Lesson 10: Efficiency and Types of Movement

Vocabulary

Linear - a type of movement where the end tool moves to a point in a straight line.
Joint - a type of movement where the end tool moves to a point through the fewest movements.
Point - a user-defined and recallable position for the ORA.
Efficiency - measured in cycle time, the time taken to complete a product from start to finish.

Direct Instruction

Engage with ORA Motion Types

Open the **EngageL10Codes** and have them ready to run on connected ORA. Run **EnageL10Code_Joint** (Share Code: y7cgto6) and allow students to observe the movements. Then, run **EnageL10Code_Linear** (Share Code: v9hhhzd) for students to observe. Allow students to view the code and see that the code is very similar except for the movement type. Direct students to the Engage section of the Student Activity Sheet and have students compare and contrast the two different movement types.

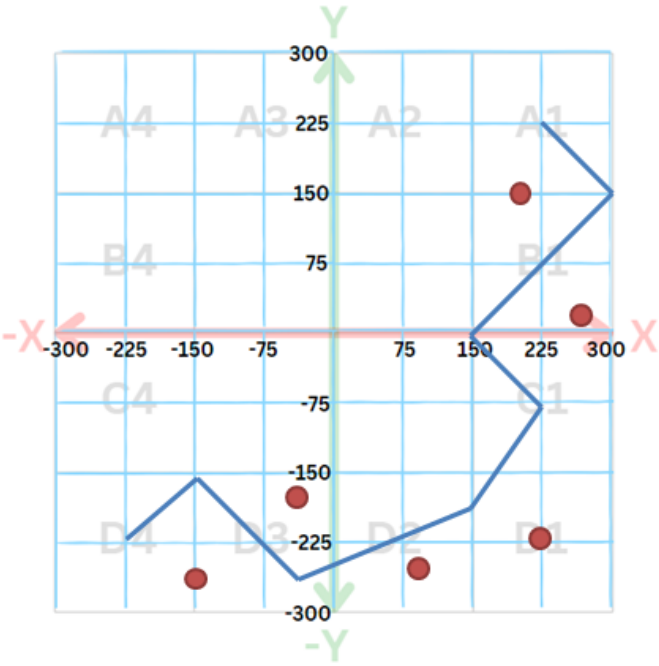
Next, show the Lesson 10 video.

Explore the Different ORA Motion Types

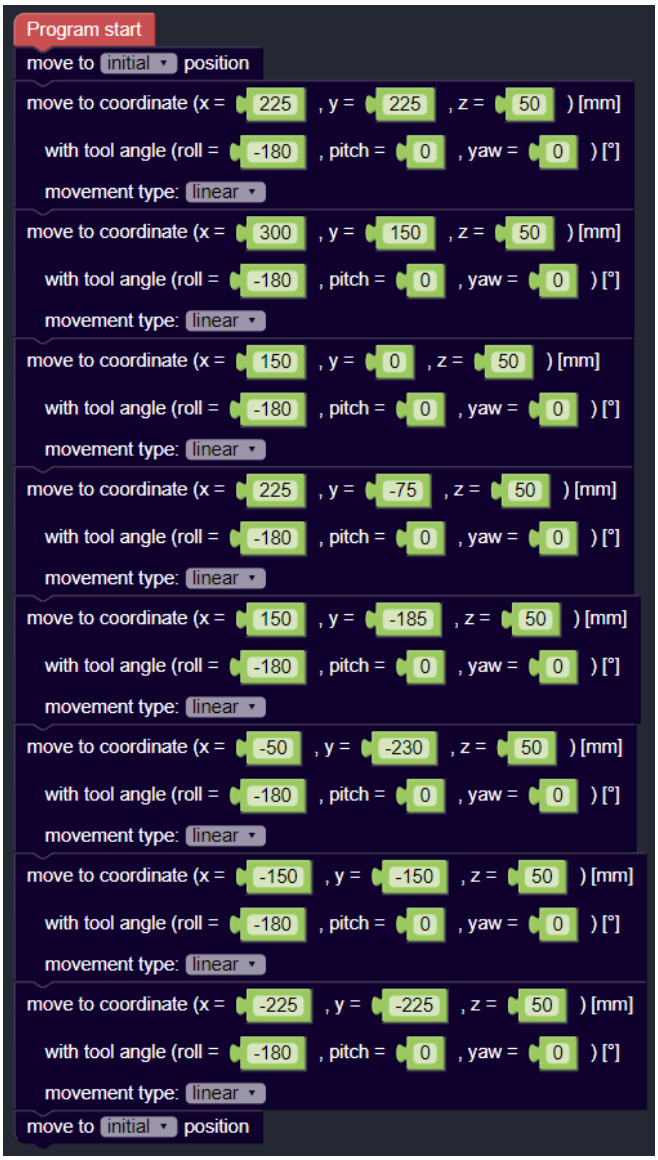
Place the six red cylinders randomly in quadrants A1, B1, C1, D1, D2, D3, and D4. Working in Level 5 of the Ozobot Editor, tell the students that, after moving to the initial position, the ORA must start in A1 and move through B1, C1, etc, to D4 at a fixed Z height of 50mm without knocking over any of the red cylinders. They will try to accomplish the task using linear motion and then joint motion (note: if time is limited, assign half the groups to joint and the other half to linear, then have the groups collaborate). After they test their code, direct students to the Explore section of the Student Activity Sheet to answer questions.

Example task and code

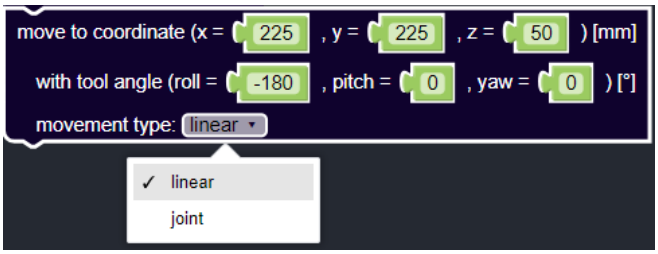
In the example quadrant map on the right, six red cylinders were randomly placed on the quadrant map. The blue line represents the end tool path in linear mode



Below is an example code for the linear movement type.



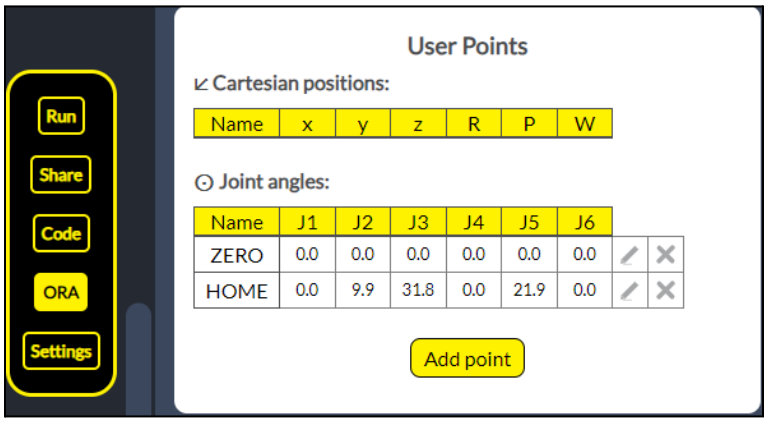
Changing from linear to joint is accomplished by selecting the drop-down next to movement type in the move to coordinate block.



Explain Custom Points with the ORA Editor.

Ask students, "How have you been determining what X, Y, and Z coordinates you need to put in the Ozobot Editor?" Allow a few moments to get responses. Ask the students, "Has this been easy or difficult?" After a few more responses, ask the students, "Don't you wish it was as simple as putting the ORA where you want it and running the program?"

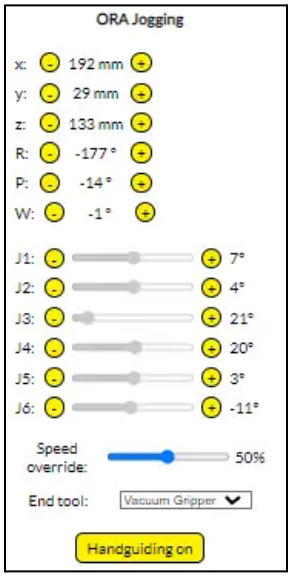
Open up the Ozobot Editor. On the right-hand side, select ORA. Here, you can see User Points. Under Cartesian Positions, points are named and have X, Y, Z, Roll, Pitch, and Yaw attributes. Note that the Yaw is a W so as not to confuse the Y of the XYZ coordinates and the Yaw.



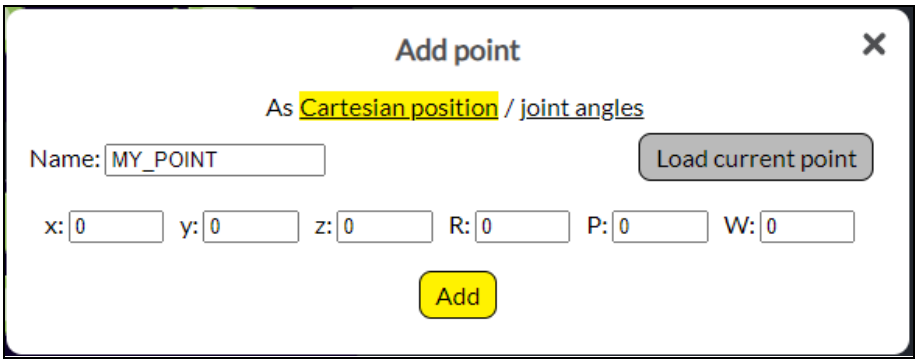
Under the Cartesian Positions are Joint Angle-defined positions. Let the students know that they will not be using joint angles, but there are two pre-defined points for Zero and Home.

Remind students about manually moving the ORA around. On the right-hand side, select ORA. Then, select "Handguding On". When the 'handguide' button on the ORA is pressed and held (the same button used during calibration), a person may physically move the ORA around. The Ozobot Editor updates the X, Y, and Z coordinates in real time.

When the desired position is reached, select the Add Point Button under "User Points".

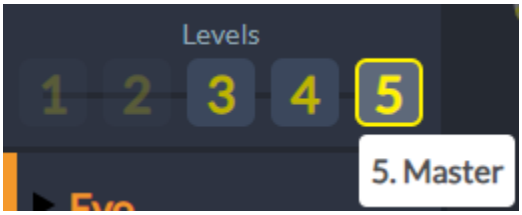


On the popup, students can select to enter the point as a Cartesian position or with joint angles. Keep the Cartesian position highlighted. On the left, students can name the point. Let students know that "My Point1" is probably okay if they only have one point. But it is good practice to get into naming the points something that makes it easier to remember the point's purpose. So something like "A4_Cup_Grab" has a quick descriptor of what the point does.

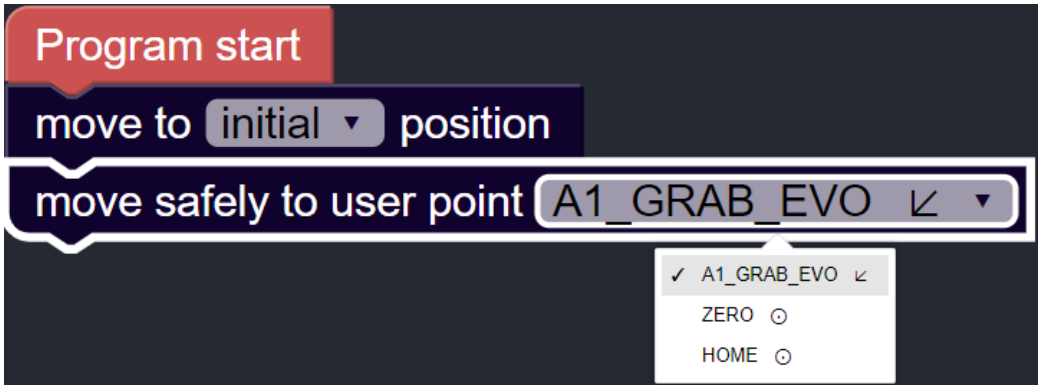


After the point is named, students can manually add the

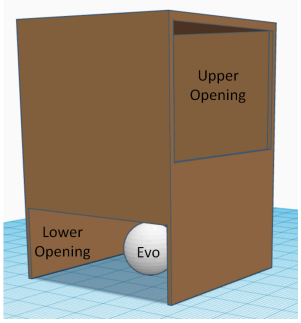
X, Y, Z, Roll, Pitch, and Yaw coordinates, or they can press "Load Current Point." This will save the ORA's position to be easily recalled.



So how do students recall a position? In the ORA Editor, students need to ensure they are in Level 5 of the Editor. Click on the ORA Tab on the left. The very top option is **move safely to user point**. The drop-down menu of this block shows all custom points (Cartesian and Joint). If students create multiple points, they will all be available in the drop-down menu.

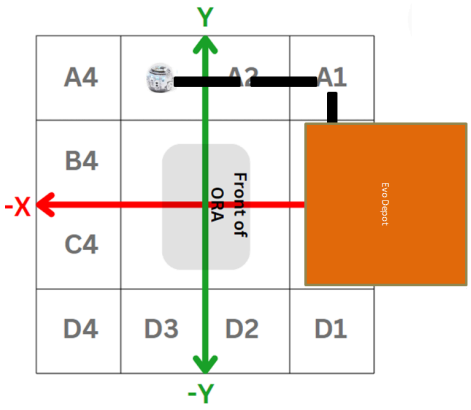


Elaborate: ORA-Evo Concurrency in Hard-to-Reach Locations



In this challenge, students will revisit Evo concurrency while utilizing user-created points to access an Evo in a hard-to-reach location.

Setup: Place Evo in A3. Use Line Cutouts to make a path from A3 to the "Evo Depot." Alter a cardboard box as shown (box dimensions approximately 30x30x60 cm). Place it on quadrants A2 and B2 with the upper opening facing ORA.

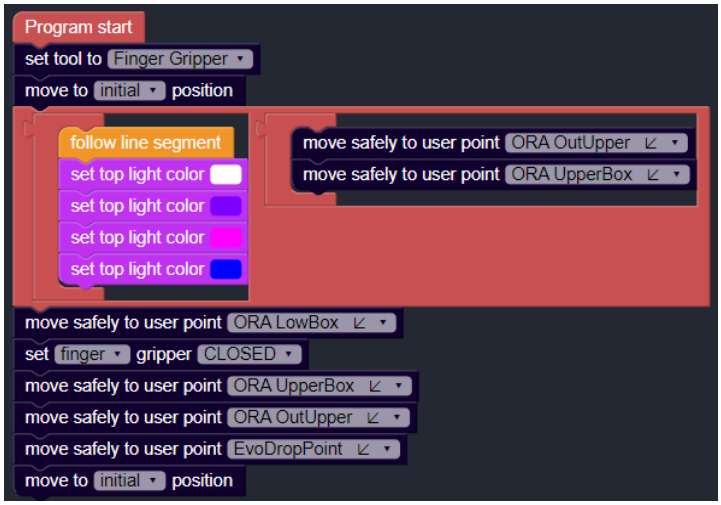


Task Requirements:

1. ORA must start in the Initial Position.
2. Students must run ORA and Evo concurrently.
3. Evo will follow the line from A3 into the "Evo Depot" and begin to flash the top lights.

4. Students will manually move ORA to create custom waypoints to guide the robot arm into the box and down to pick up the Evo.
5. Students will then return Evo to A3 to continue following the line.
6. ORA must not touch the "Evo Depot" Box.
7. ORA must finish in the Initial Position.

Example code



Direct students to plan this code in the Elaborate section of their Student Activity Sheet. After creating the pseudocode, direct them to create this code on their devices in the ORA Editor. Ensure students are connected to your shared workspace.

As students complete their code, they will test their code. Click on a given student's or student group's tab on the ORA Editor. Click "Run" to execute their code. Reset the Evo between each code run (if needed). Remember to follow the Safety Protocol established in Lesson 1.

If students are waiting for the ORA, they could:

- Answer Questions on the Student Activity Sheet
- Complete the Exit Ticket.

Evaluate: Exit Ticket

Students will fill out and submit their Evaluate section of the Activity Sheets.

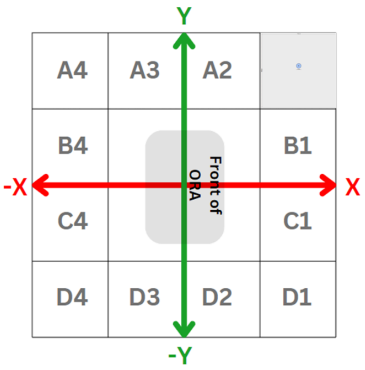
Optional Extension: Robotics Calibration Engineer

In this activity, students will assume the role of a Robotics Calibration Engineer. The calibration engineer is a subset of several different roles, including robotics technician, mechatronics engineer, controls engineer, mechanical engineer, or robotics engineer.

This person is critical to the calibration of a robot because the key aspect of any robotic device is the ability to repeat processes reliably. Without a properly calibrated robot, the end product would be ruined!

Source: Canva

In this task, students will modify the code from the Engage section to include a repeatability test. This will include having the ORA pick up a felt-tipped marker and using the ORA to make a dot in the center of the Quadrant Map Bullseye cutout in A1. Then, students will move to the other quadrants with a higher Z value to avoid marking the Quadrant Map. Run the code until the marker dot is at least 5 mm away from the center dot (touching the line of the outer circle). In this task, students will investigate how many cycles (A1, D1, D4, A4, D4, D1, and A1 are considered one cycle) it takes for each type of movement to drift from the center point.



Setup: Place Bullseye cutout in A1. Place afelt-tipped marker in ORA's finger grippers to start.

Task Requirements:

1. ORA must start in the Initial Position.
2. ORA must be coded to include a loop including:
    a. ORA must move above, then lower to touch the center of the A1 bullseye with the marker, then move straight back up.
    b. ORA must then visit D1, D4, A4, D4, D1 and then A1 again.
3. ORA must return to the Initial Position when complete.
4. Students will complete the task with both Joint and Linear motion.