OzoBlockly Lesson:
Evo Simulates a Garden Robot

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

Introduction

Let's say you enjoy eating fresh vegetables from a home garden, but don't like the work associated with gardening. Or maybe you're concerned that not knowing how the veggies in the grocery store are grown could lead to heath issues in addition to long-term environmental problems. Addressing these kinds of concerns was in the mind of Rory Aronson, the founder of a garden robot called FarmBot (https://farmbot.io). He describes FarmBot as "humanity's first open-source CNC farming machine". CNC is short for **C**omputer **N**umeric **C**ontrol, and *open-source* means that all blueprints and software are available free to the public for building the robot or making any desired modifications. This robot is designed for a small home garden outside or on a rooftop or in a greenhouse, allowing you to grow fresh food where it is convenient for you. It is even available as a kit.

FarmBot, controlled by an app, allows you to decide which crops to grow, where to grow them in your garden, plant the seeds, and give them the water in the exact amounts that they need. It allows you to grow a variety of crops in the same area simultaneously, typical of the small home garden. A camera detects weeds and FarmBot kills them by burying them into the ground. The camera also monitors the health of the plants. You can use pesticides, but only if you want to. You can measure soil properties such as pH and temperature, giving you a great deal of control on how to grow your crops. The app manages FarmBot's weekly schedule, so you can go ahead and take that family vacation to Yellowstone National Park or the Grand Canyon that you have been putting off for years! It does appear that FarmBot is likely to attract individuals who are somewhat technologically oriented.

In this lesson, students will make humanity's first Evo-based garden robot simulation. No, it won't have all of the features of FarmBot, but it will simulate planning the location of crops, planting seeds, and watering. This will be accomplished via an OzoBlockly program that teaches modular design using subroutines. The students also *simulate a two-dimensional array* representing the crop layout plan, using a one-dimensional OzoBlockly array in a way that reminds one of Facebook's famous Farmville game. Hopefully, Evo will avoid the dying of many acres of digital farmland as occurs with some Farmville players!

Figure 1 shows a bird's eye view of the planned layout design for our OzoBlockly crop. The light brown shaded region represents the box containing the soil for our garden. Our garden has eight rows with five plant

locations per row. We'll grow squash in rows 1 and 2, and peas in row 3. Row 4 will have two pea plants and three cabbage plants. Row 5 will be carrots, and rows 6, 7, and 8 will all be tomatoes.

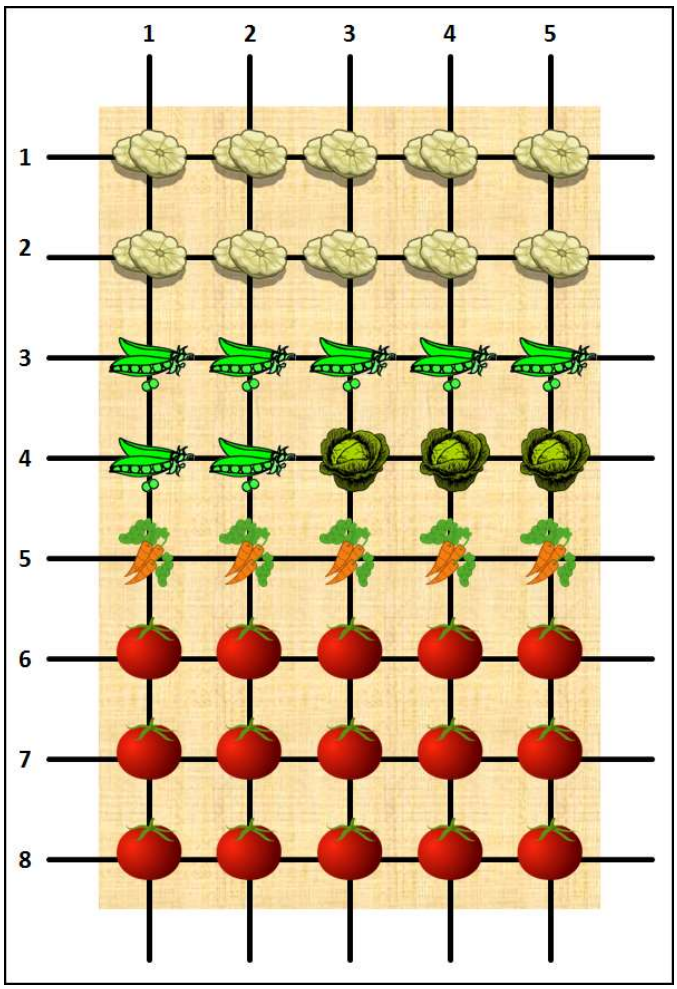


Figure 1

The Ozomap and Evo's Movement on the Map

Figure 2 shows a small copy of the Ozomap for the garden robot simulation. It should be noted that the final page of this document contains a full size Ozomap, suitable for copying for student use with Evo. Evo is placed at his home start location on the grid, facing the direction shown by the arrow. Evo's top light will always show white while he is moving. The grid contains eight numbered rows, each with five numbered positions for plants. Evo first plants row 1 from left to right. He then moves over to row 2, repeating this process. After completing the planting of all eight rows, Evo returns to his home start location and waters each of the plants in the same order as he planted them. Finally, he returns to his home location, stops, and turns off.

While planting the seeds, Evo's top light shows colors as follows−red for tomatoes, orange for carrots, yellow for squash, green for peas, and purple for cabbage. Evo also beeps momentarily while planting each seed at its planned location. While watering the seeds, Evo's top light shows a blue color and Evo plays a fast octave of notes intended to simulate the sound of dripping water.

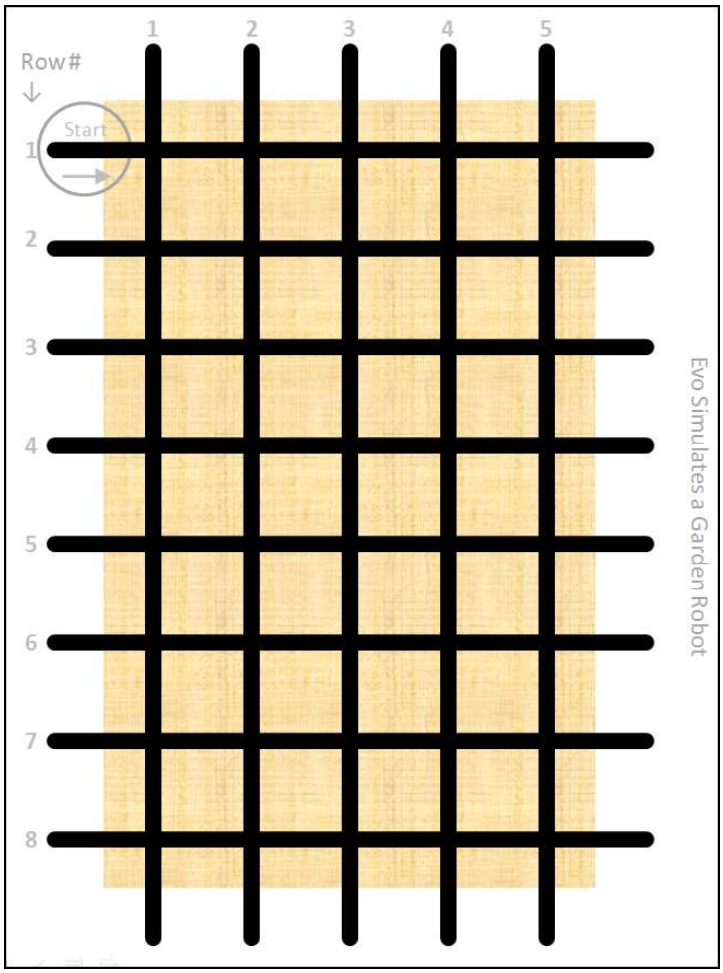


Figure 2

Running the OzoBlockly Program

1. Load the OzoBlockly program *GardenRobotEvo.ozocode* onto Evo. This program is one of the files accompanying this document.
2. With Evo turned off, press the button once to turn Evo on. Evo's front light should show light blue.
3. While the lights are light blue, double-press the button. The top light should turn white.
4. While it is white, you have three seconds to place Evo on the start location on the Ozomap facing the direction shown by the arrow.
5. Evo will then plant the seeds and water them, as discussed in the previous section of this document.

Discussion of the OzoBlockly Program

Figure 3 shows the structure of the OzoBlockly program, including the main program in gray and 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.

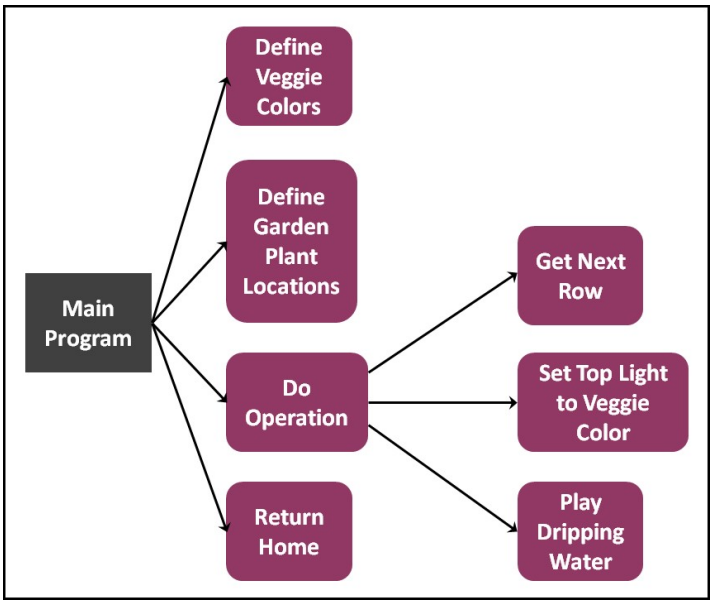


Figure 3

The Main Program

Figure 4 shows the blocks making up the main program. The number of rows in the garden is set to 8, and the number of columns (*i.e.*, plant locations per row) is set to 5. The operation of planting the seeds is set to the value 1, and the operation of watering the plants is set to 2. This allows us to refer to the operations by name rather than by number throughout the program, promoting self-documentation and ease of maintenance. The top light is then set to the color white, and the user is given 3 seconds to place Evo on the start location of the Ozomap. Subroutines are then called in the order needed, and the line-following speed is set to 45 mm/s.

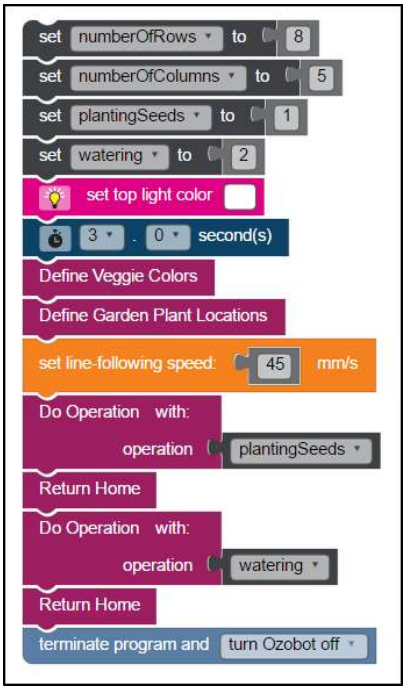


Figure 4

The "Define Veggie Colors" Subroutine

The blocks in the "Define Veggie Colors" subroutine are shown in Figure 5. First, the numbers 1 through 5 are assigned to the colors red, orange, yellow, green, and purple, respectively. Doing this allows us to refer to colors by name rather than number, promoting self-documentation and easing program maintenance. Then each of the vegetables in the garden is assigned its colors, which is the color that Evo will display when planting the seed at its assigned location in the garden grid.

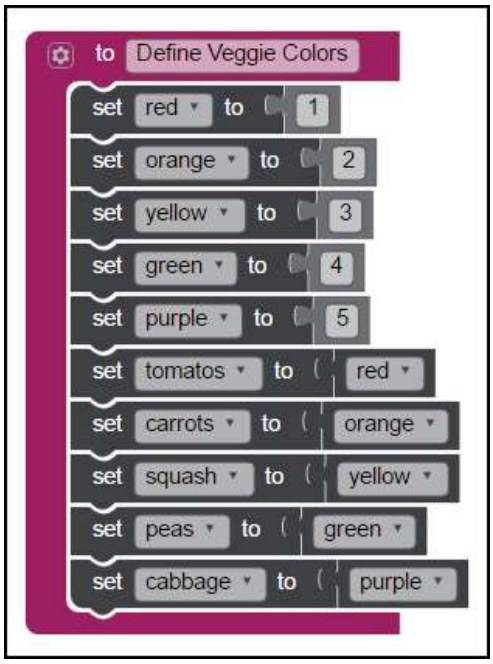


Figure 5

The ***Veggie Array and "Define Garden Plant Locations" Subroutine***

Figure 6 shows the *veggie* array declaration and the first half of the "Define Garden Plant Locations" subroutine. The reason for setting the size of the array to 86 elements will become clear during our discussion. The subroutine simply consists of one *in array … set element* block for each plant location in the garden grid. The element number is the index into the one-dimensional OzoBlockly array *veggi*e. In order to simulate a two-dimensional array like our rectangular garden grid, we treat an index such as 32, as an example, as meaning row 3, position 2 in the grid. Therefore, the first five *set element* blocks in the subroutine indicate that all five positions in row 1 will be planted with squash. The same holds for row 2. All five positions in row 3 will contain peas. The first two positions of row 4 will contain peas, while the remaining three positions of that row will contain cabbage. The last position in row 8, the final row, will have an index into the array *veggies* of 85. This is the highest numbered index we will need, and since arrays start with an index of zero, the number of elements in our array is 86. The disadvantage of simulating a two-dimensional array as we have done is that we have reserved memory for 86 array elements but have actually used only 8x5=40 of these locations. With this technique, the maximum number of rows, each with five columns, would be 12, since OzoBlockly allows for an array whose maximum length is 127. In this case, the last array index actually used for our garden would be 125 (row 12, position 5).

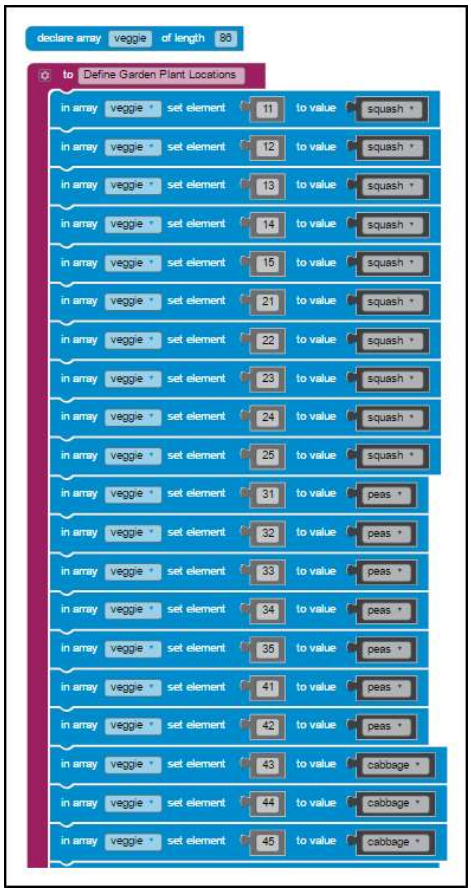


Figure 6

The "Do Operation" Subroutine

Figure 7 contains the blocks for the "Do Operation" subroutine. This subroutine handles both the planting of seeds and watering the plants, dependent upon whether its input operation is *plantingSeeds* or *watering*, respectively. The subroutine contains double-nested *count with* loops, with the outer loop varying from 1 to the number (8) of rows in the garden grid, and the inner loop varying from 1 to the number (5) of columns (positions in the row). Inside these loops is an *if…else if* block. The *if* clause handles the planting operation, while the *else if* clause handles the watering operation. Regardless of the operation, Evo needs to move to the next intersection in the garden grid and pick the direction *straight*, so these two orange-colored blocks precede the *if* block.

First, we look at the blocks for the planting operation. The index into the veggie array is found by multiplying the row number by 10 and then adding the column number. Then the veggie color is set to the value of the array element whose index is *veggieIndex*. A subroutine is then called to set the top light to the veggie color. A note is played for 0.2 seconds while Evo remains at the plant location for one second.

There are only two blocks for the operation of watering. The top light is set to the color blue (representing water) and a subroutine is called to play the sound of dripping water. Finally, regardless of whether the operation is planting or watering, the top light is returned to the color white as Evo resumes moving. When the inner loop is completed, a subroutine is called to move Evo to the next row in the garden grid.

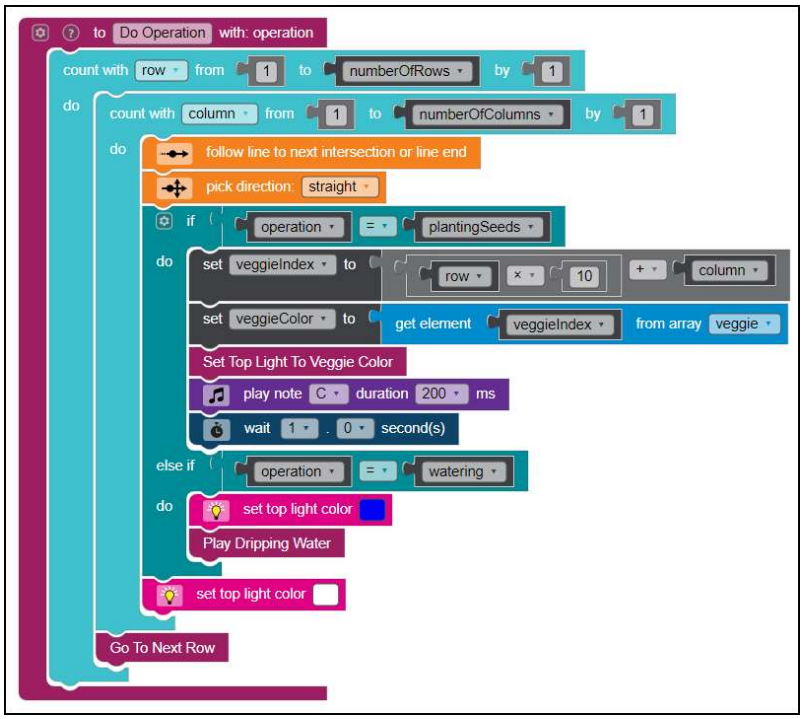


Figure 7

The "Set Top Light To Veggie Color" and "Play Dripping Water" Subroutines

These two subroutines are shown in Figure 8. Both are quite straight forward and require minimal discussion. In the "Set Top Light to Veggie Color" subroutine, the top light is set to a color similar to that of the vegetable itself via an *if* block with multiple *else if* clauses. The "Play Dripping Water" subroutine uses a sequence of *play note* blocks running down an octave. Each note is played for 0.7 seconds, and Evo pauses for 1 second at each plant location. It should be mentioned that you need to have waits after each *play note* block that are at least as long as the time duration of the note.

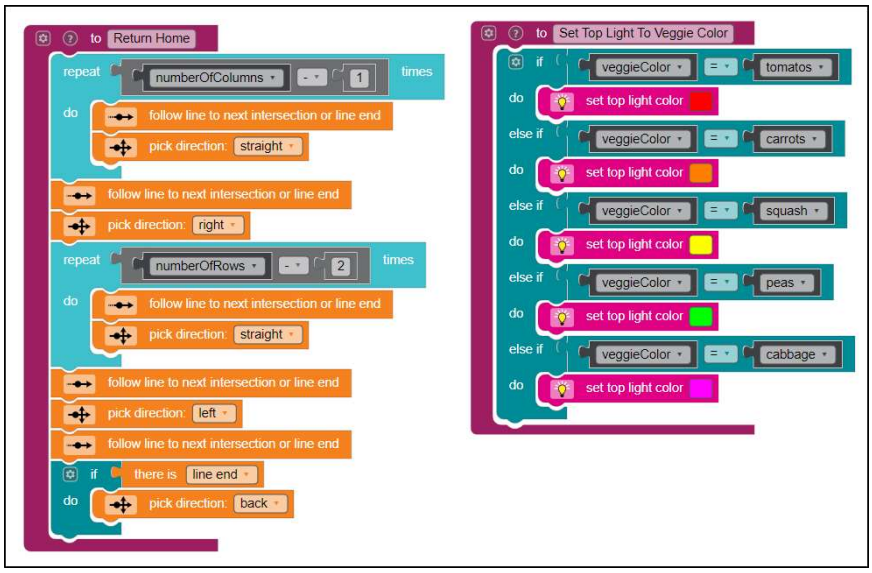


Figure 8

The "Go To Next Row" Subroutine

This subroutine is shown in Figure 9. The purpose of this subroutine is to get Evo to move from point A on one row to point B on the next row, in the manner shown by the arrowed red line in Figure 10. He accomplishes this by making U-turns at the ends of each of the rows. With the red line of Figure 10 in view, it should become clearer what the blocks are accomplishing in the subroutine. Note that the *if* block keeps Evo from going to the next row if he has just finished the ***final*** row in the grid, for in this case there is no next row.

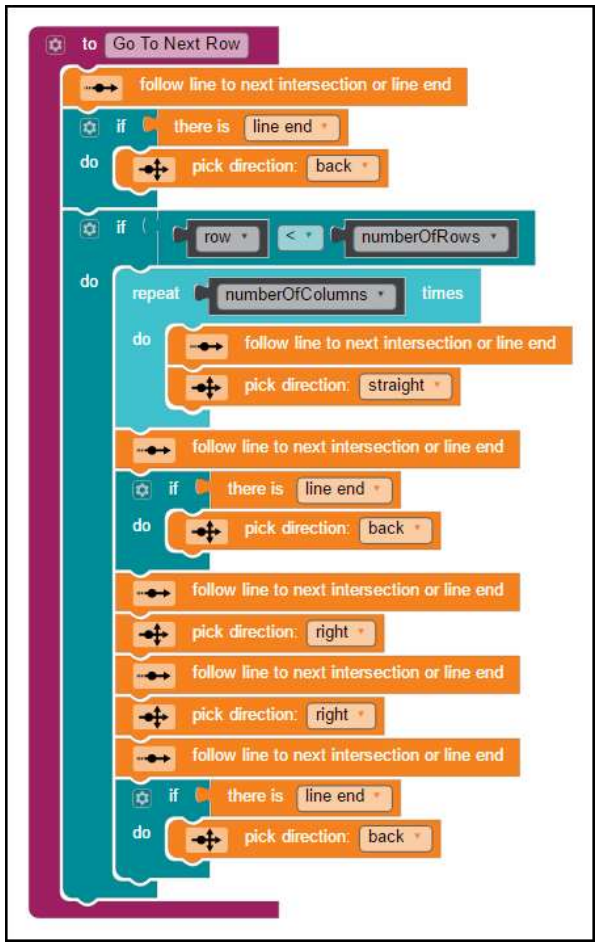


Figure 9

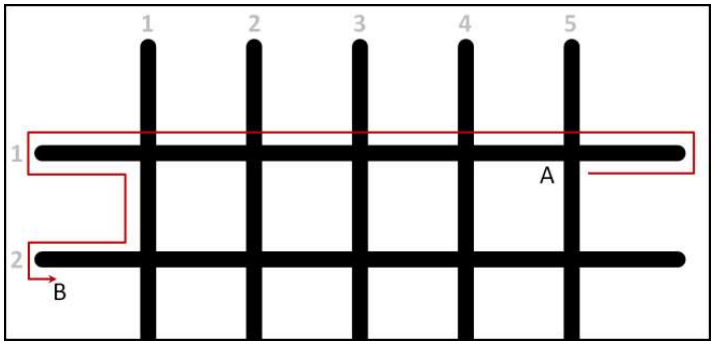


Figure 10

The "Return Home" Subroutine

This subroutine appears in Figure 11. The purpose of this subroutine is to get Evo to move from point C on row 8 to point D on the row 1, in the manner shown by the arrowed red line in Figure 12. With the red line of Figure 11 in view, it should become clearer what the blocks are accomplishing in the subroutine.

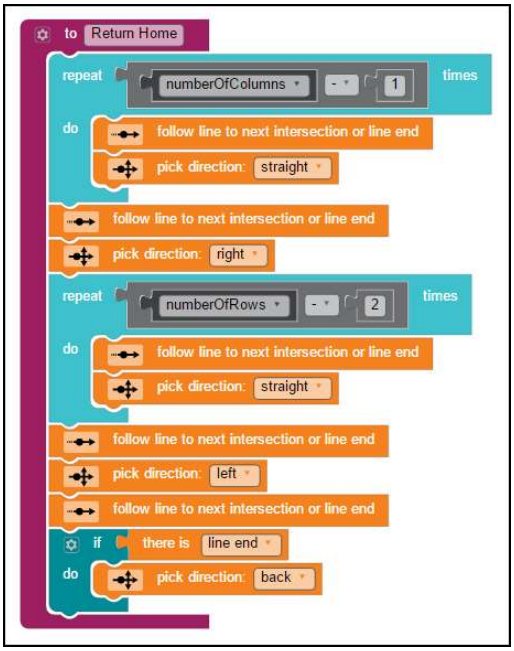


Figure 11

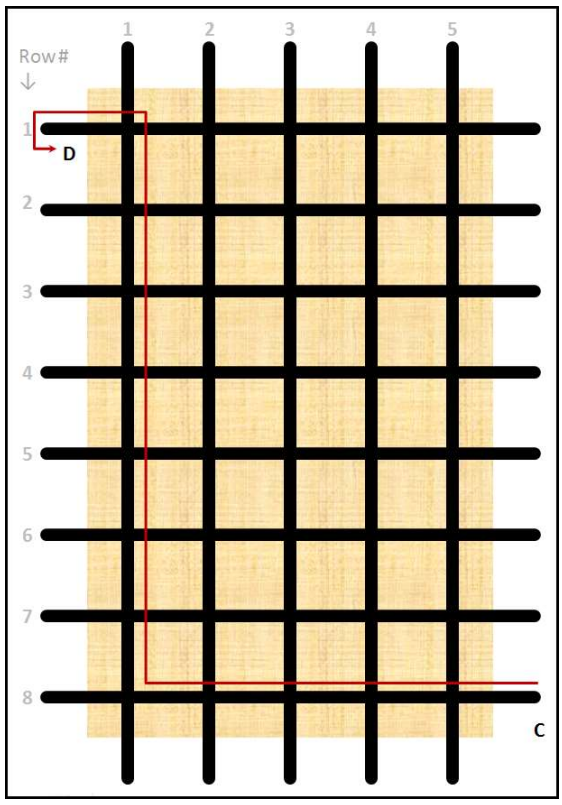
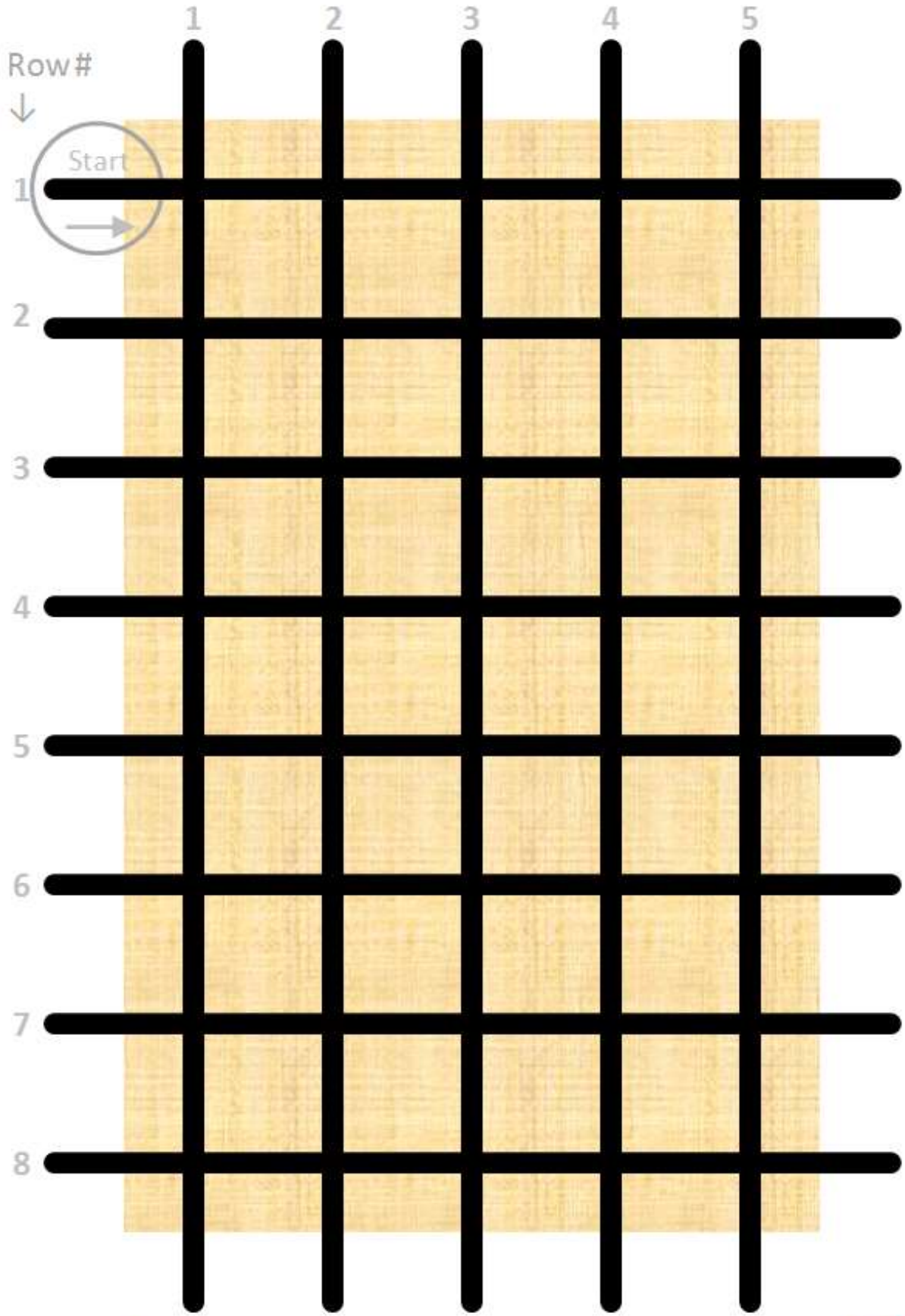


Figure 12

Student Activities

- (Easy) Change the grid layout so that the first three positions of row 6 grow carrots and the entire row 8 is cabbage. Then load and run your modified program to see if it is correct.
- (Medium) Make modifications to the program so that it will handle an additional vegetable—beans. Row 8 should be changed from tomatoes to beans. Beans should be displayed by the top light showing the color aqua. Then load and run your modified program to see if it is correct.
- (More challenging) The OzoBlockly program we discussed in this document was written for Evo. Make all necessary modifications to that the program can run on Ozobot Bit. Keep the number of planting positions per row at 5. You will probably have to reduce the number of rows significantly, but should keep this *as large as possible* without exceeding Bit's memory of 1024 bytes. You can check memory usage by clicking the gears tab on the far right of the OzoBlockly screen. Since Bit has no sound capability, Bit should just display vegetable colors while planting and show blue when watering plants. Load and run your modified program on Bit to see if it is correct.



Evo Simulates a Garden Robot