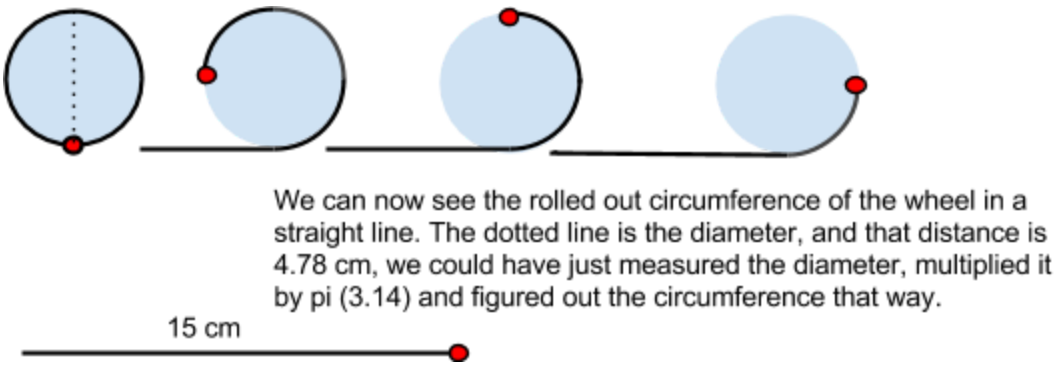
# Understanding Turning Radius and Driving in Convex Polygon Paths in Introductory Robotics

George K. Thiruvathukal, Loyola University Chicago; Dave Garcia, Independent; Ronald I. Greenberg, Loyola University Chicago

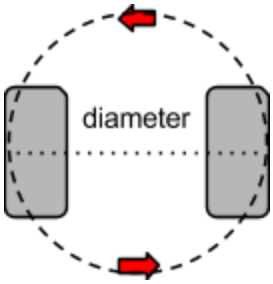
Since most robots are built differently and have different shapes and sizes, you need to calibrate the turning radius for each robot that you want to use.

Start by programming a (Lego) move block to rotate the motors one (1) rotation in a straight line (circle with arrow that has a # inside). Then measure how far the robot travels. Let's say it travelled 15 cm.




By knowing how far a wheel travels when it rotates once, we can program the robot to travel any desired distance we want without ever thinking about this number again. If the robot travels 15 cm for every 1 rotation of the motor, then we can easily program the robot to travel 45 cm. The distance we wish to travel divided by the distance for one rotation equals the number of rotations. 45cm ÷ 15cm = 3.

When we set the steering section (up arrow symbol) in the move block to -100 or 100, the robot will spin in place. That spinning motion creates an imaginary circle pattern on the floor. This circle's circumference is also the distance the wheels are traveling in order to spin 360 degrees. Once we know what the circumference of that imaginary circle is, we can use our formula from above to program the motors to rotate enough to spin 360 degrees. Since we can't straighten out this circle, we will have to figure out the circumference by measuring the diameter. What is




the diameter? When the robot is spinning around, the midpoint between the wheels is the center of the circle, so if we measure the distance between the wheels, we will have the diameter of this circle.

Measure the distance between the wheels (I usually measure from the outer edge of one wheel to the outer edge of the other). Let's say the diameter is 19.11 cm. Multiply this number by $\pi$ and we have the circumference. 19.11 cm x 3.14 cm = 60 cm If the circumference of the spinning circle is 60 cm, then all we need to do is use our original formula: distance needed(60 cm)$\div$distance from one rotation(15 cm)=number of rotations(4). The result shows how many rotations are needed to spin the robot in place for 360$\circ$. In the move block, set the steering (up arrow symbol) to either -100 or 100, this will set the robot to spin in place. In the same move block, set the number of rotations (circle with arrow and # inside) to the result from your last equation. Since the measurement may not be perfect, you may need to tinker with the final number of rotations in order to turn it exactly 360 degrees.

So now that you know the number of rotations it takes to turn the robot 360 degrees, here are a couple of exercises you can do to add turns to your robotics explorations:

- (easy) Write a function (using a My Block) that turns a specific number of degrees. Since the above exploration resulted in knowing how to turn 360 degrees, you can make a parameter in your function to turn by a specified number of degrees and use an arithmetic block to divide by 360 so you can compute the correct number of rotations. You should take care to ensure that the calculation is being performed in decimal (floating point).

- (easy) Using the function you developed in the previous step, write a program that drives the robot from any starting point and follows the imaginary legs of an equilateral triangle. Recall that an equilateral triangle has equal-length sides and 60-degree angles between each edge. The pseudocode to draw this triangle is to do the following:
    - drive straight K rotations
    - turn 60 degrees
    - drive straight K roations
    - turn 60 degrees
    - drive straight K rotations

    - (optionally) turn 60 degrees (to point in the same direction the robot was originally facing)
- (easy) Write a program that drives a square (equal-length sides and 90-degree angles) pattern instead of a triangle. The pseudocode for this is similar to the equilateral triangle but uses 90-degree turns instead of 60-degree turns

- (intermediate) Write a program that can drives in a general convex polygon pattern. Recall that for a convex polygon, the interior angle calculation is (n-2) x 180 / n. Given what we did in the two previous exercises, for n=3, this formula gives 180 / 3 = 60. For n=4, this formula gives (4-2) x 180 / 4 = 360 / 4 = 90. These are the familiar angles for equilateral triangles and squares, respectively. To do this exercise, you will find it convenient to create a function that performs the interior angle calculation. You will also find it convenient to use a loop that iterates over the number of sides. You might find it convenient to first “test” this idea by modifying the two previous exercises to use a loop instead of explicit driving and turning n times.

- (intermediate) Rewrite all of your programs to appear on separate “tabs” (programs) in your Lego EV-3 development environment. Create a tab “triangle” to draw a triangle. Use your function to drive in a convex polygon with parameter n=3. Repeat for “square” with n=4. Try to make some additional convex polygons (octagon, n=8). Each of these tabs should be a single block that is calling your general polygon driving function (My Block).

- Other more advanced options may be added, pending acceptance of our Nifty Assignment.