C: > Users > ryanh > Downloads > JS dsdn_142_p1.js > ...

```
// parameters ---------------------------------

// set cell_size
let cell_size = 200.0;

// initialise cell width and cell height (Don't change these)
let cell_width = cell_size;  // re-initialised lower
let cell_height = cell_size;  // re-initialised lower

// This is used solely for logic purpose: DO NO CHANGE THIS!!!
let reset_cell_width = cell_width;
let reset_cell_height = cell_height;

// adjusts the scale of the design elements relative to the cell size
let design_scale = 50;  // changes the scale of the design
design_scale = 100.0 / design_scale; // this is used for logic purposes: DO NOT CHANGE THIS!!!

// select color pallete
let color_p1 = false;
let color_p2 = false;
let color_p3 = false;
let color_p4 = true;
let color_p5 = false;
let color_p6 = false;
let color_p7 = false;
let color_p8 = false;
let color_p9 = false;

//-----------------------------------------------
// grid element settings

// Turn grid design on/off
let is_grid = true;

// select number of primitive lines of grid element (1 through 10)
let prim_lines = 4.0;

// set number of lines in each grid element piece (quarter)
let ge_lines_iter = 500.0;

// rotate the entire gride (degrees)
let rotate_grid = 45; // default = 45

// translates the grid's position
let translate_grid_x = 0.0;
let translate_grid_y = 0.0;

// twists the grid element
let rotate_grid_elements = 1.9;

// sets the spacing between the grid elements
let corner_spacing = 0.001;

//--------------------------------------------
// center piece settings

// turns center piece on = true off = false
let is_center_piece = false; // true = center piece on; false = center_piece off;

// turns center piece from koru spiral = true; circular piece = false
let is_cp_spiral = true;

// adjusts the size of the center piece relative to the grid element
let cp_size_factor = 2.5;  // sets the size of the spiral relative to cell_size

// sets the number of lines in center piece
let cp_lines_iter = 200.0;

// rotates center piece
let rotate_cp = 180.0 // default = 180

// translate center piece position
let translate_cp_x = 147;
let translate_cp_y = 150;

// set center piece twist (lower twists more) when koru spiral is on (is_cp_spiral = true)
let cp_twist_factor = 0.97

// set spiral gradient: false = dark; true = bright
let is_cp_gradient_dark = true;

// sets center piece's rate of color gradient dropoff
let cp_color_decrement = 1; // sets rate of spiral gradient

// adjust the brightness of the spiral
let spiral_brightness = 1.0;

// alter the color of the spiral: will have inverse behaviour if gradient is dark or not
let red_decr_factor = 1.0;   // alter these values to change spiral color
let green_decr_factor = 1.0;
let blue_decr_factor = 1.0;

// used solely for calibrating the brightness: don't adjust these values!!!!!!
red_decr_factor *= spiral_brightness;
green_decr_factor *= spiral_brightness;
blue_decr_factor *= spiral_brightness;

// recalibrate to keep color calibration consistent when gradient is dark: don't adjust these values!!!!!
if (is_cp_gradient_dark) {
  red_decr_factor = 1.0 / red_decr_factor;
  green_decr_factor = 1.0 / green_decr_factor;
  blue_decr_factor = 1.0 / blue_decr_factor;
  spiral_brightness = 1.0 / spiral_brightness;
}


let rotate_spiral = 90;

// end of center piece adjustment variables
//---------------------------------------------
// set colors based on parameters already defined above
if (!is_cp_gradient_dark) {
  cp_color_decrement *= -1;
}

if (color_p1) {  // Color Palette 1: gray/blue
  red_back = 47;
  green_back = 72;
  blue_back = 88;
}

else if (color_p2) {  // Color Palette 2: white
  red_back = 253.0;
  green_back = 255.0;
  blue_back = 252.0;
}


else if (color_p3) {  // Color Pallette 3:
  red_back = 53;  // azure blue
  green_back = 125;
  blue_back = 237;

}

else if (color_p4) {  // Color Pallette 4:
  red_back = 98;  // blue
  green_back = 187;
  blue_back = 193;

}

else if (color_p5) {  // Color Pallette 5:
  red_back = 246;  // blue
  green_back = 247;
  blue_back = 235;

}

else if (color_p6) {  // Color Pallette 6:
  red_back = 0;  // black
  green_back = 0;
  blue_back = 1;

}

else if (color_p7) {  // Color Pallette 7:
  red_back = 35;  // dark maroon
  green_back = 0;
  blue_back = 30;

}

else if (color_p8) {  // Color Pallette 8:
  red_back = 157;    // dull green
  green_back = 217;
  blue_back = 210;

}


else if (color_p9) {  // Color Pallette 8: gray/brown
  red_back = 78.0;     // forrest green
  green_back = 110.0;
  blue_back = 88.0;

}
// ------------------------------------------------
Qodo Gen: Options | Test this function
function setup_wallpaper(pWallpaper) {
  pWallpaper.output_mode(GRID_WALLPAPER); // GRID_WALLPAPER, GLIDE_WALLPAPER, DEVELOP_GLYPH
  pWallpaper.resolution(NINE_LANDSCAPE);
  pWallpaper.show_guide(false); //set this to false when you're ready to print

  //Grid settings
  pWallpaper.grid_settings.cell_width  = reset_cell_width;
  pWallpaper.grid_settings.cell_height = reset_cell_height;
  pWallpaper.grid_settings.row_offset  = 0;
}

Qodo Gen: Options | Test this function
function wallpaper_background() {
  background(red_back, green_back, blue_back); // sky blue: 16, 126, 230
}

Qodo Gen: Options | Test this function
function my_symbol() { // do not rename this function. Treat this similarly to a Draw function
// let cell_size_slider = creatSlider(50.0, 500.0, 300.0);
// reset_cell_width = cell_size_slider.value();
// end of GUI
//----------------------------------------------------------
  cell_width = reset_cell_width;
  cell_height = reset_cell_height;

//-----------------------------------------------------

  if (is_cp_spiral) {
    cp_twist_factor = 0.976;
  } else {
    cp_twist_factor = 1.0;
  }

  if (color_p1) {  // Color Pallette 1: blue, purple, orange, green

    red_spiral = 134.0;    // light blue
    green_spiral = 187.0;
    blue_spiral = 216.0;

    red_spiral = 47;
    green_spiral = 72;
    blue_spiral = 88;


    red_left = 246.0;    // gold yellow
    green_left = 174.0;
    blue_left = 45.0;

    red_right = 242.0;     // bright orange
    green_right = 100.0;
    blue_right = 25.0;
  }

  else if (color_p2) {  // Color Pallette 2: max purple, dark purple, china pink

    red_left = 65.0;    // turquoise
    green_left = 234.0;
    blue_left = 212.0;


    red_spiral = 255.0;    // bright red
    green_spiral = 0.0;
    blue_spiral = 34.0;

    red_spiral = 253.0;
    green_spiral = 255.0;
    blue_spiral = 252.0;


    red_right = 1.0;   // black
    green_right = 22.0;
    blue_right = 39.0;

  }


  else if (color_p3) {  // Color Pallette 3:
    red_spiral = 86.0;    // light blue
    green_spiral = 238;
    blue_spiral = 244.0;

    red_spiral = 53;  // azure blue
    green_spiral = 125;
    blue_spiral = 237;



    red_left = 84;    // purple
    green_left = 56;
    blue_left = 220;

    red_right = 50;   // light green
    green_right = 232;
    blue_right = 117;

  }

  else if (color_p4) {  // Color Pallette 4:
    red_spiral = 98;  // blue
    green_spiral = 187;
    blue_spiral = 193;

    red_left = 1;    // black
    green_left = 4;
    blue_left = 1;

    red_right = 255;   // white
    green_right = 251;
    blue_right = 252;

  }

  else if (color_p5) {  // Color Pallette 5:
    red_spiral = 246;  // off white
    green_spiral = 247;
    blue_spiral = 235;

    red_left = 57;    // dark gray
    green_left = 62;
    blue_left = 65;

    red_right = 233;   // light orange
    green_right = 79;
    blue_right = 55;

  }

  else if (color_p6) {  // Color Pallette 6:
    red_spiral = 0;  // black
    green_spiral = 0;
    blue_spiral = 1;

    red_left = 220;    // pale yellow
    green_left = 204;
    blue_left = 163;

    red_right = 130;   // light pink
    green_right = 76;
    blue_right = 113;

  }

  else if (color_p7) {  // Color Pallette 7:
    red_spiral = 35;  // dark maroon
    green_spiral = 0;
    blue_spiral = 30;

    red_left = 255;    // dull yellow
    green_left = 186;
    blue_left = 73;

    red_right = 239;   // dull pink
    green_right = 91;
    blue_right = 91;

  }

  else if (color_p8) {  // Color Pallette 8:
    red_spiral = 157;    // dull green
    green_spiral = 217;
    blue_spiral = 210;

    red_right = 57;   // purple
    green_right = 47;
    blue_right = 90;

    red_left = 255;  // dark yellow
    green_left = 136;
    blue_left = 17;

  }


  else if (color_p9) {  // Color Pallette 8:
    red_spiral = 78.0;    // forrest green
    green_spiral = 110.0;
    blue_spiral = 88.0;

    red_right = 122.0;   // dark mint green
    green_right = 223.0;
    blue_right = 187.0;

    red_left = 166.0;  // light mint green
    green_left = 236.0;
    blue_left = 224.0;

  }


//-----------------------------------------------------

function create_square(cell_width1, cell_height1, divisor1) {
  let p1 = createVector(cell_width1 / divisor1); // top point
  let p2 = createVector(cell_width1 / divisor1, cell_height1 / divisor1); // right point
  let p3 = createVector(cell_width1 / divisor1, cell_height1 / divisor1);  // bottom point
  let p4 = createVector(0, cell_height1 / divisor1);  // left point
  let p5 = createVector(0, cell_height1 / divisor1*2.0);
  let p6 = createVector(0, cell_height1 / divisor1*3.0);
  let p7 = createVector(0, cell_height1 / divisor1*4.0);
  let p8 = createVector(0, cell_height1 / divisor1*5.0);
  let p9 = createVector(0, cell_height1 / divisor1*6.0);
  let p10 = createVector(0, cell_height1 / divisor1*7.0);

  if (prim_lines >= 1) {
    line(p1.x, p1.y, p2.x, p2.y);
  }
  if (prim_lines >=2) {
    line(p2.x, p2.y, p3.x, p3.y);
  }
  if (prim_lines >= 3) {
    line(p3.x, p3.y, p4.x, p4.y);
  }
  if (prim_lines >= 4) {
    line(p4.x, p4.y, p1.x, p1.y);
  }
  if (prim_lines >= 5) {
    line(p4.x, p4.y, p5.x, p5.y);
  }
  if (prim_lines >= 6) {
    line(p5.x, p5.y, p6.x, p6.y);
  }
  if (prim_lines >= 7) {
    line(p6.x, p6.y, p7.x, p7.y);
  }
  if (prim_lines >= 8) {
    line(p7.x, p7.y, p8.x, p8.y);
  }
  if (prim_lines >= 9) {
    line(p8.x, p8.y, p9.x, p9.y);
  }
  if (prim_lines >= 10) {
    line(p9.x, p9.y, p10.x, p10.y);
  }
}

// start of grid creation
if (is_grid) {
  rotate(rotate_grid);

  translate(translate_grid_x, translate_grid_y);

  cell_width = reset_cell_width;
  cell_height = reset_cell_height;

  strokeWeight(0.5);
  translate(cell_width * 0.5, cell_height * 0.5);
  for (let i = 0; i < ge_lines_iter; i++) {   // top left
    rotate(rotate_grid_elements);
    stroke(red_left, green_left, blue_left);
    create_square(cell_width, cell_height, design_scale);
    cell_width *= 0.97;
    cell_height *= 0.97;
  }
  rotate(rotate_grid_elements * ge_lines_iter * -1);


  cell_width = reset_cell_width;
  cell_height = reset_cell_height;
  translate(cell_width *-corner_spacing, 0);
  rotate(90);

  for (let i = 0; i < ge_lines_iter; i++) {   // top left
    rotate(rotate_grid_elements);
    stroke(red_left, green_left, blue_left);
    create_square(cell_width, cell_height, design_scale);
    cell_width *= 0.97;
    cell_height *= 0.97;

  }
  rotate(rotate_grid_elements * ge_lines_iter * -1);


  cell_width = reset_cell_width;
  cell_height = reset_cell_height;
  translate(cell_width *-corner_spacing, 0);
  rotate(90);

  for (let i = 0; i < ge_lines_iter; i++) {   // top left
    rotate(rotate_grid_elements);
    stroke(red_right, green_right, blue_right);
    create_square(cell_width, cell_height, design_scale);
    cell_width *= 0.97;
    cell_height *= 0.97;
  }
  rotate(rotate_grid_elements * ge_lines_iter * -1);

  cell_width = reset_cell_width;
  cell_height = reset_cell_height;
  translate(cell_width *-corner_spacing, 0);
  rotate(90);

  for (let i = 0; i < ge_lines_iter; i++) {   // top left
    rotate(rotate_grid_elements);
    stroke(red_right, green_right, blue_right);
    create_square(cell_width, cell_height, design_scale);
    cell_width *= 0.97;
    cell_height *= 0.97;
  }
  rotate(rotate_grid_elements * ge_lines_iter * -1);


  rotate(-rotate_grid); // counter rotation so spiral is not affected
}
// end of grid creation
//------------------------------------------------------------

if (is_center_piece) {
  // -------------------------------------------------------
  // Koru sprial code

  cell_width = reset_cell_width * cp_size_factor;
  cell_height = reset_cell_height * cp_size_factor;

  rotate(rotate_spiral);
  translate(translate_cp_x, translate_cp_y);

  // spiral
  //translate(cell_width / 2.0, cell_height / 2.0);
  stroke(red_spiral, green_spiral, blue_spiral);
  rotate(rotate_cp);
  for (let i = 0; i < cp_lines_iter; i++) {   // top left
    rotate(-9);
    create_square(cell_width, cell_height, design_scale * 5);
    cell_width *= cp_twist_factor;
    cell_height *= cp_twist_factor;
    red_spiral -= cp_color_decrement * red_decr_factor;
    green_spiral -= cp_color_decrement * green_decr_factor;
    blue_spiral -= cp_color_decrement* blue_decr_factor;
    stroke(red_spiral, green_spiral, blue_spiral);
    }

  }

}
```