ELEMENTARY COMPUTER SCIENCE WITH GAME DESIGN

DANGER ZONES

LESSON 3

Summary

Aim Ozobot toward the winning color while avoiding the losing color in a game that teaches how to program winning and losing game mechanics.

CS Topics: Loops and breaks, conditional logic, (if, + else if), + Boolean logic, sensor input/output	**Learning Outcomes:** Alternative "if" statements, called "else if", for multiple "conditions".

Grades: 2-5	**Coding Level:** Beginner	**Map Style:** Free Movement	**Game Mechanic:** winning, + losing

Introduction

Welcome to the third 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. To learn more about this series of lessons, visit the Elementary Computer Science, with Game Design README.

The game "Danger Zones" shows students how winning and losing can be programmed to make a game for Bit or Evo. In this game, the player must set their bot on the right course to avoid the losing colors, a little bit like a pinball machine. Ozobot must travel up a corridor and bounce around a map and hopefully reach the winning color. If it reaches the losing color, the game ends on a sad note.

The new concept for this game is 'losing', which adds a new term to our coding vocabulary: 'else if'. Conditional logic in programming languages give us 'if' to check for a certain condition to trigger specific code. It's often paired with 'else if', which gives coders a second, or third, or fourth (or more) condition to check for. In this program, we want both a winning and a losing reaction, as well as instructions to bounce off other colors.

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

'COLOR SEARCH' OZOBLOCKLY PROGRAM ozo.bot/danger-zones

REQUIRED MATERIALS

- **FOR DEMO:** 1 Bit or Evo (pre-programmed, link above) & 1 printed map.
- 1 Bit or Evo per group,
- 1 printed game map 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 the simple map, maintaining line thicknesses of the original.
- 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**: Aim Ozobot to bounce off colors to land on the winning color (red) and avoid the losing color (black). Winning will require some thoughtful planning and multiple tries to learn the best angles to choose.
- Begin your programmed Ozobot on the Start symbol, aiming it to bounce off of blue (right turn) and green (a left turn), and watch it move autonomously, hoping it lands on red.
- Once Ozobot lands on red, it will do a victory dance. On black, it will do a 'fail dance'.

DEMONSTRATION SETUP

- Set up your projector, or a main table, to show your Ozobot running the game program on the map so all students can see the bot's movements and reactions.

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

'DANGER ZONES' OZOBLOCKLY PROGRAM ozo.bot/danger-zones

The coding concepts explained below make it possible for Ozobot to 'bounce' off of blue and green, and either win or lose the game by landing on red or black. Read on to learn more.

LOOPS

Loops allow a section of code to be repeated a certain number of times, or forever until a 'break' code is called. In this program, we use "**break**". This code helps create autonomous movement and the ability to win.

To play these games, we need Ozobot to check the colors underneath it regularly. The best way to do that is to write a short movement code followed by code to check the color underneath (Conditional Logic). By using a loop with these two codes, we can create autonomous movement where Ozobot reacts to a map made of color often.

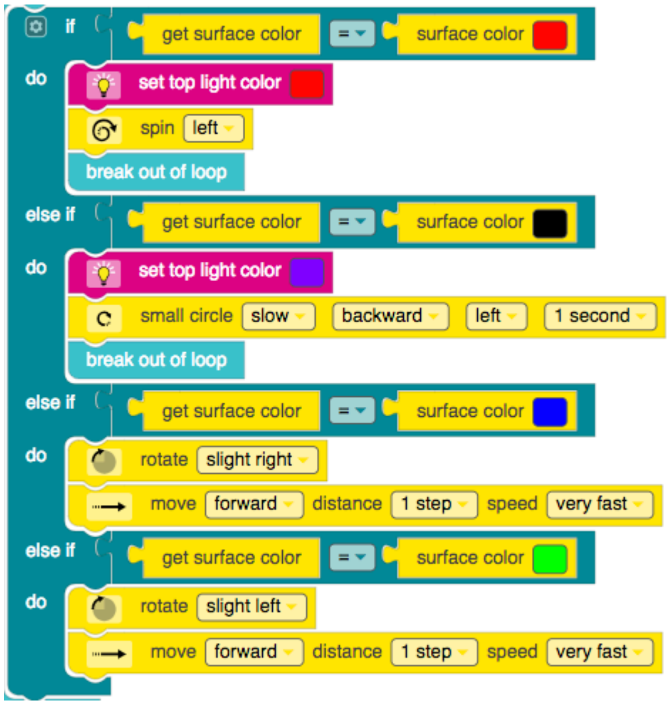
For winning and losing, we need to have the '**break'** code for both conditions. But, we don't want the same victory dance reaction, so the win/lose reactions have to be inside each condition, instead of one victory dance below the loop. There are two 'break' codes, but only one will ever be called per game. You can, in theory, use as many breaks as you need, since only one will ever run.

CONDITIONAL LOGIC

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

There are many times when there are multiple conditions to check. We could accomplish this by using 'if' multiple times, but the proper way is to use **else if** along with **if**. (Today's If/Else is a Mode 4 Logic block. Students aren't required to change this block, but you can add more 'else if' code using the gear and dragging the codes in the window left or right.)

If/else codes always start with **if**, and every extra condition is **else if**. (If we wanted a unique reaction to



```
if      get surface color  =  surface color 🟥
do      set top light color 🟥
        spin left
        break out of loop
else if get surface color  =  surface color ⬛
do      set top light color 🟪
        small circle slow backward left 1 second
        break out of loop
else if get surface color  =  surface color 🟦
do      rotate slight right
        move forward distance 1 step speed very fast
else if get surface color  =  surface color 🟩
do      rotate slight left
        move forward distance 1 step speed very fast
```

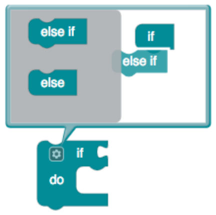
run if ALL of these conditions are false, we could end this code with **else**, which means 'every other possible condition,' but this program doesn't require it.)

The blue and green reactions have turns as well as a forward step. This is because we want to move the Ozobot off of the color. Otherwise, it might just spin there forever.

We are used to breaking out of the loop and then doing our victory dance. As explained in LOOPS, above, we need two different reactions for the two possible game-end conditions - winning and losing. That's why we put any victory dance (or fail dance) movements inside of the if statements for red and black, and before the **break** code.

HOW TO EDIT THE 'IF' BLOCK

To add, or remove, 'else if' code in the if block, click on the gear on the block. This opens a sort of mini OzoBlockly.

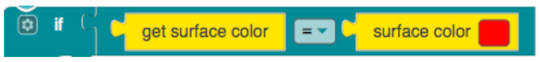


- Drag 'else if' from the left and attach it on the right to add.
- Drag 'else if' from the right and into the left to delete it.
- To close this window, click the gear again.

Students are not required to edit this block in this lesson.

BOOLEAN LOGIC (NEW)

As is visible in the screen shot above there are a more new codes. They are the 'equals' logic check, the 'get surface color' and 'surface color x'.



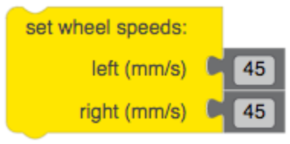
In English, this line of code means "If the surface color that the color sensor sees (get) <u>is equal to</u> a red surface color..." This line of code will return either 'true' or 'false'. If true, the code inside 'do' will run. If false, it will not run, and the next "else if" code will check the surface color (if there are more else ifs).

This program uses only the "=" code, and students don't need to edit this. The only other meaningful version of this block for this program is ≠ which means 'not equal to'. This isn't used in this program, but students may find it useful in their own creations. Other possibilities are <, ≤, ≥ and >.

MOVEMENT

The new code for movement in this program is 'set wheel speeds' from Mode 4 Movement. This code is being used instead of a step because it will see any changes in surface color very quickly.

If students change the speed of their program, both blocks for left and right motor speeds should be edited. Otherwise, you'll get some kind of turn (which could be useful in some conditions…).


```
set wheel speeds:
      left (mm/s)  45
     right (mm/s)  45
```

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 autonomously and either win on red or lose on black.*
3. **Hand out** the student worksheet to each student. (Or plan to do Q&A on the board.)
4. **Demonstrate** the game program as many times as you like, showing Ozobot reacting to reach color. Explain how the game is played: *Start the program and set Ozobot at the bottom of the corridor, angled to bounce up and into the map.*
5. **Students complete** sections A and B of the worksheet about Ozobot's movements and reactions on the map. We'll compare the answers to how the program is written, next.

DISCUSS HOW THE PROGRAM WORKS - 15 minutes

1. **Open** the OzoBlockly program (ozo.bot/danger-zones) on your main screen.
2. Use ABOUT THE CODE to help explain the different code types and what they do in the program.
3. **Point out** code for autonomous movement and winning or losing: Loops;
   a. Explain that the loop makes the code inside run over and over so Ozobot makes lots of short movements and sees colors often.
   b. Point out that there are two breaks for two possible endings – winning and losing.
4. **Point out** the codes that make Ozobot move: 'set wheel speeds';
   a. Explain that this new movement block allows Ozobot to see color more quickly than a step.
   b. Explain that to change the speed of Ozobot's movement, both number blocks need to be edited.
5. **Point out** the codes that make Ozobot react to color: Conditional Logic;
   c. Explain what conditional logic is for, and how 'else if' is for multiple situations.
   d. Point out that students will not need to edit the if block.
   e. Point out the new way to check color with the equals block, which does not need to be edited, and that the line of code means "if the surface color Ozobot sees is red, do…".
6. **Point out** the codes inside red and black: the victory dance and break;
   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.
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 mechanics of Winning and Losing!

STUDENTS EDIT THE PROGRAM – 25 minutes

1. Before having students edit the program, **seed** some programming ideas to make some meaningful edits, like make Ozobot go faster, make the turns wider, set LED colors in if statements.
2. **Hand out** one computer/tablet, Ozobot and map to each group of students.
3. **Students navigate** to the program link (or set each device to that page prior to class).
4. **Students load** the original program to their Ozobot without changing the code, and play.
5. **Students plan edits** on paper. If writing will take too long, students can plan edits verbally.
6. If there is time, **students can create** their own unique maps, which will allow them to explore how the program works a bit more.
7. **Ask** students to explain what edits they made to their program, and why.

DANGER ZONES

STUDENT WORKSHEET

A. What does Ozobot do?

On blue, Ozobot ____

On green, Ozobot ____

On black, Ozobot ____

On red, Ozobot ____

B. Pseudocode

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

```
Set LED to yellow.
Wait 2 second.
Repeat forever:

    Set both wheel speeds to 45 mm/s.
    If Ozobot is on red,

        ______________________,____________________,______________________

    If Ozobot is on black,

        ______________________,____________________,______________________

    If Ozobot is on blue,

        ________________________________,________________________________

    If Ozobot is on green,

        ________________________________,________________________________
```