

OZOBOT'S OZOGROOVE INTRO 2

CODING AND GEOMETRY

What students will learn

- Programming Ozobot using moves/functions
- Analyze and decompose geometric figures and translate them into Ozobot's movements

Topics

- Computer science: programs and functions
- Robotics: programming Ozobot and executing the program
- Math: geometry, decomposing a geometric drawing into a series of movements

Challenge

Write, debug and run a program composed of several functions to make Ozobot go in a path that forms the letters OZO

Real-life connection

What are the jobs computer scientists do?

Age

Grades 3-12

Ozobot skill level

Beginner

STEM topics

- Computer science: programs and subroutines
- Inter-disciplinary concept: use robotics to study programming and math

Common Core Standards

CCSS.MATH.PRACTICE.MP1 Make sense of problems and persevere in solving them.

CCSS.MATH.PRACTICE.MP2 Reason abstractly and quantitatively.

CCSS.MATH.PRACTICE.MP5 Use appropriate tools strategically.

CCSS.MATH.CONTENT.3.G.A.2 Reason with shapes and their attributes.

CCSS.MATH.CONTENT.4.G.A.2 Draw and identify lines and angles, and classify shapes by properties of their lines and angles.

Materials

- Ozobots (about 1 per group of 3 students), charged
- Digital tablet (iOS or Android operating system), charged and screen brightness set to 100% (one tablet per ozobot/group of students)
- OzoGroove app (download for free on iOS app store or google play, go to settings and select: "keep tablet awake")
- Printout #1, one per group
- Optional: printout of lesson PDF, one per group, if students are learning self-guided

Note

Students need to see the diagrams in order to do the exercises. You can print out the entire lesson (without printout #1) which includes the diagrams if you want the students to read along. Otherwise, just print out printout #1 only, without the lesson plan.

Age

Grades 3-12

Ozobot skill level

Beginner

STEM topics

- Computer science: programs and subroutines
- Inter-disciplinary concept: use robotics to study programming and math

Estimated duration

1 or 2 class sessions

LESSON

Differentiation: This lesson can be taught to students of many different ages – all of them should be very comfortable with the programming even if they don't have any prior programming knowledge. You can vary the level of difficulty by giving students none, a few, or all of the hints in chapter 3 and for the challenge.

Note: Prerequisite for this lesson is lesson 4 – Coding Basics

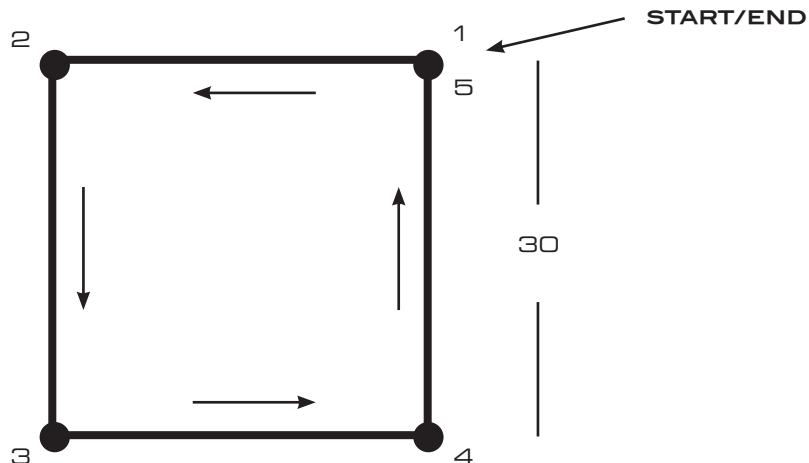
1. Make a square!

This lesson is all about making Ozobot go in certain formations: squares, letters and so on. It's much like choreographing a dance we did in the last lesson, but without the music. In fact, we will be using the same app OzoGroove to program Ozobot!

Get your tablet ready and open OzoGroove. Don't forget to set the screen brightness to 100% and calibrate your Ozobot on the tablet (just in case you don't remember, the calibration can be done in the Ozobot Tuneup in OzoGroove).

We are going to use the move creator to make Ozobot go in a square, so go to OzoGroove's main menu and choose "Move Creator" on the bottom of the screen. In the creator, tap "Create a new move" in the popup and select "No music" in the next popup.

We want to make Ozobot move according to the following diagram (also on printout #1, part 1)



Ozobot is supposed to start at point 1 and go in a square counter-clockwise to end back at the start. Each side of the square is to be 30 units long.

Suppose Ozobot starts out at point 1 looking west, i.e in direction of point 2. The first move should be going forward 30 units.

Choose the "Forward" category in the editor and you will see the instruction "Forward 30" in the menu below. Drag and drop the instructions into the timeline, right after the start.

At this point, Ozobot would arrive at point 2, but to continue the square, we have to make Ozobot turn to face point 3. If Ozobot turns left, then Ozobot would have to turn a right angle, which is 90 degrees (a quarter of a full 360 degree circle).

Choose the "Left Turns" category and drag and drop the instruction "Left 90" into the timeline, right after the "Forward 30" instruction.

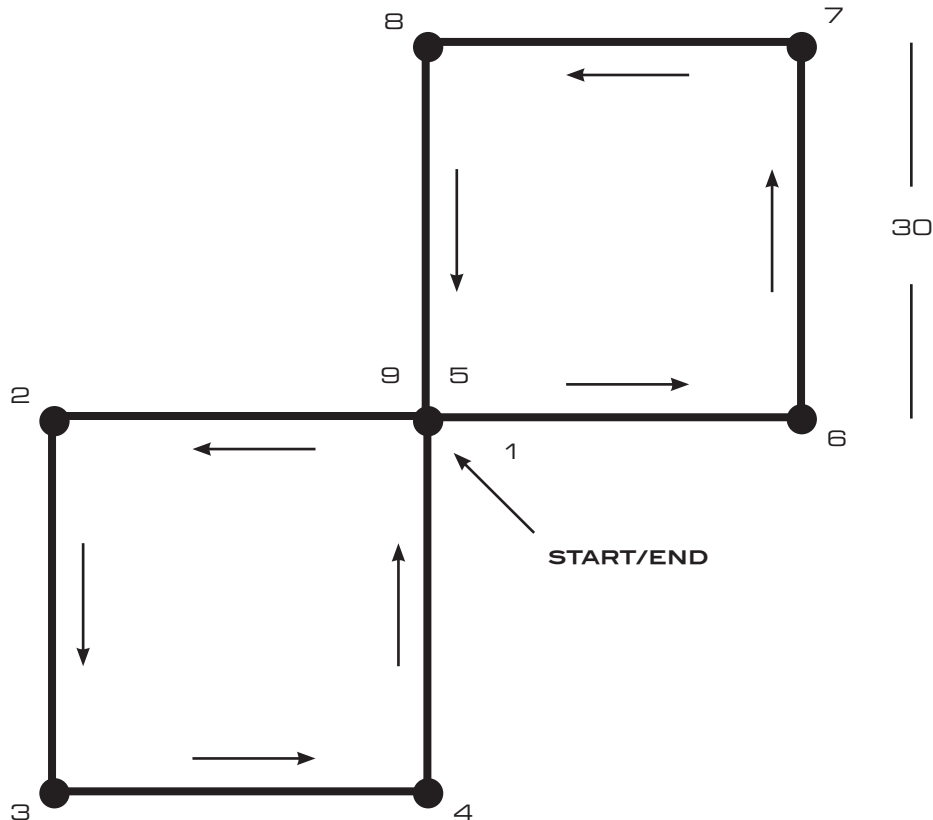
Continue assembling your move in this manner until your instructions make Ozobot reach point 5. Note that you don't have to turn Ozobot when reaching point 5.

To test if your program is correct, save it first (for example under the name "Square 1" in a new category "lesson 5"), then press "Dance." On the dancefloor, tap the pod you want to use and place Ozobot on the pod. Make sure that Ozobot's middle color sensor is centered on the pod, but adjust Ozobot's position to be ready to start the square. This means turn Ozobot so that Ozobot is facing west, like sitting on point 1 facing point 2. Unselect the "Loop Dance" slider since we want Ozobot to walk the square only once. Now press "Load Ozobot" and when the loading is done, tap "Dance" and watch Ozobot move.

Is Ozobot moving like you thought it would? If not, debug your program by going back to the editor, making changes, saving and returning to the dancefloor to check if your program is correct. Repeat this until you got it right. If you want to double-check your program, you can compare it to solution 1 at the end of this lesson.

2. Now make a butterfly!

Now we are ready to make Ozobot go in a more complex shape like this (also on printout #1, part 2)



If you use your imagination you may be able to see the wings of a butterfly, so let's call it that for now.

This time we will construct a program using the "Square 1" move we just created. So go to the main menu and select the "Dance Editor". As before, create a new dance and select "No Music".

Take a look back at the butterfly diagram above. The first 5 points are arranged in exactly the same way as in the move "Square 1", so it would be good if we could reuse this move here.

To add the move to your program, scroll the categories all the way to the right end to find "Lesson 5". Select it and you will see the "Square 1" move right below. You can now drag and drop this move into the timeline right after the start.

Now imagine how Ozobot moves along the path. After completing "Square 1", Ozobot will arrive at point 5 facing up to point 8. But this is not the way Ozobot is supposed to go. What kind of turn do we need to add to the timeline so Ozobot will be facing right to point 6? **If you know, drag and drop the turn into the timeline. If you need help, please see solution 2 at the end of this lesson.**

Now Ozobot is positioned properly for the second wing of the butterfly. Turn the page with the diagram upside-down. What can you see now? How does the second wing compare to the first one?

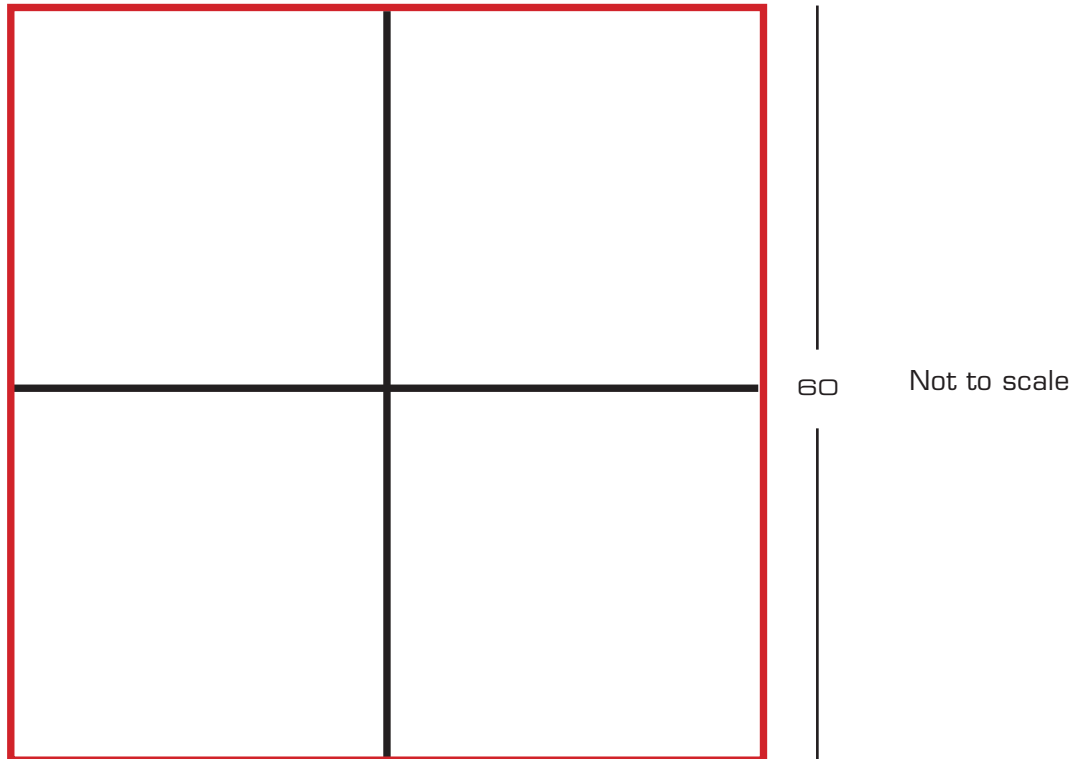
You will notice that, looking at it upside-down, the second wing is exactly the same as the first one. Imagine Ozobot executing the instructions from "square 1" starting at point 5 facing point 6. **Those moves will bring Ozobot to the end (point 9), so let's add another "Square 1" move to the timeline.**

It's now time to try out our program. Save the dance for example as "Butterfly" and try it out on the dancefloor. Remember to turn off "Loop Dance" again. Does Ozobot's move resemble the butterfly outline? If not, make changes in the editor and try again until it's just right.

Notice that we have used the "Square 1" move twice for the "Butterfly". In programming, we would call the "Butterfly" the program and "Square 1" its subroutine or function. It is a separate unit of the code that performs a specific task. It can be "called" (i.e. used) in the program whenever needed. It can be used as many times as necessary and you can see that in our case we have used it twice.

3. Back to Square 1

Imagine the following drawing shows the map of the electricity wires on a grid in your neighborhood (also on printout #1, part 3):



The electrical wires are the red lines on the big square. Ozobot is in charge of checking the wires. Ozobot's task is:

- To check the wires by going along every side of the big square
- Ozobot has to drive along the entire red outer square, but not necessarily all at once in a continuous motion (i.e. Ozobot may turn onto a black line in at any time)
- Ozobot must go on each part of the outer red lines only ONCE
- Ozobot can go on the black inner lines as often as needed
- And, making it very difficult, Ozobot can only move using the "Square 1" subroutine combined with left and right turns
- Ozobot can start and end on any corner or intersection of the grid

You are the computer programmer and in charge of programming Ozobot for this task. Assemble your program in the “Pro Dance Editor” (without music just like in the “Butterfly” exercise). Make sure Ozobot follows the guidelines above!

If you want, read the following hints. Or, for a challenge, skip ahead to the next paragraph.

- Notice that the black lines are dividing the big square into four smaller squares
- When dividing a square into four equal squares like this, all small squares are the same size
- The smaller squares are the same size as the square we used for “Square 1”
- When dividing a square into smaller squares, all angles are still right angles (90 degrees)
- You might want to start Ozobot on the center intersection

When you are ready to try out your program, save it (for example as “Electricity Square”), go to the dancefloor, turn off “Loop Dance”, and tap “Dance”. Ozobot will execute your commands on the tablet. Does it look like Ozobot is making a big square just like in the diagram? If not, debug your code as before by going back to the editor and making your adjustments.

There are several solutions to this problem. For an example of one of them, see solution 3 at the end of this lesson.

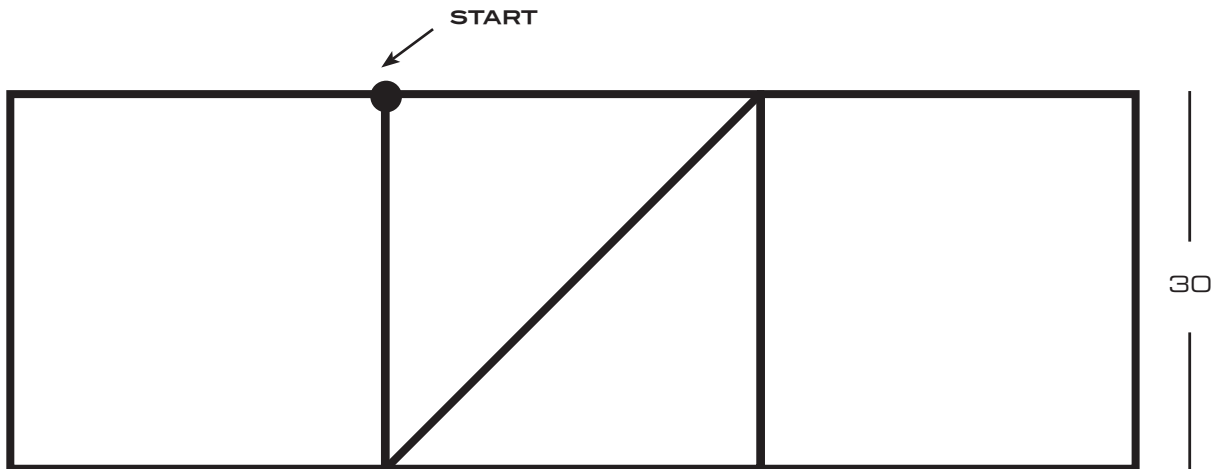
4. Computer science careers

Writing programs like we just did are the very basic forms of what programmers are doing as part of their jobs. After studying Computer Science at the university, there are many different jobs that can be done with that degree. Here are just some of them:

- *Software developers* write programs for all kinds of different purposes, for example for banks, hospitals, other companies, and for many things we use at home (like apps for computers, tablets and phones).
- *Database administrators* develop ways to store, go through and understand very big amounts of data (i.e. information).
- *Computer hardware engineers* design and test computer components.
- *Web developers* create and program the technical structure of websites and make sure that web pages function properly.

Challenge

The final challenge is for Ozobot to go in a path that resembles the outline of OZO, i.e. like in the following diagram (also on printout #1, part 4)



You are again in charge of programming Ozobot. Make sure that Ozobot traces every line ONCE and only ONCE. Ozobot has to start at the indicated starting point and face west (i.e. to the left). It will be easier if you try to use the subroutine “Square 1” as often as possible.

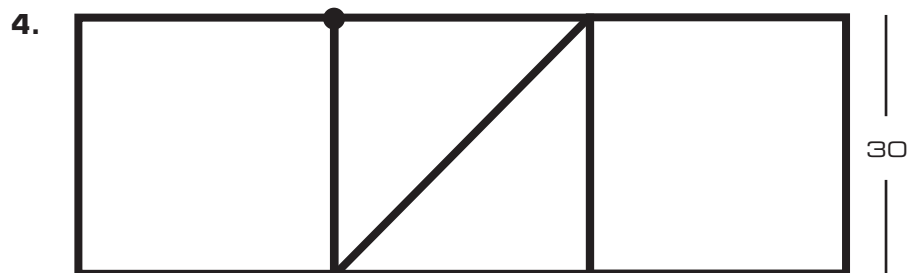
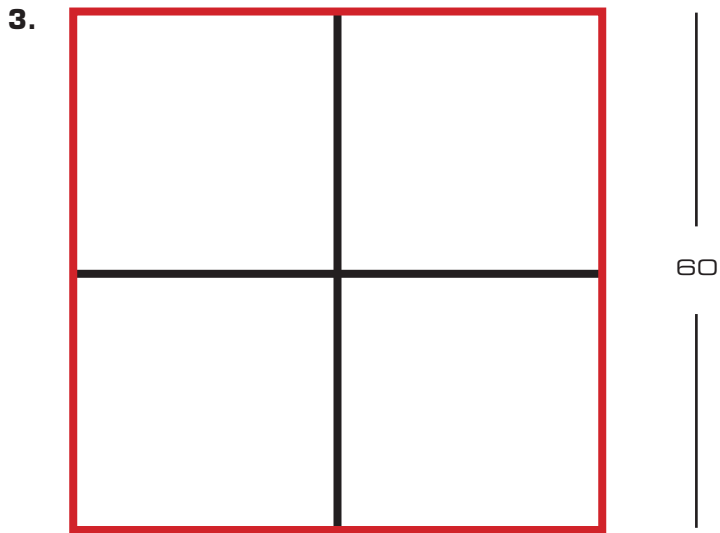
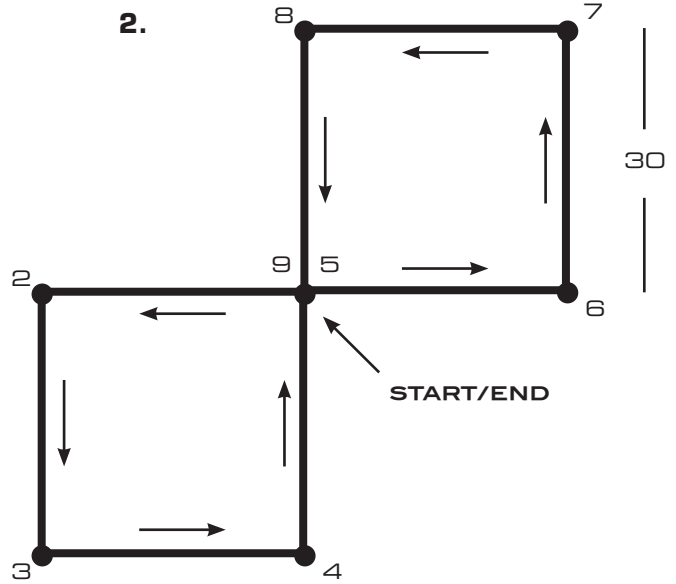
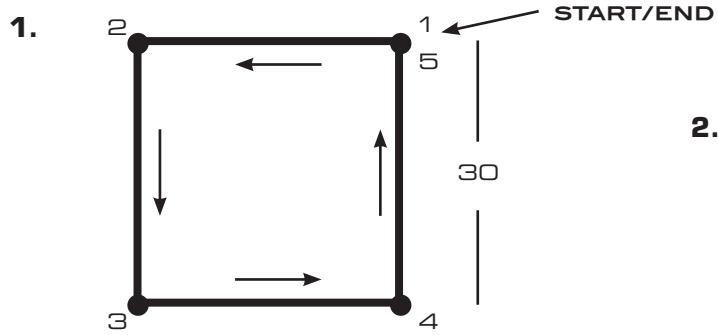
Start creating your program now using the same methods of writing and debugging as for the “Electricity Square” of chapter 3. You may want to read through the hints in the next paragraph first if you need help to get started. There are several solutions to this problem. Once you’re done, you can see an example of one of these in part 4 of the solutions at the end of this lesson.

Hints:

- There are two squares in this diagram, try to find them! Can you use your “Square 1” subroutine for both of them?
- Look at the Z in the middle. What are all of the angles? These angles determine which turns to take. You may have to use two instructions to make one turn.
- The diagonal line of the Z is actually a bit longer than 30 units (how long exactly?), but for this exercise it’s ok to use “Forward 30”

Have fun!

Printout # 1



Solutions

Solution 1

Place Ozobot looking west.

Forward 30,

Left 90,

Forward 30,

Left 90,

Forward 30,

Left 90,

Forward 30

Solution 2

Place Ozobot looking west.

Square 1,

Right 90,

Square 1

Solution 3

Place Ozobot looking west.

Square 1,

Left 180,

Square 1,

Right 180,

Square 1,

Left 180,

Square 1

Solution 4

Place Ozobot looking west.

Square 1,

Right 90,

Forward 30,

Right 90,

Right 45,

Forward 30,

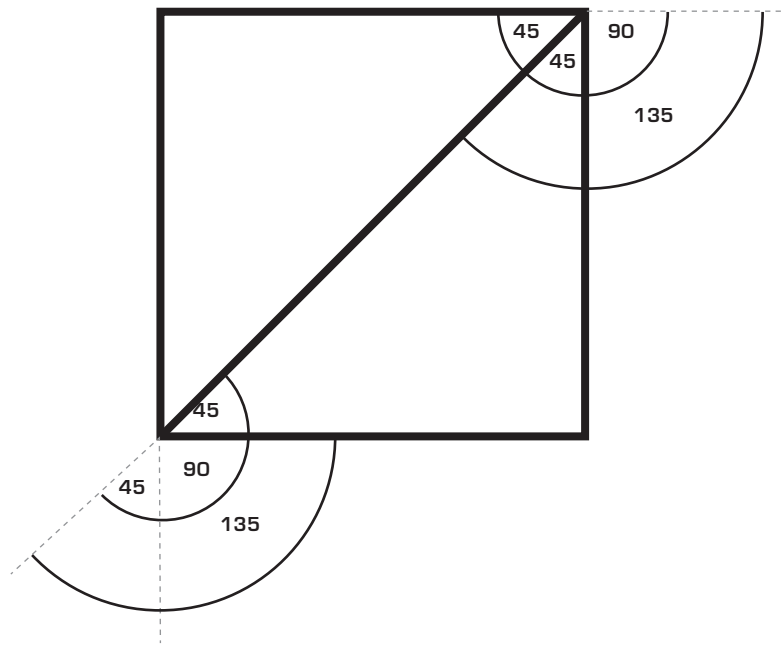
Left 45,

Left 90,

Forward 30,

Square 1

These are the angles for Z:



This is the path Ozobot is going in the solution above

