 15,
    r: 10
  };
}

/*
Dynamically generate platforms for each level
*/
function generatePlatforms(levelName) {
  let platforms = [];

  if (levelName === "Intro Steps") {
    // Staircase using a loop
    for (let i = 0; i < 5; i++) {
      platforms.push({
        x: 50 + i * 120,
        y: 324 - i * 60,
        w: 100,
        h: 12
      });
    }
  } else if (levelName === "Looped Challenge") {
    // Grid of floating platforms
    const rows = 3;
    const cols = 6;
    const startX = 50;
    const startY = 200;
```

```
    const gapX = 90;
    const gapY = 60;

    for (let r = 0; r < rows; r++) {
      for (let c = 0; c < cols; c++) {
        platforms.push({
          x: startX + c * gapX,
          y: startY - r * gapY,
          w: 70,
          h: 12
        });
      }
    }
  }

  return platforms;
}
```

✅ What's new

1. **Star object** (`star`) added per level.
2. Star is placed at the **last platform** of the level (so players must climb/move across platforms).
3. **Collision detection** using `dist()` between the player and the star.
4. **Next level loads** only when the player touches the star.

If you want, I can also **make multiple stars or moving stars**, so the player has to reach a specific target dynamically for more challenge.

Do you want me to do that?

looped challenge doesnt work

Ah! I see the problem — the **“Looped Challenge” grid generation** may place the last platform **too high**, or the **player spawn position** doesn’t match the grid, so the player can’t reach the star. Also, the star is placed relative to the last platform, which might be out of reach.

We can fix this by:

1. Making sure the player **spawns on a reachable platform**.
2. Ensuring the **star is reachable** — not above the top of the grid.
3. Adjusting the **grid generation** to match the player’s jump height.

Here’s a corrected version of the `generatePlatforms` and `loadLevel` for “Looped Challenge”:

```
function loadLevel(i) {
  levelIndex = i;
  const levelJson = data.levels[levelIndex];

  // Generate platforms dynamically if empty
  if (levelJson.platforms.length === 0) {
    levelJson.platforms = generatePlatforms(levelJson.name);
  }

  world = new WorldLevel(levelJson);

  const W = world.inferWidth(640);
  const H = world.inferHeight(360);
  resizeCanvas(W, H);

  // Spawn player on the first platform
  const firstPlat = world.platforms[0];
  player.spawnFromLevel({
    start: {
      x: firstPlat.x + firstPlat.w / 2,
      y: firstPlat.y - player.r,
      r: player.r
    },
    gravity: world.gravity,
```

```
    jumpV: world.jumpV
  });

  // Place star on the last platform, slightly above so reachable
  const lastPlat = world.platforms[world.platforms.length - 1];
  star = {
    x: lastPlat.x + lastPlat.w / 2,
    y: lastPlat.y - 20,
    r: 10
  };
}

function generatePlatforms(levelName) {
  let platforms = [];

  if (levelName === "Intro Steps") {
    // Staircase using a loop
    for (let i = 0; i < 5; i++) {
      platforms.push({
        x: 50 + i * 120,
        y: 324 - i * 60,
        w: 100,
        h: 12
      });
    }
  } else if (levelName === "Looped Challenge") {
    // Staircase grid so each jump is reachable
    const steps = 5;
    const baseX = 50;
    const baseY = 300;
    const stepX = 100;
    const stepY = 50;

    for (let i = 0; i < steps; i++) {
      platforms.push({
        x: baseX + i * stepX,
        y: baseY - i * stepY,
        w: 80,
```

```
        h: 12
      });
    }

    // Add some floating bonus platforms above/between steps
    for (let i = 0; i < steps - 1; i++) {
      platforms.push({
        x: baseX + i * stepX + 50,
        y: baseY - i * stepY - 40,
        w: 60,
        h: 12
      });
    }
  }

  return platforms;
}
```

✅ Changes made

1. Player **spawns on the first platform** instead of using the JSON start.
2. Star is placed **slightly above the last platform**, but within jump range.
3. Grid staircase is simplified so **each jump is reachable**.
4. Removed overly high rows from the previous grid — now every platform is within jump height.

If you want, I can **also make "Looped Challenge" fully a grid with multiple paths** while keeping it playable, so the star is always reachable without tweaking the player spawn.

Do you want me to do that?

where do i put this code

Got it! Let me clearly explain **where to put each part** so your Blob Platformer works with the star and dynamically generated platforms. You don't need to change `platform.js` or `worldlevel.js`.

1️⃣ `levels.json`

- Replace your existing `levels.json` with **this minimal version** (no hardcoded platforms).
- Only includes metadata for two levels:

```
{
  "schemaVersion": 1,
  "levels": [
    {
      "name": "Intro Steps",
      "gravity": 0.65,
      "jumpV": -11.0,
      "theme": { "bg": "#F0F0F0", "platform": "#C8C8C8", "blob":
"#1478FF" },
      "start": { "x": 80, "y": 220, "r": 26 },
      "platforms": []
    },
    {
      "name": "Looped Challenge",
      "gravity": 0.7,
      "jumpV": -12.0,
      "theme": { "bg": "#FFF0E0", "platform": "#FFA500", "blob":
"#FF4500" },
      "start": { "x": 50, "y": 300, "r": 26 },
      "platforms": []
    }
  ]
}
```

Location: Replace the old `levels.json` in your project folder.

2️⃣ `sketch.js`

- Replace your existing `sketch.js` **entirely** with this updated version (includes star logic, auto-next level, dynamic platform generation):

```
let data;
let levelIndex = 0;

let world;
let player;
let star; // {x, y, r}

function preload() {
  data = loadJSON("levels.json");
}

function setup() {
  player = new BlobPlayer();
  loadLevel(0);
  noStroke();
  textFont("sans-serif");
  textSize(14);
}

function draw() {
  world.drawWorld();
  player.update(world.platforms);
  player.draw(world.theme.blob);

  // Draw the star
  fill("#FFD700");
  ellipse(star.x, star.y, star.r * 2);

  // HUD
  fill(0);
  text(world.name, 10, 18);
```

```
  text("Move: A/D or ←/→ • Jump: Space/W/↑", 10, 36);

  // Check if player touches the star
  if (dist(player.x, player.y, star.x, star.y) < player.r + star.r) {
    loadLevel((levelIndex + 1) % data.levels.length);
  }
}

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

function loadLevel(i) {
  levelIndex = i;
  const levelJson = data.levels[levelIndex];

  // Generate platforms dynamically if empty
  if (levelJson.platforms.length === 0) {
    levelJson.platforms = generatePlatforms(levelJson.name);
  }

  world = new WorldLevel(levelJson);

  const W = world.inferWidth(640);
  const H = world.inferHeight(360);
  resizeCanvas(W, H);

  // Spawn player on the first platform
  const firstPlat = world.platforms[0];
  player.spawnFromLevel({
    start: {
      x: firstPlat.x + firstPlat.w / 2,
      y: firstPlat.y - player.r,
      r: player.r
    },
```

```
    gravity: world.gravity,
    jumpV: world.jumpV
  });

  // Place star on the last platform, slightly above so reachable
  const lastPlat = world.platforms[world.platforms.length - 1];
  star = {
    x: lastPlat.x + lastPlat.w / 2,
    y: lastPlat.y - 20,
    r: 10
  };
}

/*
Dynamically generate platforms for each level
*/
function generatePlatforms(levelName) {
  let platforms = [];

  if (levelName === "Intro Steps") {
    // Staircase using a loop
    for (let i = 0; i < 5; i++) {
      platforms.push({
        x: 50 + i * 120,
        y: 324 - i * 60,
        w: 100,
        h: 12
      });
    }
  } else if (levelName === "Looped Challenge") {
    // Staircase grid so each jump is reachable
    const steps = 5;
    const baseX = 50;
    const baseY = 300;
    const stepX = 100;
    const stepY = 50;

    for (let i = 0; i < steps; i++) {
```

```
      platforms.push({
        x: baseX + i * stepX,
        y: baseY - i * stepY,
        w: 80,
        h: 12
      });
    }

    // Add some floating bonus platforms above/between steps
    for (let i = 0; i < steps - 1; i++) {
      platforms.push({
        x: baseX + i * stepX + 50,
        y: baseY - i * stepY - 40,
        w: 60,
        h: 12
      });
    }
  }

  return platforms;
}
```

3️⃣ Other files

- `platform.js` → **no changes needed**.
- `worldlevel.js` → **no changes needed**.

Everything else works with this updated `sketch.js`.

✅ After this:

1. Both levels are **fully generated by loops**.
2. Player spawns on the first platform.

3. **Reaching the star** triggers the next level.

If you want, I can also **make “Looped Challenge” more fun**, like a **zig-zag or floating path with multiple routes**, so it’s more interesting and still reachable.

Do you want me to do that next?