Name: Cohen Peters

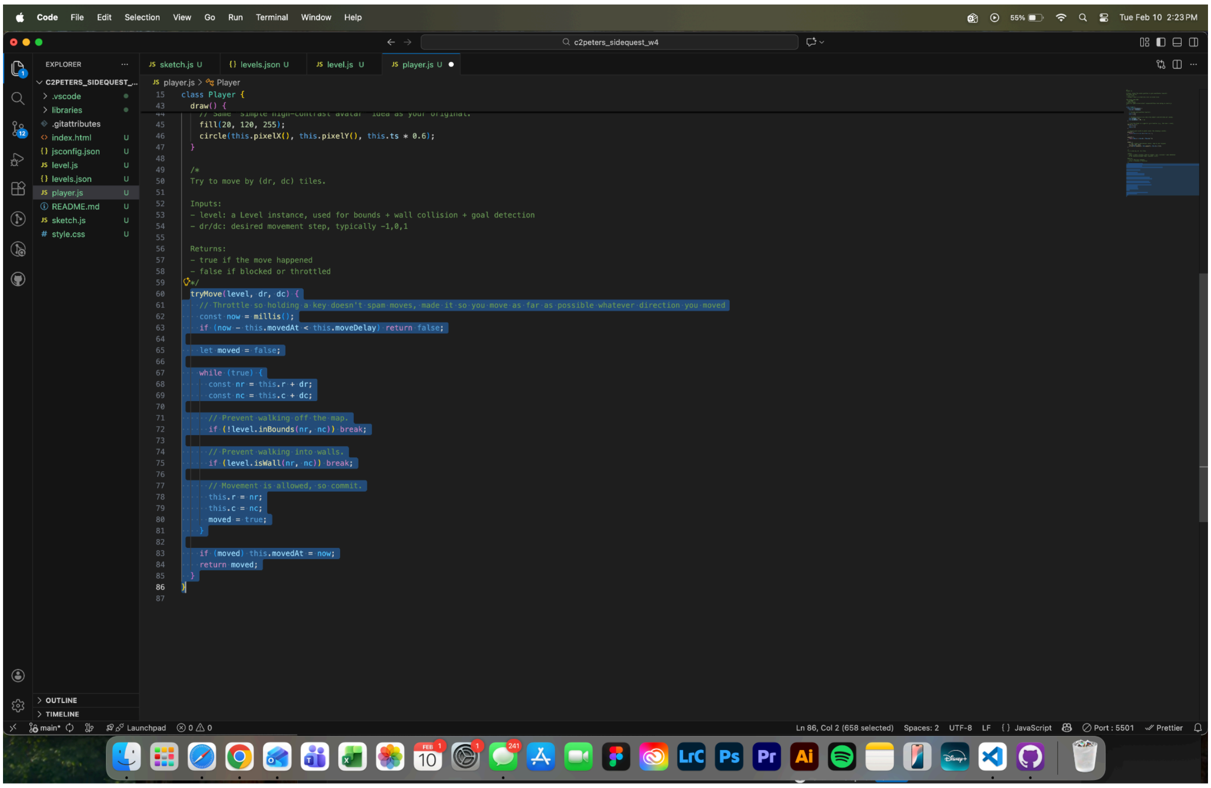
One change I made was adjusting the ball movement so that it slides until it hits a wall with each movement. I did this because it adds a different play style compared to normal maze puzzles.

Goal of work session: The goal was to make a level based game using arrays or JSON data.

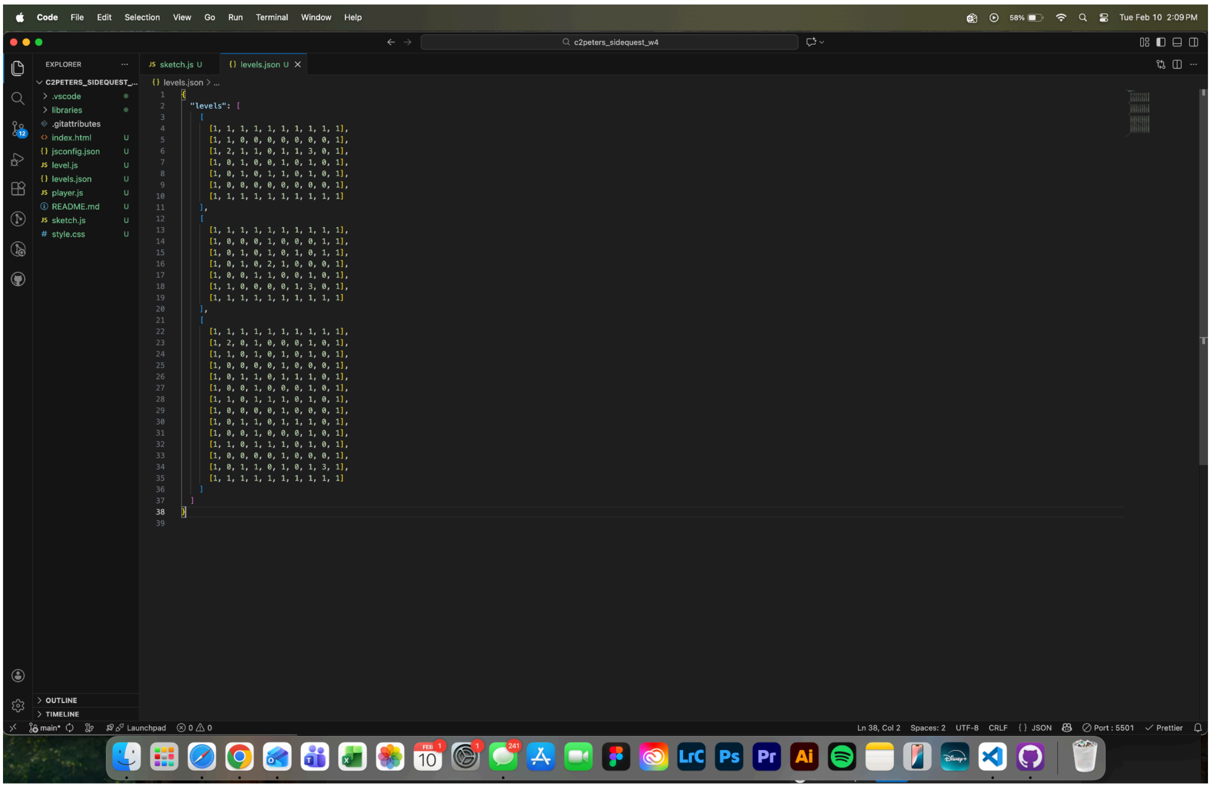
Tools used: VS code

AI documentation: No GenAI used for this task.

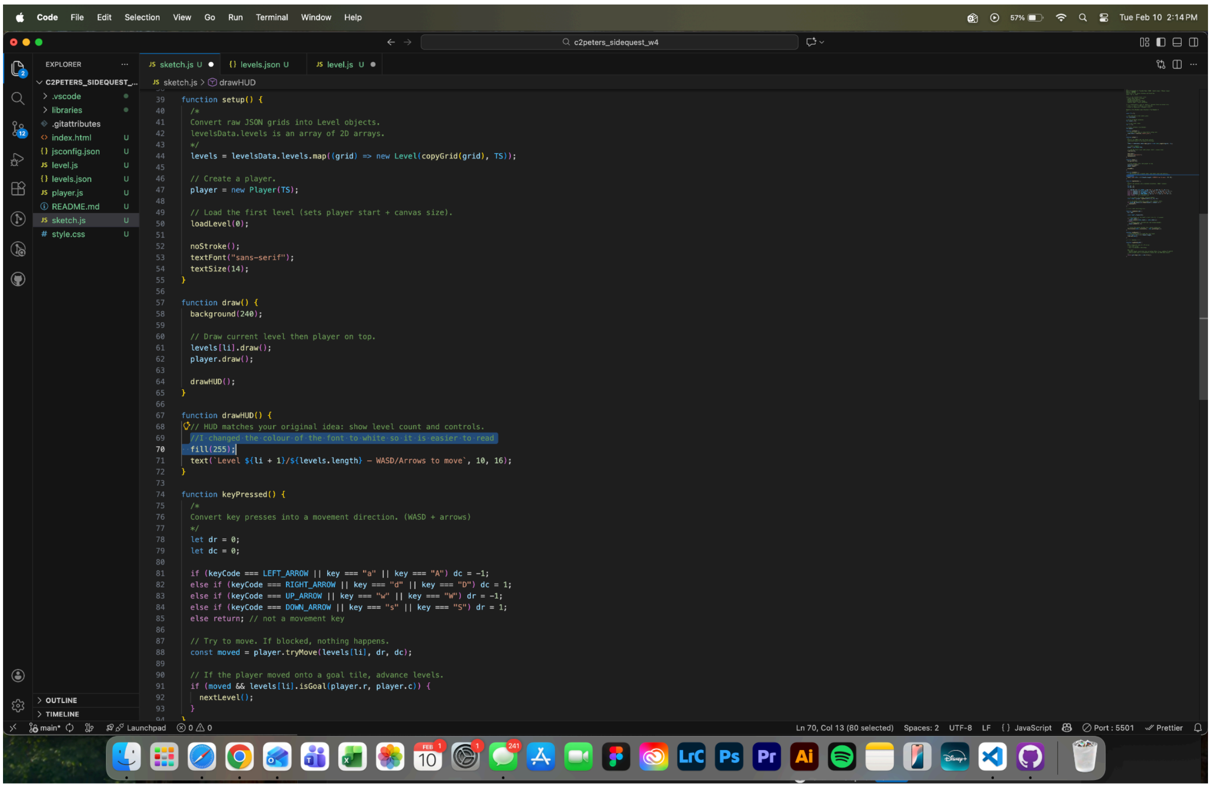
Working screenshots:



```
class Player {
  draw() {
    // Same "simple high-contrast avatar" idea as your original.
    fill(20, 120, 255);
    circle(this.pixelX(), this.pixelY(), this.ts * 0.6);
  }

  /*
  Try to move by (dr, dc) tiles.

  Inputs:
  - level: a Level instance, used for bounds + wall collision + goal detection
  - dr/dc: desired movement step, typically -1,0,1

  Returns:
  - true if the move happened
  - false if blocked or throttled
*/
  tryMove(level, dr, dc) {
    // Throttle so holding a key doesn't spam moves, made it so you move as far as possible whatever direction you moved
    const now = millis();
    if (now - this.movedAt < this.moveDelay) return false;

    let moved = false;

    while (true) {
      const nr = this.r + dr;
      const nc = this.c + dc;

      // Prevent walking off the map.
      if (!level.inBounds(nr, nc)) break;

      // Prevent walking into walls.
      if (level.isWall(nr, nc)) break;

      // Movement is allowed, so commit.
      this.r = nr;
      this.c = nc;
      moved = true;
    }

    if (moved) this.movedAt = now;
    return moved;
  }
}
```



```json
{
  "levels": [
    [
      [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
      [1, 1, 0, 0, 0, 0, 0, 0, 0, 1],
      [1, 2, 1, 1, 0, 1, 1, 3, 0, 1],
      [1, 0, 1, 0, 0, 1, 0, 1, 0, 1],
      [1, 0, 1, 0, 1, 1, 0, 1, 0, 1],
      [1, 0, 0, 0, 0, 0, 0, 0, 0, 1],
      [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
    ],
    [
      [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
      [1, 0, 0, 0, 1, 0, 0, 0, 1, 1],
      [1, 0, 1, 0, 1, 0, 1, 0, 1, 1],
      [1, 0, 1, 0, 2, 1, 0, 0, 0, 1],
      [1, 0, 0, 1, 1, 0, 0, 1, 0, 1],
      [1, 1, 0, 0, 0, 0, 1, 3, 0, 1],
      [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
    ],
    [
      [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
      [1, 2, 0, 1, 0, 0, 0, 1, 0, 1],
      [1, 1, 0, 1, 0, 1, 0, 1, 0, 1],
      [1, 0, 0, 0, 0, 1, 0, 0, 0, 1],
      [1, 0, 1, 1, 0, 1, 1, 1, 0, 1],
      [1, 0, 0, 1, 0, 0, 0, 1, 0, 1],
      [1, 1, 0, 1, 1, 1, 0, 1, 0, 1],
      [1, 0, 0, 0, 0, 1, 0, 0, 0, 1],
      [1, 0, 1, 1, 0, 1, 1, 1, 0, 1],
      [1, 0, 0, 1, 0, 0, 0, 1, 0, 1],
      [1, 1, 0, 1, 1, 1, 0, 1, 0, 1],
      [1, 0, 0, 0, 0, 1, 0, 0, 0, 1],
      [1, 0, 1, 1, 0, 1, 0, 1, 3, 1],
      [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
    ]
  ]
}
```



```javascript
function setup() {
  /*
  Convert raw JSON grids into Level objects.
  levelsData.levels is an array of 2D arrays.
  */
  levels = levelsData.levels.map((grid) => new Level(copyGrid(grid), TS));

  // Create a player.
  player = new Player(TS);

  // Load the first level (sets player start + canvas size).
  loadLevel(0);

  noStroke();
  textFont("sans-serif");
  textSize(14);
}

function draw() {
  background(240);

  // Draw current level then player on top.
  levels[li].draw();
  player.draw();

  drawHUD();
}

function drawHUD() {
  // HUD matches your original idea: show level count and controls.
  //I changed the colour of the font to white so it is easier to read
  fill(255);
  text(`Level ${li + 1}/${levels.length} — WASD/Arrows to move`, 10, 16);
}

function keyPressed() {
  /*
  Convert key presses into a movement direction. (WASD + arrows)
  */
  let dr = 0;
  let dc = 0;

  if (keyCode === LEFT_ARROW || key === "a" || key === "A") dc = -1;
  else if (keyCode === RIGHT_ARROW || key === "d" || key === "D") dc = 1;
  else if (keyCode === UP_ARROW || key === "w" || key === "W") dr = -1;
  else if (keyCode === DOWN_ARROW || key === "s" || key === "S") dr = 1;
  else return; // not a movement key

  // Try to move. If blocked, nothing happens.
  const moved = player.tryMove(levels[li], dr, dc);

  // If the player moved onto a goal tile, advance levels.
  if (moved && levels[li].isGoal(player.r, player.c)) {
    nextLevel();
  }
}
```