```
// Arrays to store user entries
var m = [];
var b = [];
var r = [];
var x = [];
var y = [];

/* Variables to store quadratic equations - Line to Circle
Note: since the graph only allows 1 line and 1 circle,
the array index can just be 0; no array needed for a1, b1, etc */
var a1;
var b1;
var c1;
var xlnc = [0,0];
var ylnc = [0,0];

/* Variables to store quadratic equations - Circle to Circle
Note: since the graph only allows 1 circle and 1 circle,
the array index can just be 0; no array needed for a1, b1, etc */
var Mnum;
var Mden;
var M;
var N;

var a2;
var b2;
var c2;

var xcc = [0,0];
var ycc = [0,0];

function lineEntry () {
  if (getText("m") != "" && getText("b") != "" &&
  (getText("m") != m[m.length-1] || getText("b") != b[b.length-1] )) {
    appendItem(m,getText("m"));
    appendItem(b,getText("b"));
  }
}

function circleEntry () {
  if (getText("r") != "" && getText("x") != "" && getText("y") != "" &&
  (getText("r") != r[r.length-1] || getText("x") != x[x.length-1] ||
  getText("y") != y[y.length-1])) {
    appendItem(r,getText("r"));
```

```
    appendItem(x,getText("x"));
    appendItem(y,getText("y"));
  }
}

function graph () {
  // Add the user entry for a line
  lineEntry();

  // Graph Line
  setStrokeWidth(3);
  line((0),(160-20*b[b.length-1])+20*8*m[m.length-1],
  (320),(160-20*b[b.length-1])-20*8*m[m.length-1]);


  // Add the user entry for a circle
  circleEntry();

  // Graph circle
  setFillColor(rgb(255,255,0,0));
  setStrokeWidth(3);
  circle(160+20*x[x.length-1],160-20*y[y.length-1],20*r[r.length-1]);
}

function circCirc () {
  if (r.length > 1) {
    //Assign values to M-numerator and M-denominator
    Mnum = (Math.pow(r[r.length-2],2)-Math.pow(r[r.length-1],2)-Math.pow(x[x.length-2],2)
    -Math.pow(y[y.length-2],2)-(-1*Math.pow(x[x.length-1],2))-(-1*Math.pow(y[y.length-1],2)));
    Mden = 2*y[y.length-1] - 2*y[y.length-2];

    //If the Denominator is not 0 (different y coordinate and different x coordinate)
    diffXY();

    //If the denominator is 0
    sameY();
  }
}

function sameXY () {
  if (2*x[x.length-1]-2*x[x.length-2] == 0) {
    setText("circlecirclesolx1", "No real solution");
    setText("circlecirclesoly1", "No real solution");
    setText("circlecirclesolx2", "No real solution");
```

```
    setText("circlecirclesoly2", "No real solution");
  }
}

function sameY () {
  if (Mden == 0) {
    //If the denominator is not 0 (same y coordinate, different x coordinate)
    if (2*x[x.length-1]-2*x[x.length-2] != 0) {
      xcc[0] = Math.round(10*(Mnum/(2*x[x.length-1]-2*x[x.length-2])))/10;
      xcc[1] = Math.round(10*(Mnum/(2*x[x.length-1]-2*x[x.length-2])))/10;
      ycc[0] = Math.round(10*(Math.pow(Math.pow(r[r.length-2],2)-
      Math.pow(xcc[0]-x[x.length-2],2),0.5) - (-y[y.length-2])))/10;
      ycc[1] = Math.round(10*(-Math.pow(Math.pow(r[r.length-2],2)-
      Math.pow(xcc[1]-x[x.length-2],2),0.5) - (-y[y.length-2])))/10;

        //Nonnegative discriminant
        if (Math.pow(r[r.length-2],2) - Math.pow(xcc[0]-x[x.length-2],2) >= 0) {
          setText("circlecirclesolx1", xcc[0]);
          setText("circlecirclesoly1", ycc[0]);
          setText("circlecirclesolx2", xcc[1]);
          setText("circlecirclesoly2", ycc[1]);
        }

        //Negative Discriminant
        if (Math.pow(r[r.length-2],2) - Math.pow(xcc[0]-x[x.length-2],2) < 0) {
          setText("circlecirclesolx1", "No real solution");
          setText("circlecirclesoly1", "No real solution");
          setText("circlecirclesolx2", "No real solution");
          setText("circlecirclesoly2", "No real solution");
        }

      }

      //If the denominator is 0 (same (x,y) center position)
      sameXY();
    }

  }

function diffXY () {
  if (Mden != 0) {
    //Assign values to Quadratic variables
    M = Mnum/Mden;
    N = (2*x[x.length-1]-2*x[x.length-2])/Mden;
```

```
    a2 = 1-(-Math.pow(N,2));
    b2 = -2*x[x.length-2]-2*M*N-(-2*N*y[y.length-2]);
    c2 = Math.pow(x[x.length-2],2)-(-1*Math.pow(M,2))-2*M*y[y.length-2]
    - (-1*Math.pow(y[y.length-2],2))-Math.pow(r[r.length-2],2);

    xcc[0] = Math.round(10*((-1*b2-(-1*Math.pow(Math.pow(b2,2)-4*a2*c2, 0.5)))/(2*a2)))/10;
    xcc[1] = Math.round(10*((-1*b2-Math.pow(Math.pow(b2,2)-4*a2*c2, 0.5))/(2*a2)))/10;
    ycc[0] = Math.round(10*(M-N*xcc[0]))/10;
    ycc[1] = Math.round(10*(M-N*xcc[1]))/10;

      //Nonnegative discriminant (no Complex roots)
      if (Math.pow(b2,2)-4*a2*c2 >= 0) {
        setText("circlecirclesolx1", xcc[0]);
        setText("circlecirclesoly1", ycc[0]);
        setText("circlecirclesolx2", xcc[1]);
        setText("circlecirclesoly2", ycc[1]);
      }

      //Negative Discriminant (Complex roots)
      if (Math.pow(b2,2)-4*a2*c2 < 0) {
        setText("circlecirclesolx1", "No real solution");
        setText("circlecirclesoly1", "No real solution");
        setText("circlecirclesolx2", "No real solution");
        setText("circlecirclesoly2", "No real solution");
      }

    }
}

// Creating the graph
createCanvas("Canvas",320,320);

  // Horizontal lines
  for (var h = 16; h > 0; h--) {
    line (0 ,20*h, 320, 20*h);

    //Horizontal Axis
    if (20*h == 160) {
      setFillColor('black');
      rect(0 , 157, 320, 6);
    }
  }

  //Vertical Lines
```

```
  for (var v = 16; v > 0; v--) {
    line (20*v, 0, 20*v, 330);

    //Vertical Axis
    if (20*v == 160) {
      setFillColor('black');
      rect(157 , 0, 6, 320);
    }
  }

  //Origin
  circle(160,160,10);


// Graph function
onEvent("graphbutton", "click", function () {

  graph();

});


// Solve for Solutions
onEvent("solve", "click", function () {

  // Add the user entry for a line
  lineEntry();

  // Add the user entry for a circle
  circleEntry();


  //Only Solve for solutions where there are enough graphs
  if (m.length + r.length > 1) {
    setScreen("solutionsscreen");

    //Line to Line intersections
    if (m.length > 1) {

      //Non parallel lines
      if (m[m.length-1] != m[m.length-2]) {
        setText("linesolx", Math.round(10*(b[b.length-1]-b[b.length-2])
        /(m[m.length-2]-m[m.length-1]))/10);
```

```
    setText("linesoly", Math.round(10*(m[m.length-2]*((b[b.length-1]-b[b.length-2])
    /(m[m.length-2]-m[m.length-1]))-(-1*b[b.length-2])))/10);
  }

  //Parallel lines
  if (m[m.length-1] == m[m.length-2]) {
    setText("linesolx", "No Solution");
    setText("linesoly", "No Solution");
  }

}

//Line to Circle intersections
if (m.length > 0 && r.length > 0) {

  //Assign values to Quadratic variables
    a1 = (1+Math.pow(m[m.length-1],2));
    b1 = (2*m[m.length-1]*(b[b.length-1]-y[y.length-1])-2*x[x.length-1]);
    c1 = (Math.pow((b[b.length-1]-y[y.length-1]),2)
    - (-1*Math.pow(x[x.length-1],2)+Math.pow(r[r.length-1],2)));

    xlnc[0] = Math.round(10*((-1*b1-(-1*Math.pow(Math.pow(b1,2)-4*a1*c1, 0.5)))/(2*a1)))/10;
    xlnc[1] = Math.round(10*((-1*b1-Math.pow(Math.pow(b1,2)-4*a1*c1, 0.5))/(2*a1)))/10;
    ylnc[0] = Math.round(10*(m[m.length-1]*xlnc[0] - (-1*b[b.length-1])))/10;
    ylnc[1] = Math.round(10*(m[m.length-1]*xlnc[1] - (-1*b[b.length-1])))/10;

  //Nonnegative discriminant (no Complex roots)
  if (Math.pow(b1,2)-4*a1*c1 >= 0) {
    setText("linecirclesolx1", xlnc[0]);
    setText("linecirclesolx2", xlnc[1]);
    setText("linecirclesoly1", ylnc[0]);
    setText("linecirclesoly2", ylnc[1]);
  }

  //Negative discriminant (has Complex roots)
  if (Math.pow(b1,2)-4*a1*c1 < 0) {
    setText("linecirclesolx1", "No real solution");
    setText("linecirclesoly1", "No real solution");
    setText("linecirclesolx2", "No real solution");
    setText("linecirclesoly2", "No real solution");
  }
}

//Circle to Circle intersections
```

```
    circCirc();

  }

  //If there are less than 2 graphs
  if (m.length + r.length < 2) {
    setScreen("graphscreen");
    setText("solve", "Not enough graphs!");
  }

});

//Return to Graph screen
onEvent ("mainscreen", "click", function () {
  setScreen("graphscreen");
});
```