Teacher Guide

Red-Green-Blue OzoBlockly Intersection Tally Challenge

by Richard Born

Associate Professor Emeritus

Northern Illinois University

rborn2@niu.edu

It is assumed that you have read the 1-page student document that describes this challenge. Let's take a look at a variety of ways that you can use this lesson in your computer science class.

1. Simply give the student handout to students and turn them loose on the challenge. To save class time, you might want to make this a take-home programming project. Then have the students test and debug their programs with Evo the next day in class. Ask each student or group to show and explain their OzoBlockly solution to the challenge. A class discussion of the advantages and disadvantages of the solutions would then be in order.
2. You could provide students with the author's solution and ask them to do a code review. You might also ask them to add appropriate comments to key blocks in the program. Students might even come up with ways to improve the author's solution!
3. When assigning the challenge, you could encourage the use of:
    a. An array to keep track of the tallies for red, green, and blue intersections that are encountered by Evo, rather than three separate named variables.
    b. Use of functions (subroutines) where appropriate to handle specific tasks, *e.g.*, initialization, speaking the color of the intersection, and speaking tallies.
    c. Array and variable names that promote self-documentation, and short function names that describe the purpose of the function.

Comments on the Author's Solution to the Challenge

Figure 1 shows the main program. An array called *Tally* is defined to keep track of red, green, and blue intersection counts. *Tally[0]* is not used. *Tally[1]* will keep track of red, *Tally[2]* green, and *Tally [3]* blue intersections. The main program begins with a call to the initialization subroutine. Then there is a loop where Evo travels consecutively to the 20 intersections. This loop calls the subroutine in which Evo speaks the color of each intersection as it is

encountered. Finally, the "Speak Tallies" subroutine moves Evo to the three tally intersections where he speaks the number of intersections encountered with each of the three colors.

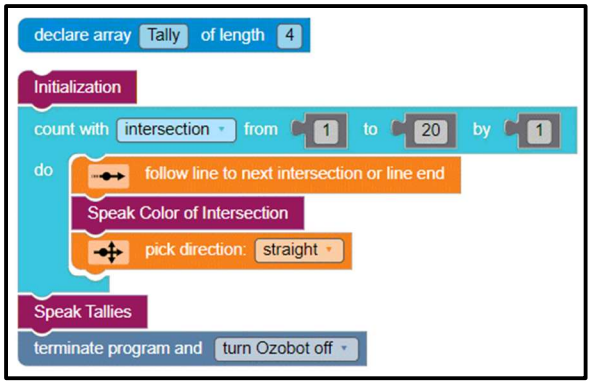

Figure 1

Figure 2 shows the initialization subroutine. First, the line-following speed is set to the highest allowed for Evo. This keeps the time to traverse the intersections as low as possible. Next, three variables—*red*, *green*, and *blue*—are defined with values 1, 2, and 3. This makes it possible to refer to colors by their names, rather than by numbers, significantly improving self-documentation of the program. Finally, the top light is set to white, and a 2-second pause allows placing Evo at the start location before Evo takes off on his journey. When Evo starts moving, his top light is turned off.

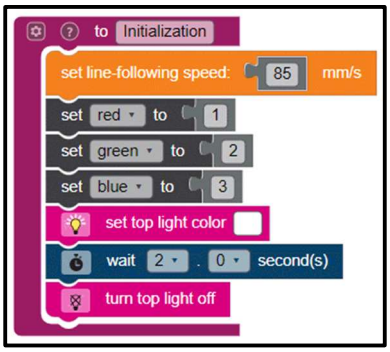

Figure 2

Figure 3 shows the subroutine "Speak Color of Intersection". It begins by setting the top light to the color of the intersection and speaking the color. The light is then turned off before proceeding to the next intersection. An *if* block then sets the variable color to either red, green, or blue. (Remember that the variables *red*, *green*, and *blue* have values 1, 2, and 3, respectively.) Finally, the appropriate element in the array *Tally* is incremented by 1.

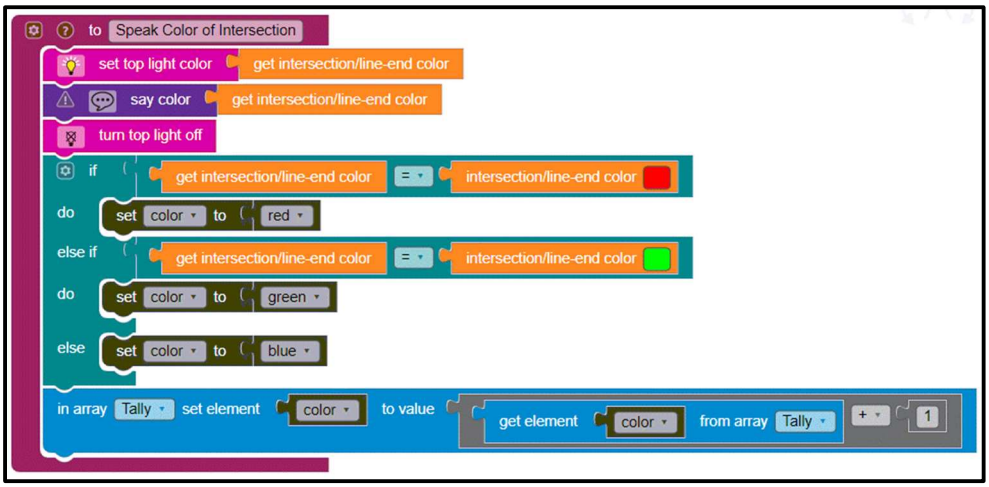

Figure 3

Figure 4 shows the subroutine "Speak Tallies". A loop counter *color* varies from *red*, to *blue*, to *green*, *i.e.*, from 1 to 3 by steps of 1. Within the loop, Evo travels to each of the final three intersections, setting the top light to the intersection color, speaking the tally value followed by the name of the color of the intersection. The light is turned off before proceeding to the next intersection.

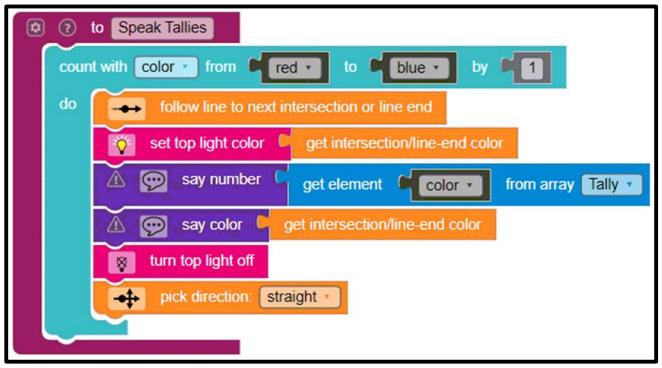

Figure 4