OzoBlockly Editor Challenge:

Precision Placement – ORA Places Marbles on a Special Pallet

Grades 9-12

Teacher Guide

by Richard Born, Associate Professor Emeritus, Northern Illinois University

Introduction

Welcome to this OzoBlockly Editor challenge, an innovative project where students will dive into robotics by programming Ozobot ORA. This challenge focuses on developing precise control as students learn to use ORA's vacuum gripper to pick up marbles and accurately place them onto a specially designed 3D-printed pallet with two rows and four columns.

Through this engaging activity, students will cultivate essential skills in block-based programming, logical thinking, and fine-tuning robotic movements. The guide emphasizes efficient coding and meticulous execution, preparing them for real-world robotics applications. Crucially, the challenge also integrates vital safety lessons, highlighting the importance of operating at reduced speeds and having immediate access to the emergency stop button. Get ready to guide your students through a rewarding experience that blends technology, problem-solving, and practical skill development!

If you have not already done so, you should now read through the *Student Guide* for this challenge for details on what is expected from the students and view the 8-second accompanying time-lapse video.

Share Code for the Solution – zeevonw

Link to 3D STL File – Marble Ramp
Link to 3D STL File – Marble Pallet

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. Images of the modules follow with a discussion explaining the purpose of the Ozobot Editor blocks that make up each module. A separate included PDF file also contains images of these modules, in the event that you want to share any of them with your students.

Figure 1 shows the main program and the "*Initialization*" function. All blocks, with the exception of the last block in the initialization function, are self-explanatory. The centers of the marble locations

in the marble pallet are all separated by 32 mm, both in the *x* and *y* directions. The *placeIncrement* variable will be used to determine the exact location for placing each marble.

The main program begins by calling the "*Initialization*" function. Then a double-nested "count with" loop varies the row from 0 to 1 with the loop variable x. Meanwhile, the inner loop varies the column from 0 to 3 with the loop variable y. For each combination, the "*Pick Marble*" function first picks the marble from the end of the marble ramp. The "*Place Marble*" function is then called, with inputs *x* and *y*. After the pallet has been filled, ORA's arm moves to the *Home* location and the program terminates.

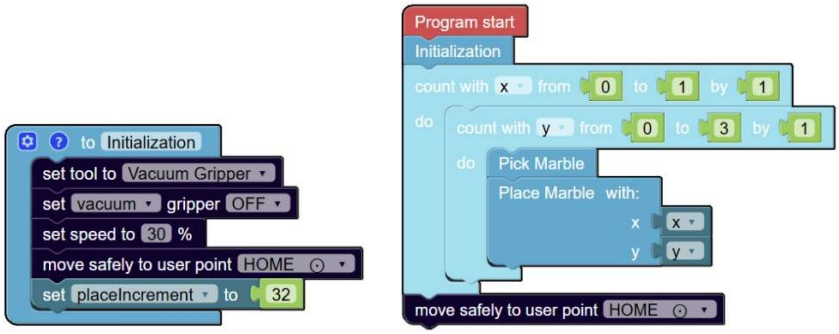

Figure 1

Figure 2 shows the "*Pick Marble*" function. This function directs ORA to pick up the marble that is currently at the end of the downward sloping marble ramp at (*x, y*) = (195, -78). The first *move to coordinate* block moves ORA's vacuum gripper to a height of 60 mm above the pick-up location. The second block lowers the gripper so that it contacts the marble at a height of 20 mm. The gripper is then turned on, and the marble is lifted back to a height of 60 mm above the pick-up location.

Figure 2

Figure 3 shows the "*Place Marble*" function. The *xPlaceLocation* and *yPlaceLocation* variables are determined by the values of the input variables *x* and *y* and the *placeIncrement* variable. The first marble is placed at location (175, 95) since the x and y inputs are both 0. The first *move to coordinate* block moves the marble at a height of 60 mm to the point (*xPlaceLocation, yPlaceLocation*) above the marble pallet. The second *move to coordinate* block lowers the marble to the drop height, and then the vacuum is turned off to release the marble. The third *move to coordinate* block raises the gripper back to a height of 60 mm above the tabletop.

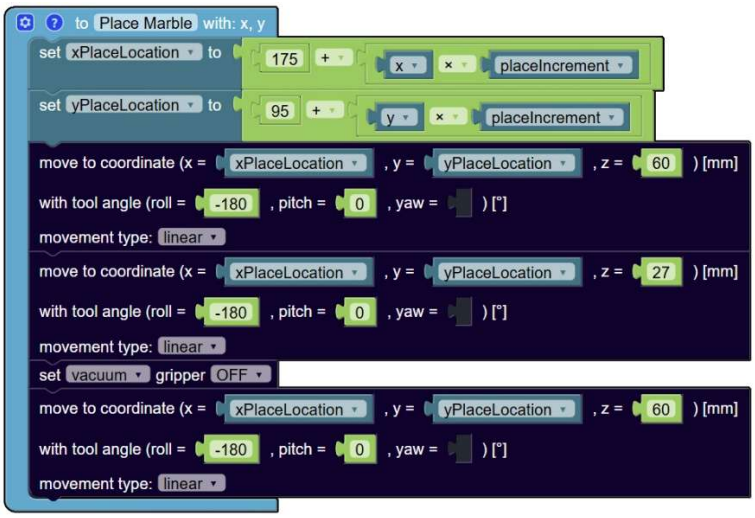

Figure 3

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-08	Describe how artificial intelligence drives many software and physical systems.
3B-AP-14	Construct solutions to problems using student-created components, such as procedures, modules and/or objects.