THIRD PART - LOCATION OF A CIRCLE

(LINK)

This project is a programmatic interpretation of Sol LeWitt's "Location of a Circle", realized through p5.js. LeWitt's original instructions, notoriously intricate and recursive, demand the construction of a circle using only geometric points derived from the structure of a square. As stated by LeWitt:

> "A circle whose radius is equal to half the distance between two points, the first point is found where two lines would cross if the first line were drawn from a point halfway between a point halfway between the center of the square and the upper right corner and the midpoint of the topside…"

In my code, I first define the square and all its key vertices and midpoints within the draw() function. To translate LeWitt's layered instructions, I repeatedly use a custom midpoint(p1, p2) function, which calculates the midpoint between two p5.Vector points. The construction then proceeds by creating further midpoints of midpoints, reflecting LeWitt's recursive "halfway between a point halfway…" structure.

Crucially, to find the specific points called for by the instructions, I make use of the function lineIntersection(p1, p2, p3, p4), which computes the intersection of two lines defined by pairs of points. This is how the two "special points" of the construction are found:
the first at the intersection of the "first set" of lines, the second at the intersection of the "second set," exactly as described in the original text.

To determine the center of the final circle, I use the function circumcenter(A, B, C). This calculates the circumcenter of a triangle whose vertices are:

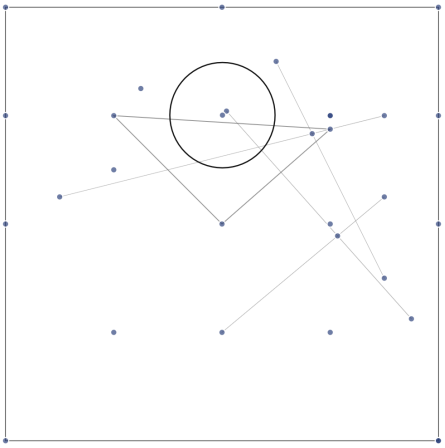
- the center of the square,
- a midpoint between the center and the upper left corner,
- and a further midpoint constructed between lines—matching LeWitt's phrasing, "the center is located equidistant to three points…"

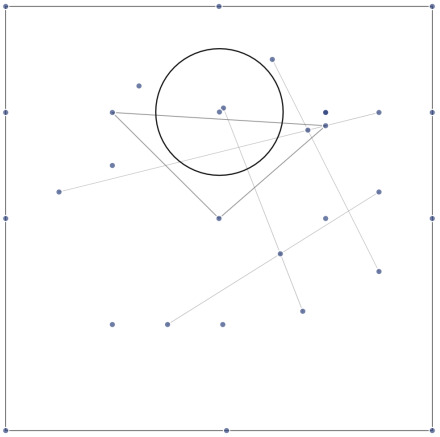
All the geometric points and construction lines are visualized step-by-step, and a further function, drawPoint(pt, sz, col), is used to render each relevant point as a small circle on the canvas.

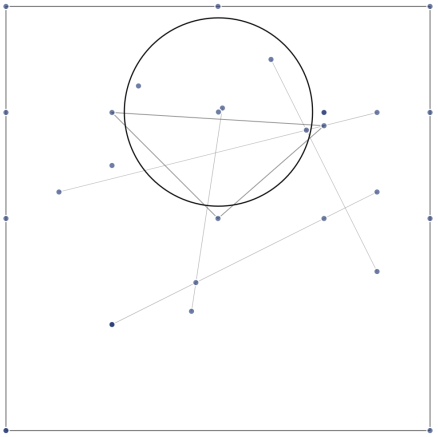
A notable addition in this implementation is the interactivity: the point "MB," representing the midpoint on the bottom edge, can be moved by the user along the square's base. While LeWitt does not prescribe this, I believe it embraces his spirit of openness—leaving room for variation and the interpreter's initiative.

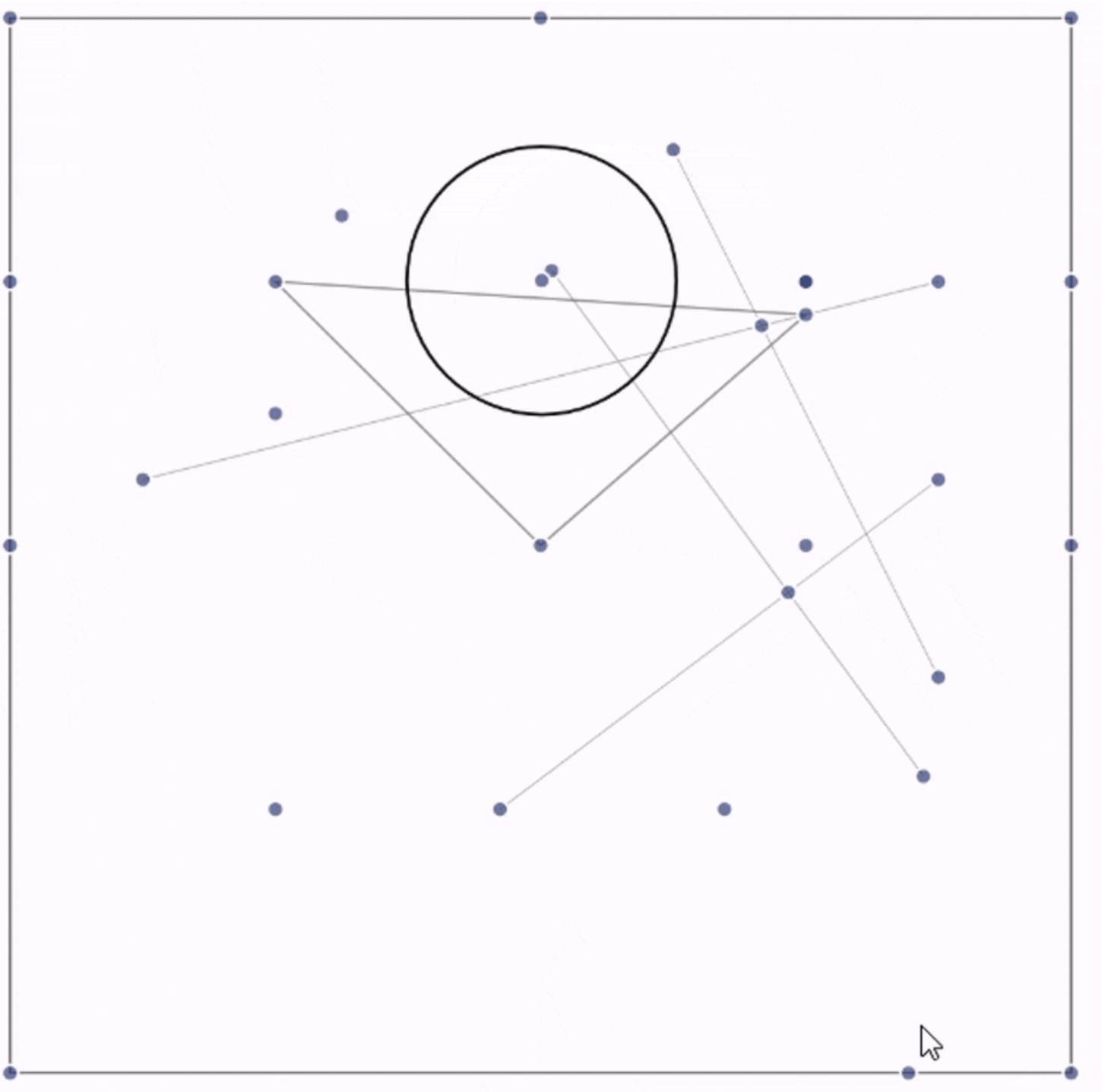
Overall, the code balances mathematical rigor and creative freedom, staying as faithful as possible to LeWitt's detailed instructions, while acknowledging the ambiguities and liberties his conceptual art always allowed. The translation of the written score into code makes both the logic and the inherent interpretative space of LeWitt's process visible and experiential.

```
// ------------------------------------------------------------------
// Summary:
// This p5.js sketch constructs, according to Sol LeWitt's instructions,
// a circle inscribed through geometric construction starting from a square.
// All construction points are visualized, and the point MB on the lower
// edge can be moved with the mouse. The final circle's radius is half the
// distance between two points found as intersections of lines, and its center
// is the circumcenter of a triangle whose vertices are also determined by
// midpoints and intersections, as described in LeWitt's text.
//
// Reference text for the instructions:
// "A circle whose radius is equal to half the distance between two points,
// the first point is found where two lines would cross if the first line
// were drawn from a point halfway between a point halfway between the center
// of the square and the upper right corner and the midpoint of the baseside to
// a point halfway between a point halfway between the center of the square and
// the midpoint of the right side and a point halfway between the midpoint of
// the right side and the lower right corner, the second line of the first set
// is drawn from a point halfway between a point halfway between the center of
// the square and a point halfway between the midpoint and the left side and
// the upper left corner and the midpoint of the left side to a point halfway
// between a point halfway between the center of the square and the upper right
// corner and a point halfway between the midpoint of the right side and the
// upper right corner, the second point is found where two lines would cross
// if the first line is drawn from a point halfway between a point halfway
// between the center of the square and the midpoint of the bottom side and a
// point halfway between the center of the square and the lower left corner
// to a point halfway between the end of the first line of the first set and
// the end of the second line of the first set, the second line of the second
// set is drawn from a point halfway between the point where the first two
// lines have crossed and a point halfway between the start of the first line
// of the first set and a point halfway between the midpoint of the left side
// and the upper left corner to a point halfway between the end of the first
// line of the second set and the midpoint of the bottom side; all whose center
// is located equidistant to three points, the first of which is located at
// the center of the square, the second point is located at a point halfway
// between a point halfway between the center of the square and the upper left
// corner, the third point is located halfway between the start of the first
// line of the first set and the end of the first line of the second set."
// ------------------------------------------------------------------

let sqX, sqY, sqL;

function setup() {
  createCanvas(windowWidth, windowHeight);
}

function windowResized() {
  resizeCanvas(windowWidth, windowHeight);
}

function draw() {
  background(255);

  // --- Construct the largest centered square in the window ---
  // "center of the square" and all edge references are derived from here.
  sqL = min(width, height) * 0.85;
  sqX = (width - sqL) / 2;
  sqY = (height - sqL) / 2;

  // --- Define basic points of the square (vertices and midpoints) ---
  // These points are used for all subsequent constructions.
  let UL = createVector(sqX, sqY);                    // Upper Left
  let UR = createVector(sqX + sqL, sqY);              // Upper Right
  let LR = createVector(sqX + sqL, sqY + sqL);        // Lower Right
  let LL = createVector(sqX, sqY + sqL);              // Lower Left
  let MT = createVector(sqX + sqL / 2, sqY);          // Mid Top
  let MR = createVector(sqX + sqL, sqY + sqL / 2);    // Mid Right

  // --- Interactivity: MB slides along the bottom edge of the square ---
  // "midpoint of the bottom side" is variable via mouseX
  let t = constrain(map(mouseX, sqX, sqX + sqL, 0, 1), 0, 1);
  let MB = createVector(lerp(sqX, sqX + sqL, t), sqY + sqL);  // Mid Bottom (slidable)
  let ML = createVector(sqX, sqY + sqL / 2);          // Mid Left
  let C = createVector(sqX + sqL / 2, sqY + sqL / 2);         // Center

  // Helper function to find the midpoint between two points
  function midpoint(p1, p2) {
    return createVector((p1.x + p2.x) / 2, (p1.y + p2.y) / 2);
  }

  // ------------------------------------------------
  // --- 1. FIRST SERIES OF INTERSECTIONS (POINT P1) ---
  // ------------------------------------------------

  // Line A, quotation:
  // "the first line were drawn from a point halfway between a point halfway
  // between the center of the square and the upper right corner and the
  // midpoint of the topside to a point halfway between a point halfway
  // between the center of the square and the midpoint of the right side and
  // a point halfway between the midpoint of the right side and the lower
  // right corner"
  let A1 = midpoint(C, UR);                   // halfway between center and upper right
  let A1b = midpoint(A1, MT);                 // halfway between the result and mid-top
  let A2 = midpoint(C, MR);                   // halfway between center and mid-right
  let A2b = midpoint(A2, midpoint(MR, LR));// halfway between the result and halfway between mid-right and lower right

  // Line B, quotation:
  // "the second line of the first set is drawn from a point halfway between a
  // point halfway between the center of the square and a point halfway
  // between the midpoint and the left side and the upper left corner and the
  // midpoint of the left side to a point halfway between a point halfway
  // between the center of the square and the upper right corner and a point
  // halfway between the midpoint of the right side and the upper right corner"
  let B1a = midpoint(ML, UL);                 // halfway between mid-left and upper left
  let D1 = midpoint(C, B1a);                  // halfway between center and (mid-left/upper left)
  let B1b = midpoint(D1, ML);                 // halfway between result and mid-left
  let B2a = midpoint(MR, UR);                 // halfway between mid-right and upper right
  let B2 = midpoint(C, UR);                   // halfway between center and upper right
  let B2b = midpoint(B2, B2a);                // halfway between result and (mid-right/upper right)

  // Intersection point of the two lines above: FIRST POINT
  // "the first point is found where two lines would cross..."
  let P1 = lineIntersection(A1b, A2b, B1b, B2b);

  // ------------------------------------------------
  // --- 2. SECOND SERIES OF INTERSECTIONS (POINT P2) ---
  // ------------------------------------------------

  // Line C, quotation:
  // "the first line is drawn from a point halfway between a point halfway
  // between the center of the square and the midpoint of the bottom side and
  // a point halfway between the center of the square and the lower left corner
  // to a point halfway between the end of the first line of the first set and
  // the end of the second line of the first set"
  let C1a = midpoint(C, MB);                  // halfway between center and mid-bottom
  let C1b = midpoint(C, LL);                  // halfway between center and lower left
  let C1 = midpoint(C1a, C1b);                // halfway between the two results
  let C2 = midpoint(A2b, B2b);                // halfway between the ends of the two lines of the first series

  // Line D, quotation:
  // "the second line of the second set is drawn from a point halfway between
  // the point where the first two lines have crossed and a point halfway
  // between the start of the first line of the first set and a point halfway
  // between the midpoint of the left side and the upper left corner to a
  // point halfway between the end of the first line of the second set and
  // the midpoint of the botton side"
  let D1a = midpoint(A1b, midpoint(ML, UL));      // halfway between start of first line and (mid-left/upper left)
  let D1 = midpoint(P1, D1a);                     // halfway between intersection1 and D1a
  let D2 = midpoint(C2, MB);                      // halfway between end of line C and mid-bottom

  // Intersection point of these two lines: SECOND POINT
  // "the second point is found where two lines would cross..."
  let P2 = lineIntersection(C1, C2, D1, D2);

  // ------------------------------------------------------------------
  // --- 3. TRIANGLE FOR THE CENTER OF THE CIRCLE (CIRCUMCENTER, point O) ---
  // ------------------------------------------------------------------

  // "all whose center is located equidistant to three points,
  // the first of which is located at the center of the square,
  // the second point is located at a point halfway between a point halfway
  // between the center of the square and the upper left corner,
  // the third point is located halfway between the start of the first line
  // of the first set and the end of the first line of the second set."
  let pt2 = midpoint(C, UL);                      // halfway between center and upper left
  let pt3 = midpoint(A1b, C2);                    // halfway between start of line A and end of line C
  let O = circumcenter(C, pt2, pt3);              // circle center: circumcenter of these three points

  // ------------------------------------------------------
  // --- 4. CALCULATION OF THE FINAL CIRCLE RADIUS ---
  // ------------------------------------------------------
  // "A circle whose radius is equal to half the distance between two points..."
  let r = dist(P1.x, P1.y, P2.x, P2.y) / 2;   // radius: half the distance between the two intersection points

  // ------------------------
  // --- FINAL DRAWING ---
  // ------------------------

  // Draw the base square
  stroke(30);
  strokeWeight(0.8);
  noFill();
  rect(sqX, sqY, sqL, sqL);

  // Draw all construction lines
  stroke(120, 120, 120, 140);
  strokeWeight(0.6);
  line(A1b.x, A1b.y, A2b.x, A2b.y);
  line(B1b.x, B1b.y, B2b.x, B2b.y);
  line(C1.x, C1.y, C2.x, C2.y);
  line(D1.x, D1.y, D2.x, D2.y);

  // Draw the triangle that determines the circumcenter
  stroke(30, 90, 90, 160);
  strokeWeight(1);
  noFill();
  beginShape();
  vertex(C.x, C.y);
  vertex(pt2.x, pt2.y);
  vertex(pt3.x, pt3.y);
  endShape(CLOSE);

  // Draw the final circle
  stroke(20);
  strokeWeight(1.4);
  noFill();
  ellipse(O.x, O.y, 2 * r, 2 * r);

  // Draw all construction points
  let allPoints = [
    UL, UR, LR, LL, MT, MR, MB, ML, C,
    A1, A1b, A2, A2b,
    B1a, B1, B1b, B2a, B2, B2b,
    P1, C1a, C1b, C1, C2, D1a, D1, D2, P2,
    pt2, pt3, O
  ];
  for (let pt of allPoints) {
    drawPoint(pt, 7, color(40, 60, 120, 170));
  }

  // Instructions at the bottom for interactivity
  noStroke();
  fill(50, 90);
  textSize(15);
  textAlign(CENTER);
  text("Sol LeWitt - All construction points (MB moves along the bottom edge)", width / 2, sqY + sqL + 28);
}

// --- Function for intersection between two (infinite) lines ---
// Used to find P1 and P2
function lineIntersection(p1, p2, p3, p4) {
  let denom = (p1.x - p2.x) * (p3.y - p4.y) - (p1.y - p2.y) * (p3.x - p4.x);
  if (denom === 0) return createVector(0, 0);
  let x = ((p1.x * p2.y - p1.y * p2.x) * (p3.x - p4.x) - (p1.x - p2.x) * (p3.x * p4.y - p3.y * p4.x)) / denom;
  let y = ((p1.x * p2.y - p1.y * p2.x) * (p3.y - p4.y) - (p1.y - p2.y) * (p3.x * p4.y - p3.y * p4.x)) / denom;
  return createVector(x, y);
}

// --- Function to calculate the circumcenter of a triangle ---
// Used to find the center of the circle (O)
function circumcenter(A, B, C) {
  let D = 2 * (A.x * (B.y - C.y) + B.x * (C.y - A.y) + C.x * (A.y - B.y));
  let Ux = ((A.x * A.x + A.y * A.y) * (B.y - C.y) +
            (B.x * B.x + B.y * B.y) * (C.y - A.y) +
            (C.x * C.x + C.y * C.y) * (A.y - B.y)) / D;
  let Uy = ((A.x * A.x + A.y * A.y) * (C.x - B.x) +
            (B.x * B.x + B.y * B.y) * (A.x - C.x) +
            (C.x * C.x + C.y * C.y) * (B.x - A.x)) / D;
  return createVector(Ux, Uy);
}

// --- Draw a point as a small circle ---
// Used to visualize all intermediate and final points
function drawPoint(pt, sz, col) {
  push();
  stroke(255);
  strokeWeight(1.2);
  fill(col);
  ellipse(pt.x, pt.y, sz, sz);
  pop();
}
```









Sol LeWitt - LOCATION OF A CIRCLE