From a Diagram to Motion with an LLM Bot and Camera

Ozobot Editor Lesson—Student Guide

by Richard Born, Associate Professor Emeritus, Northern Illinois University

Introduction

A Large Language Model (LLM) is a language model that has been trained using self-supervised machine learning. Its training is based on an extremely large amount of text. In this lesson, you will couple an LLM Bot with a camera using the **Open AI standard** model. This LLM is *multimodal* as it can analyze *both* text and images and provide textual responses. Among many other things, the response can be a detailed description of the image as well as a detailed answer to a numerical question related to the image.

Lesson Goals

1. Input an image with a circle similar to that shown in Figure 1 using the Ozobot Editor's LLM Bot and camera, determining the degrees clockwise from point A to B, and
2. Make Ari or Evo travel from point A to point B following the circle in the direction shown by the arrow, stopping when reaching point B. You will be using Evo/Ari **Movement** blocks, **not** Line Navigation. With Figure 1, Ari or Evo would have to move 3/4 of a circle = 0.75 x 360° = 270°.

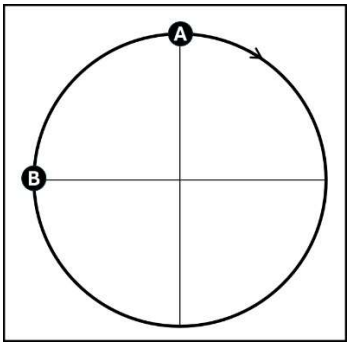

Figure 1

How To Get Evo or Ari to Follow a Circular Path

Accomplishing this requires using two blocks from the *Movement* category, as shown in Figure 2. Making the number of mm in the *move distance* block smaller will reduce the size of the circle. The number of degrees in the *rotate angle* block will cause Ozobot to rotate that number of degrees following each distance move. A positive speed of rotation will result in a counterclockwise rotation, as viewed looking down on Ozobot. A negative speed will result in a clockwise rotation.

Since Figure 1 shows a clockwise rotation, you will want to make the *rotate speed* negative in your program.

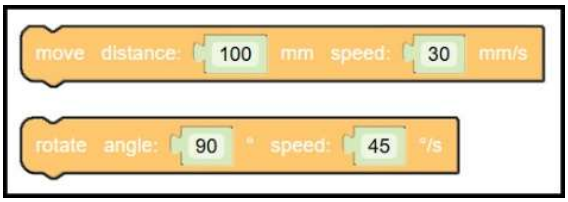

Figure 2

Figure 3 shows that increasing the number of sides from 12 to 18 to 36 makes the trace closer and closer in appearance to a circle. In your program, you will go with 36 sides. This means that after each distance move of several mm, you will want to make a rotation of 360°/36 = 10°.

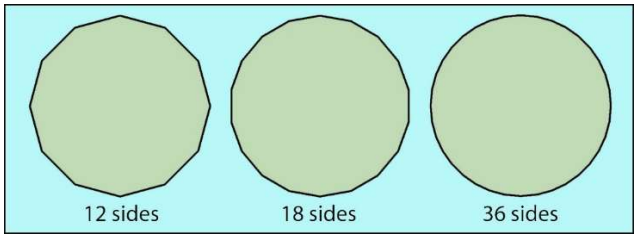

Figure 3

Addressing Goal #1 -- *Input an image with a circle similar to that shown in Figure 1 using the Ozobot Editor's LLM Bot and camera, determining the degrees clockwise from point A to B*.

Figure 4 shows the blocks used to accomplish this goal. The prompt simply poses the question asking for the number of degrees between points A and B on the circle. The result, returned in a string variable called *angularDegrees*, is shown in the INPUT/OUTPUT console. You can see that the LLM has done a very good job analyzing the image and has come up with the correct answer of 270°. **Be certain that you use the Open AI (Standard) model**—it is the best of those available and most likely to provide correct responses to all of the images that are provided by your teacher. You may get a differently worded explanation than shown in Figure 4, however.

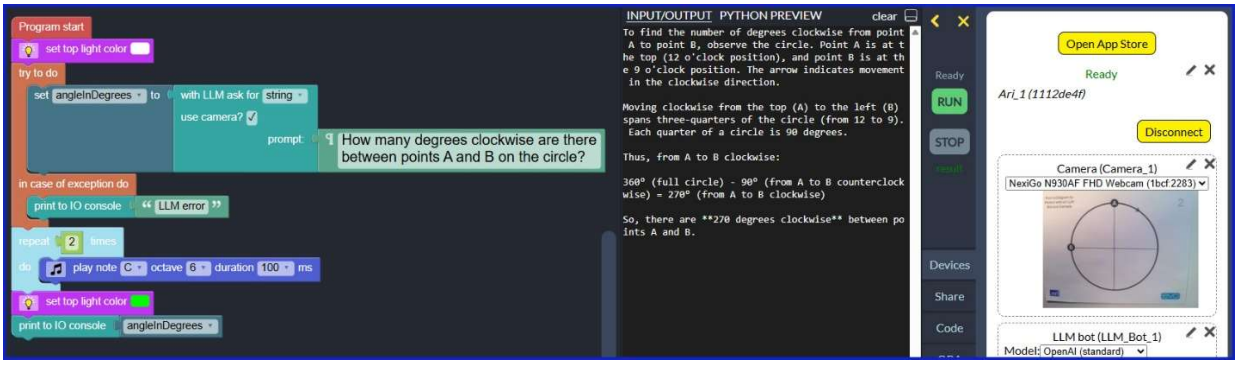

Figure 4

The prompt used in Figure 4 was quite useful, as it gives you a good feeling for the detailed logic used by the LLM to arrive at its answer. However, we only need the *number of degrees* clockwise from point A to point B when we get to solving goal #2. We don't want to worry about extracting this number from the detailed explanation provided in the response in Figure 4—that would be a rather complicated task!

So, we modify the prompt by adding "Please respond only with the numerical answer." to the end of the prompt, as shown in Figure 5. We now see that the I/O console contains only the number 270. This *number* is stored as a *string* in the variable *angleInDegrees,* which is printed to the I/O console.

The repeat loop causes two beeps to sound, just to let us know *audibly* that the LLM has determined the angle's value. The top light of Ari/Evo turns green to give a *visual* indication that the angle's value has been determined. We are now ready to address the second goal.

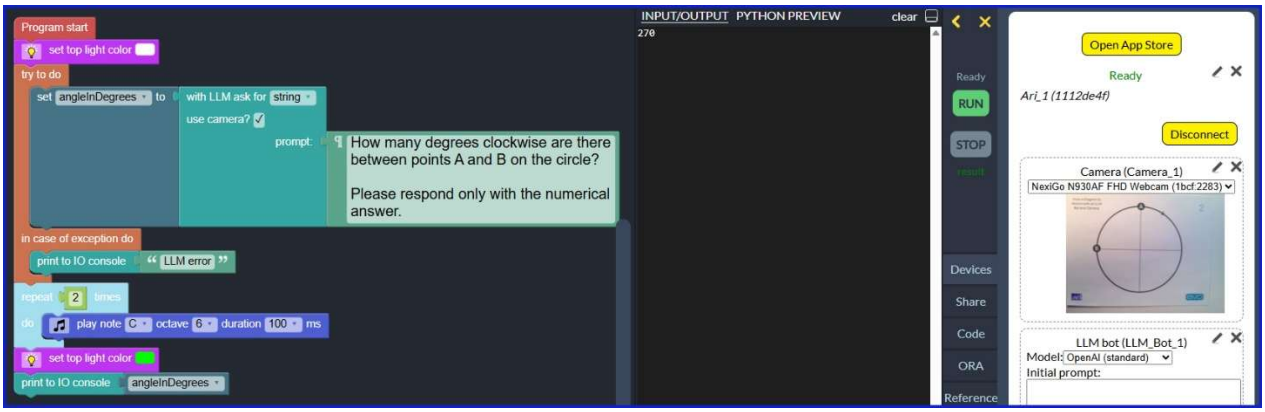

Figure 5

Addressing Goal #2 -- *Make Ari or Evo travel from point A to point B following the circle in the direction shown by the arrow, stopping when reaching point B*.

Figure 6 shows the blocks used to accomplish this goal. These blocks follow those shown in Figure 5 to complete the program. Recall that in Figure 5, the variable *angleInDegrees* was defined to be a *string* variable. We first need to convert this angle to type *number* as we will be doing some math operations with it. The alert message will appear on Ari's display or on the Editor's Terminal if using Evo. The purpose of this message is to provide the user with whatever time is needed to place Ari/Evo at point A on the circle, facing the direction of the arrow.

Finally, let's investigate the *count with* loop. An example will help you understand it. Suppose that the *angleInDegrees* is 270. That would mean that we want Ari/Evo to make 27 steps of 10° each in order to traverse from point A to point B on the circle. The move distance can be adjusted by trial and error so that Ari/Evo's motion matches the size of the circle well.

In addition, the rotate angle can be adjusted from the nominal 10° by trial and error so that point B is not overshot or undershot by Ari/Evo.

You may be wondering why you need to adjust for overshoot or undershoot. The Editor's Reference tells us that "Ozobot performs the rotations with a limited precision due to traction differences between different operating surfaces (wheel slippage), inertia, speed calibration, and its hardware design limitations. The over- and under-rotation errors can be also varied when you change the rotation speeds."

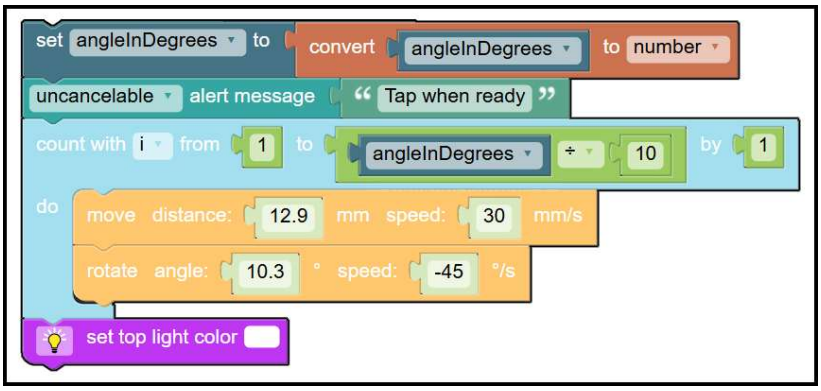

Figure 6

Exercises

Your teacher will provide you with four sheets of paper with circles already drawn, each circle having point B located at a different location on the circle. The circles are numbered 1, 2, 3, and 4.

On circle 4, there is only a point A drawn. On this circle, Ozobot is to move clockwise from point A all the way around the circle and then stop at point A. You will need to make a slight change in the prompt in order for Ozobot to travel clockwise around the circle when doing circle 4.

Try out the program on all four circles, adjusting the move distance and rotate angle until you are satisfied that Ozobot is following the circles well. Demonstrate your program for circle 3 to your teacher.

One Final Note

Our LLM does quit well with angles that are multiples of 90°. But it should be pointed out that LLMs generally have a difficult time measuring angles because they lack true geometric reasoning and spatial awareness. Any geometric concepts that they learn while being trained on large datasets are inferred from *language patterns*, not precise visual understanding.