Process & Decision Documentation

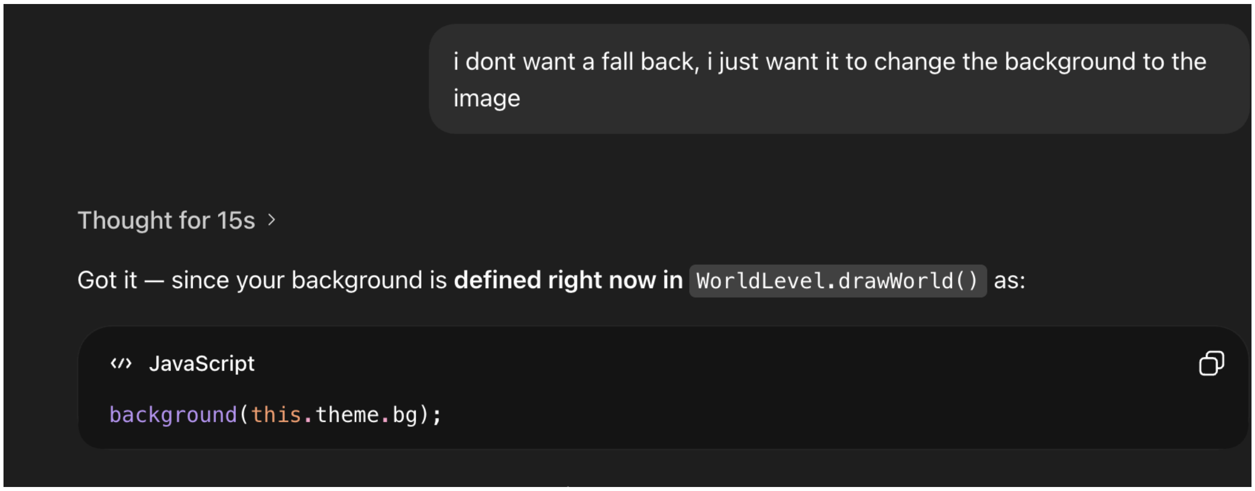
Project/Assignment Decisions

For this assignment I created a simple horror game experience by altering the code example provided in class. I started off by creating a background using pixilart, it was a simple forest background in which I also added a ghost at the end of the background so that the player would get an element of horror. I also chose to add red eyes in the background to create more of a scary environment. After this, I took the background and wanted to implement it into the code example that was provided in class. I wanted to do this change myself but when looking at the code I noticed that the background color and properties were implemented in a different way from what I was used to. (using objects) due to this I had a bit of trouble changing the background into an image as it would have been different from changing the color value itself. At this point, I decided to use AI to help me understand how it worked, and also what changes to make in order to have the background be an image. I looked through the specific snippets and added them into my code, constantly checking if it was working.

At this point, I had to do a bit of debugging as there were certain things that weren't working and the changes were not visible. I made sure to ask the AI for assistance where needed but tried to understand it myself just to make it easier for myself later on. After going back and forth for a while, I finally successfully added the background, after this I wanted to make some subtle changes such as changing the color of the platforms and blob, which wasn't too difficult as now I had a better understanding of the object properties. One thing I noticed was the platform sizes and positioning did not align with the background, as a result I noticed odd gaps within the ground that the user could not see since I had changed the platform color to be black to match with the background. To fix this, I went to the platforms and altered the y position and length of the bottom platform so that there were no more holes at the bottom. I made sure to keep playtesting so I could notice any more things that needed to be changed. One other thing that needed to be changed was the color of the platforms, although the bottom one was black and matched the background, the rest of the platforms were blending in with black background. To fix this, I added a colour field within the platform object, and made just the background black with the platforms as a dark grey color to make them more visible. I noticed the color for the blob was white for my first iteration, which felt too harsh and I had changed it to be more off-white to make it easier on the eyes. For my final change, I made the text at the top left corner white as it also lost visibility by blending into the background.

Role-Based Process Evidence

I used GenAI to help me understand certain parts of the code to make changes, and also for clarification on why certain changes didn't work for me.

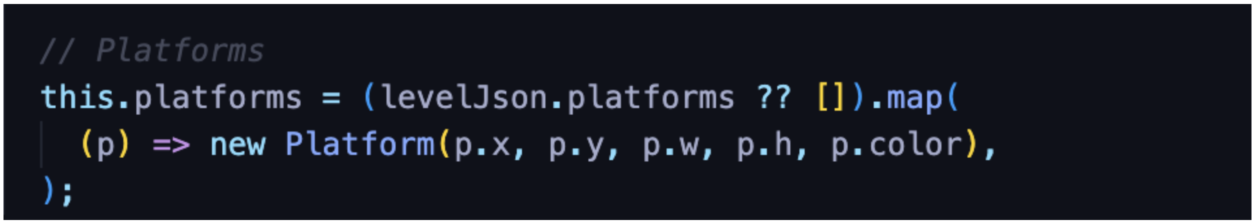


i dont want a fall back, i just want it to change the background to the image

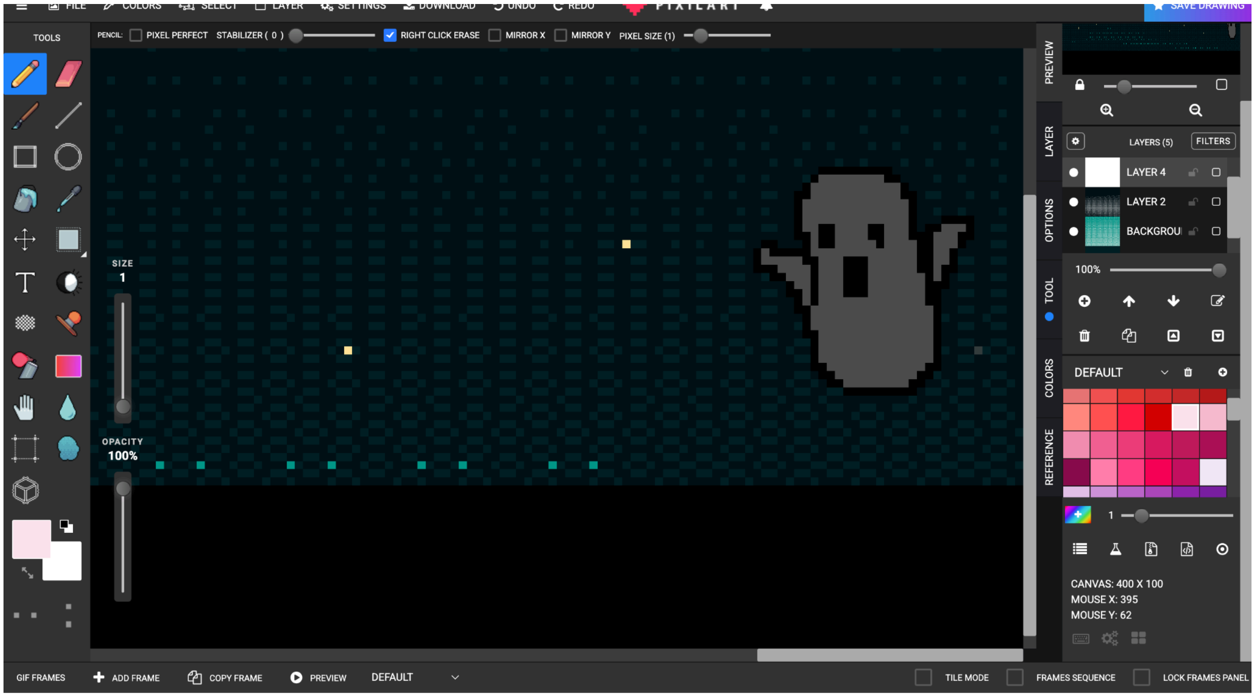
Thought for 15s ›

Got it — since your background is **defined right now in** `WorldLevel.drawWorld()` as:

```javascript
background(this.theme.bg);
```



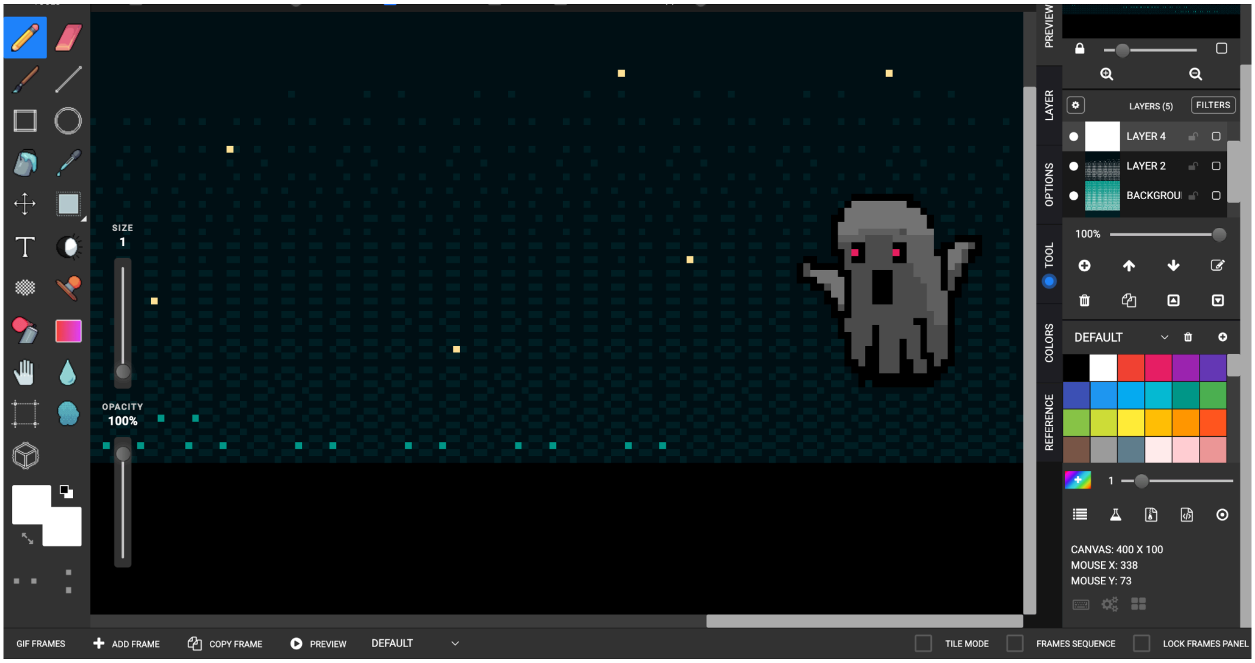
```
// Platforms
this.platforms = (levelJson.platforms ?? []).map(
  (p) => new Platform(p.x, p.y, p.w, p.h, p.color),
);
```



PENCIL: ☐ PIXEL PERFECT STABILIZER (0) ☑ RIGHT CLICK ERASE ☐ MIRROR X ☐ MIRROR Y PIXEL SIZE (1)

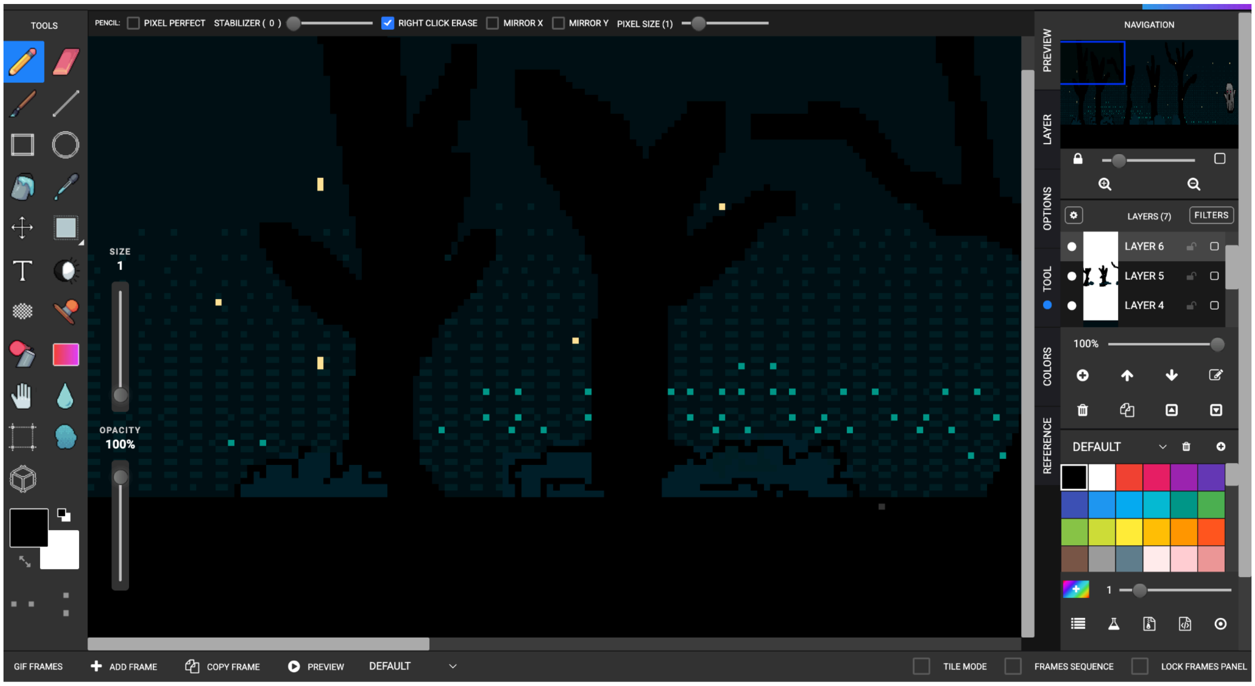
CANVAS: 400 X 100
MOUSE X: 395
MOUSE Y: 62

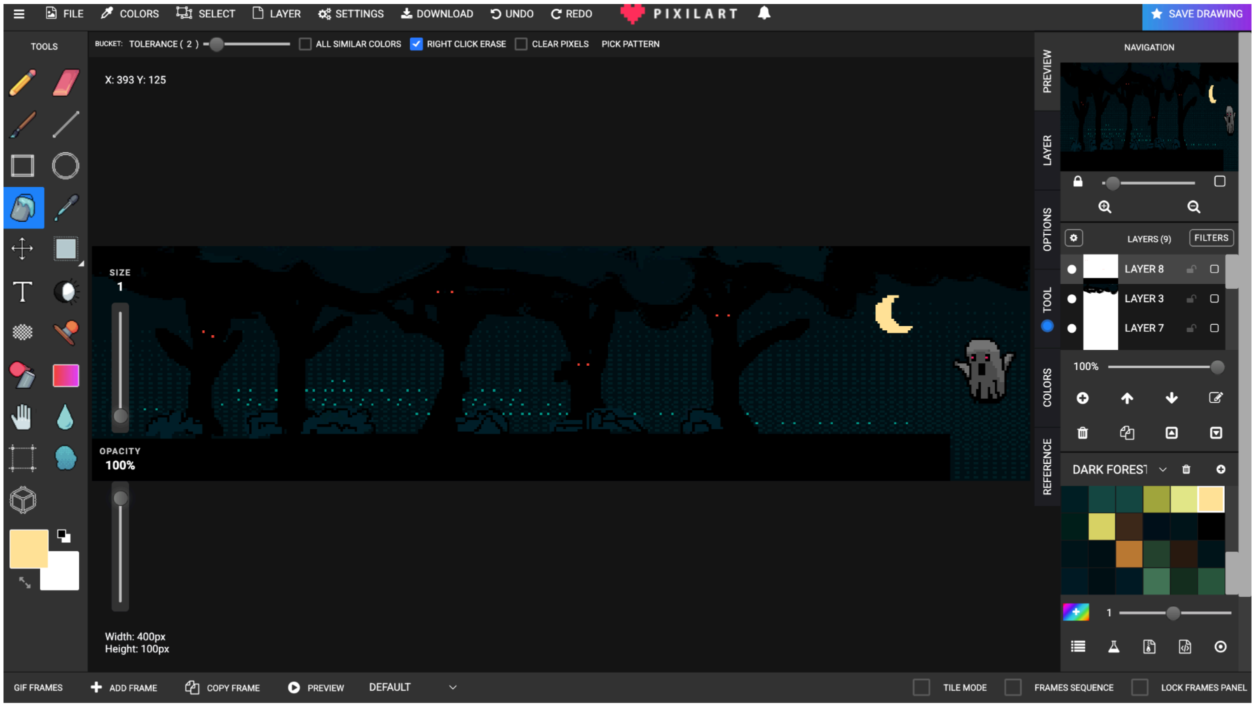
☐ TILE MODE ☐ FRAMES SEQUENCE ☐ LOCK FRAMES PANEL

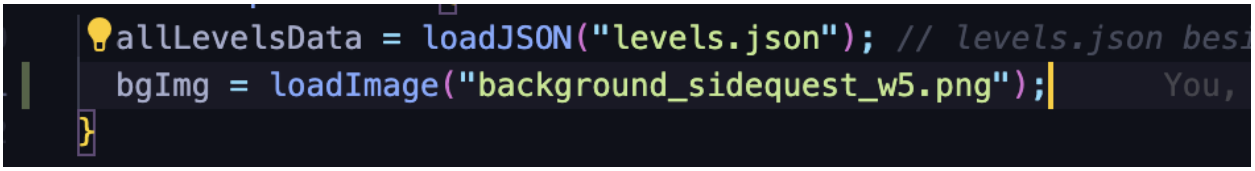


CANVAS: 400 X 100
MOUSE X: 338
MOUSE Y: 73

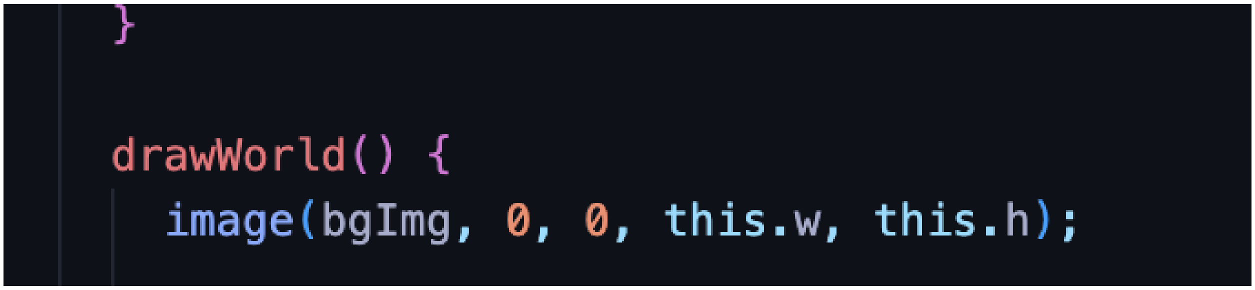
☐ TILE MODE ☐ FRAMES SEQUENCE ☐ LOCK FRAMES PANEL



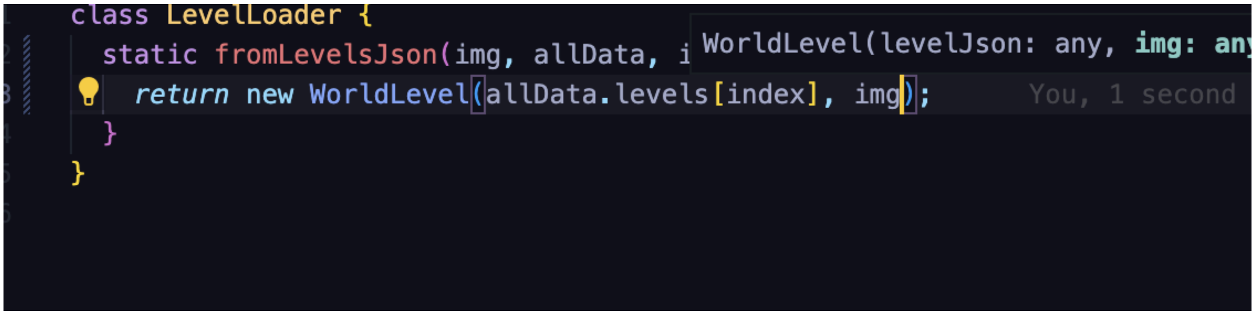




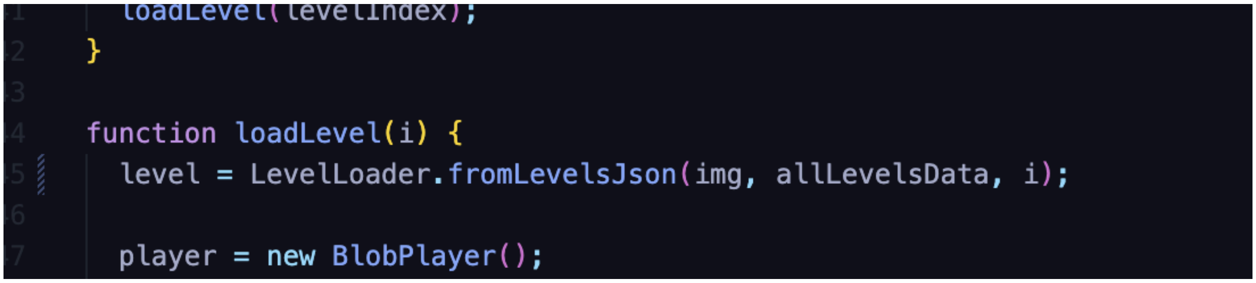
```
allLevelsData = loadJSON("levels.json"); // levels.json besi
bgImg = loadImage("background_sidequest_w5.png");        You,
}
```



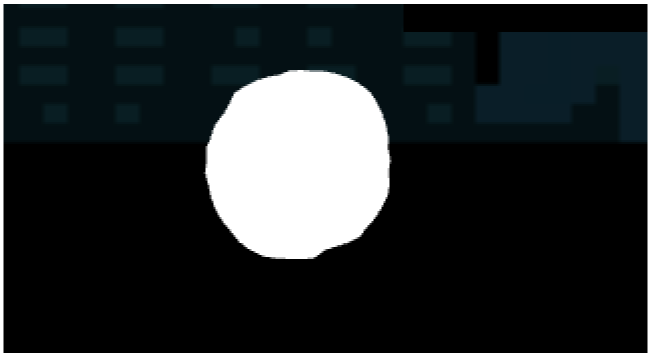
```
  }

  drawWorld() {
    image(bgImg, 0, 0, this.w, this.h);
```

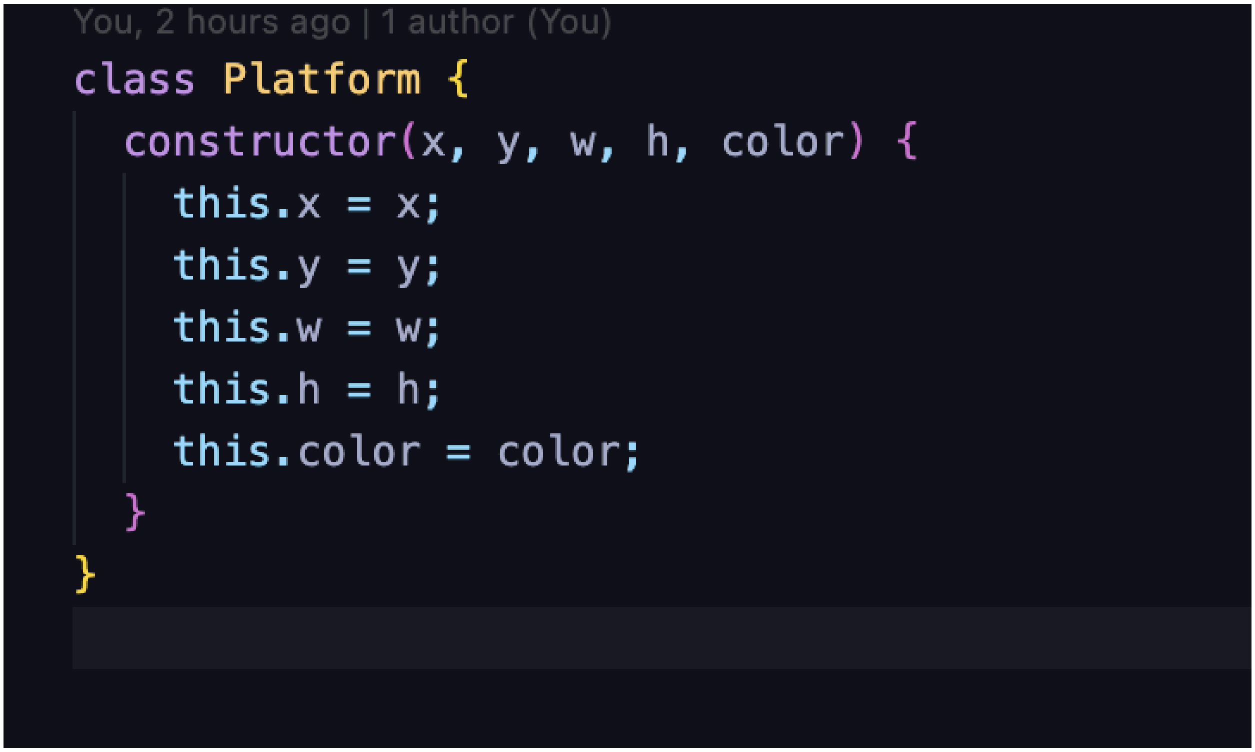


```
class LevelLoader {
  static fromLevelsJson(img, allData, i
    return new WorldLevel(allData.levels[index], img);
  }
}
```

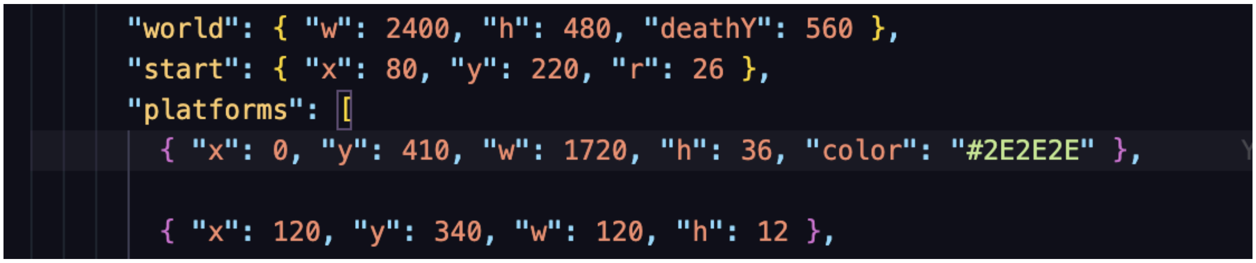


```
    loadLevel(levelIndex);
  }

  function loadLevel(i) {
    level = LevelLoader.fromLevelsJson(img, allLevelsData, i);

    player = new BlobPlayer();
```

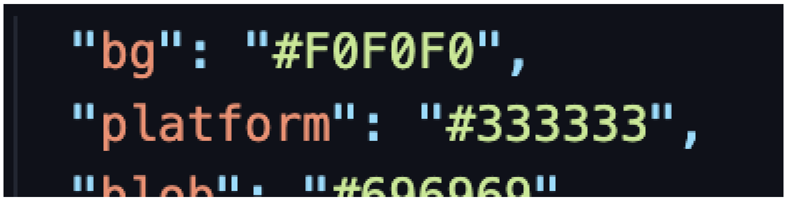




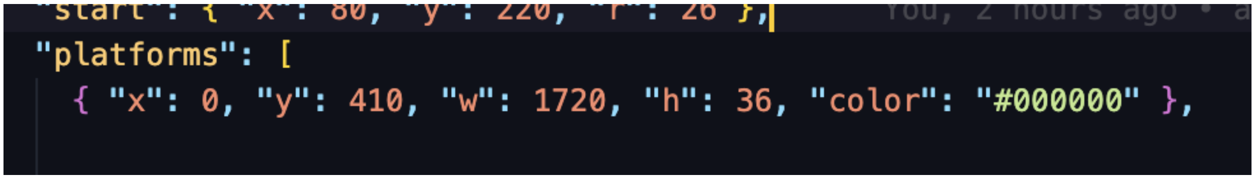
```
You, 2 hours ago | 1 author (You)
class Platform {
  constructor(x, y, w, h, color) {
    this.x = x;
    this.y = y;
    this.w = w;
    this.h = h;
    this.color = color;
  }
}
```



```
"world": { "w": 2400, "h": 480, "deathY": 560 },
"start": { "x": 80, "y": 220, "r": 26 },
"platforms": [
  { "x": 0, "y": 410, "w": 1720, "h": 36, "color": "#2E2E2E" },

  { "x": 120, "y": 340, "w": 120, "h": 12 },
```



```
"bg": "#F0F0F0",
"platform": "#333333",
"blob": "#696969"
```



```
"start": { "x": 80, "y": 220, "r": 26 },        You, 2 hours ago • a
"platforms": [
  { "x": 0, "y": 410, "w": 1720, "h": 36, "color": "#000000" },
```

Goal of Work Session

The goal of this work session was to alter existing code in order to create an experience as the player moves around the space. I created a short experience with an element of horror at the end where the player is met with a ghost, and falls into eternal darkness. I did the following tasks

- I drew a background using pixilart by myself
- I used AI to help assist me in changing that image into the background
- I then made some changes to the platform size, position, and color
- I also changed the blob color.
- Throughout this process, I debugged both by myself and at times with the assistance of the AI.

Tools, Resources, or Inputs Used

- ChatGPT 5.2

GenAI Documentation

Date Used:

Tool Disclosure: ChatGPT 5.2

Purpose of Use: I needed help understanding certain parts of the code and suggestions for changing the visuals.

Summary of Interaction: I asked the AI to help me understand how I can change the background to be an image instead of the color, since the code I was modifying was not my own, I was having some trouble figuring out where certain things went.

Human Decision Point(s): At times I had to ask it for a more simpler solution, as it was overcomplicating its response. I also noticed that the AI was using concepts that I wasn't as familiar with, so I had to ask it to simplify it further so it made it easier to debug for myself.

Integrity & Verification Note: I made sure to double check my code and make sure I understood where the changes were being made, and what they were doing. Instead of pasting the whole output, I also kept checking if each of my changes were working by running the live server.

Scope of GenAI Use: For my background drawing I had done the whole drawing myself and by hand. I also implemented certain parts of the code by myself such as adding the color field for setting the color of the ground platform to be different from the rest.

Limitations or Misfires: It feels like there were multiple times where I had to ask the AI for various iterations and had to take parts here and there from the code within the outputs. The challenge it was having was to not implement it in a simple manner, and also was creating a lot of unnecessary fail edge cases and jargon. For example, initially it said to have either the background image if a condition is met, or if it fails to select the color instead. I had to fix this as I wanted the image to be the sole background without any condition.

Summary of Process (Human + Tool)

I had to go back and forth with the AI as I found it quite challenging to implement its suggestions and I wanted to make sure I understood where the changes were being made so it was easier to debug later on. For example, when I had first made my changes I realized the image was not sized properly based on what the AI had told me to do, I then looked deeper into the code and set the sizing to be correct so that the full image was visible instead. I had a few moments like this where I had to clarify certain parts or things that I had tried on my own didn't work so I needed further assistance.

Decision Points & Trade-offs

I had to make various design decisions throughout this process, one change I had to make once I had added the background. I noticed the based platform did not align with the ground that I had drawn for the background, this made it confusing for me when playing it as I noticed it threw me off and I kept forgetting there was a whole there, as a result I ended making the entire bottom platform one long flat platform that the user could continue to land on, aside from the very last moment where they fall (a part of the experience)

Verification & Judgement

Most of my decisions were changed through playtesting, certain things didn't stand out to me until after I played the game. As mentioned before, the length of the ground platform and making sure it aligns with the background I had drawn. Another example would be the color choices for platforms. I changed it to be black to match the background but noticed the upper platforms made it harder to see, as a result I changed only the bottom platform to be black and the rest to be slightly more visible. I also noticed the text on the top left hand side was black and hard to see, so I changed it to white to make it more visible. All these changes were noticed when testing the game out.

Limitations, Dead Ends, or Open Questions

This was my first time using AI to help assist me with writing the code and I think that I found it quite challenging. I could have spent most of that time learning the concepts myself rather than trying to get various outputs.

Appendix

BlobPlayer.js

JavaScript

Camera2D.js

JavaScript

index.html

File

jsconfig.json

File

LevelLoader.js

JavaScript

levels.json

File

Platform.js

JavaScript

sketch.js

JavaScript

style.css

File

WorldLevel.js

JavaScript

can you show me at where in the code I can make changes to the background if i wish to make it an image(background_sidequest_w5.png)?

Thought for 5s

Yep — **your background is currently set in `WorldLevel.drawWorld()`** with this line:

```
background(this.theme.bg);
```

That’s inside `WorldLevel.js`.

WorldLevel

Option A (quickest): Hard-code a background image in `WorldLevel.js`

1. Add a global for the image (top of file or near top of project):

```
let bgImg;
```

2. Load it in `preload()` inside `sketch.js`:

```
function preload() {
 allLevelsData = loadJSON("levels.json");
 bgImg = loadImage("background_sidequest_w5.png");
}
```

(`preload()` already exists in `sketch.js`.)

sketch

3. Replace the background line in `WorldLevel.drawWorld()`:

```
drawWorld() {
 if (bgImg) {
  image(bgImg, 0, 0, this.w, this.h);   // draw image as the world background
 } else {
  background(this.theme.bg);
 }


 push();
 rectMode(CORNER);
 noStroke();
 fill(this.theme.platform);
 for (const p of this.platforms) rect(p.x, p.y, p.w, p.h);
 pop();
}
```

This edits the exact spot where your background is drawn.

WorldLevel

> Make sure `background_sidequest_w5.png` is in the same folder as `index.html` (or adjust the path).

Option B (clean / data-driven): Set the background image per-level via `levels.json`

Right now, `levels.json` only has a color like `"bg": "#F0F0F0"` in the theme.

levels

You can add something like:

```
"theme": {
 "bg": "#F0F0F0",
 "bgImage": "background_sidequest_w5.png",
 "platform": "#C8C8C8",
 "blob": "#1478FF"
}
```

Then in `WorldLevel.js`, store `bgImage` and draw it (you'll still load it with `loadImage(...)` somewhere, usually in `preload()`).

If you tell me which option you want (A quick hard-code vs B per-level in JSON), I can paste the exact final code changes for just those files.

Sources

is there another way to do this?

Yes — a few other clean ways, depending on what you want the background to *behave* like.

1) CSS background behind the canvas (no p5 `image()` call)

If you just want a static background and your canvas sits on top, you can set the background on the **canvas element** (or its container) in `style.css`.

Example:

```
/* style.css */
canvas {
 background: url("background_sidequest_w5.png") no-repeat center center;
 background-size: cover; /* or contain */
}
```

Pros: super simple, no loading in JS.
Cons: it's "UI background" — it won't move with your camera/world if you want parallax or world-scrolling.

---

2) Set the canvas DOM style from JS (still no p5 drawing)

In `setup()` (in `sketch.js`), right after `createCanvas(...)`:

```
const c = createCanvas(W, H);
c.elt.style.backgroundImage = 'url("background_sidequest_w5.png")';
c.elt.style.backgroundSize = "cover";
c.elt.style.backgroundPosition = "center";
```

Same pros/cons as CSS, just controlled in code.

3) “One-time load” in `setup()` instead of `preload()`

If you don’t want to touch `preload()`, you can load the image once in `setup()`:

```
let bgImg;

function setup() {
 // ...
 bgImg = loadImage("background_sidequest_w5.png");
}
```

Then in `drawWorld()` you still do `image(bgImg, ...)` once it’s ready.

Pros: avoids preload workflow.
Cons: the image may pop in a moment later (first frames might draw the fallback color).

4) Draw the background in screen-space (parallax / not affected by camera)

If your camera transforms the world (translate/scale), and you want the background to *stay fixed* (like a skybox), draw it **before** you apply camera transforms. That means: draw the image in `draw()` in `sketch.js` (or wherever you’re not inside the camera transform), then draw the world after.

Pros: perfect for “static sky” effect.
Cons: requires knowing where your camera transform is applied in your draw loop.

Quick question (so I recommend the best one):

Do you want the background to **scroll with the level** (like it’s part of the world), or **stay fixed behind everything** (like a wallpaper)?

I am unsure as to where the image is being used as the background, look back at where the background is defined, and show me how to implement it based on that

Thought for 23s

Yep — I checked your files, and the background is being set **right here**:

✅ Where the background is defined

In **`/mnt/data/WorldLevel.js`**, inside `drawWorld()`:

```
drawWorld() {

 background(this.theme.bg);

 ...

}
```

So your “background” right now is just a **solid color** from `this.theme.bg`.

How to switch that exact spot to an image (`background_sidequest_w5.png`)

Step 1) Load the image in `preload()` (in `sketch.js`)

Your `preload()` currently only loads `levels.json`.

Add one line:

```
let bgImg;

function preload() {
 allLevelsData = loadJSON("levels.json");
 bgImg = loadImage("background_sidequest_w5.png");
}
```

> This assumes `background_sidequest_w5.png` is beside `index.html` (same place as `levels.json`).

Step 2) Replace the `background(this.theme.bg)` line in `WorldLevel.drawWorld()`

In **`WorldLevel.js`**, change `drawWorld()` to:

```
drawWorld() {
 // draw image background if available, otherwise fallback to theme color
 if (typeof bgImg !== "undefined" && bgImg) {
   image(bgImg, 0, 0, this.w, this.h);
 } else {
   background(this.theme.bg);
 }

 push();
 rectMode(CORNER);
 noStroke();
 fill(this.theme.platform);
```

for (const p of this.platforms) rect(p.x, p.y, p.w, p.h);

pop();

}

That's it — you're still using the **same background "location"** in the code (`drawWorld()`), you're just swapping the **fill method** from a color to an image.

Important note (why this works with your camera)

In `sketch.js`, you do:

cam.begin();

level.drawWorld();

player.draw(...);

cam.end();

So whatever you draw in `drawWorld()` is part of the **world** (it scrolls with the camera). Your background image will scroll too, because it's being drawn in world space.

If you want the background to **stay fixed** while the world scrolls (like a sky), tell me and I'll show the tiny change for that too.

i dont want a fall back, i just want it to change the background to the image

Thought for 15s

Got it — since your background is **defined right now in `WorldLevel.drawWorld()`** as:

background(this.theme.bg);

…we'll replace that with drawing your image **in the exact same place**.

---

1) `sketch.js`: load the background image in `preload()`

Add a global variable near the top (with your other `let`s):

```
let bgImg;
```

Then update `preload()` to load the image:

```
function preload() {
 allLevelsData = loadJSON("levels.json");
 bgImg = loadImage("background_sidequest_w5.png");
}
```

> Make sure `background_sidequest_w5.png` is in the same folder as `index.html` and `levels.json`.

---

2) `WorldLevel.js`: replace the background color with the image

In **`drawWorld()`**, replace:

```
background(this.theme.bg);
```

with:

```
image(bgImg, 0, 0, this.w, this.h);
```

So your full function becomes:

```
drawWorld() {
 image(bgImg, 0, 0, this.w, this.h);

 push();
 rectMode(CORNER);
 noStroke();

 fill(this.theme.platform);


 for (const p of this.platforms) rect(p.x, p.y, p.w, p.h);

 pop();

}
```

That’s all you need — no fallback, it will always draw the image as the background.

One important note

Because `drawWorld()` is called **inside** **`cam.begin()`** **/** **`cam.end()`**, this background image will be drawn in **world space** (it will move with the camera and cover the whole level using `this.w` and `this.h`). If you instead want it to stay fixed on the screen, tell me and I’ll show the 2-line change for that.

it seems my image height does not align with the canvas height, what is the canvas height?

Thought for 6s

Your canvas height is **480 pixels**.

It’s defined at the top of **`sketch.js`**:

```
const VIEW_H = 480;

...

createCanvas(VIEW_W, VIEW_H);
```

So:

- **Canvas width:** 800
- **Canvas height:** 480

there is still an issue with the code, it looks odd, how can I change it so that my image aligns with the left corner, it is starting towards the right

Thought for 40s

What you’re seeing makes sense with your camera setup.

Right now your background image is drawn at **world position (0,0)** *inside* `cam.begin()` / `cam.end()`, and your camera does:

```
translate(-this.x, -this.y);
```

So if `cam.x` is, say, 300, the left edge of the screen is actually showing **world x = 300**, which makes the background *look like it “starts to the right”*. (It’s just being viewed through a shifted camera.)

Fix: draw the background at the camera’s world position

This pins the image to the **top-left of the canvas** (screen space), even though you’re inside the camera transform.

In `WorldLevel.js`, change your background line to:

```
image(bgImg, cam.x, cam.y, width, height);
```

So `drawWorld()` becomes like:

```
drawWorld() {
 image(bgImg, cam.x, cam.y, width, height);  // <- aligns to canvas top-left


 push();
 rectMode(CORNER);
 noStroke();
 fill(this.theme.platform);
 for (const p of this.platforms) rect(p.x, p.y, p.w, p.h);
 pop();
}
```

Why this works: inside the camera transform, screenX = worldX − cam.x.
If you draw at worldX = cam.x, it lands at screenX = 0 (left edge).

Even cleaner alternative (same result)

Draw the background **before** `cam.begin()` in `sketch.js`:

```
image(bgImg, 0, 0, width, height);


cam.begin();
level.drawWorld();
player.draw(level.theme.blob);
cam.end();
```

If you do this, remove the `image(bgImg, ...)` from `drawWorld()` so it isn't drawn twice.

Either approach will make it align with the left corner; the first one is the smallest change to your current setup.

what if i wish to change the platform and blob color?

Thought for 15s

Yep — both of those colors are already **data-driven** through `theme`.

Where platform color is applied

In **`WorldLevel.js`** → **`drawWorld()`**:

```
fill(this.theme.platform);
```

That's what colors the platforms.

Where blob color is applied

In **`sketch.js`** → **`draw()`**:

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

That `level.theme.blob` is passed into `BlobPlayer.draw(colHex)` and used to fill the blob.

The easiest way to change them: edit `levels.json`

Each level can define:

"theme": {

"bg": "#F0F0F0",

"platform": "#C8C8C8",

"blob": "#1478FF"

}

So just change the hex codes:

- `platform` → platform color
- `blob` → blob color

Example:

"theme": {

"bg": "#F0F0F0",

"platform": "#3B3B3B",

"blob": "#FF4D6D"

}

That will apply automatically when the level loads.

If you want to change the *default* colors for any level that doesn't specify them

In **`WorldLevel.js` constructor**, the defaults are here:

{ bg: "#F0F0F0", platform: "#C8C8C8", blob: "#1478FF" }

Change those hex values and any level missing a `theme` field will inherit them.

If you want, paste the `theme` block from your `levels.json` and tell me what vibe you want (dark, neon, pastel, etc.) and I'll give you a few good color combos.

"platforms": [{ "x": 0, "y": 424, "w": 640, "h": 36 }, { "x": 120, "y": 354, "w": 120, "h": 12 }, { "x": 300, "y": 304, "w": 90, "h": 12 }, { "x": 440, "y": 244, "w": 130, "h": 12 }, { "x": 520, "y": 354, "w": 90, "h": 12 }, { "x": 570, "y": 164, "w": 30, "h": 12 }, { "x": 720, "y": 424, "w": 420, "h": 36 }, { "x": 760, "y": 364, "w": 110, "h": 12 }, { "x": 930, "y": 324, "w": 110, "h": 12 }, { "x": 1090, "y": 284, "w": 110, "h": 12 }, { "x": 1200, "y": 424, "w": 520, "h": 36 }, { "x": 1400, "y": 344, "w": 140, "h": 12 } can you change the platform so the base main platform is slightly raised, and can you make sure there are no holes that fall below the canvas? I just want platforms for some interaction and to jump on top of

Thought for 10s

Yep — the "holes" are happening because your **ground is split into 3 chunks** with gaps:

- ends at `x=640`, next starts at `x=720` → gap **640–720**
- ends at `x=1140`, next starts at `x=1200` → gap **1140–1200**

To **raise the main base platform** *and* ensure there are **no gaps to fall through**, the simplest fix is to make the ground **one continuous platform** and (optionally) shift the other platforms up by the same amount so everything keeps the same relative spacing.

Here's an updated `platforms` array that:

- raises the ground from `y=424` to `y=410` (14px up)
- removes the gaps by making the ground continuous from `x=0` to `x=1720`
- shifts your jump platforms up by 14px too (so the "feel" stays the same)

"platforms": [

{ "x": 0, "y": 410, "w": 1720, "h": 36 },

{ "x": 120, "y": 340, "w": 120, "h": 12 },

```
 { "x": 300, "y": 290, "w": 90,  "h": 12 },
 { "x": 440, "y": 230, "w": 130, "h": 12 },
 { "x": 520, "y": 340, "w": 90,  "h": 12 },
 { "x": 570, "y": 150, "w": 30,  "h": 12 },

 { "x": 760, "y": 350, "w": 110, "h": 12 },
 { "x": 930, "y": 310, "w": 110, "h": 12 },
 { "x": 1090, "y": 270, "w": 110, "h": 12 },

 { "x": 1400, "y": 330, "w": 140, "h": 12 }
]
```

If you want the ground to cover **your entire level width** (in case the level is wider than 1720), tell me what the level width is set to in your level JSON (or paste that part) and I'll match it exactly.

how can i change the color of just the ground platform?

Right now **every platform uses the same fill color** because `WorldLevel.drawWorld()` does this once:

```
fill(this.theme.platform);
for (const p of this.platforms) rect(p.x, p.y, p.w, p.h);
```

So to color **just the ground**, you need a way to **identify the ground platform** and then set a different `fill()` just for that one.

Here are the 2 cleanest ways:

Option 1 (best): Give the ground its own color in `levels.json`

1) Add a color field to the ground platform only

Example (ground is your big base platform):

```
{ "x": 0, "y": 410, "w": 1720, "h": 36, "color": "#2E2E2E" }
```

Leave the other platforms without `"color"`.

2) Update `WorldLevel.drawWorld()` to use per-platform color

In **`WorldLevel.js`**, change your platform drawing loop to:

```
for (const p of this.platforms) {
 fill(p.color ?? this.theme.platform);
 rect(p.x, p.y, p.w, p.h);
}
```

Now only the ground (the one with `"color"`) will be different.

Option 2: Detect the ground by size/position (no JSON changes)

If your ground is always the **tallest** platform (like `h === 36`), do:

```
for (const p of this.platforms) {
 const isGround = p.h >= 30; // or p.h === 36
 fill(isGround ? "#2E2E2E" : this.theme.platform);
 rect(p.x, p.y, p.w, p.h);
}
```

This works, but it’s a bit “hacky” if you later add other thick platforms.

If you paste your current ground platform line (the `{ "x": ..., "y": ..., "w": ..., "h": ... }` one), I’ll show you exactly how to add the `"color"` field for it.

thank you

Anytime 🙂 If you run into another weird visual thing (camera, scaling, platforms, collisions), paste the snippet and I’ll point to the exact line to tweak.