ELEMENTARY COMPUTER SCIENCE WITH GAME DESIGN

LINE-END SEARCH

LESSON 2

Summary

Ozobot follows the lines of a maze until it sees the red line-end, where the program ends. Students learn how to program line navigation in OzoBlockly.			
CS Topics: Loops and breaks, conditional logic, sensor input/output, + line navigation		**Learning Outcomes:** Program line following commands and color sensors with conditional logic.	
Grades: 2-5	**Coding Level:** Beginner	**Map Style:** Free Movement	**Game Mechanic:** winning

Introduction

Welcome to the second lesson in Ozobot's Elementary Computer Science with Game Design! Students will quickly develop their coding knowledge and intuition by 'deconstructing' a game's program to see what type of code creates each action, then create their own program. Leaders in computer science teach themselves new skills through exploring and modifying sample programs this way. These lessons are also great for teachers still learning coding, themselves. Read the Elementary CS with Game Design README for more about this series.

In this game called "Line-End Search", Bit or Evo is programmed to travel down branches of a maze, always taking right turns, until it finds the red line-end. To reach the maze's solution with the shortest program, we blend in 'if' statements to see the colors of line-ends and to see the intersection shape. (All codes are explained in ABOUT THE CODE, below.) Students will learn how to use Ozobot's 'line navigation' code blocks, which are used to choose what decisions Ozobot will make at intersections and line-ends.

To show how the logic of this program works, print the game's map and black out the red. You will run Ozobot on the map and see how it will continue through the map forever. The reason that a program like this works is because of the shape of the maze, called a 'binary tree' because every branch splits in two. Next, you will play the program on an unedited first map to show how the program helps Ozobot end on red. After

students see how Line-End Search is played, the class will compare the actions of the bot to the program itself.

There is a second, optional version of the program that is 'hard coded', which means that each intersection decision was made in the program ahead of time. This can make for a more direct path to the maze finish, but increases the possibility for bugs and errors. Showing this alternative program is for showing other possible ways to play this maze, and show why the original program is most useful.

The task for students is to edit the given program to help Ozobot reach the blue line end. You can offer either example program, or both, for students to edit. If there is time, students can make their own "binary tree" maps (where each line branches into two lines) with programs to help their bot explore it.

To get started, read each section of this lesson, below, before teaching the lesson. SETUP lists the preparation steps needed for this lesson, ABOUT THE CODE explains the programming concepts used, and LESSON OUTLINE is a simple step-by-step walkthrough for teaching this lesson. We recommend the teacher tries out the program and map before class to clarify the coding concepts and game play.

Questions or comments about this lesson? We'd love to hear from you! Email us at ozoedu@ozobot.com

SETUP

'LINE-END SEARCH' OZOBLOCKLY PROGRAM ozo.bot/line-end-search-game
OPTIONAL HARD CODED VERSION ozo.bot/line-end-search-hardcoded
OPTIONAL CHALLENGE MAP SOLUTION ozo.bot/line-end-search-challenge
OPTIONAL CHALLENGE HARD CODED SOLUTION ozo.bot/line-end-search-challenge-hardcoded

REQUIRED MATERIALS

- **FOR DEMO:** 1 Bit or Evo (pre-programmed, link above) & 2 printed maps.
- 1 Bit or Evo per group,
- 1 printed copy of both game maps per group (attached),
- 1 computer or tablet with wifi per group,
- 1 printed student worksheet per student (or write Q&A on the board) (attached).

Time-saving tip
Set all devices to the program link, above, and calibrate all bots to the screen before class

STUDENT GROUPING

- Groups of 2-3 students are recommended, but 1:1 student to robot ratio is fine.

MAP SETUP

- Use one printed copy of the attached map. If you can't print in color, use markers to recreate it
- Remember to calibrate your bot to the black circle on paper before starting; this makes your bot able to see the colors properly. You don't need to calibrate again unless your bot isn't seeing the colors properly.

GAME PLAY STEPS

- **OBJECTIVE**: Program Ozobot to complete the maze on its own.
- Start the program and set your bot immediately on the starting line, facing towards the branches of lines.
- Watch as your Ozobot chooses right at each intersection, then stop and dance on the red line-end.
- **For the Challenge Map**: The program for this map should have left turn decisions at each intersection so that it gets to the top branches. It should also **break** when it sees blue.

DEMONSTRATION SETUP

- Set up your projector, or a main table, to show your demo so all students can see.
- Print two copies of the maze and black out the red of one of them. You will demo the program on this uncolored map before showing how Ozobot wins on red.

TIME MANAGEMENT

- This lesson is planned for 50 minutes. See the LESSON OUTLINE headers for a breakdown.
- If your classes are short (30 mins), or very busy, split the lesson in half; do the demo and code explanation first, and the code editing and playing second (with a little refresher on the program).

ABOUT THE CODE

'LINE-END SEARCH' OZOBLOCKLY PROGRAM ozo.bot/line-end-search-game

The coding concepts explained below make it possible for Ozobot to follow lines and make decisions at intersections (Line Navigation), recognize colors at line-ends or intersections (Conditional Logic), and move continuously until a certain color is seen (Loops). Read on to learn more.

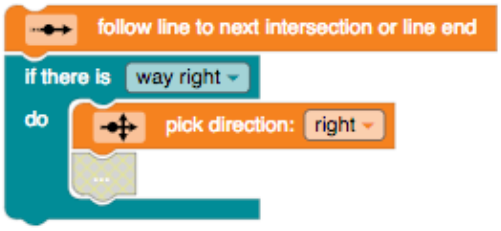
LINE NAVIGATION

These codes are unique to Ozobot. Use them to tell Ozobot to follow a line and what to do at an intersection or line-end.

There are some important things to know about Line Navigation codes:

- You can only tell Ozobot what to do at an intersection or line-end, and NOT while it's walking on a line. That means it cannot see color of a line not in an intersection or line end, or make a turn off of the line like the "Line Jump" color code.
- A right angled turn (or any turn) is not an intersection; Ozobot sees it as the same line continuing.
- An intersection can be a cross (+) or T shape.
- You can't mix Line Navigation movement (orange blocks) with "Free Movement" (yellow blocks), unless you are transitioning between the two movement types. You can't use steps on a line.
- If Ozobot doesn't see a line or that possible intersection choice when a Line Navigation code runs, you will see red-blue lights flashing and the program will end.

The game "Line-End Search" repeats the 'follow a line until the next intersection or line end' code block and an if statement for what to do at an intersection.



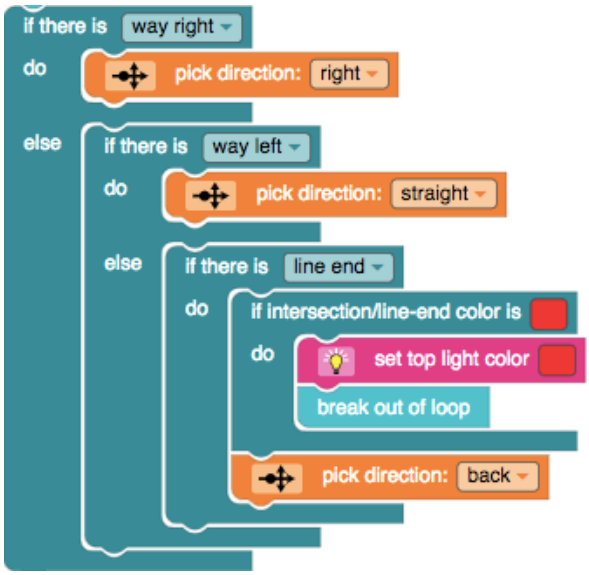
Using a 'follow line' code block with an **if** statement about what to do at an intersection within a loop creates autonomous line following behavior. See Conditional Logic below for more.

CONDITIONAL LOGIC

In programming, sometimes we want certain code to run only in a specific situation, or 'condition'. Programming languages make this easy by giving us '**if**' statements.

Just like in 'Lesson 1 - Color Search', Evo and Bit can be programmed to check condition in its sensor after performing a movement. Last time, Ozobot ran the **if statement** after a forward step. In this lesson, Ozobot runs it at a line-end or intersection. We can also check what type of intersection Ozobot has reached.

(In Mode 3 of OzoBlockly there are blocks for checking line colors and for checking surface colors - be sure to notice the difference when building or editing a program!)



PROGRAM LOGIC: This program is organized so that taking a right turn when possible is always first (and Ozobot will return to the top of the loop to follow that new line). The second priority is when Ozobot needs to walk back on a line. We want it to go straight in those conditions. Last is the end of the line reaction. We prioritize checking if it's red; if it is, end the loop, and if it's not, do nothing. Then, choose to go back.

TIP If Ozobot is given a line follow or direction choice command, but it doesn't see the line, it will 'fail' by flashing red and blue, and turn off. If it seems to be failing in places it shouldn't, recalibrate Ozobot to paper using the black circle.

LOOPS

Loops allow a section of code to be repeated either a certain number of times, or until a 'break' code is called. In this program, we use "break" to tell Ozobot when to end the loop.

Bit or Evo will run the **'break'** code when it stops at a line-end or intersection and sees that the color of the line is red. You will see the bot will not stop as soon as it walks on the red, because, as explained in Line Navigation, above, the bots only execute code once it knows it is at a line-end or intersection.

LESSON OUTLINE

DEMONSTRATE THE GAME - 10 minutes

Have one map and one programmed Ozobot ready to demonstrate to the class (see SETUP, above).

1. **Organize** students into pairs or groups.
2. **Explain** today's lesson to your students: *We will play a programmed game for Ozobot, then discover how it was coded to make Ozobot walk in a maze until it finds the winning color.*
3. **Hand out** one worksheet (attached) to each student (or do the Q&A on a board or screen.)
4. **Demonstrate** the game program on the blacked out map, and then on a red map. Explain how the game is played: *Start the program and set Ozobot on the start line. Ozobot will like follow until it lands on a red line.*
5. **Students complete** sections A and B of the worksheet. We'll compare the answers to the program next.

DISCUSS HOW THE PROGRAM WORKS - 15 minutes

1. **Open** the OzoBlockly program (ozo.bot/line-end-search-game) on your main screen.
2. **Point out,** using ABOUT THE CODE, the codes that make Ozobot follow lines: Line Navigation;
   a. Explain the 'Important Points' from ABOUT THE CODE, especially that 'follow the line' is a code that runs and can't be interrupted until Ozobot reaches an intersection or line-end.
   b. Point out that the L shaped turns on the map are not seen as intersections, but a continuing line.
3. **Point out,** using ABOUT THE CODE, the codes that make Ozobot choose actions: Conditional Logic;
   a. Explain the Program Logic for the instructions if there is a way right, left, or straight.
4. **Point out** the codes that make Ozobot walk around by itself: The Loops;
   a. Point out that the program can run forever because it will loop through a line follow code and a decision at an intersection or line-end over and over. (Do **Optional Demo** by putting your demo bot on a hand-drawn black plus sign – it will walk on the lines counter-clockwise forever.)
5. **Point out** the codes after the loop: the victory dance;
   a. A victory dance can be lights, movements, and sounds for Evo. Put lights before movement.
   b. Light animations (police, rainbow, etc.) take longer to load, so use only one in a program.
6. *Optional*: Show the alternative program that is **hard-coded**. This program is what new students might make to solve this maze from scratch. It's correct, but it's not autonomous and leaves room for error.
7. **Allow** for any questions from students.
8. **Summarize** the program to remind students that they can now code autonomous movement, surface color reactions and the game mechanic of Winning!

STUDENTS EDIT THE PROGRAM – 25 minutes

1. To get students to make meaningful changes to this program we've supplied a second maze map, called CHALLENGE. Point out that there is now a blue line end on the second branch from the top.

2. **Ask students** what changes to the original program are needed to make Ozobot go straight to that line and see blue. Either give the answers, or have students discuss and test in groups.
    a. ANSWER: Change "if there is a way right" and "pick direction right" both to left. Change line end color from red to blue. Change "if there is a way left, pick straight" to "if there is a way right, pick straight". Remember, this is a backup code whose importance will be more apparent if and when students make their own binary tree maps.
3. **Hand out** one computer/tablet, Ozobot and both maps to each group of students.
4. **Students go** to the program link (or set each device to that page prior to class). Students should load original program to test on original map before making changes to the program for the Challenge map.
5. **Students reprogram** their Ozobot to get to the blue line more directly, and end the program there.
6. Once the Challenge is complete, **students can draw** their own "binary tree" mazes with different colored line ends, and program Ozobot to reach the different colors.
    a. Remember that if students make a new program with line follow codes, Ozobot may flash red-blue (a fail indication) if the bot does not see the line or intersection it expects! The original program is least likely to fail.
    b. Students can exchange maps with classmates for others to solve!

LINE-END SEARCH

STUDENT WORKSHEET

A. What does Ozobot do?

At the first intersection, Ozobot _____

At the second intersection, Ozobot _____

On a black line-end, Ozobot _____

On a red line-end, Ozobot _____

B. Pseudocode

This 'pseudocode' is a simple English version of the program for the Line-End Search game. Read through, then fill in the blanks with the words you think belong there.

```
Set LED to yellow.
Wait 2 seconds.
Repeat forever:
     Follow line to the next intersection or line end,
     If there is a way right,
          _____
     If there is a way left,
          _____
     If there is a line end,
          If the line-end is red,
               _____
          _____
Move forward 1 step.
Set LED to pink.
Spin left.
```