Recognizing Damaged Items with an LLM Bot and Camera

Ozobot Editor Challenge—Teacher's Guide

by Richard Born, Associate Professor Emeritus, Northern Illinois University

Introduction

If you haven't already done so, you should read the two-page student guide that accompanies this lesson. It provides all of the information needed by you and the student to thoroughly understand the challenge.

Challenge Summary

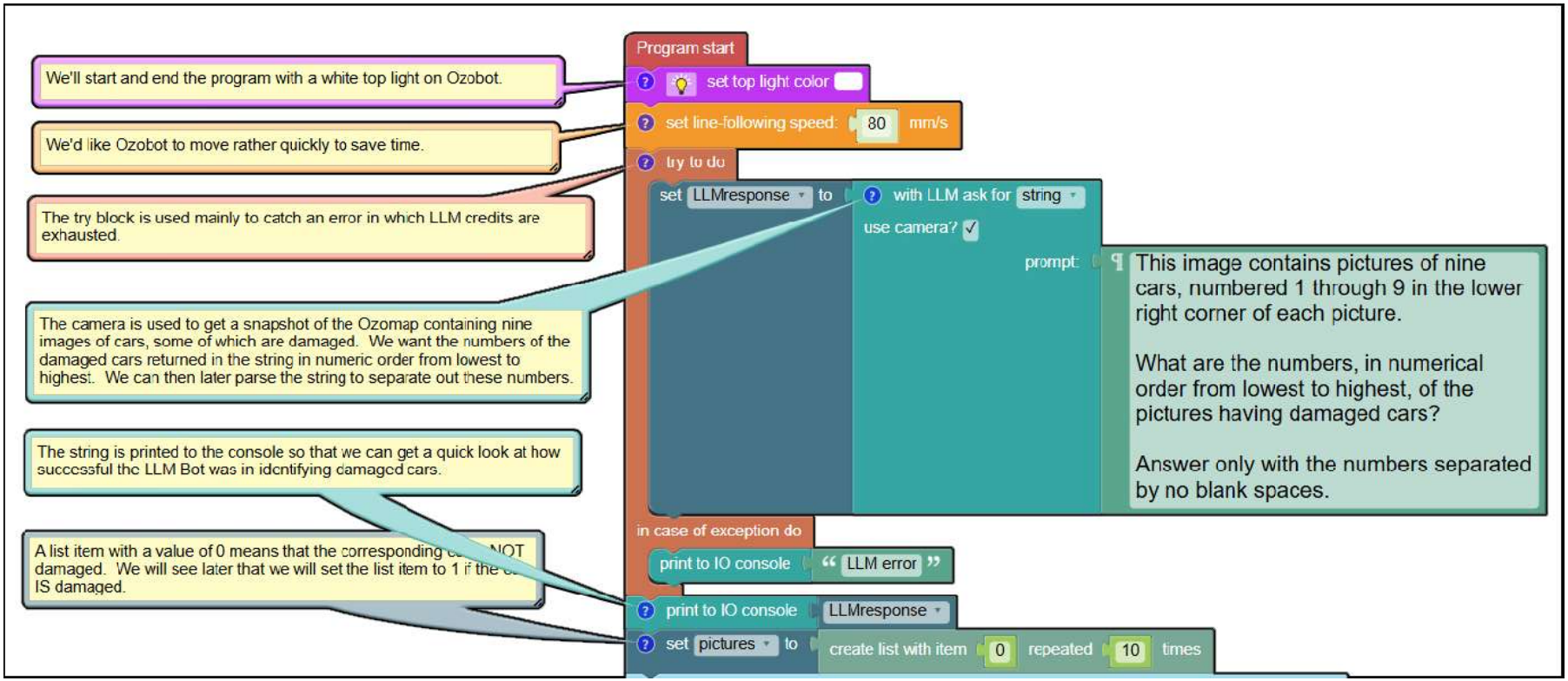
This challenge involves using a large language model (LLM) with a camera to identify damaged cars on an Ozomap containing nine vehicles, some collision-damaged and others not. Since cars are larger and feature clearer damage signs, they are easier to recognize than small warehouse items with subtle damage. Participants will capture images of the Ozomap, then design a block-based program for an Ozobot to follow the path through all nine cars, displaying a red light at intersections with damaged vehicles and a green light otherwise, based on information from an LLM prompt. The process requires creating a universal prompt that works for different Ozomaps (A and B) without referencing their labels, leveraging the **Google (mini) LLM for optimal accuracy**, and ensuring the program operates correctly for both maps.

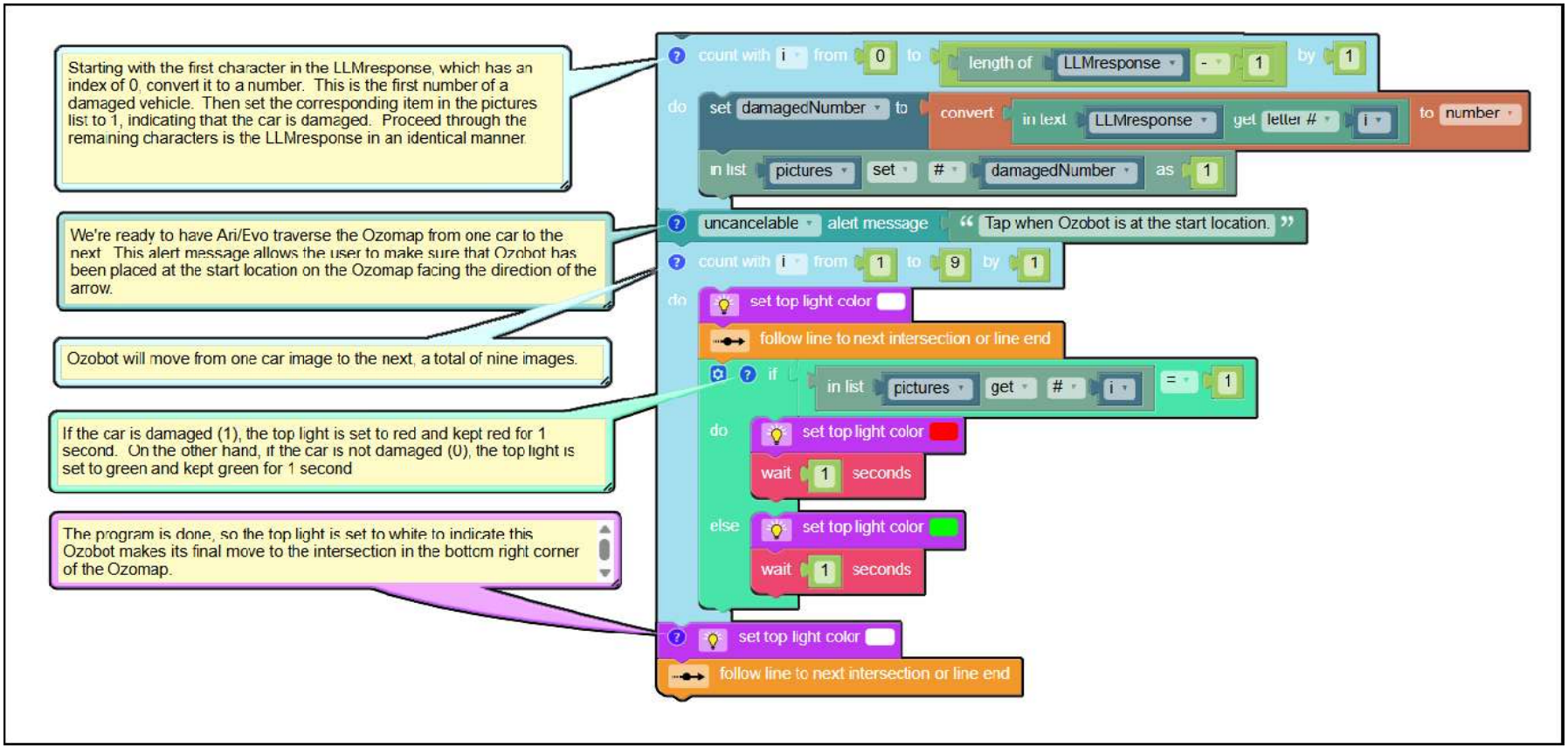
Computer Science Programming Concepts in This Lesson

- The use of variables to hold the values of program constants
- The use of loops to handle tasks that must be repeated several times
- The use of conditionals, as needed, to control the flow of logic
- Variable names that support self-documentation
- Extracting information from a string by parsing
- The use of LLM (Large Language Models) with a camera

Complete Program with Comments

On the next two pages, you will find the complete Ozobot Editor solution to the challenge as prepared by the author. The wording of the LLM prompt is a key factor in achieving success in the challenge. The prompt may require several iterations before the students get the desired response from the LLM. The solution also involves parsing the response to obtain the identification number of any cars that are damaged. The desired *LLMresponse* variable for Ozomap A is the string 13569, indicating that cars 1, 3, 5, 6, and 9 are damaged. The desired *LLMresponse* variable for Ozomap B is the string 23568, indicating that cars 2, 3, 5, 6, and 8 are damaged.


We'll start and end the program with a white top light on Ozobot.
We'd like Ozobot to move rather quickly to save time.
The try block is used mainly to catch an error in which LLM credits are exhausted.
The camera is used to get a snapshot of the Ozomap containing nine images of cars, some of which are damaged. We want the numbers of the damaged cars returned in the string in numeric order from lowest to highest. We can then later parse the string to separate out these numbers.
The string is printed to the console so that we can get a quick look at how successful the LLM Bot was in identifying damaged cars.
A list item with a value of 0 means that the corresponding
NOT
damaged. We will see later that we will set the list item to 1 if the
IS damaged.
Program start
set top light color
set line-following speed: 80 mm/s
try to do
set LLMresponse to
with LLM ask for string
use camera? ✓
prompt:
This image contains pictures of nine cars, numbered 1 through 9 in the lower right corner of each picture.
What are the numbers, in numerical order from lowest to highest, of the pictures having damaged cars?
Answer only with the numbers separated by no blank spaces.
in case of exception do
print to IO console " LLM error "
print to IO console LLMresponse
set pictures to
create list with item 0 repeated 10 times


Starting with the first character in the LLMresponse, which has an index of 0, convert it to a number. This is the first number of a damaged vehicle. Then set the corresponding item in the pictures list to 1, indicating that the car is damaged. Proceed through the remaining characters is the LLMresponse in an identical manner.
We're ready to have Ari/Evo traverse the Ozomap from one car to the next. This alert message allows the user to make sure that Ozobot has been placed at the start location on the Ozomap facing the direction of the arrow.
Ozobot will move from one car image to the next, a total of nine images.
If the car is damaged (1), the top light is set to red and kept red for 1 second. On the other hand, if the car is not damaged (0), the top light is set to green and kept green for 1 second
The program is done, so the top light is set to white to indicate this Ozobot makes its final move to the intersection in the bottom right corner of the Ozomap.
count with i from 0 to length of LLMresponse - 1 by 1
do
set damagedNumber to convert in text LLMresponse get letter # i to number
in list pictures set # damagedNumber as 1
uncancelable alert message " Tap when Ozobot is at the start location. "
count with i from 1 to 9 by 1
do
set top light color
follow line to next intersection or line end
if in list pictures get # i = 1
do
set top light color
wait 1 seconds
else
set top light color
wait 1 seconds
set top light color
follow line to next intersection or line end

***Share Code* – nh29h3p**

Alternative Ways to Use this Challenge

Instead of using this lesson as a challenge, the teacher may elect to present the solution to students after they have read the Student Guide. A teacher walkthrough of the program with comments, allowing student interaction and questions from students, may be more appropriate in some situations. Following the walkthrough, students could then work with the program using the accompanying Ozomaps A and B, containing damaged and undamaged cars.

Example Video

Accompanying this lesson is a video entitled *Ari Identifies Damaged Vehicles.m4v*. You may find it useful to show this short video to your students before they begin the challenge. It shows Ari moving on Ozomap A, displaying a red top light for damaged cars and a green top light for undamaged cars.

Applicable CSTA (Computer Science Teachers Association) Standards

2-CS-02	Design projects that combine hardware and software components to collect and exchange data.
2-AP-11	Create clearly named variables that represent different data types and perform operations on their values.
2-AP-19	Document programs in order to make them easier to follow, test, and debug.
3A-DA-11	Create interactive data visualizations using software tools to help others better understand real-world phenomena.
3B-CS-02	Illustrate ways computing systems implement logic, input, and output through hardware components.
3B-AP-08	Describe how artificial intelligence drives many software and physical systems