COMP 140: Circles (Writeup)

Due on September 7, 2017 at 1:00 PM

Rixner

Joel Abraham

Recipe

Describe the recipe for computing the center and radius of a circle given three points. Keep in mind the five functions you wrote in part 3. You must refer (by name) to these functions here!

Solution

Three points (A, B, and C) are passed as an input to the make_circle() function – the main function which returns the center and radius of the unique circle formed by these three points. This function first calculates the slopes of lines AB and AC via calls to the slope() function. The slope() function takes two points, $P_1 = (x_0, y_0)$ and $P_2 = (x_1, y_1)$ as an input and returns the slope of the line passing through the two points, using the slope formula, given by $m = \frac{y_1 - y_0}{x_1 - x_0}$. The midpoints M_1 of AB and M_2 of AC are calculated via calls to the midpoint() function. The midpoint() function takes two points, $P_1 = (x_0, y_0)$ and $P_2 = (x_1, y_1)$, as an input and returns the midpoint of the line formed by the two points (represented as a 2-tuple) using the midpoint formula, given by $M = (\frac{x_0 + x_1}{2}, \frac{y_0 + y_1}{2})$. The center is determined to be the intersection of the perpendicular bisectors of AB and AC, passing through M_1 and M_2, respectively. Thus, the slopes of the perpendicular bisectors are calculated via calls to the perp() function. The perp() function takes the slope of a line, m as an input and calculates the slope of a perpendicular line using the following equation: $m_p = -\frac{1}{m}$. Now, the perpendicular bisectors are uniquely defined by the midpoints of AB and AC coupled with their respective perpendicular slopes. The center is now computed via a call to the intersect() function, which takes two lines that intersect exactly once as an input and returns their unique point of intersection as a 2-tuple. The inputs to the intersect() function are two lines, each formed by a point and a slope: $P_1 = (x_0, y_0)$ with slope m_1 and $P_2 = (x_1, y_1)$ with slope m_2. The x-coordinate and y-coordinate of intersection are determined by:

$$x_{int} = \frac{m_1 x_0 - m_2 x_1 + y_1 - y_0}{m_1 - m_2}, \quad y_{int} = m1(x_{int} - x_0) + y_0.$$

Finally, the radius is calculated by determining the distance from O to A, using a call to the distance() function, which takes two points, $P_1 = (x_0, y_0)$ and $P_2 = (x_1, y_1)$ as an input and returns the distance between the two points based on the distance formula, given by $d(P_1, P_2) = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}$. The make_circle() function then returns the center and radius of the resulting circle as a 3-tuple, consisting of the center's x-coordinate, y-coordinate, and the circle's radius.

Discussion

The strategy we followed to calculate the center and radius of the circle is not the most efficient. We used a strategy that you would use on paper in order to introduce you to Python and computational problem solving. However, there are better ways for a computer to solve this problem. Describe (in English) a method for doing so. Your description can be brief (2 or 3 sentences), just give the idea, you do not need to work out all of the math and Python.

Solution

One solution would be to use the equation of a circle. Since any circle can be represented by $Ax + By + C = 0$, where $C = x^2 + y^2 + D$ for real numbers A, B, and D, we can create a system of equations by plugging in points. For each of the three given points, we get three distinct equations. These can be combined to solve the system of equations. Now, using a 3 x 3 matrix, we can solve this systems of equations by multipying both sides by the inverse of the left hand size matrix to obtain values for A, B, and C. Then, plugging the values back into the general equation and completing the square yields the equation of a circle that reveals the circle's center and radius.