ORA: The Tetris Kitting Challenge

Grades 9-12

Teacher Guide

by Richard Born, Associate Professor Emeritus, Northern Illinois University

Introduction

The Tetris Kitting Challenge, an engaging, hands-on lesson, is designed to bridge the gap between abstract geometric logic and real-world industrial robotics. In this challenge, students are tasked with programming ORA using the OzoBlockly Editor to identify, pick up, and precisely place four distinct Tetris blocks—the L, J, I, and S tetrominoes—into a perfect 4x4 square grid. While the final destination of each piece is fixed, the starting orientations are intentionally mismatched. This requires students to think like both programmers and geometricians as they calculate the precise change in *yaw*, *i.e.,* rotation around the vertical axis, needed to align each piece correctly.

This exercise serves as a powerful simulation of modern *Bin Picking* and *Kitting* operations found in smart factories today. By navigating ORA's 6-axis flexibility to adjust skewed parts in mid-air, students practice the same high-level spatial reasoning and robotic path planning used by engineers in automotive and electronics manufacturing. As an instructor, you will guide students through the complexities of coordinate-based movement and rotational physics, helping them see the direct link between a global gaming phenomenon and the automation of assembly lines.

Safety and precision are central themes of this teacher guide. You will help students implement critical programming safeguards, such as initializing ORA at slower speeds to prevent collisions and ensuring the proper use of emergency stop procedures. By mastering these robotic movements to solve the Tetris puzzle, your students are not just playing a game. They are developing the technical literacy and safety-conscious mindset required for the next generation of smart manufacturing.

If you haven't already done so, you are encouraged to view the accompanying 79-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 1provides a reference copy of the black-and-white ORA workspace sheet. The manually placed locations of the four tetrominoes, I, L, S, and J, from top to bottom, are shown on the left. The final ORA-placed locations on the 4x4 square are shown on the right.

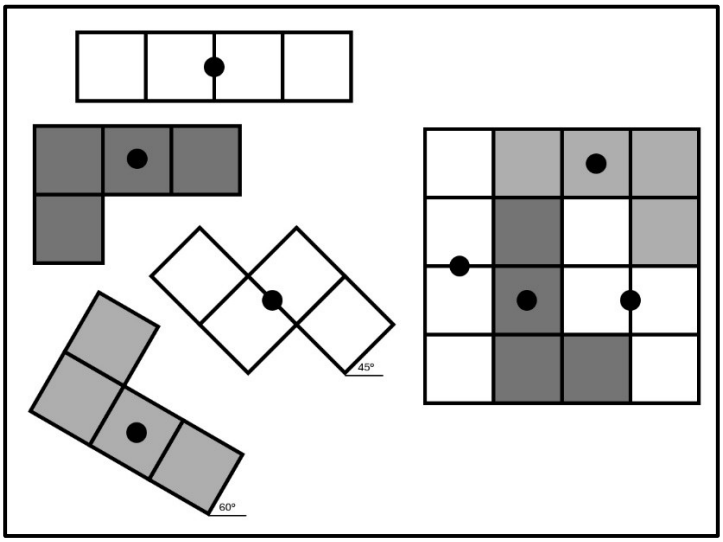

Figure 1

Figure 2 shows the main program, with function names that are self-documenting. The *count with* loop varies the loop variable *i* from 0 to 3 by steps of 1. The values 0, 1, 2, and 3 refer to tetrominoes I, L, S, and J, respectively. Each time through the loop, a tetromino is picked up by ORA and then placed on the 4x4 square after adjusting the yaw. When the task is compete, ORA moves to the Zero position and the program ends.

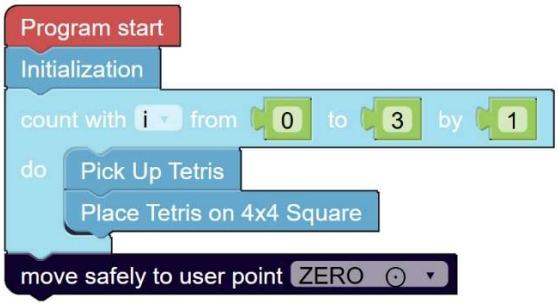

Figure 2

Figure 3 shows the "*Initialization*" function. ORA's tool is set to the vacuum gripper and is OFF. ORA's speed is set to 50%, though it is a good idea to set this much lower, say 20% during program development, for safety. Five lists are then declared, each with indices ranging from 0 through 3, corresponding to the four tetrominoes I, L, S, and J. The first two lists, *xPickLocations* and *yPickLocations*, are initialized with the ORA (x, y) locations of the black dots where the tetrominoes were manually placed. The next two lists, *xPlaceLocations* and *yPlaceLocations*, are initialized with the ORA (x, y) locations of the black dots where ORA places the tetrominoes into the 4x4

square. The final list contains yaw rotations in degrees that orient each tetrominoe for proper fit into the square. Positive is counter-clockwise and negative is clockwise, as seen looking down on ORA's base plane from above.

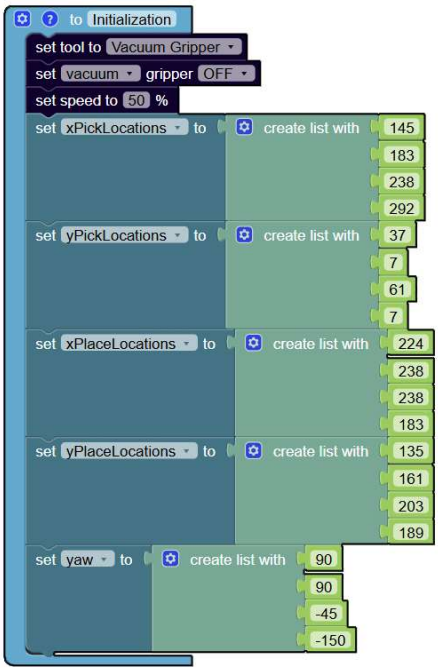

Figure 3

Figure 4 shows the "*Pick Up Tetris*" function. The first *move* block moves the vacuum gripper to a height of 20 mm above the pickup location. The second *move* block lowers it to -4 mm so that it contacts the tetrominoe. The vacuum gripper is then turned on to grab the Tetris. The third *move* block raises the gripper and the tetrominoe back to a height of 20 mm. The final *move* block adjusts the yaw so that the tetrominoe is in the correct orientation for the 4x4 square.

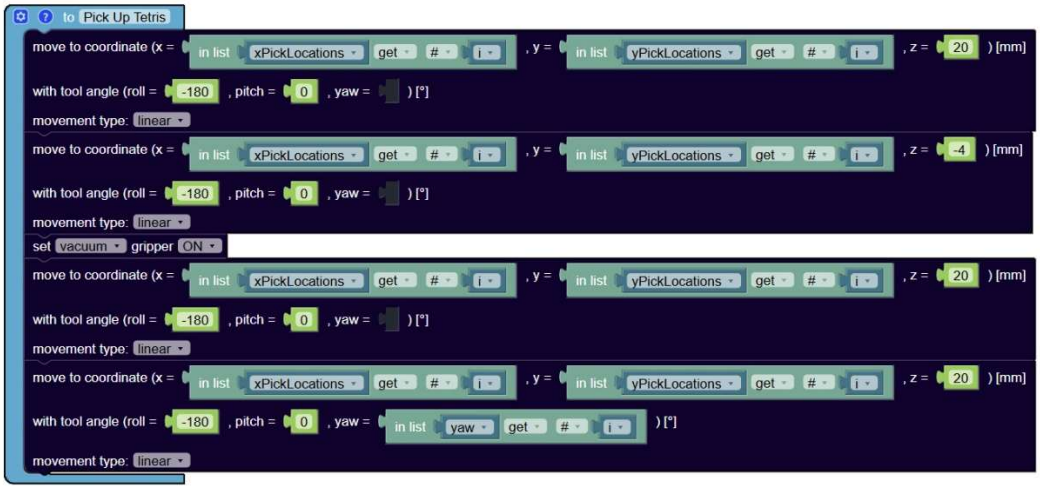

Figure 4

Figure 5 shows the "*Place Tetris on 4x4 Square*" function. The first *move* block moves the vacuum gripper to a height of 20 mm above the place location on the square. The second *move* block lowers the gripper so that it is just in contact with the square, where the gripper is then turned OFF and the tetrominoe is released. The final move block raises the gripper back to a height of 20 mm above the place location. Note that the yaw value is kept throughout the three moves.

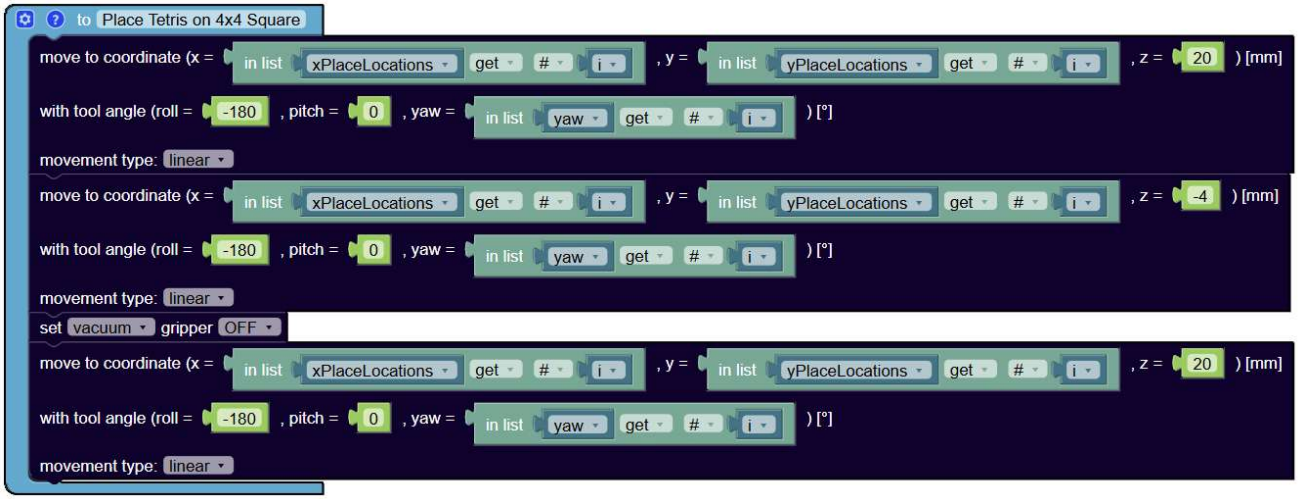

Figure 5

Share Code for this Lesson – 43crf54

Additional Tip

1. It is strongly recommended that the "printable" documents (grids, Tetrominoes) included with this lesson be printed on card stock paper.
2. The exact values for the pick and place location lists may need to be adjusted depending on the size of the printables when printed on your printer.

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-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.