OzoBlockly Editor Challenge:

ORA Unstacks Disks and Places Them On a Vertical Pegboard

Grades 9-12

Teacher Guide

by Richard Born, Associate Professor Emeritus, Northern Illinois University

Introduction

The teacher guide is designed to support this ORA challenge, a project that bridges the gap between classroom robotics and real-world industrial automation. By guiding students through the process of unstacking disks and placing them onto a vertical pegboard, you are helping them simulate tasks common in automotive and gear-manufacturing facilities. This challenge highlights the precision and flexibility of the ORA cobot, specifically focusing on the complex "peg-in-hole" insertion technique.

Your role is to facilitate an environment where students can explore spatial reasoning and coordinate geometry while prioritizing safety. The guide emphasizes the use of OzoBlockly to manage movement speeds and the importance of the Emergency Stop Button during the development phase. By following the structured setup, including a time-lapse video for visualization, students will gain hands-on experience in modern automation. Use this guide to help students troubleshoot their coordinate values and understand the significance of flexible motion in global manufacturing.

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 11-second accompanying time-lapse video.

Share Code for the Solution – 94mz8gz

Link to 3D STL File – Peg Wall
Link to 3D STL File – Disk with Hole x6

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. The first three blocks of the initialization module are self-explanatory. The *placeIncrement* variable specifies the distance in mm between each peg of the pegboard, both horizontally and vertically. The *pickUpZValue* variable is the height in mm where the top disk in the vertical stack is to be extracted. The *diskThickness* variable is the thickness of each disk.

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 z. The bottom row is row 0, and the top row is row 1. Meanwhile, the inner loop varies the column from 0 to 2 with the loop variable x. For each combination, the "*Pick Up Disk*" function first picks the top disk from the stack of disks. The "*Place on Pegboard*" function is then called, with inputs *x* and *z*. After the pallet has been filled, ORA's arm moves to the *Home* location and the program terminates.

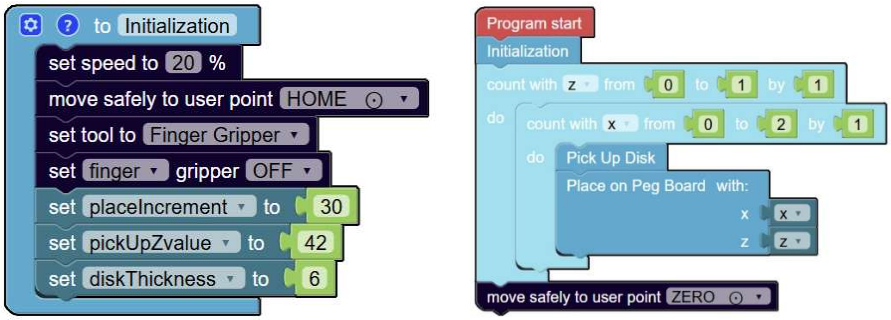

Figure 1

Figure 2 shows the "*Pick Up Disk*" function. This function directs ORA to pick up the disk that is currently at the top of the stack of disks at (*x, y*) = (241, -243). The first *move to coordinate* block moves ORA's vacuum gripper to a height of 60 mm above the pick-up location. The finger gripper is then opened. The second move block lowers the gripper so that it can grip the disk at a height of *pickUpZvalue* mm. The gripper is then closed, and the disk is lifted back to a height of 60 mm above the pick-up location. The *pickUpZvalue* variable is then decreased by the thickness of a disk so that it is ready for the *next* disk to be picked up. Finally, ORA's arm is moved to the Home location.

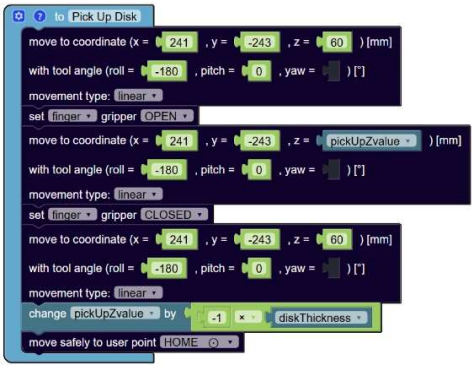

Figure 2

Figure 3 shows the "*Place on Pegboard*" function. The *xPlaceLocation* and *zPlaceLocation* variables are determined by the values of the input variables *x* and *z* and the *placeIncrement* variable. The first disk is placed on the peg at location (x, z) = (200, 50) since the x and z inputs are both 0. The first *move to coordinate* block moves the disk to the point (*xPlaceLocation, 0, yPlaceLocation*) adjusting the pitch and yaw by 90°. The second *move to coordinate* block moves the disk so that its hole is centered over the peg. Then the finger gripper is OPEN to release the disk onto the peg, and the gripper is turned off. The third *move to coordinate* block backs the gripper away from the pegboard to y = 140 mm. Finally, ORA's arm is moved to the Home location.

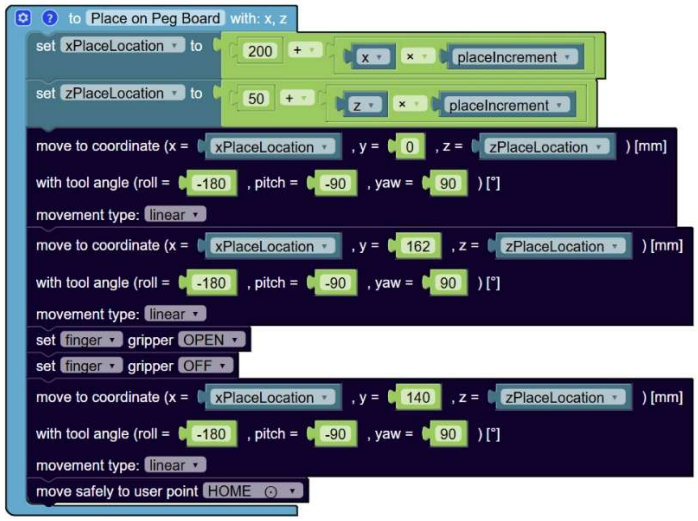

Figure 3

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-14	Construct solutions to problems using student-created components, such as procedures, modules and/or objects.