ORA and Ari Collaborate as a Candy Bar Confectioner

Grades 9-12

Teacher Guide

by Richard Born, Associate Professor Emeritus, Northern Illinois University

Introduction

In the dynamic field of robotics, hands-on learning is essential for nurturing the next generation of innovators. This OzoBlockly Editor Challenge, featuring Ari and ORA, presents an exciting opportunity for students to delve into the practical applications of programming and robotics within a confectionery context. In this lesson, students will take on the role of confectioners, using interactive technology to bring their creativity to life by programming Ari to allow customers to select their favorite Hershey mini candy bars, including Milk Chocolate, Dark Chocolate, Krackel, and Mr. Goodbar. This engaging challenge not only captures students' imaginations but also embeds fundamental programming concepts, such as user input, decision-making, and automated movement.

As students navigate this challenge, they will learn how robots like ORA assist in real-world processes, demonstrating the transformative power of automation in various industries, from manufacturing to logistics. By simulating the process of candy selection and packaging, this lesson mirrors everyday tasks that robots are now capable of performing, highlighting their role in enhancing productivity and safety in modern workplaces. Additionally, students will collaborate and innovate, mirroring the work of engineers and technologists as they solve problems and create functional prototypes.

This guide aims to equip educators like you with the necessary insights to facilitate this challenge effectively. By fostering an environment that emphasizes exploration and creativity, your students will be inspired to embrace the future of technology and engineering, preparing them for potential careers in a rapidly evolving landscape. Help your students embark on this journey of discovery, innovation, and fun!

If you haven't already done so, you are encouraged to view the accompanying 73-second video to get a good feeling for what is expected of your students from this challenge.

Example Solution to this Challenge

The solution to this challenge provides the opportunity for your students to develop an OzoBlockly Editor program that is modular, dividing the program logic into functions each having a specific purpose. In the solution presented here, variable and function names have been created in a way that promotes self-documentation.

Figure 1 shows a diagram of the candy grid for this challenge. It is placed on the base plane (z=0) of ORA Quadrant I with the center of the top upper left cell located at ORA (x, y) = (*upperleftORAx, upperleftORAy*). The values of these coordinates as well as the width of each cell *yORAincrement* and height *xORAincrement* are specified in the "*Initialization*" function to be discussed next.

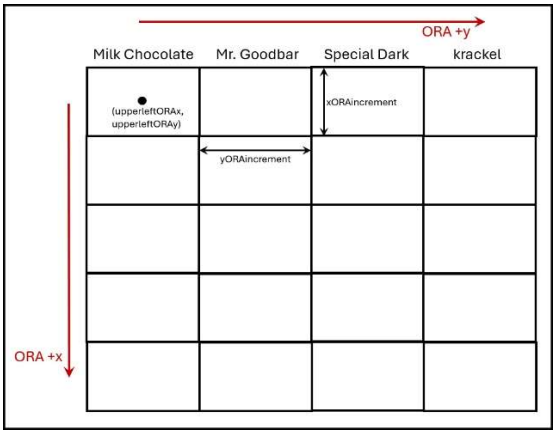

Figure 1

The "*Initialization*" function is shown in Figure 2. Ari's top color is set to white, eliminating the system information that appears automatically. ORA begins in the ZERO position. ORA's speed is set to 85% but should be kept at a much lower speed, say about 20%, during program development for safety. The vacuum gripper is used by ORA to move the candy. The nonporous foil wrappers of the Hershey's miniatures provide for a perfect vacuum fit. The values of the four variables discussed in Figure 1 are then set. These may need adjusting depending upon where the grid is placed on ORA's base plane and the exact size of the grid after it has been printed. Arrays used in the program are then initialized. The *candyName* list (length 4) contains the names of the miniatures in indices 0 through 3. The list *orderQuantity* (also length 4) will be filled with the number of each kind of miniature candy ordered by the customer. Indices for this list also are 0 through 3.

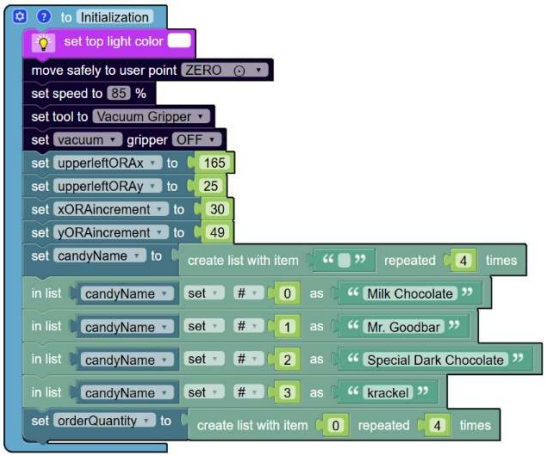

Figure 2

Figure 3 shows the "*Customer Selects Desired Candy*" function. This function is where the order quantity for each of the four kinds of Hershey's miniatures is captured from the user. This is accomplished via a *count with* loop in which the loop variable *candyNumberID* varies from 0 to 3. An uncancelable prompt is used to provide the user with the available options. They range from 0 (no candy of that type) to a maximum of 5, set by the size of the grid on which the candies are placed. The *print to IO console* block is useful during program development, but can be removed in the final program if desired.

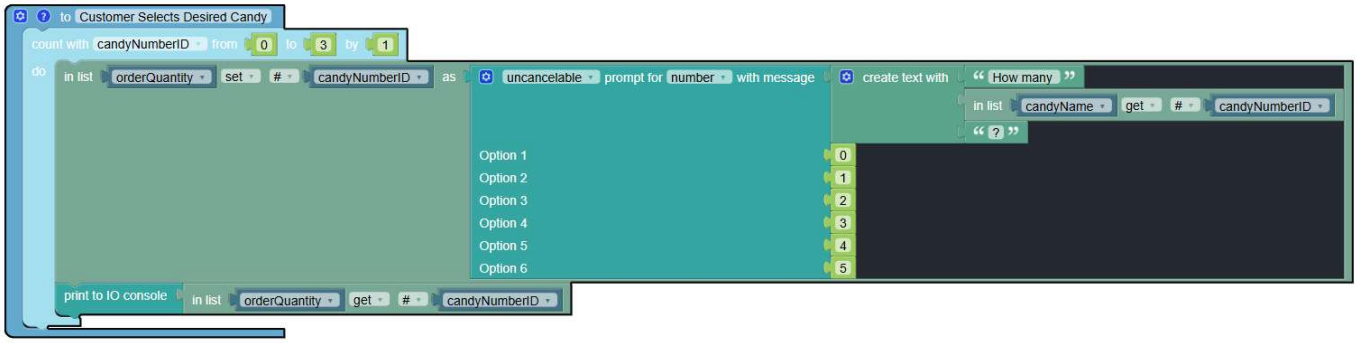

Figure 3

The main program is shown in Figure 4.

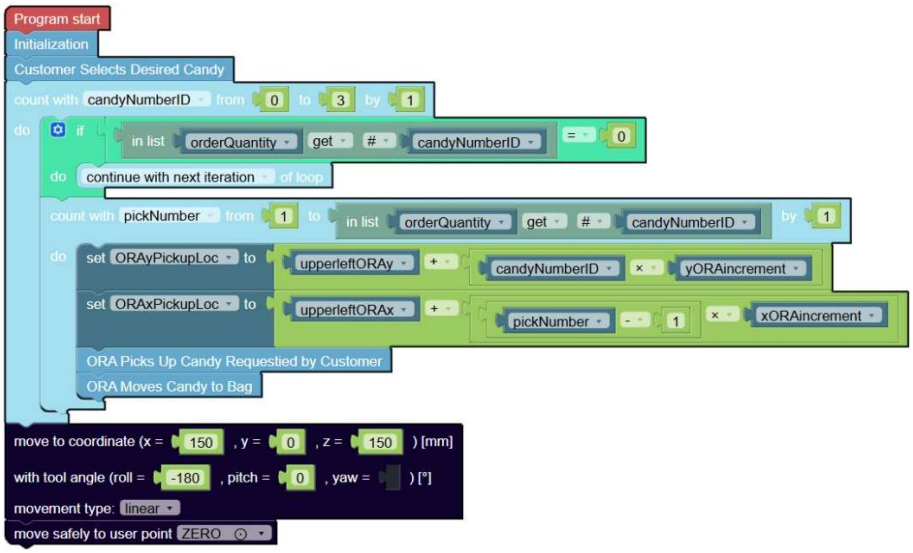

Figure 4

As shown in Figure 4, the program begins with calls to the "*Initialization"* and "*Customer Selects Desired Candy*" functions. The bulk of the program is found in the *count with* loop, in which the loop variable *candyNumberID* varies from 0 to 3. The first item in the loop is an *if-do* block that continues with the next iteration of the loop if no candy of that kind has been ordered by the customer. Then an inner loop varies from 1 to the number of items ordered for that candy. Within that loop, the ORA x and y pick up locations are computed based on the variables *upperleftORAy*,

upperleftORAx, yORAincrement, and xORAincrement variables discussed in Figures 1 and 2. The functions "*ORA Picks Up Candy Requested by Customer*" and "*ORA Moves Candy to Bag*" are then called, ending the inner loop's current iteration. After all desired candy have been placed into the bag, ORA's arm first moves to (x, y) = (150, 0) and then to the ZERO location before the program ends.

Figure 5 shows the "*ORA Picks Up Candy Requested by Customer*" function. The pick up location (x, y) = (*ORAxPickupLoc, ORAyPickupLoc*) were computed in the main program. ORA's arm first moves to that location at a height of 150 mm. It is then lowered to 4 mm where the vacuum gripper makes contact with the candy wrapper. The vacuum is turned ON, and the gripper is raised back to a height of 150. This height may need to be adjusted to a little higher that the height of the bag.

Figure 5

Figure 6 shows the *"ORA Moves Candy to Bag"* function. The center of the bag is resting at ORA (x, y) = (250, -150). The vacuum is then turned OFF releasing the candy into the bag.

Figure 6

Share Code for this Lesson – gka9gy3

ISTE Standards (International Society for Technology and Education)

Empowered Learner 1.1.d: Students understand the fundamental concepts of technology operations, demonstrate the ability to choose, use and troubleshoot current technologies and are able to transfer their knowledge to explore emerging technologies.

Knowledge Constructor 1.3.d: Students build knowledge by actively exploring real-world issues and problems, developing ideas and theories and pursuing answers and solutions.

Computational Thinker 1.5.c: Students break problems into component parts, extract key information, and develop descriptive models to understand complex systems or facilitate problem-solving.

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-12	Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
2-AP-13	Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
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.
3A-AP-17	Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
3B-CS-02	Illustrate ways computing systems implement logic, input, and output through hardware components.
3B-AP-14	Construct solutions to problems using student-created components, such as procedures, modules and/or objects.