

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 an advanced camera experience that triggered and/or represented an emotional response.

Goal of Work Session: During this work session, I focused on making changes to this game world by making independent and GenAI changes to the camera experience by slowing down the pace and motion to represent tranquility.

GenAI tools (if used):

- Chatbot on Visual Studio Code 6

GenAI Documentation

Date Used: February 12th, 2026

Tool Disclosure: Chatbot on Visual Studio Code 6

Purpose of Use: To help add code that was too complex for my knowing.

Summary of Interaction:

- Visual code helped fill in my let breathe function halfway through
- Asked the Visual Studio Chatbot to make an effect that zooms into the other coloured squares when the small blue square is located within them
- Asked the Visual Studio Chatbot to make an out of bounds film over the camera

Human Decision Point(s):

- I had to overrule one AI decisions
 - In iteration three the Visual Studio Chatbot made changes to the code that caused an error and when I ran it the screen was just white

Integrity & Verification Note:

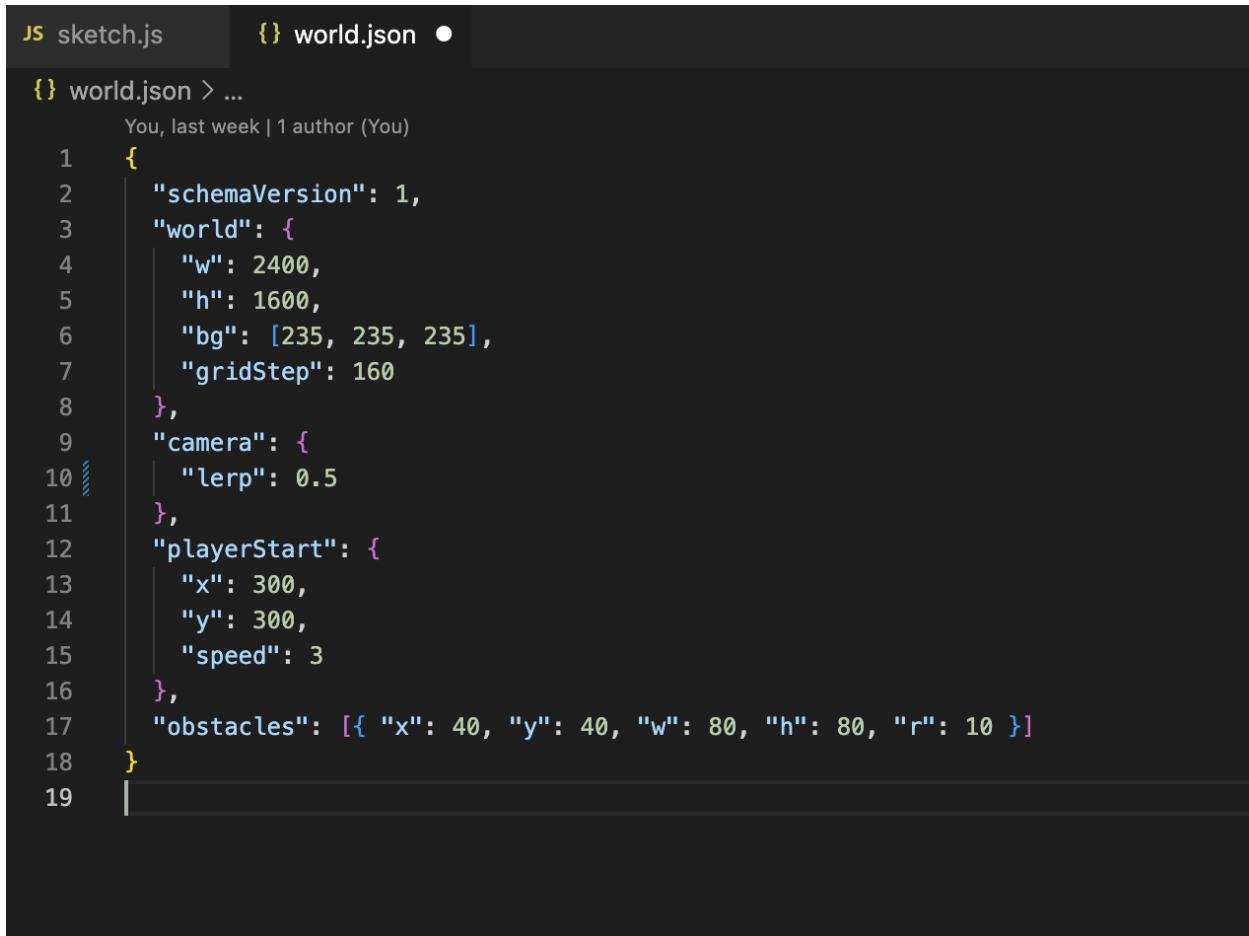
All decisions were verified using:

- 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*:

- Make the original change the lerp so it is slower and more calming
- Make the original changes to the push and pop section



The screenshot shows a code editor window with a dark theme. The title bar indicates the file is named "world.json". The code itself is a JSON configuration object with the following structure:

```
{} world.json ●
{} world.json > ...
You, last week | 1 author (You)
1 {
2   "schemaVersion": 1,
3   "world": {
4     "w": 2400,
5     "h": 1600,
6     "bg": [235, 235, 235],
7     "gridStep": 160
8   },
9   "camera": {
10    "lerp": 0.5
11  },
12   "playerStart": {
13     "x": 300,
14     "y": 300,
15     "speed": 3
16   },
17   "obstacles": [{ "x": 40, "y": 40, "w": 80, "h": 80, "r": 10 }]
18 }
19 |
```

```

    targetX = constrain(targetX, 0, maxCamX);

    // Smooth follow using the JSON knob
    const camLerp = level.camLerp; // ← data-driven now
    camX = lerp(camX, targetX, camLerp);
    camY = lerp(camY, targetY, camLerp);

    level.drawBackground();

    push();
    translate(width / 2, height / 2);
    scale(breathe);
    translate(-camX - width / 2, -camY - height / 2);
    level.drawWorld();
    player.draw();
    pop();

    level.drawHUD(player, camX, camY);
}

```

Limitations or Misfires:

There were two misfires with AI's addition to my codes

1. The chatbot caused the screen to be white and nothing else was being drawn following the second iteration
2. The chatbot did not make significant changes to the extreme zoom in function I wanted to add that happens when the square is within another square

Summary of Process (Human + Tool)

Four iterations with GenAI:

- Added code for the rest of my let statement (code screenshots below)
- Asked AI to code a slow and extreme zoom in function when it is within another square but the code had errors and nothing but white was being drawn (code screenshots below)
- Asked AI to try again without making the code just white and the visual difference was very minimal, the zoom in didn't work as hoped (code screenshots below)

- Asked AI to add an out of bounds film that fades in and out slowly if the player is near the edges of the world, which it did successfully (code screenshots below)

Decision Points & Trade-offs

I chose to make the breathe function independently to express calmness while moving around. I then made the choice to give AI the opportunity to do an extreme zoom function that would interrupt the wave breathing briefly. However, I wasn't happy with how that fit in and minor changes were made so I had to move on from that idea and stop prompting the AI to modify the code for that specific function.

Verification & Judgement

I evaluated the appropriateness of my changes by:

- Reviewing notes from CS105 and CS106
- Reviewing Class 5 lecture by David

Limitations, Dead Ends, or Open Questions

- All issues were resolved successfully by the chatbot but the extreme zoom in function that I prompted in iteration 2 did not come out as expected and it seemed like it wasn't very achievable since I had asked twice

Appendix - Visual Studio Code 6 Chatbot

#1 Visual code helped me fill the rest automatically as I was typing the breathe formula

```
JS sketch.js > draw
32  function setup() {
33    camX = player.x - width / 2;
34    camY = player.y - height / 2;
35  }
36
37  function draw() {
38    player.updateInput();
39
40    // Keep player inside world
41    player.x = constrain(player.x, 0, level.w);
42    player.y = constrain(player.y, 0, level.h);
43
44    // Target camera (center on player)
45    let targetX = player.x - width / 2;
46    let targetY = player.y - height / 2;
47
48    // Let's add a breathing effect
49    let breathe = scale(1 + sin(frameCount * 0.02) * 0.1); // breathing effect
50
51    // Clamp target camera safely
52    const maxCamX = max(0, level.w - width);
53    const maxCamY = max(0, level.h - height);
54    targetX = constrain(targetX, 0, maxCamX);
55    targetY = constrain(targetY, 0, maxCamY);
56
57    // Smooth follow using the JSON knob
58    const camLerp = level.camLerp; // ← data-driven now
59    camX = lerp(camX, targetX, camLerp);
60    camY = lerp(camY, targetY, camLerp);
61
62    level.drawBackground();
63
64    push();
65    translate(width / 2, height / 2);
66    scale(breathe);
67    translate(-camX - width / 2, -camY - height / 2);
68    level.drawWorld();
69    player.draw();
70    pop();
71
72
73
74
75
76
77
78
```

#2 Could you make adjustments to the code please so that when the small square is within the big square the camera slowly zooms in on the big square. When the player exists the big square it can return back to the normal rhythm

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

Summarized conversation history

Read WorldLevel.js, lines 40 to 54

Summarized conversation history

To implement the camera zoom functionality based on the player's position relative to the big square, we can make the following adjustments to the WorldLevel class in WorldLevel.js.

1. Add properties for the big square's dimensions and position.
2. Modify the camera zoom logic in the drawWorld method.
3. Implement a smooth zoom effect when the player is inside the big square.

Here's how the updated code will look:

```
class WorldLevel {

  constructor(json) {

    this.schemaVersion = json.schemaVersion ?? 1;

    this.w = json.world?.w ?? 2400;

    this.h = json.world?.h ?? 1600;

    this.bg = json.world?.bg ?? [235, 235, 235];

    this.gridStep = json.world?.gridStep ?? 160;

    this.obstacles = json.obstacles ?? [];

    // NEW: camera tuning knob from JSON (data-driven)

    this.camLerp = json.camera?.lerp ?? 0.12;

    // NEW: Big square properties

    this.bigSquare = {

      x: 800, // X position of the big square

      y: 600, // Y position of the big square
```

```
size: 400 // Size of the big square

};

this.cameraZoom = 1; // Initial camera zoom level

}

drawWorld(player) {

// Check if the player is within the big square

const inBigSquare = (

player.x > this.bigSquare.x &&

player.x < this.bigSquare.x + this.bigSquare.size &&

player.y > this.bigSquare.y &&

player.y < this.bigSquare.y + this.bigSquare.size

);

// Adjust camera zoom based on player's position

if (inBigSquare) {

this.cameraZoom += (0.01 - this.cameraZoom) * this.camLerp; // Zoom in slowly

} else {

this.cameraZoom += (1 - this.cameraZoom) * this.camLerp; // Return to normal zoom

}

// Apply camera zoom

scale(this.cameraZoom);

noStroke();
```

```
fill(this.bg[0], this.bg[1], this.bg[2]);

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, 190, 210);

for (const o of this.obstacles) rect(o.x, o.y, o.w, o.h, o.r ?? 0);

// Draw the big square for visualization

fill(255, 0, 0, 100); // Semi-transparent red

rect(this.bigSquare.x, this.bigSquare.y, this.bigSquare.size, this.bigSquare.size);

}

drawHUD(player, camX, camY) {

noStroke();

fill(20);

text("Example 4 - JSON world + smooth camera (lerp).", 12, 20);

text(

"camLerp(JSON) : " +

this.camLerp +

" Player: " +

(player.x | 0) +

```

```

    ", " +
    (player.y | 0) +
    " Cam: " +
    (camX | 0) +
    ", " +
    (camY | 0),
    12,
    40,
  );
}

}

```

Key Changes:

- **Big Square Properties:** Added a `bigSquare` object to define its position and size.
- **Camera Zoom Logic:** Adjusted the `drawWorld` method to check if the player is within the big square and modify the `cameraZoom` accordingly.
- **Scale Application:** Used the `scale()` function to apply the zoom effect.

I'll now make these changes in the `WorldLevel.js` file.

Edited

`WorldLevel.js+80-1`

Summarized conversation history

The adjustments have been successfully made to the `WorldLevel.js` file. The camera will now slowly zoom in on the big square when the player is within it and return to normal when the player exits.

If you need any further modifications or have additional requests, feel free to ask!

#3 Please try that again, those changes and addition to the code made the entire screen white and I couldn't see anything

I'll open `WorldLevel.js` to inspect the current camera and drawing logic before reapplying a safe zoom.

I'll open `sketch.js` to see how camera and player are managed.

I'll update both WorldLevel.js (add big-square and zoom state + helper) and sketch.js (fix breathing calc and apply smooth zoom) in one patch.

Edited

WorldLevel.js+17-0

I'll update the breathing calculation and add smooth zoom math in sketch.js.

Edited

sketch.js+9-2

Done — I fixed the breathing bug and re-added a safe, smooth zoom.

Changes:

- **Updated** sketch.js: replaced the erroneous scale(...) call used to compute `breathe` with `breatheFactor`, added smooth zoom lerp (target 1.2 when inside big square), and applied `totalScale = cameraZoom * breatheFactor`.
- **Updated** WorldLevel.js: added bigSquare defaults, `cameraZoom` state, a visual big-square rect, and `isPlayerInBigSquare(player)` helper.

What I fixed that caused the white screen:

- The previous code called `scale(...)` while computing `breathe`, which mutated the canvas transform unexpectedly. Now transforms are only applied inside the `push()/pop()` with a computed numeric scale.

If you want different zoom strength or speed, tell me the desired target scale (e.g. 1.1–1.5) or lerp speed and I'll tune it.

```

46 function draw() {
47   // Keep player inside world
48   player.x = constrain(player.x, 0, level.w);
49   player.y = constrain(player.y, 0, level.h);
50
51   // Target camera (center on player)
52   let targetX = player.x - width / 2;
53   let targetY = player.y - height / 2;
54
55   // breathing factor (no canvas transform here)
56   const breatheFactor = 1 + sin(frameCount * 0.02) * 0.1;
57
58   // Clamp target camera safely
59   const maxCamX = max(0, level.w - width);
60   const maxCamY = max(0, level.h - height);
61   targetX = constrain(targetX, 0, maxCamX);
62   targetY = constrain(targetY, 0, maxCamY);
63
64   // Smooth follow using the JSON knob
65   const camLerp = level.camLerp; // ~ data-driven now
66   const camX = lerp(camX, targetX, camLerp);
67   const camY = lerp(camY, targetY, camLerp);
68
69   level.drawBackground();
70
71   push();
72   translate(width / 2, height / 2);
73   scale(breathe);
74   translate(-camX - width / 2, -camY - height / 2);
75 }

```



```

46 function draw() {
47   // Keep player inside world
48   player.x = constrain(player.x, 0, level.w);
49   player.y = constrain(player.y, 0, level.h);
50
51   // Target camera (center on player)
52   let targetX = player.x - width / 2;
53   let targetY = player.y - height / 2;
54
55   // breathing factor (no canvas transform here)
56   const breatheFactor = 1 + sin(frameCount * 0.02) * 0.1;
57
58   // Clamp target camera safely
59   const maxCamX = max(0, level.w - width);
60   const maxCamY = max(0, level.h - height);
61   targetX = constrain(targetX, 0, maxCamX);
62   targetY = constrain(targetY, 0, maxCamY);
63
64   // Smooth follow using the JSON knob
65   const camLerp = level.camLerp; // ~ data-driven now
66   const camX = lerp(camX, targetX, camLerp);
67   const camY = lerp(camY, targetY, camLerp);
68
69   // Smooth zoom: when player is inside the big square, zoom in
70   const inBig = level.isPlayingInBigSquare(player);
71   const zoomTarget = inBig ? 1.2 : 1.0;
72   level.cameraZoom = lerp(level.cameraZoom ?? 1, zoomTarget, 0.1);
73
74   const totalScale = level.cameraZoom * breatheFactor;
75
76   level.drawBackground();
77
78   push();
79   translate(width / 2, height / 2);
80   scale(totalScale);
81   translate(-camX - width / 2, -camY - height / 2);
82 }

```

JS WorldLevel.js > CAM.js

```

1 class WorldLevel { You, 2 hours ago • added files ...
2   constructor(json) {
3     // Big square and camera zoom state
4     this.bigSquare = json.bigSquare ?? { x: 800, y: 600, size: 400 };
5     this.cameraZoom = 1;
6   }
7
8   drawBackground() {
9     background(220);
10  }
11
12  drawWorld() {
13    noStroke();
14    fill(this.bg[0], this.bg[1], this.bg[2]);
15    rect(0, 0, this.w, this.h);
16
17    stroke(245);
18    for (let x = 0; x <= this.w; x += this.gridStep) line(x, 0, x, this.h);
19    for (let y = 0; y <= this.h; y += this.gridStep) line(0, y, this.w, y);
20
21    noStroke();
22    fill(170, 190, 210);
23    for (const o of this.obstacles) rect(o.x, o.y, o.w, o.h, o.r ?? 0);
24
25    // Draw big square for visualization (semi-transparent)
26    noStroke();
27    fill(255, 0, 0, 80);
28    rect(this.bigSquare.x, this.bigSquare.y, this.bigSquare.size, this.bigSquare.size);
29  }
30
31  isPlayerInBigSquare(player) {
32    return (
33      player.x > this.bigSquare.x &&
34      player.x < this.bigSquare.x + this.bigSquare.size &&
35      player.y > this.bigSquare.y &&
36      player.y < this.bigSquare.y + this.bigSquare.size
37    );
38  }
39}

```

#4 Can you add an out of bounds film that fades in and out slowly if the player is near the edges of the world

I'll add an edge-fade overlay: plan the changes, inspect current WorldLevel.js and sketch.js, then update both files to compute and render a smooth edge film.

Edited

WorldLevel.js+23-0

Edited

sketch.js+10-0

Progress update: I added a smooth edge-fade "film" that appears when the player nears world edges.

What I changed

- WorldLevel.js
 - Added edge-film parameters: edgeFadeDistance, edgeFadeMaxAlpha, edgeLerp, edgeFilmAlpha.
 - Added computeEdgeTargetAlpha(player) and updateEdgeFilm(player) helpers to compute and smooth the overlay alpha.
- sketch.js
 - Each frame calls level.updateEdgeFilm(player) and renders a full-screen overlay rect(0,0,width,height) with fill(0, alpha) to create the fade effect.

Tuning

- Defaults: edgeFadeDistance = 160px, edgeFadeMaxAlpha = 160 (out of 255), edgeLerp = 0.08.
- Tell me if you want the film darker, the fade to start sooner/later, or the fade smoothing faster/slower — I can adjust the defaults or expose them in world.json.

-
-
-

GPT-5 mini • 1x

```
JS WorldLevel.js > ⌂ WorldLevel > ⌂ drawWorld
  1  class WorldLevel {
  2    isPlayerInBigSquare(player) {
  3      player.x > this.bigSquare.x &&
  4      player.x < this.bigSquare.x + this.bigSquare.size &&
  5      player.y > this.bigSquare.y &&
  6      player.y < this.bigSquare.y + this.bigSquare.size
  7    );
  8  }
  9
 10
 11 // Compute target alpha (0..edgeFadeMaxAlpha) based on distance to closest world edge
 12 computeEdgeTargetAlpha(player) {
 13   const dl = player.x; // distance to left edge
 14   const dr = this.w - player.x; // right
 15   const dt = player.y; // top
 16   const db = this.h - player.y; // bottom
 17   const minD = min(dl, dr, dt, db);
 18   if (minD >= this.edgeFadeDistance) return 0;
 19   const t = constrain(1 - minD / this.edgeFadeDistance, 0, 1);
 20   return t * this.edgeFadeMaxAlpha;
 21 }
 22
 23 // Smoothly update internal film alpha (call each frame)
 24 updateEdgeFilm(player) {
 25   const target = this.computeEdgeTargetAlpha(player);
 26   this.edgeFilmAlpha = lerp(this.edgeFilmAlpha ?? 0, target, this.edgeLerp);
 27 }
 28
 29 drawHUD(player, camX, camY) {
 30   noStroke();
 31   fill(20);
 32   text("Example 4 - JSON world + smooth camera (lerp)", 12, 20);
 33
 34 class WorldLevel {
 35   constructor(json) {
 36     this.bg = json.world?.bg ?? [235, 235, 235];
 37     this.gridStep = json.world?.gridStep ?? 160;
 38
 39     this.obstacles = json.obstacles ?? [];
 40
 41     // NEW: camera tuning knob from JSON (data-driven)
 42     this.camLerp = json.camera?.lerp ?? 0.12;
 43     // Big square and camera zoom state
 44     this.bigSquare = json.bigSquare ?? { x: 800, y: 600, size: 400 };
 45     this.cameraZoom = 1;
 46
 47     // Edge-film (out-of-bounds) defaults
 48     this.edgeFadeDistance = json.camera?.edgeFadeDistance ?? 160; // px from edge where film starts
 49     this.edgeFadeMaxAlpha = json.camera?.edgeFadeMaxAlpha ?? 160; // max alpha (0-255)
 50     this.edgeLerp = json.camera?.edgeLerp ?? 0.08; // smoothing for fade
 51     this.edgeFilmAlpha = 0; // current alpha state (0-255)
 52   }
 53
 54   drawBackground() {
 55     background(220);
 56   }
 57
 58   drawWorld() {
 59     noStroke();
 60     fill(this.bg[0], this.bg[1], this.bg[2]);
 61     rect(0, 0, this.w, this.h);
 62
 63     stroke(245);
 64     for (let x = 0; x <= this.w; x += this.gridStep) line(x, 0, x, this.h);
 65     for (let y = 0; y <= this.h; y += this.gridStep) line(0, y, this.w, y);
 66   }
 67 }
 68
 69 I'll add a new file
 70 > WorldLevel.js
 71 > render()
 72 > main()
 73 > edit()
 74 > edit()
 75 > progress()
 76 > player()
 77 > What I'm doing
 78 > > >
```

```
0 function draw() {
1   const zoomTarget = lmbg : 1.2 * 1.0,
2   level.cameraZoom = lerp(level.cameraZoom ?? 1, zoomTarget, camLerp);
3   const totalScale = level.cameraZoom * breatheFactor;
4
5   level.drawBackground();
6
7   push();
8   translate(width / 2, height / 2);
9   scale(totalScale);
10  translate(-camX - width / 2, -camY - height / 2);
11  level.drawWorld();
12  player.draw();
13  pop();
14
15  level.drawHUD(player, camX, camY);
16
17 // Update edge-film alpha and draw screen-space overlay if near edges
18 level.updateEdgeFilm(player);
19 if ((level.edgeFilmAlpha ?? 0) > 1) {
20   push();
21   noStroke();
22   fill(0, level.edgeFilmAlpha);
23   rect(0, 0, width, height);
24   pop();
25 }
26 }
```

