"Evodot": The Making of an OzoBlockly Game

By Richard Born
Associate Professor Emeritus
Northern Illinois University
rborn2@niu.edu

Introduction

So you want to develop an exciting OzoBlockly game for Evo! First, ask yourself what features of the game would make it exciting for players. You decide that the game should make use of free-movement, not line following. It would be nice if one or multiple players could play. By pressing Evo's button, the player should be able to select from several levels of difficulty before play starts. By using Evo's IR sensors, there should be lots of non-contact interaction between players and Evo. Players should be able to accumulate points during play. Points earned should relate to the complexity of the interactions on the game board (Ozomap). The length of the round of play in seconds should be quickly adjustable in the OzoBlockly program before loading the program to Evo. At the end of a round of play, the player should be informed by Evo speaking the total points earned in that round. Evo should respond differently to different colors on the board, with its top light flashing. The game should work on an appropriate map drawn on paper as well as the provided OzoMap. Provision should be made in the program and on the OzoMap to keep Evo from moving off the game board.

Wow—this sounds like a lot to ask! Well, the game does make use of Evo's button for input as well as its color and IR sensors. To do this requires features of OzoBlockly advanced (Mode 4) and master (Mode 5). But this need not be a reason to back away in fear of the program being too difficult to understand. Using subroutines (known as functions in OzoBlockly) to break the program down into more manageable chunks, the program's logic becomes much clearer. This document explains the entire program in detail with the hope of preparing readers to develop their own Evo games.

The OzoMap for this Game

Figure 1 shows a small version of the OzoMap created for use with the ***Evodot*** game. A thick black border helps to keep Evo from leaving the area of play. Evo is placed on the light gray circle labeled *Start*, facing any direction desired by the player. After selecting a level of difficulty from easy, medium and challenging, corresponding to three Evo speeds, Evo begins moving. The player, without touching Evo, guides Evo using the two front IR sensors. The goal is to obtain as high a score as possible in the game time that has been set in the OzoBlockly program (defaulted at 30 seconds). Points are earned by getting Evo's color sensor to move over circles of varying sizes, with the largest circles (red) earning 1 point, the green circles earning 2 points, and the small blue circles earning three points. Players must be aware of the fact that the program does not award points for crossing circles of the same color *two or more times in succession*—points are earned only for the first crossing. (This makes it more challenging for the player, but the real reason behind this rule is to

avoid receiving points more than once while crossing a given circle.) Evo stops momentarily and speaks the points earned for a circle when crossing the circle. When the game time has elapsed, Evo stops and speaks the total score earned by the player.

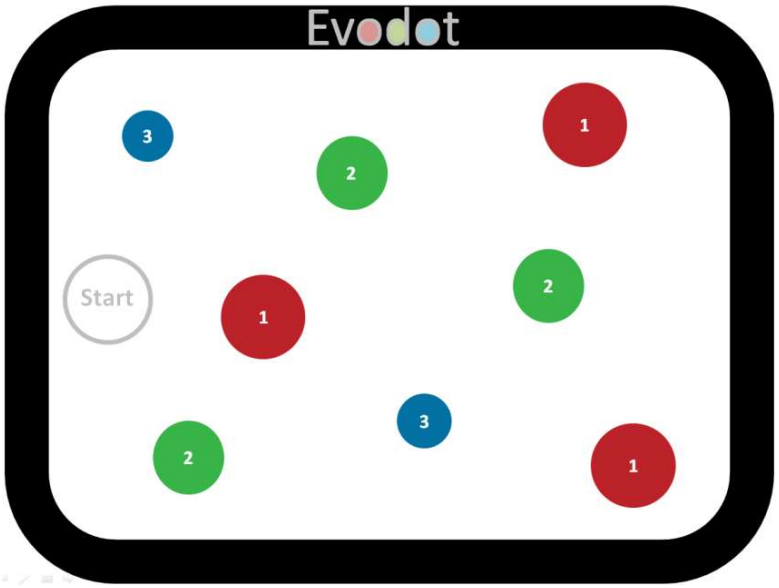

Figure 1

The OzoBlockly Program

Figure 2 shows the structure of the OzoBlockly program, including the main program in gray and ten subroutines in purple, with arrowed lines representing subroutine calls. In order to promote "self-documentation", subroutines are given names symbolic of what they accomplish. In the next several sections of this document, we will discuss in detail the blocks comprising the main program and each of the subroutines.

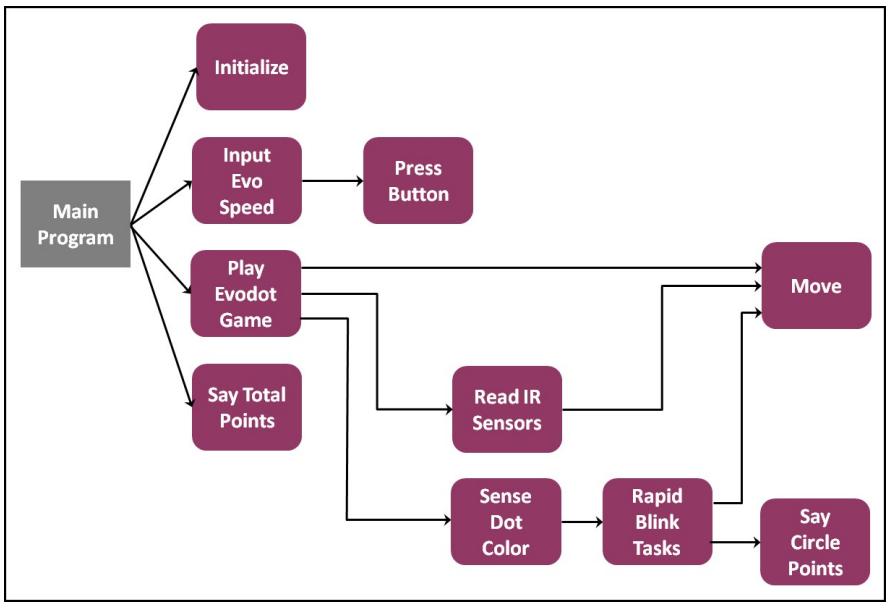

Figure 2

The Main Program

Figure 3 shows the blocks found in the main program. After some initialization of variables and inputting Evo's speed, the top light is set to the color white for 3 seconds, giving the player time to set Evo on the start location of the OzoMap. Just before Evodot game play begins, the timer value is set to 0 seconds. Following this zero setting, the system timer automatically increases by 1 during each second of play. After game play has concluded, there is a 1 second delay before the total points earned by the player are indicated.

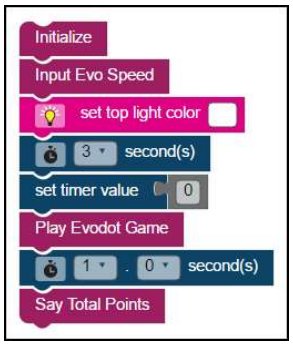

Figure 3

The "Initialize" Subroutine

Figure 4 shows the blocks making up the "initialize" subroutine. In this subroutine, all of the program variables are initialized at the desired values. Doing this makes it easier to find and adjust the values during program maintenance, and is preferred over "hard-coding" numbers throughout the program.

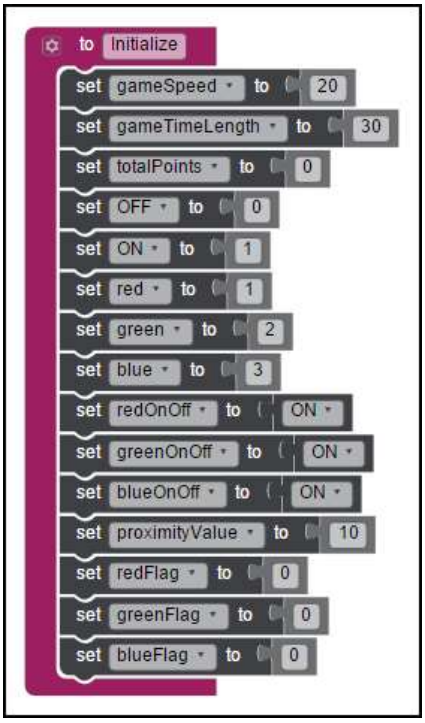

Figure 4

The variable *gameSpeed* is initialized at 20 mm/s. However, the user can make it higher before game play begins, to increase the level of difficulty. The variable *gameTimeLength* is initialized at 30 seconds. This can be adjusted to as high as 127 seconds, however, before loading the program into Evo. The variable *totalPoints* is initialized at 0 and keeps track of the points earned by the player as he or she traverses the colored dots on the OzoMap. The variables *OFF* and *ON*, having the value 0 and 1, respectively, are used to improve self-documentation in the program—the variable names are more meaningful than their values. The variables *red*, *green*, and *blue* have the values 1, 2, and 3, respectively, and represent the points earned by a player when crossing dots of those colors. The variables *redOnOff*, *greenOnOff*, and *blueOnOff* are used to enforce the game rule that a player is not allowed to earn points by crossing two dots consecutively of the same color—only the first crossing earns points. The variable *proximityValue* is used to hold the value received by an IR sensor on Evo. This value lies between 0 and 127, with 0 meaning there is no obstacle detected and increasing values meaning an object is becoming closer to the Ozobot, as indicated in the online OzoBlockly Reference. The variables *redFlag*, *greenFlag*, and *blueFlag* are used in conjunction with the *set light color* block. These variables have either the value 0 or 1, but only one of these variables has the value 1 at any given time (except for during initialization, when they all have the value 0). For example, if *greenFlag* is 1, then the *set light color* block multiples *greenFlag* by 127 to make the top light green while the red and blue components are 0x127=0.

The "Input Evo Speed" and "Press Button" Subroutines

Since these two subroutines are closely related, they are discussed together. The "Input Evo Speed" subroutine calls the "Press Button" subroutine, which counts button presses by the player to set the level of difficulty (Evo speed) of game play. The variable *presses* could in theory be any non-negative integer value. If the player doesn't press the button at all or presses it once, then the variable *gameSpeed* is set to value 20. If the player presses the button twice, the game speed is set to the value 40. If the button is pressed three or more times, then the game speed is set to the value 60.

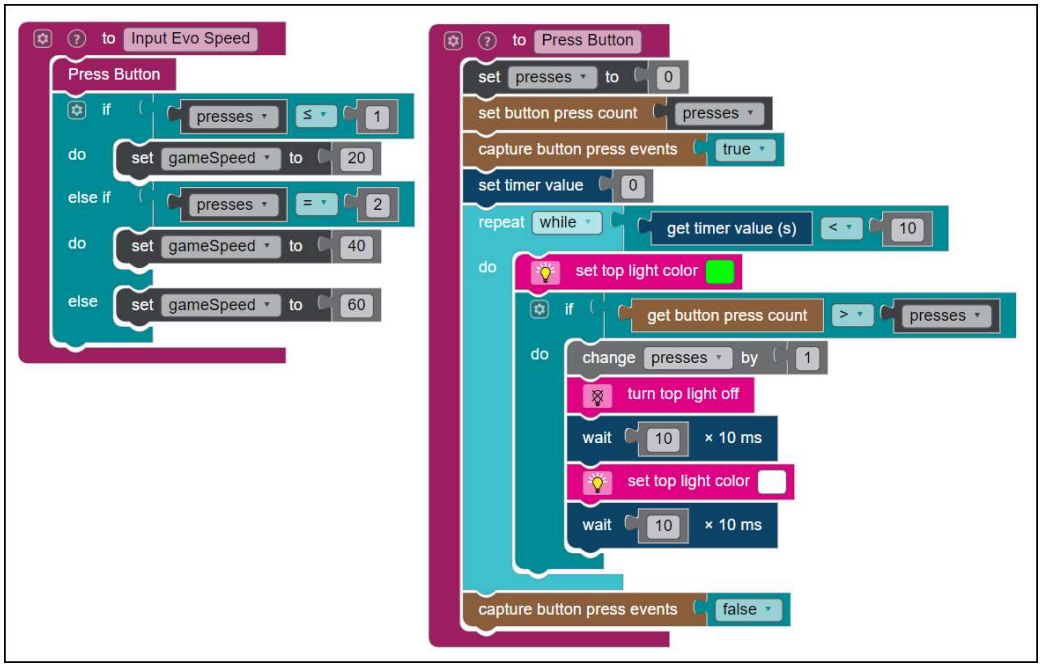

Figure 5

This subroutine provides the strength that allows button presses to be recognized and counted. It begins with a block that sets the presses to zero. The button press count is then explicitly set to presses. Knowledge of this block is important as you may need to reset the button press count to zero at various points in a program if program logic requires you to do so. Capturing button press counts is set to true, and the set time value is set to zero.

The next block is a repeat while loop, which repeats as long as the timer is less than 10 seconds. Inside the repeat while loop, the top light is set to the color green. If the button press count is greater than presses, presses is increased by 1 and the top light blinks white quickly to let us know that the press was recognized.

Once we exit the *repeat while* loop, we set "capture button press events" to false and are ready for Ozobot to follow the path selected by the user in the button presses.

The "Play Evodot Game" and "Move" Subroutines

These two subroutines are shown in Figure 6. The game begins by telling Evo to move. This means simply to set the wheel speeds to the value of the variable *gameSpeed*, which was discussed earlier. This is done in a separate subroutine called *Move* as there are several places in the program where we need to tell Evo to move. The game continues while the timer value (in seconds) is less than or equal to the game time length. The game consists of repeatedly reading the values of the front IR sensors and sensing dot colors.

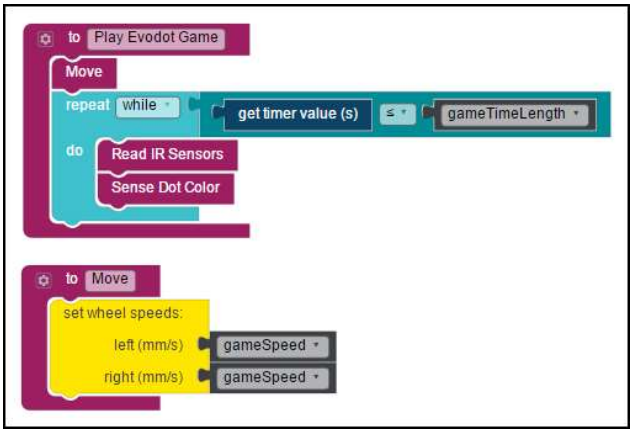

Figure 6

The "Read IR Sensors" Subroutine

The "Read IR Sensors" subroutine consists of two *if* blocks. One *if* checks the value of the left front IR sensor. If it is greater than the value of the *proximityValue* variable set in the initialization subroutine, then the player has indicated Evo should turn slightly to the right. This is accomplished by having Evo rotate -30 degrees. Note that negative rotations mean right turns, according to the OzoBlockly Reference Manual. Following the slight right turn, Evo is informed to keep moving by a call to the "Move" subroutine. Similar logic applies to the second if block, but dealing with the right front sensor.

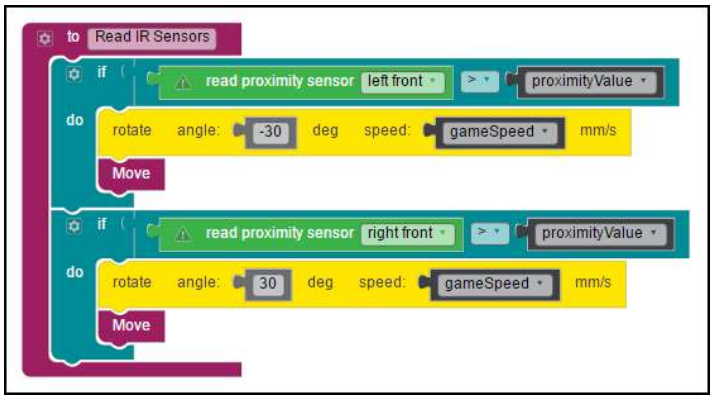

Figure 7

The "Sense Dot Color" Subroutine

The "Sense Dot Color" subroutine appears in Figure 8. It contains four *if* blocks pertaining to what Evo should do upon sensing a red dot, a green dot, a blue dot or the black border on the OzoMap. Consider the first *if*, used for sensing red. There is an *and* clause used to enforce the rule that red sense is allowed only if the previous sensed color was not also red, in which case *redOnOff* would be ON. In the event that both conditions of the *and* are true, then the "Rapid Blink Tasks" subroutine is called with the input argument color red. Similar logic is used in the *if* blocks for sensing green and blue.

The final *if* in this subroutine indicates what Evo should do upon encountering the black border of the OzoMap. The author found that a 180° rotation, accomplished by two 90° rotations (since values cannot exceed 127 with OzoBlockly), works "pretty well" to keep Evo from escaping the black border. Following the 180° rotation, Evo is told to move one centimeter (10 mm) to help get him back in the playing field. Finally, Evo must be instructed to "Move" again.

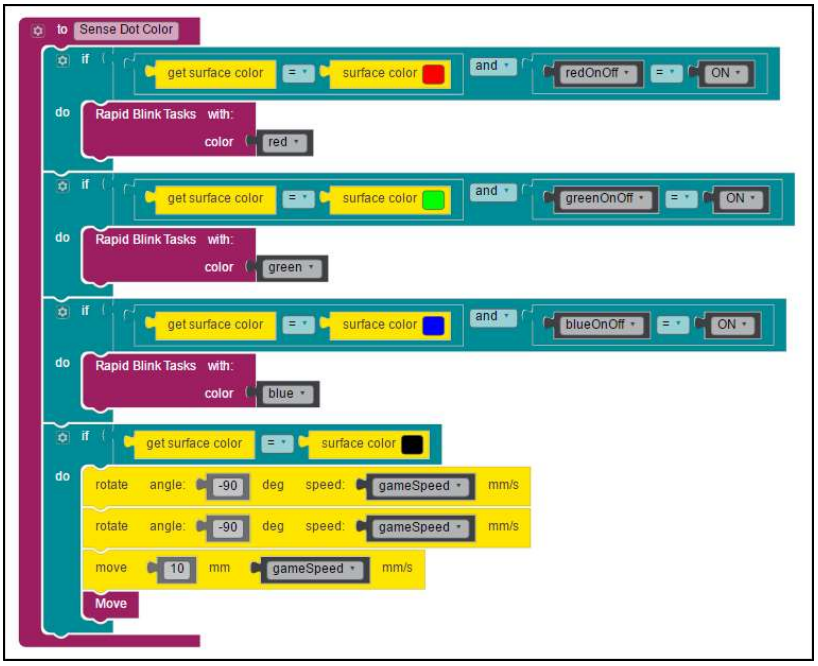

Figure 8

The "Rapid Blink Tasks" Subroutine

The "Rapid Blink Tasks" subroutine is the longest and most complex subroutine in the program. It deals with a variety of tasks necessary when Evo has sensed one of the colored dots. These tasks include blinking the top light several times to quickly let the player know that points have been earned and updating the total points earned, in addition to a number of other tasks to be discussed in detail.

The first part of this subroutine is shown in Figure 9 and the remaining part in Figure 10. The motors need to be stopped while performing these tasks—accomplished by the stop motors block, a built-in OzoBlockly block. Then depending on the color of the dot sensed, the flag for this color is set to 1 so that Evo can blink the dot color. The flags for the other two colors will have the value 0. A *repeat…times* loop then quickly blinks the top light the color of the dot on and off 5 times, with 50 ms (milliseconds) between each on or off. Note the exclamation point (!) on the set light color block. This will be seen only if you have OzoBlockly set to Bit mode, warning you that this block is not compatible with Ozobot Bit. You will also get a similar warning if you try to load this program onto Ozobot Bit, and the program will not load.

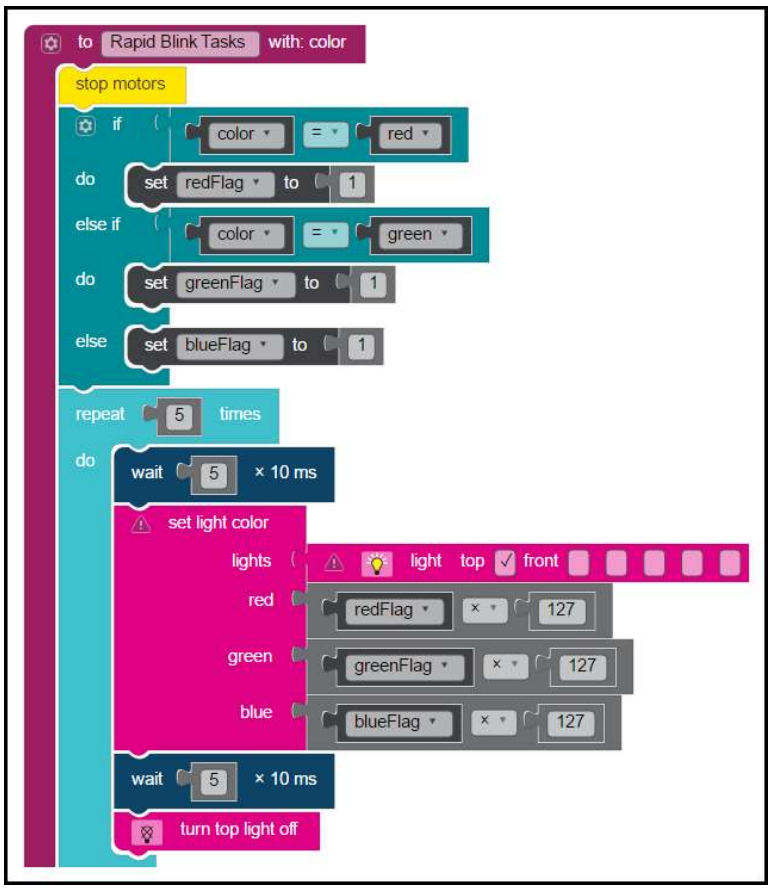

Figure 9

Now let's look at the remainder of the "Rapid Blink Tasks" subroutine, shown in Figure 10. The first thing you see is a call to the "Say Circle Points" subroutine, as the player has just crossed a circle and is being assured that the points are awarded. Next you see a lengthy *if…else if…else* block. Depending on the color (1=red,

2=green, 3=blue) of the dot sensed, tasks specific to the color are executed. These tasks are similar in nature for each of the three colors. For example, let's study the tasks for the red *do* portion of the *if* block.

The flag that was set to 1 at the start of this subroutine is reset to the value 0, so that all three flags are back to the value 0 before the next dot is encountered. Then the *redOnOff* flag is turned OFF since no points are allowed for two or more red dots in a row. *greenOnOff* and *blueOnOff* are turned ON since these color dots are now allowed to earn points for the player. Finally, the total points are increased by 1 for crossing a red dot.

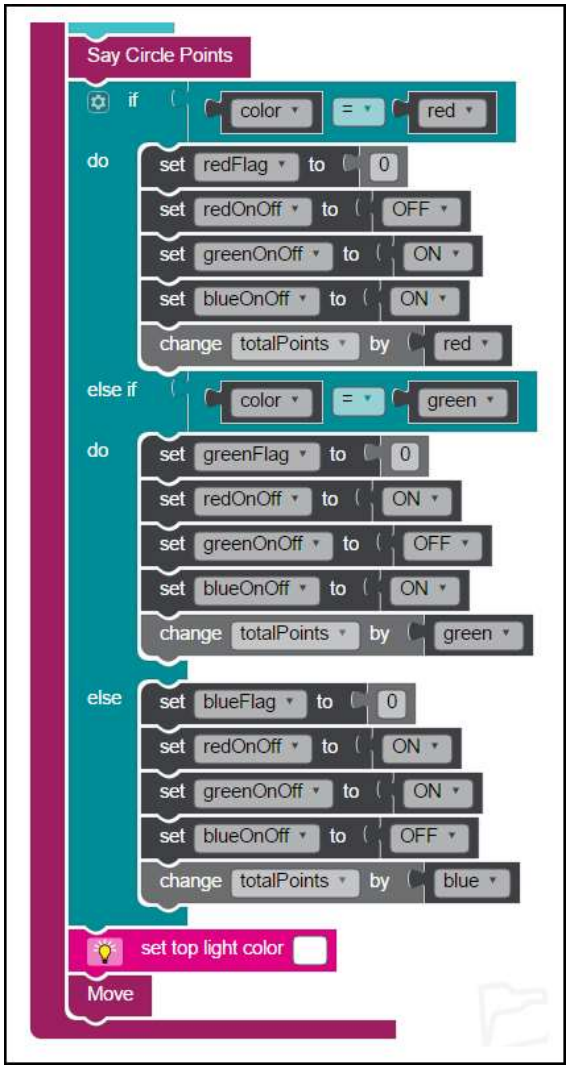

Figure 10

The "Say Total Points" Subroutine

The "Say Total Points" subroutine is shown in Figure 11. Since the game play has just completed, the motors are stopped. Then Evo says the total points scored 3 times, with 1.5 seconds between each saying of the total score.

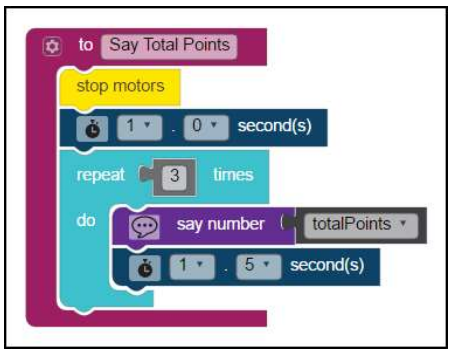

Figure 11

Running the OzoBlockly Program

1. Load *EvodotGame.ozocode* into your classroom Ozobot Evo robots and print copies of the maze on the next page for each group of students.
2. To run the loaded program on Evo, begin with Evo powered down. Press Evo's button once to turn Evo on. Then when the front lights show blue, double-press the button. Evo's top light should then be a green color. The OzoBlockly program has started running.
3. You then have 10 seconds to set the level of difficulty. None or one press of Evo's button sets Evo's speed to the lowest (easy), two presses for medium speed, and 3 or more presses for the fastest speed (challenging).
4. While the top light is still green, place Evo on the start location, facing any direction of your choice. Evo's top light will turn white and begin moving forward.
5. Guide Evo with your fore-finger near but not touching Evo's front IR sensors.
6. When Evo crosses a circle, he will stop momentarily, rapidly blink the color of the circle he has just crossed, say the point awarded for crossing the circle, and then continue moving according to your fore-finger commands.
7. When the allotted game time is over (defaulted at 30 seconds in the OzoBlockly program), Evo will stop, speak your total score three times, and then turn off.

Classroom Activity

After discussing the OzoBlockly program with the class, provide each student or student group with a copy of the *EvodotGame.ozocode* file and let them load the program onto their Ozobots and play with the program for a while. Then have them perform the following program maintenance tasks:

1. Modify the program so that Evo is "steered" by the rear sensors instead of the front sensors.
2. Design a subroutine that will allow the user to input one of three possible game times: 30 seconds, 45 seconds, or 1 minute. This subroutine should call the existing "Press Button" subroutine. Input of the game time should occur following input of Evo's speed. The top light should be red for three seconds between input of Evo's speed and input of the game time.