PROBLEM 1 - OPULENT TILING

(LINK)

This project creates a richly subdivided grid, where each subcell is filled with a color from a carefully chosen blue and gold palette, and contains either a diamond or an ellipse shape whose color is the exact inverse of the background. The key data structure, defined at the start (`let gridData;`), is a deeply nested array that holds the x, y, width, height, and color index for every subcell, as constructed by the `buildGrid()` function (lines 37–67).

The grid is initialized in the `setup()` function (lines 22–28), where the canvas size is set to the window dimensions and `buildGrid()` is called to generate the multi-level array. Each main grid cell (rows and columns set by `let rows` and `let cols`) is itself subdivided into a random number of subcells both vertically and horizontally (`minSub` and `maxSub`). For each subcell, its position, size, and a random color from the `paletteRGB` array are stored.

Rendering is handled by `drawGrid()` (lines 70–112), which loops through every subcell in the grid. For each, it first fills the background rectangle with its palette color. Then, based on the global boolean `showAllEllipse` (line 20), either a diamond (created by connecting the midpoints of each side using `beginShape()`/`endShape()`) or an ellipse (drawn with `ellipse(cx, cy, w - 2, h - 2)`, so it is almost perfectly inscribed in the cell) is drawn at the center. The shape uses the inverse of the background color to create a visually strong contrast.

Interactivity is added with the `mousePressed()` function (lines 115–148). A left mouse click on any subcell changes its color to a different palette value, by updating the fifth element of that subcell's data array. A right click anywhere toggles `showAllEllipse` globally, causing all diamonds to instantly switch to ellipses (and vice versa). To prevent the default context menu on right click, `document.oncontextmenu = () => false;` is used.

Overall, this sketch combines randomization, interaction, and color theory to produce a modernist, modular composition reminiscent of contemporary graphic design, with every detail (color, shape, and behavior) controlled by simple, readable code and explained through comments in each block.

PSEUDOCODE - FLOWCHART

1. Define a palette of six RGB colors (gold and blue tones)
2. Set up the grid parameters: number of main rows and columns, and subcell count range
3. Create a global variable for the 5D grid array
4. Create a boolean flag to track whether ellipses or diamonds are displayed

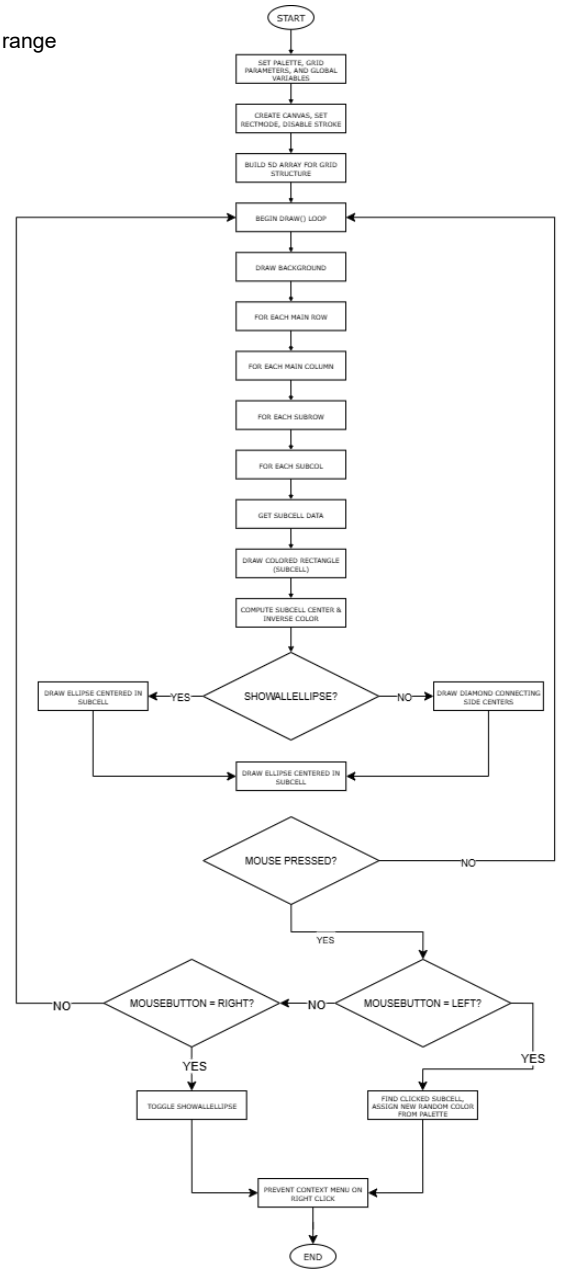
5. On program start:
6. - Create a canvas as big as the window
7. - Set rectangles to be drawn from top-left corner
8. - Disable outline stroke by default
9. - Call a function to build the grid data structure and store in global variable

10. Function to build the grid data structure:
11. - For each main grid row:
12. -- For each main grid column:
13. --- Pick a random integer for the number of subrows (between min and max)
14. --- Pick a random integer for the number of subcolumns (between min and max)
15. --- Calculate the width and height of each subcell in this cell
16. --- For each subrow:
17. ---- For each subcolumn:
18. ----- Calculate the x and y position for this subcell
19. ----- Set width and height for the subcell
20. ----- Choose a random color index from the palette
21. ----- Store [x, y, width, height, colorIndex] for the subcell in the 5D array

22. Every draw frame:
23. - Fill the whole background with dark gray
24. - For each main row in the grid:
25. -- For each main column:
26. --- For each subrow:
27. ---- For each subcolumn:
28. ----- Get the subcell's [x, y, width, height, colorIndex] from the array
29. ----- Lookup the RGB color for this cell from the palette
30. ----- Draw a rectangle at (x, y) with width and height, filled with the palette color
31. ----- Compute the center (cx, cy) of the subcell
32. ----- Compute the color inverse of the fill for the inner shape
33. ----- If ellipses mode is active:
34. ------ Draw an ellipse at the center, width = w-2, height = h-2, with inverse color
35. ----- Else (diamonds mode):
36. ------ Compute the 4 midpoints of the sides of the subcell
37. ------ Draw a diamond shape (polygon) connecting those 4 points, with inverse color

38. When the mouse is pressed:
39. - If the left mouse button:
40. -- Loop through all subcells
41. --- If mouse coordinates are inside the subcell's bounds:
42. ---- Pick a new random palette color for this subcell (not repeating the current color)
43. ---- Update colorIndex for this subcell in the array
44. ---- Stop checking further subcells (only one is changed per click)
45. - If the right mouse button:
46. -- Toggle the ellipses/diamonds mode for all subcells
47. -- Prevent the browser context menu from showing

48. On any right-click anywhere in the document, always suppress the default context menu



```
// This sketch builds a subdivided grid where each cell contains either a diamond or an ellipse,
// using a blue & gold palette. Clicking toggles colors or shapes interactively.

/* --- GLOBAL VARIABLES --- */
let gridData;           // Will store the 5D grid array: [row][col][subrow][subcol][cellData]
let rows = 6;           // Number of main grid rows
let cols = 8;           // Number of main grid columns
let minSub = 2, maxSub = 6; // Range for random subcells per cell

// Cinema Blue & Gold palette in RGB: rich, elegant color choices
let paletteRGB = [
  [202,151,33],   // Chinese Gold
  [213,173,54],   // American Gold
  [232,217,155],  // Deep Champagne
  [68,136,191],   // Cyan-Blue Azure
  [38,89,153],    // Cyan Cobalt Blue
  [219,206,191]  // Pastel Gray
];

let showAllEllipse = false; // If true: show ellipses everywhere, otherwise diamonds

/* --- SETUP: create the canvas and generate the grid --- */
function setup() {
  createCanvas(windowWidth, windowHeight);
  rectMode(CORNER); // Rectangles are drawn from their upper-left corner
  noStroke();
  gridData = buildGrid(rows, cols, minSub, maxSub); // Build the initial grid structure
}

/* --- DRAW: render the whole grid each frame --- */
function draw() {
  background(30); // Deep gray to let gold/blue pop
  drawGrid(gridData);
}

/* --- Build the grid and subdivide each cell into subcells --- */
function buildGrid(rows, cols, minSub, maxSub) {
  let arr = [];
  let cellW = width / cols;
  let cellH = height / rows;
  for (let r = 0; r < rows; r++) {
    arr[r] = [];
    for (let c = 0; c < cols; c++) {
      arr[r][c] = [];
      // Choose a random number of subrows and subcolumns for each cell
      let subRows = int(random(minSub, maxSub + 1));
      let subCols = int(random(minSub, maxSub + 1));
      let subW = cellW / subCols;
      let subH = cellH / subRows;
      for (let sr = 0; sr < subRows; sr++) {
        arr[r][c][sr] = [];
        for (let sc = 0; sc < subCols; sc++) {
          // Compute position and size for each subcell
          let x = c * cellW + sc * subW;
          let y = r * cellH + sr * subH;
          let w = subW;
          let h = subH;
          // Assign a random color from the palette (as an index)
          let colorIndex = floor(random(paletteRGB.length));
          // Store: x, y, width, height, colorIndex
          arr[r][c][sr][sc] = [x, y, w, h, colorIndex];
        }
      }
    }
  }
  return arr;
}

/* --- Render the grid: fill each subcell and draw either a diamond or ellipse at its center --- */
function drawGrid(grid) {
  for (let r = 0; r < grid.length; r++) {
    for (let c = 0; c < grid[r].length; c++) {
      for (let sr = 0; sr < grid[r][c].length; sr++) {
        for (let sc = 0; sc < grid[r][c][sr].length; sc++) {
          let cell = grid[r][c][sr][sc];
          let rgb = paletteRGB[cell[4]];
          let x = cell[0], y = cell[1], w = cell[2], h = cell[3];

          // 1. Draw subcell rectangle
          fill(rgb[0], rgb[1], rgb[2]);
          stroke(80);
          strokeWeight(1.5);
          rect(x, y, w, h);

          // 2. Draw a diamond or ellipse in the center, using the color inverse for contrast
          let cx = x + w / 2, cy = y + h / 2;
          let inv = [255 - rgb[0], 255 - rgb[1], 255 - rgb[2]];
          fill(inv[0], inv[1], inv[2]);
          stroke(255, 90);
          strokeWeight(1.1);

          if (showAllEllipse) {
            // Ellipse: perfectly inscribed in the cell (with tiny margin for stroke)
            ellipse(cx, cy, w - 2, h - 2);
          } else {
            // Diamond: connects midpoints of each side
            let top    = [cx, y];
            let right  = [x + w, cy];
            let bottom = [cx, y + h];
            let left   = [x, cy];
            beginShape();
            vertex(top[0], top[1]);
            vertex(right[0], right[1]);
            vertex(bottom[0], bottom[1]);
            vertex(left[0], left[1]);
            endShape(CLOSE);
          }
        }
      }
    }
  }
}

/* --- Handle mouse clicks for interactivity --- */
function mousePressed() {
  let changed = false;
  // Loop over all subcells to check which one was clicked
  for (let r = 0; r < gridData.length; r++) {
    for (let c = 0; c < gridData[r].length; c++) {
      for (let sr = 0; sr < gridData[r][c].length; sr++) {
        for (let sc = 0; sc < gridData[r][c][sr].length; sc++) {
          let cell = gridData[r][c][sr][sc];
          let [x,y,w,h,ci] = cell;
          if (mouseX >= x && mouseX < x + w && mouseY >= y && mouseY < y + h) {
            if (mouseButton === LEFT) {
              // Left click: change only this subcell's color, never to the same color
              let newIndex;
              do {
                newIndex = floor(random(paletteRGB.length));
              } while (newIndex === ci);
              cell[4] = newIndex;
            }
            changed = true;
            break;
          }
        }
        if (changed) break;
      }
      if (changed) break;
    }
    if (changed) break;
  }
  if (mouseButton === RIGHT) {
    // Right click: toggle between diamonds and ellipses in ALL subcells
    showAllEllipse = !showAllEllipse;
    return false; // Prevents browser context menu
  }
}

//prevent context menu on right-click everywhere
document.oncontextmenu = () => false;

/*
  INTERACTION:
  - Left click: changes the color of the clicked subcell (with no color repetition).
  - Right click: toggles ALL shapes between diamonds and ellipses across the grid.
  - Visuals: Blue & gold palette, modern & cinematic. Shapes invert their color for contrast.
  - Structure: Each grid cell is subdivided into a random grid of subcells.
*/
```