

2019

Assignments

FOR VISUAL COMPONENTS 4.1 PREMIUM
ELEC-C1320 ROBOTICS

Contents

Read me first!.....	5
Basic Assignments.....	5
Basic 1: Getting started.....	6
Overlook.....	6
Camera.....	7
Robot in workspace	7
Save.....	8
Move and rotate	9
Measure	11
Snap.....	12
Drag.....	13
Attach.....	13
PNP.....	14
Analysis	15
Basic 2: Coordinates.....	16
Initialization and recap.....	16
World and parent coordinates.....	16
Align	16
Object coordinates.....	18
Analysis	19
Basic 3: Frames	20
Initialization.....	20
Frames of the tool.....	20
Base and Tool frame	21
Create your own frame.....	21
Analysis	21
Basic 4: Robot program.....	22
Initialization.....	22
Frames initialization.....	22
Subprograms	23
Dynamical base frame set for approach point.....	24
Motion statements	24
Output statement and action configuration.....	26
Release the base	26
Place the cube	27

Analysis	28
Basic 5: Tasks, signals, actions and while.....	29
Initialization.....	29
Tasks.....	30
Signals	31
Grasp detection & while + robot program recap.....	32
Analysis	33
Basic 6: Multi grasping	34
Initialization.....	34
Multi grasping and volume detection.....	34
Program the robot (recap)	35
Analysis	35
Basic 7: Recap	36
Initialization.....	36
Tool frame.....	37
Signal setup	38
Program editor initialization	39
Motion statements	41
Analysis	42
Basic 8: Welding with Delphi Add-on	43
Initialization.....	43
Frame initialization	44
Welding with Delphi Add-on.....	44
Analysis	48
Basic 9: Singularity and collision detection.....	49
Initialization	49
Singularity	49
Collision detector	49
Alternative collision detector.....	50
Analysis	50
Basic 10: Robot programming with Python	51
Accessing to Python script editor	51
Initialization.....	52
Tool Frame	52
Python script	53
Analysis	53

Main assignment.....	54
Known problems and possible fixes.....	58
My software crashed and I didn't save.....	58
The search can't find the object form eCatalog, even it is presented in assignment instructions...	58
Some of my object were lost after I saved and opened the software again	58
The robot's initial state is wrong	58
The robot started welding to arbitrary location/ weld itself when I used Delfoi arc	58
Appendices.....	60
Python example	60
Water heater model	64

Read me first!

Congratulations, you are now probably taking your first steps to get to know the Visual components premium 4.1 offline robot programming software. As you may have noticed, this document is quite long. As it states on the cover of the Hitchhiker's guide to the galaxy: "don't panic". The document is long because it has a great number of figures and the instructions try to be really thorough.

This document includes 10 basic assignments that introduce "good to know" –features. The philosophy of the assignments is to support learning by doing. After the basics, the main assignment challenges the student to recap what has been learned in the basic assignments.

In Aalto University's ELEC-C1320 Robotiikka and ELEC-D1320 Robotics courses, the main assignment is compulsory part of the course. The students are allowed to do the main assignment in the groups (max 3 group members/group).

The basic assignments are not compulsory but by completing the 10 basic assignments, the student gets the knowledge needed in the main assignment. To encourage the students to complete the basics, there is a quiz in MyCourses, that has 18 quick question to be answered. By getting more than 60% right in three tries, the student gets +2 extra points in main assignment. The extra points are personal.

The recommended way to do the offline programming exercise is to complete the basic assignments individually and the main assignment together in a group. If you and your group want to "optimize" the usage of time then probably the best optimization is that you all do basics 1-5 and 7 and share the rest of the basics within the group.

Notice that if you get stuck with the basic assignments or especially with the main assignment, you should do the following:

- 1) Read the instructions again and try to find the error on your doing and Re-do the assignment or part of it
- 2) Check the known problems and possible fixes. We update this section during the course
- 3) Ask your group members or other students on the course if they have faced the same problem and if so, what they did to avoid the problem
- 4) Come to the exercise sessions (Tuesdays 16-18 o'clock in AS1) and ask for help or send e-mail to the teaching assistant (harri.aaltonen@aalto.fi)

Commercial usage of this document without separate permission from the owners of the document is forbidden.

Basic Assignments

The basic assignments support the understanding of how visual components 4.1 can be used. The assignments gives the students organized learning path of features, know-how and most importantly know-why.

At the beginning of every assignment there is an objective determination. At least at the end of the assignment, the expected output is explained. There are usually also expected output after every few steps.

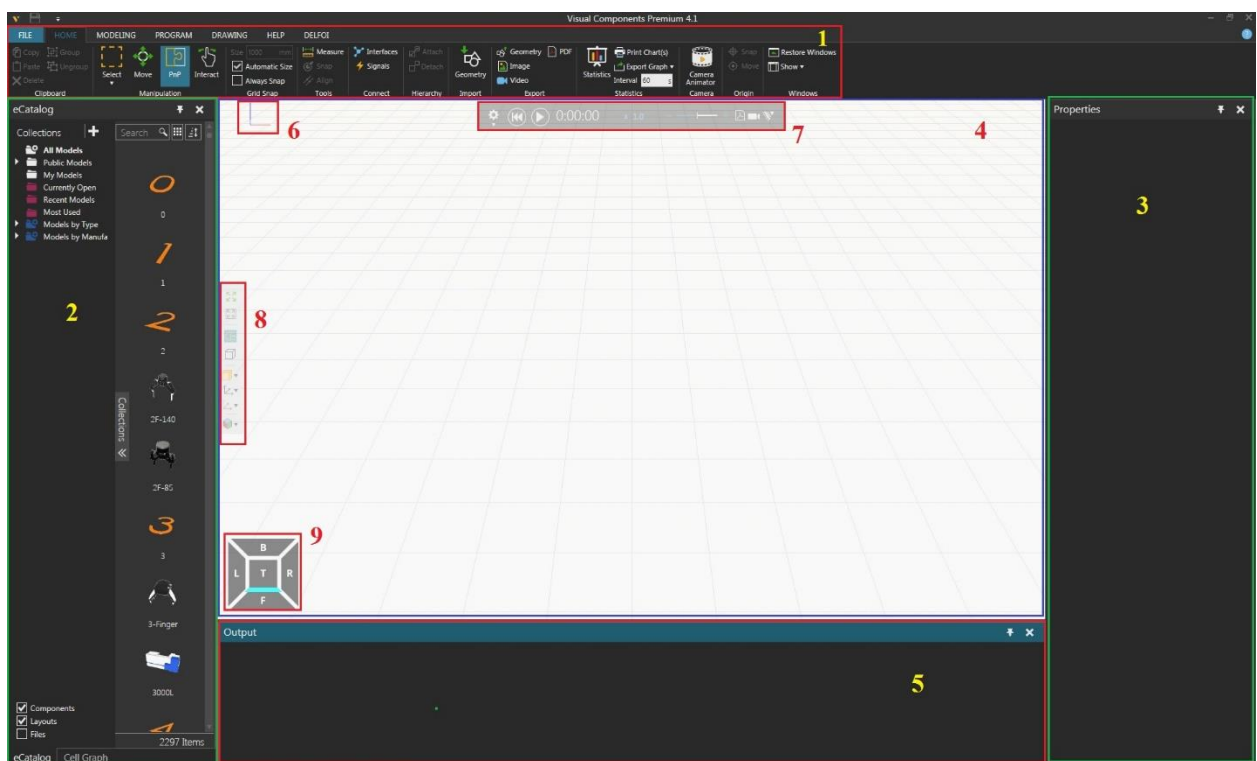
Basic 1: Getting started

The objective of the first assignment is to get known with the home view. In this assignment, we introduce next features:

- Navigation
- Operating
- eCatalog
- Cell graph
- Move
- Rotate
- Component Properties
- Measure
- PNP
- Snap

Overlook

The overlook of the visual components 4.1 is following:



The Visual Components 4.1 window consists of:

1. Menu on top/top panel
2. Panels on the left
3. Properties on right
4. Layout/ workspace
5. Output
6. World coordinate system
7. Simulation control
8. World view side buttons
9. View selector

1. The menu on top is the way to navigate between different views. Under different views, the tools and the panels on the left and right may differ by the provided functions
2. The panels on the left include the main functionality of the view.
3. The properties on the right allow the user to change the properties of selected object or the tool that is in use
4. You may build your simulation model to layout/ workspace
5. To output it is possible to print error messages etc.
6. World coordinate system is the main reference coordinate system
7. The simulation control includes the simulation control tools (play and reset) and the video recording tools
8. World view side includes shortcut commands for navigation, frame visibility controllers etc.
9. View selector includes the preset views.

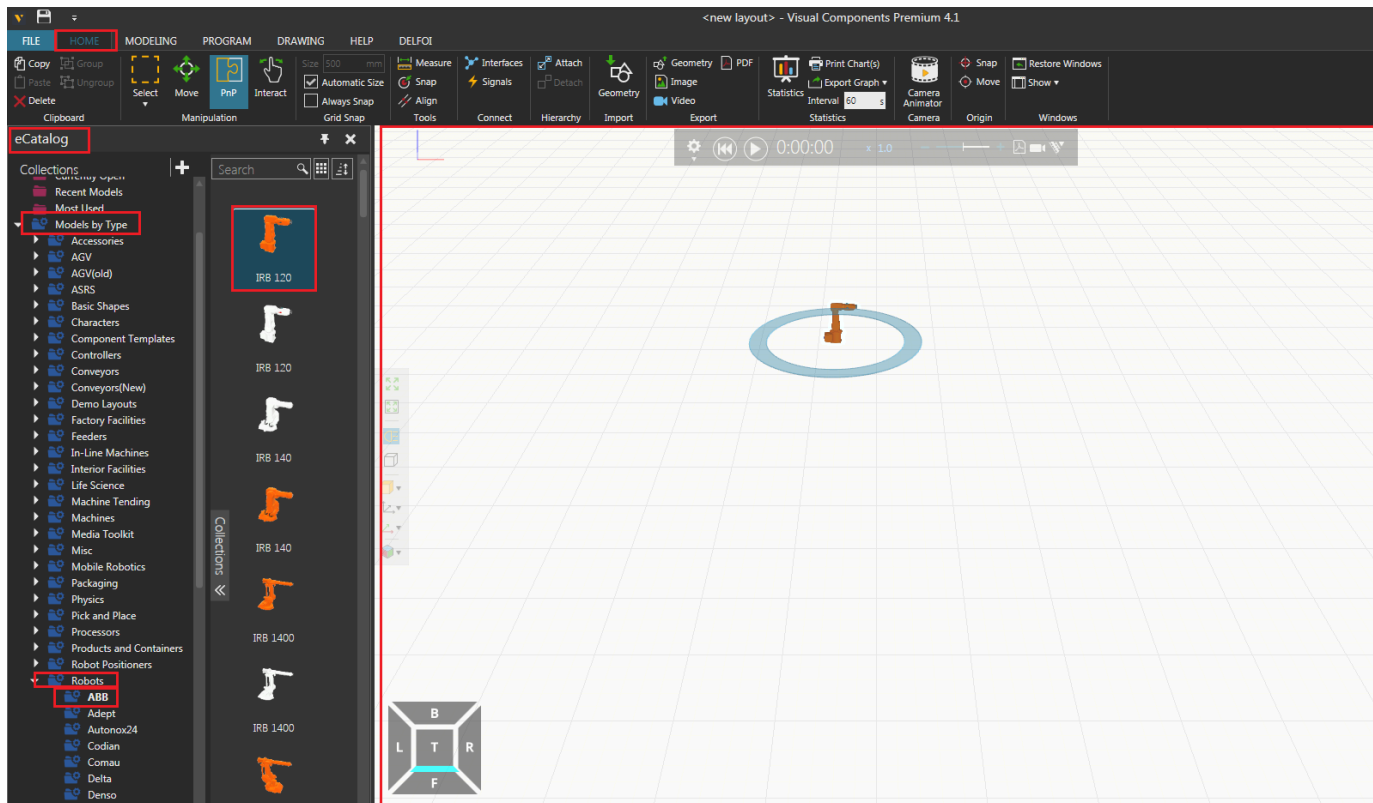
Camera

The camera of the 3D and drawing worlds can be interacted with by using a mouse.

- Right mouse button (RMB) will rotate the camera
- Left and right mouse buttons (LMB+RMB) will pan the camera along the horizontal and vertical axes of the viewport.
- To zoom the camera, you can rotate a mouse wheel or hold down SHIFT and the right mouse button (SHIFT+RMB)

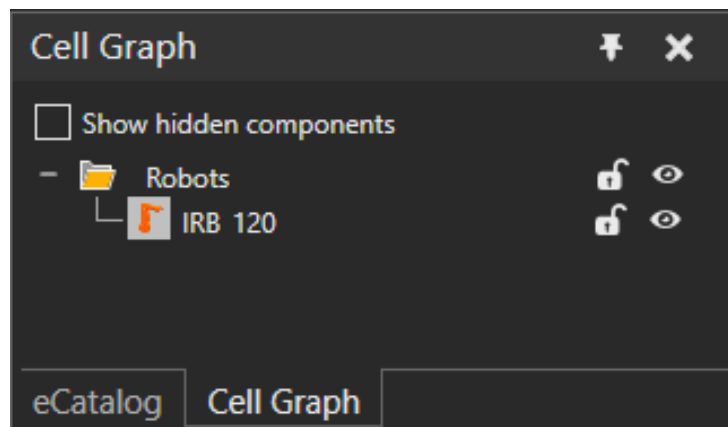
Robot in workspace

1. In home view go to Home/eCatalog/models by Type/Robots/ABB/IRB 120. Drag and drop it to work space.



2. From home view's left panel. In the bottom of the left panel is a tab called "cell graph". Click it and you should see the robot you just added to work space.
 - What happens if you click the eye next to the component?
 - What happens if you click the lock?

Make the robot visible and unlocked before next step



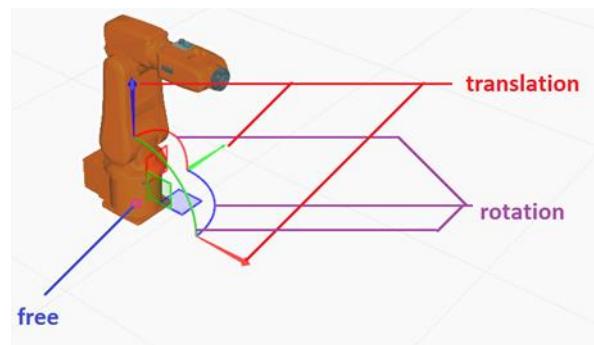
Save

3. Now we have done something concrete, it is good to save your work. Go to file view on top panel and choose save as.
4. On the right panel, give the project name and author. Check also the **Include components** from the top of the right panel. To avoid problems of missing components, always **include components, when saving the first time.**

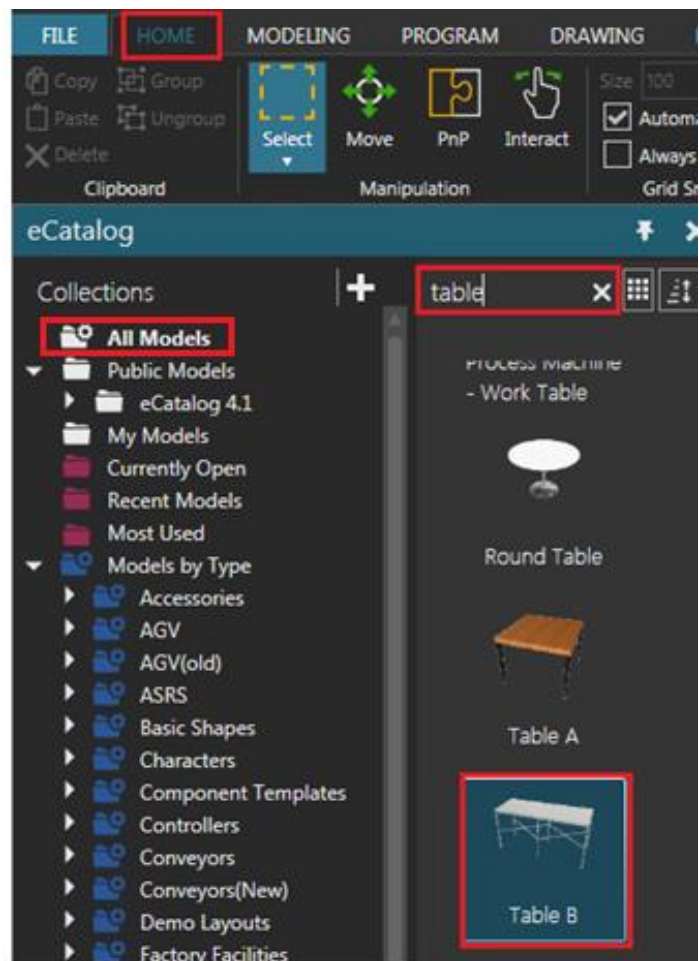
5. Save your file to path home.org.aalto.fi<user name>\data\Documents\Visual Components\4.1\My Models by clicking computer and My Models folder after you have chosen Save as. By this you have all your models in one place and you can use them easily later if and when needed

Move and rotate

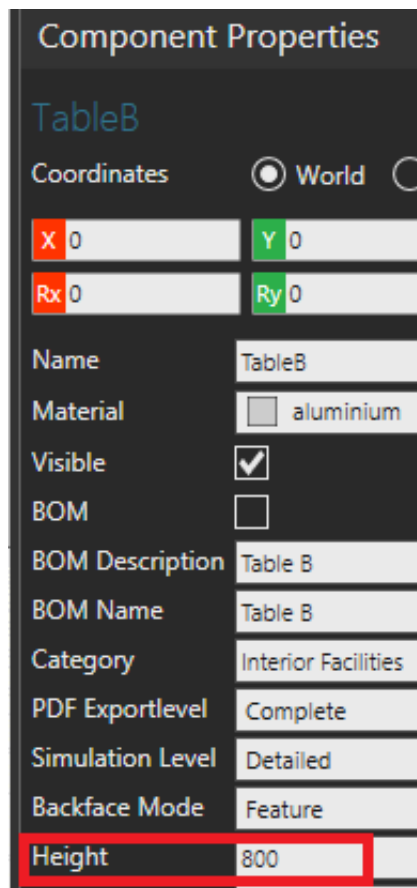
6. If you dragged and dropped the robot, it is now at arbitrary place. The robot can be moved and rotated in home view/top panel/manipulation/move. The moving can be done by dragging along axis, along plane or freely. The rotation can be also done by dragging around X-, Y- and Z- axis.



7. The other alternative is to change the location and orientation of the robot from home view/right panel/component properties (the robot must be selected) by changing the numeric values on the top of the component properties (X, Y, Z, Rx, Ry, Rz).
8. New location and orientation is $(X, Y, Z, Rx, Ry, Rz) = (0,0,0,0,0,0)$. If you click the corresponding coordinate letter (X,Y,Z,Rx,Ry,Rz), it will reset the value. Make sure that you use World coordinates (is selected as default in object properties in the panel on the right). The other two coordinate systems are presented in basics 2.
9. We want to operate the robot on the top of the table. Search from home view/panel on the left/ eCatalog and type to search "table B". Click the All models so you can search Table B from all models. Drag and drop it to work space and move it to the origin too. The alternative way is to double click the table in eCatalog to summon it to workspace. In this case, the table will automatically go to origin.



10. Operating the robot under the table is possible, but stupid. Move the robot up by changing the robot's z-coordinate. Click the table and look at the component properties on right panel. The height of the table is supposed to be 800 as default. Change the robot's coordinate to match the table height.



There seems to be a problem in robot's location. We have now five options to get the robot on the top of the table

- guess the height of the table top and change the robot's z-value from properties
- click the move on home tab and drag the robot from the blue arrow (via z-axis) and estimate the height by visual approximation
- measure the tabletop height and do the first one
- snap the robot to the table
- click the move on home tab and drag the robot from pink circle

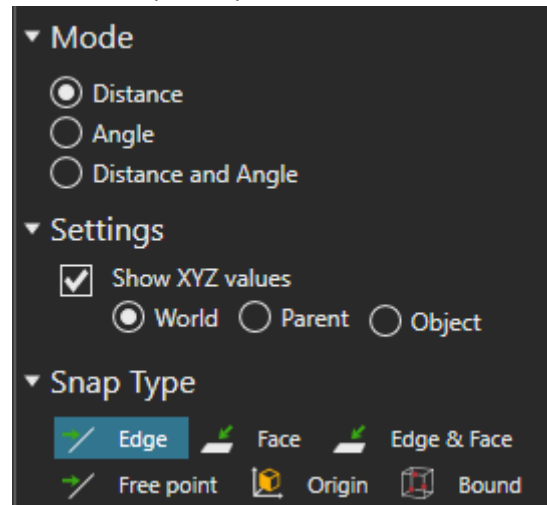
The two first options are based on guessing and hence they are not valid (at least alone).

Measure

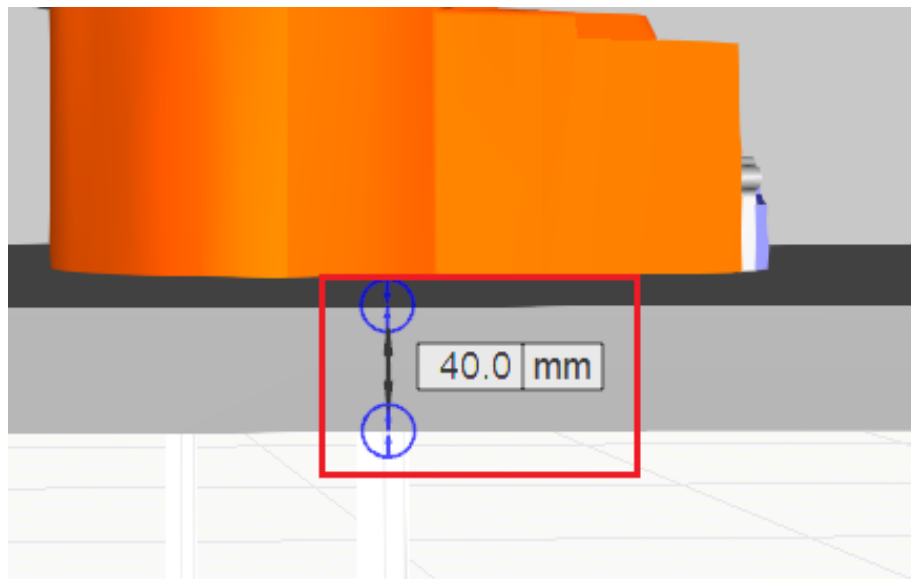
11. Measure the height of the tabletop and change the robots properties based on the measurement. Look the table from the front. You can use View selector in the left bottom corner of the workspace (figure below).



12. Select “measure” from home view/ top panel/Tools. It opens to panel on the right “the measure option”. Choose the options presented below:

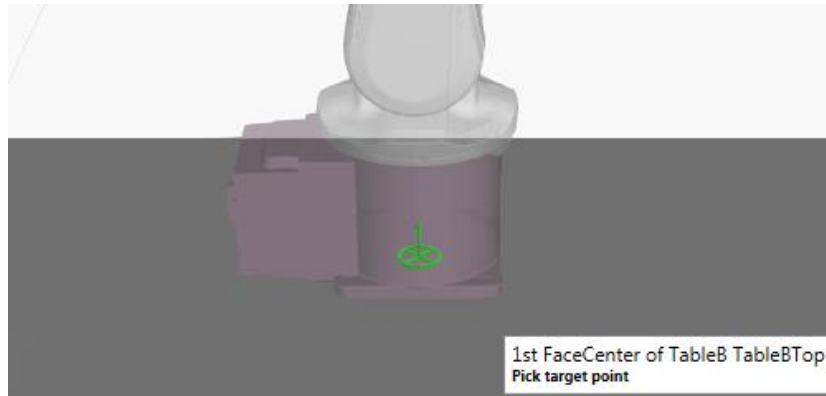


Click the top and the bottom of the tabletop. You should get the measurement. Add the correct height to robot’s z-coordinate. The expected output is presented below.



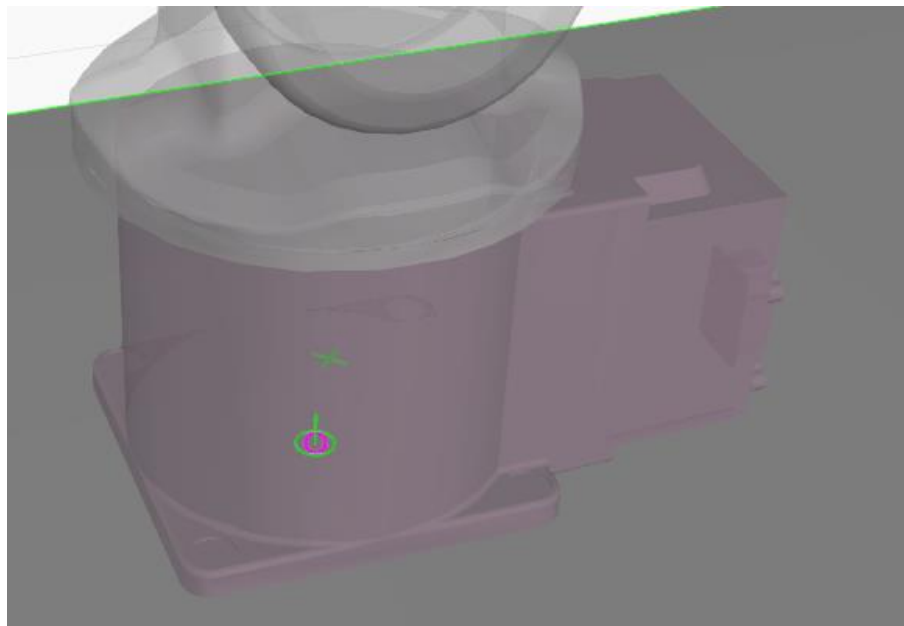
Snap

13. The other option is to snap the robot to wanted position. Move the robot back to origin by resetting the z-value from right panel/components properties. Select the robot. From home view/top panel/tools take the snap. It will open to the right side of the workspace.
14. Choose 1 Point Mode, all settings chosen and Center as snap type. Snap the robot in to middle of the table top. For some reason the robot’s x- value is non-zero. Reset the x-value.



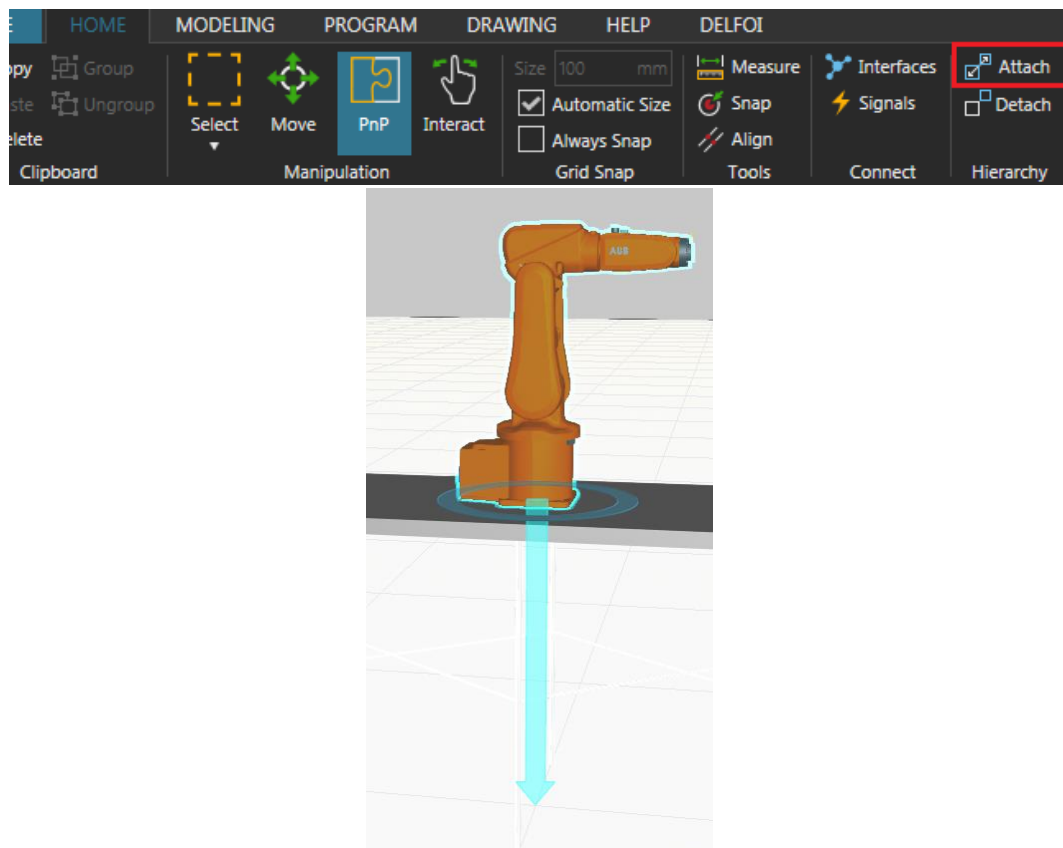
Drag

15. Return the robot to origin again. The last presented alternative is to choose the move from home view/top panel/manipulation and drag from the pink circle to green cross. The green cross turns visible when the robot is close enough to table. The center point is again in $(x,y)=(0,-0.379)$. Reset the x value from robot's properties. Robots final location values should be $(x,y,z)=(0,0,840)$.



Attach

16. The robot should be on the table. The problem is that if we move the table, the robot stays on its current position and we have to move them separately. We want that the robot follow the table automatically. Use the attachment feature by
 - select the robot (child)
 - choose Attach from home view/top panel/Hierarchy
 - select the table (parent).



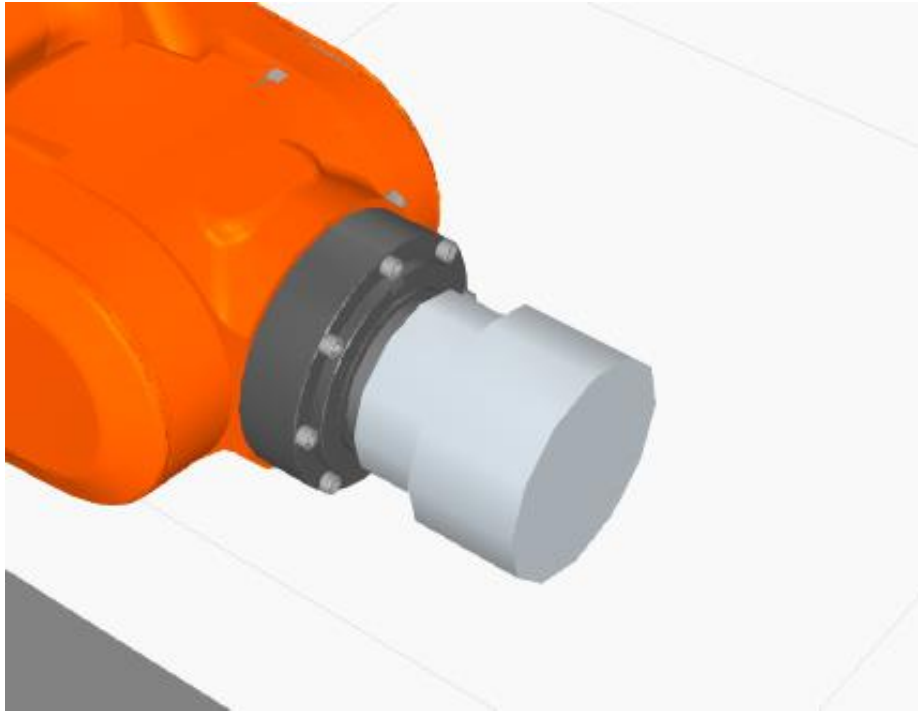
- Fill the table

Action	Output
Move the robot	
Move the table	

Return the table to origin and the robot to the center of the tabletop.

PNP

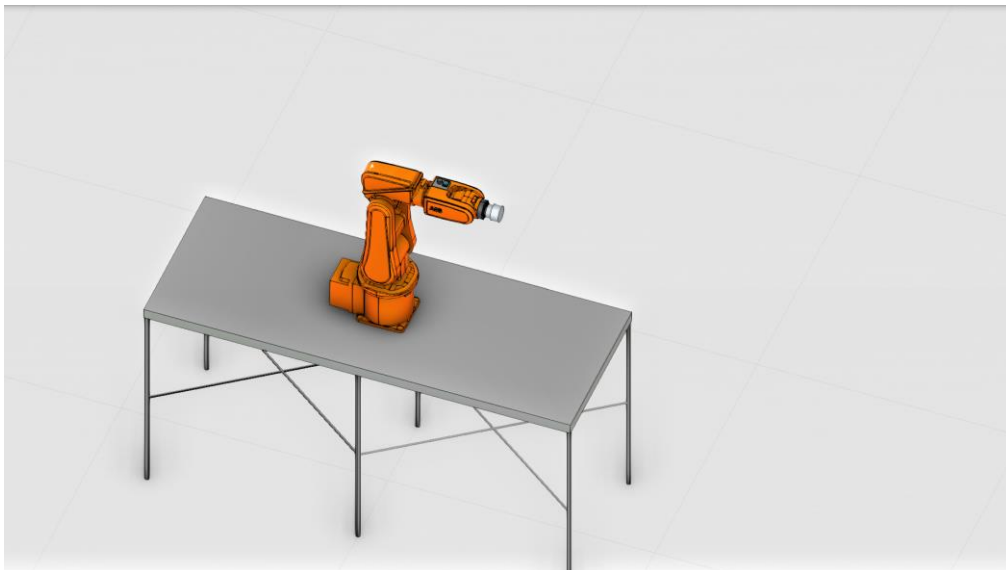
17. The last thing of the first assignment is the PNP (plug n play). Go to Home/eCatalog/Tools/Visual Components and choose Generic vacuum Gripper. Drag and drop it to workspace. From home view/right panel/component properties, move the gripper to $(x,y,z)=(500,0,1470)$.
 18. Select the tool. Choose from the home view/Manipulation the PNP. Then drag the tool close to the robot's tip so that the tool is snapped to the robot. Notice that the tool won't snap if it is too far away from the target. Select the gripper and click home view/top panel/Hierarchy/attach and look to the panel on the right.
- Is the tool attached to the robot?



Analysis

19. Recap the features

- How can you move and rotate the object
- For what the attachment tool is good for



Basic 2: Coordinates

In previous assignment was a mention of different coordinates. We used only World coordinates but they are not always the best and most efficient choice to use. In this assignment, we try to get familiar with two other coordinates and understand where they are efficient to use.

Initialization and recap

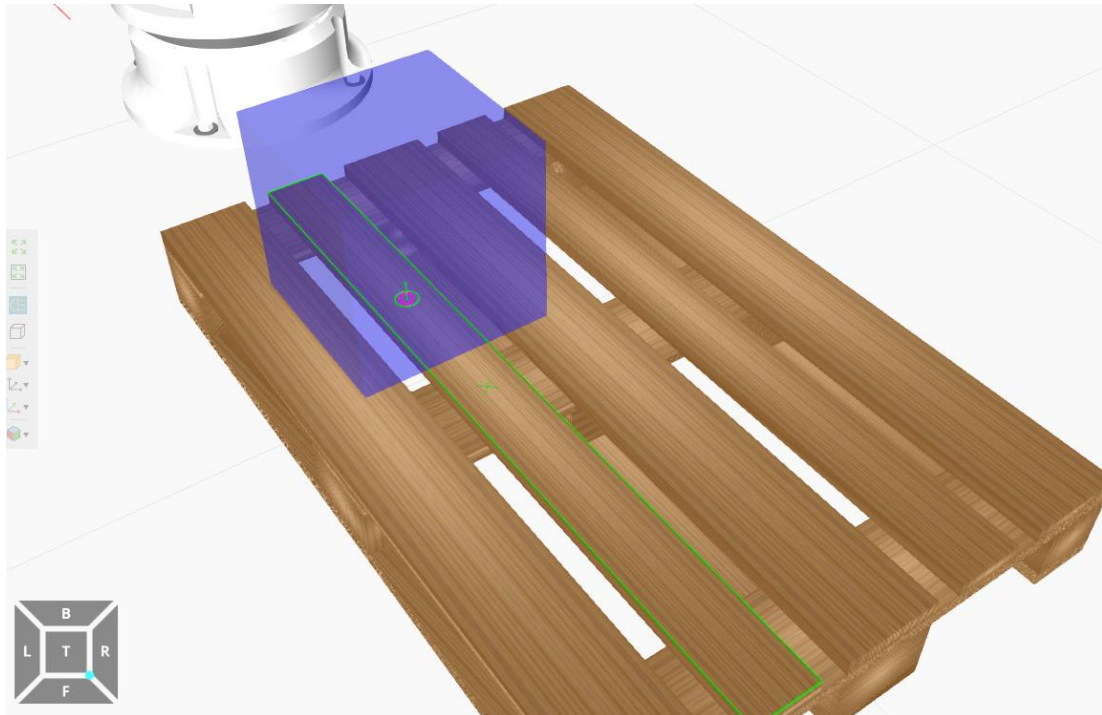
1. Start a new project. If you continue right away from basics one, save your previous work and then clear all in file view.
2. Add a "KR 100-2 P" – robot by KUKA, an "euro pallet" and a "cube"
3. Move the robot to world coordinates (0,0,0,0,0,0)
4. Move the pallet to world coordinates (1800,1200,0,0,0,0)
5. Move the cube to world coordinates (1800,-1200,0,0,0,0)
6. The cube needs to be size 300 x 300 x 300
7. Attach the euro pallet to robot with alternative method. Click the pallet and attach from home view/top panel/Hierarchy. It will open to right panel the "Attach to parent". Choose "KR 100-2 P".

World and parent coordinates

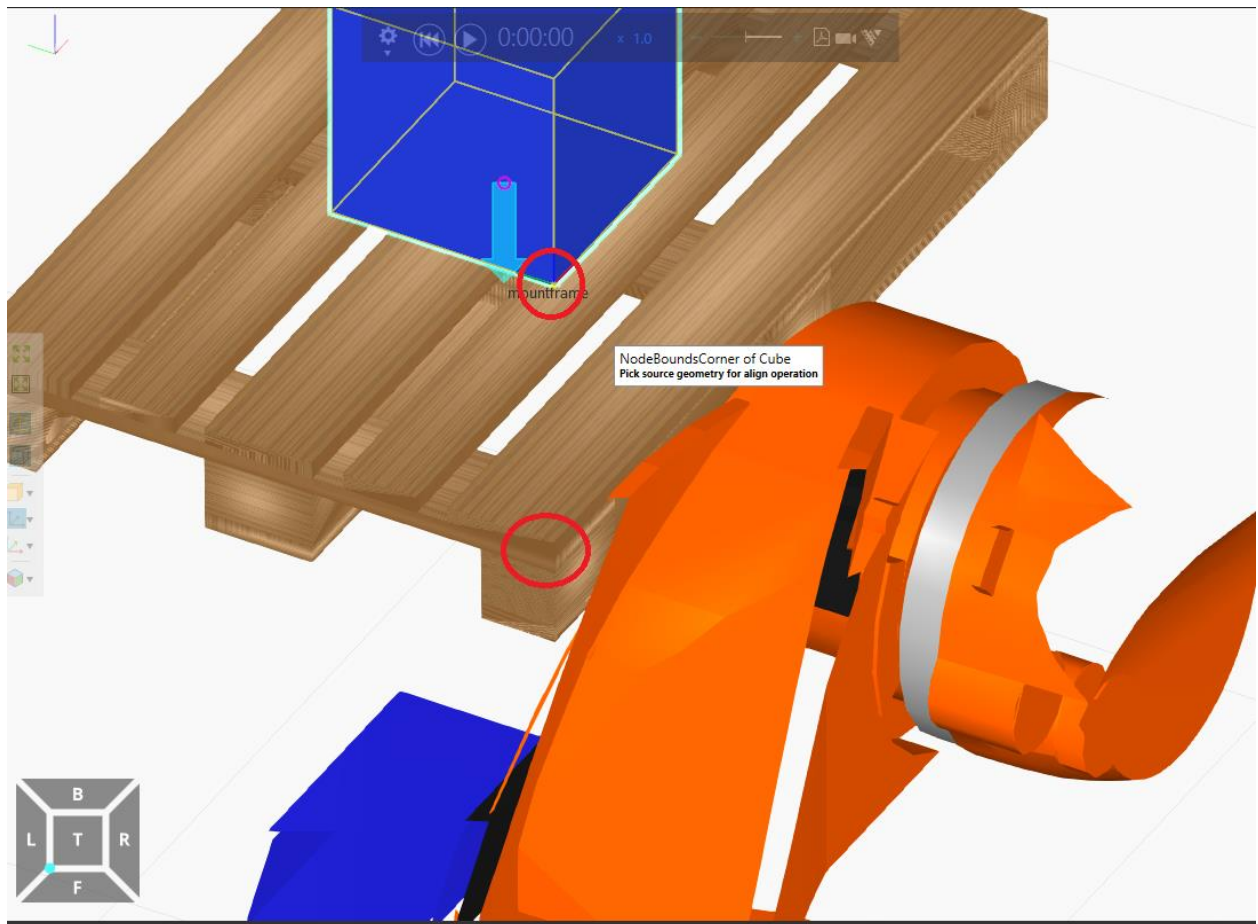
8. Move the robot to world coordinates (0,1500,0,0,0,0)
9. Click the euro pallet and change the coordinate system to be parent from component properties (panel on the right).
 - In new location
 - o What are the pallet's parent coordinates?
 - o What are the pallet's world coordinates?
 - What happens if you move the robot back to origin (0,0,0,0,0,0)
 - o What are pallet's parent coordinates?
 - o What happens in robot's world and parent coordinates if you move pallet?
10. Move the robot back to origin and pallet to (1800,1200,0,0,0,0).
 - *Explain, what are the world and parent coordinates*

Align

11. Next we want to put the cube on the pallet. Easy way is to move it from free point (pink circle in the bottom of the cube) on the pallet. The cube's z- value in world coordinates should be 144.

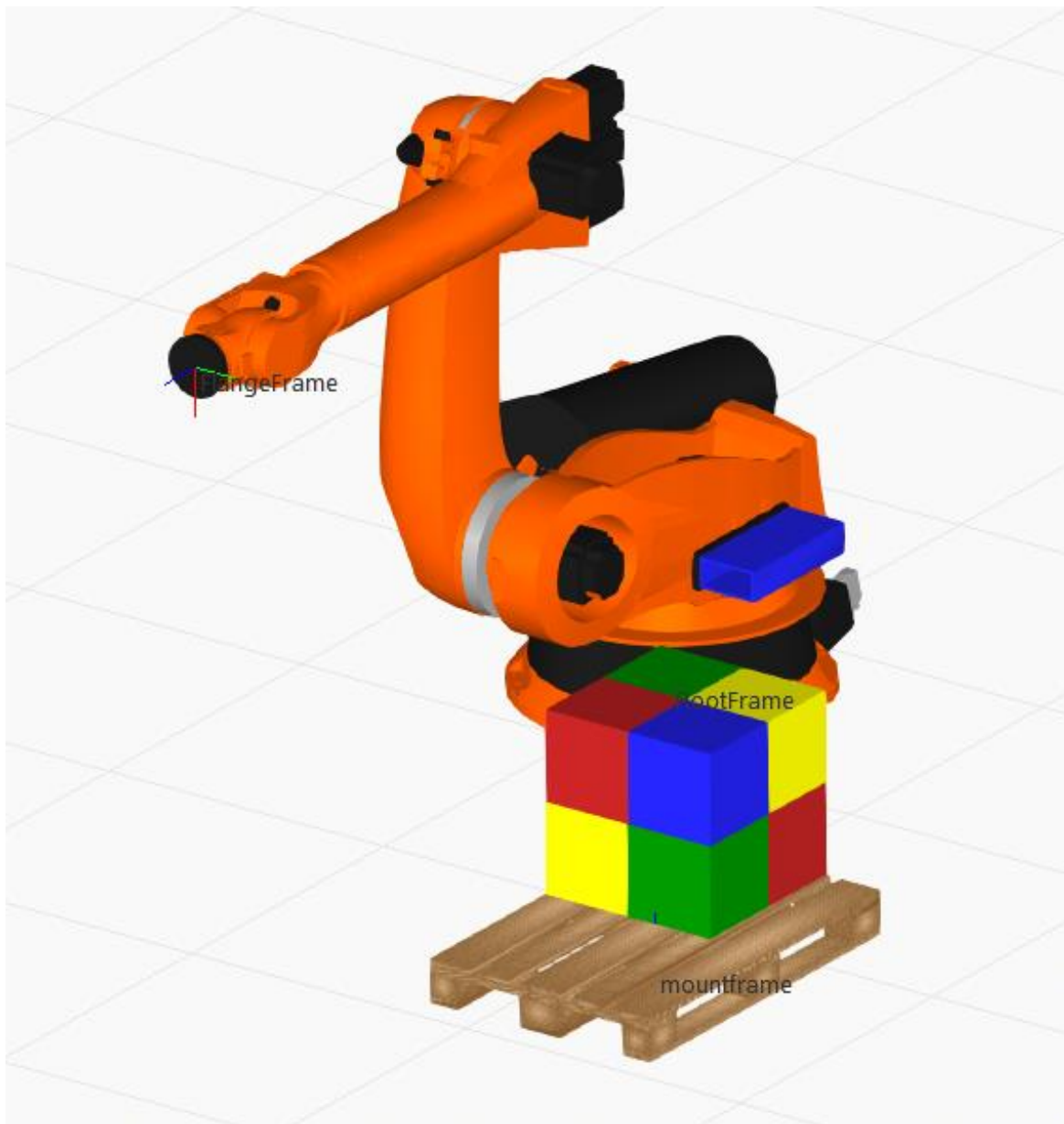


12. Attach the cube to pallet. Then use the parent coordinates and move it to parent coordinates (0,0,144,0,0,0).
13. Next we want to move the cube to corner. Click the cube and then Home/Tools/Align (snap type: Bound). Then click the bottom edge of the cube and then the corner of the pallet that is the nearest to the robot. The cube should move to the corner.



Object coordinates

14. Next we want to make 8 cubes in matrix of 2 x 2 x 2. The intuitive way is to add to the size value of the cube to world coordinates x and/or y coordinate. Instead we use the object coordinate. Copy the cube by choosing the cube in Home view/panel on the left/Cell graph and then choosing in Home view/Clipboard the copy and paste.
15. Move the new cube to the same place than original cube. Easy way is to repeat the aligning (section 13). Open the Home view/panel on the left/Cell Graph. Select move in Home view/top panel/Manipulation. Click cube #2. Change the coordinates to "object" and change the z coordinate to 300. Click the world or parent coordinates and then to object coordinates again.
 - Explain what are the object coordinates (hint: the object coordinates reset, when you change the coordinate system to parent or world or deselect the object)
16. Now create the 2 –by – 2 – by – 2 matrix from 8 cubes. Notice that you can select more than one object at the time by holding Ctrl while choosing objects in workspace. By this information, you need to copy only 3 times.



17. Save the solution

Analysis

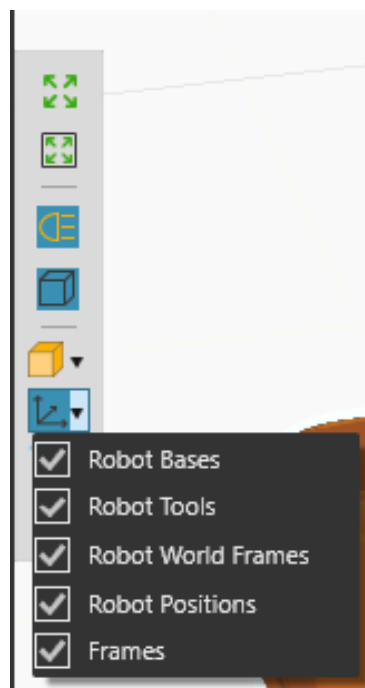
- When ready, copy the robot. What happens and why?
- What are the advantages and disadvantages of the different coordinate systems?

Basic 3: Frames

On the previous assignment, we introduced three different coordinates. In addition to coordinates, some of the objects have frames. The frames come handy, when we work with the robot and conveyors. The objective of this assignment is to understand how to use frames and where those are useful.

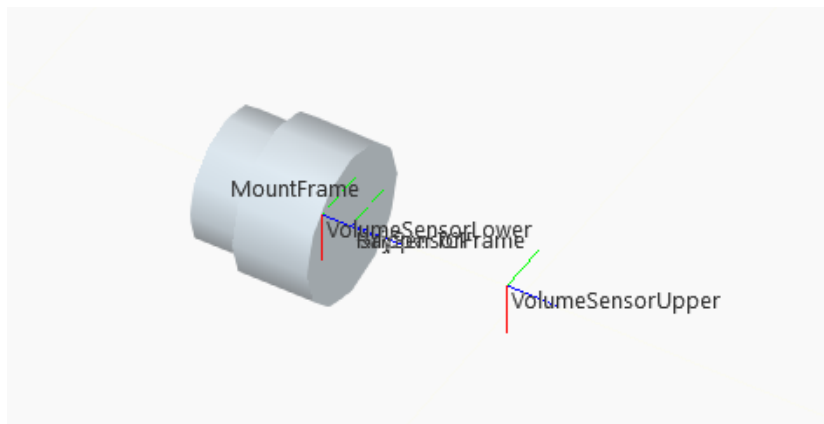
Initialization

1. Load the solution 1 and save it as separate file. You may name it as “solution3”
2. On the left side of the workspace is a vertical bar. Click the top-down bar of Frame Types and select all. Click the Frame types- icon to bring the frames visible.



Frames of the tool

3. Move the gripper tool off from the robot. It seems that there are some frames that are on the same location and we can't see them.
4. Select the tool and go to modeling view. On the left panel are features, named Root[Vacuum Gripper]. Click the plus sign above the root to expand the features. From the features we can see all the frames
5. Plug and play (PNP) the gripper to robot in home view.



Base and Tool frame

6. Click the robot and go to modelling view. Expand the features and look at the frames available. It seems that there is only Root frame available. Expand the behaviors and properties on the top of the left panel. Now you should see 16 base frames and 16 tool frames and the Gripper TCP.
7. The Base and Tool frames of the robot can be considered as local frames that simplifies the manipulation of robot's joints position and orientation. More about this in Basics 4.
8. Click the tool 1 on the left panel/Component graph. It pre-selects the "move" from manipulation. Now move the tool 1 along the parent coordinates z-axis to z=200. You can either move it by dragging it or by changing the "Tool Properties" on the right panel.
 - What are the corresponding world coordinates?
9. Go to home view/manipulation/select and click the robot. Then from the right panel/component properties, change the Axis 1 value to be 90.
 - Check from the modelling tab, what are the tool1 world and parent coordinates?
10. Move the tool 1 to its original position by resetting the tool1 parent coordinates in modelling view.
11. The base frame is very similar to tool frame. Instead of moving the base frame, we connect the robot's base frame to other object.
12. In modelling tab/left panel/Component graph/Behavior/IRC5R/Bases select wobj1 (ABB's way to name base frame 1). Next from the right panel's Tool properties change the node from Null to TaleB::TableB
13. Move the robot world coordinates origin
 - What are the world and parent coordinates of the base frame 1?

Create your own frame

14. Go to home view. Click the table and return to modelling view. Expand the bottom of the left panel (Root[TableB]). You should see that there are two Frames on the table. Go to model/Top panel/geometry/Features and choose frame. Move the created frame to $(X,Y,Z,Rx,Ry,Rz)=(0,0,840,0,0,90)$. The created frame should be now in the middle of the table. Name the frame as "middle frame"
15. Go to home view. Move the selected robot with snap tool to base frame you just created by using snap type "frame"

Analysis

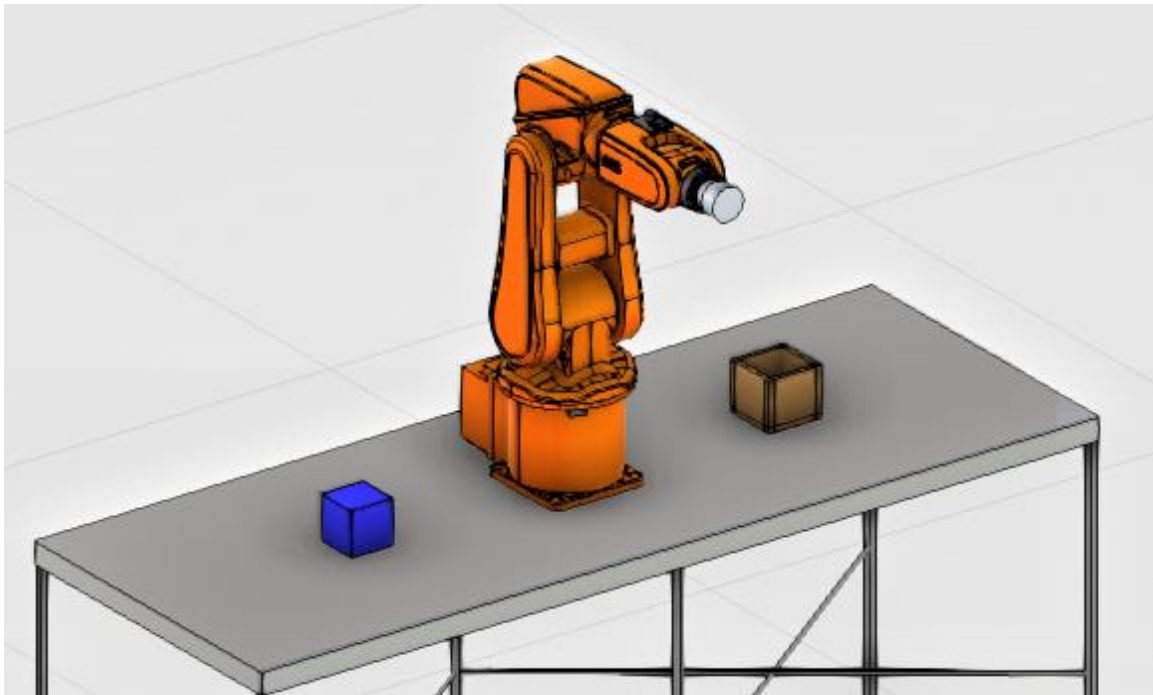
- There are coordinates and frames. Recap and explain the difference of different coordinates and frames.

Basic 4: Robot program

The purpose of this assignment is to learn how to program the robot to pick 'n place an object. Our main goal is to move the cube from table frame 1 to table frame 2. The secondary goal is to show how to program robot so that it will pick and place the cube to box in situation where the location and orientation of the cube and box is changed, without changing the program.

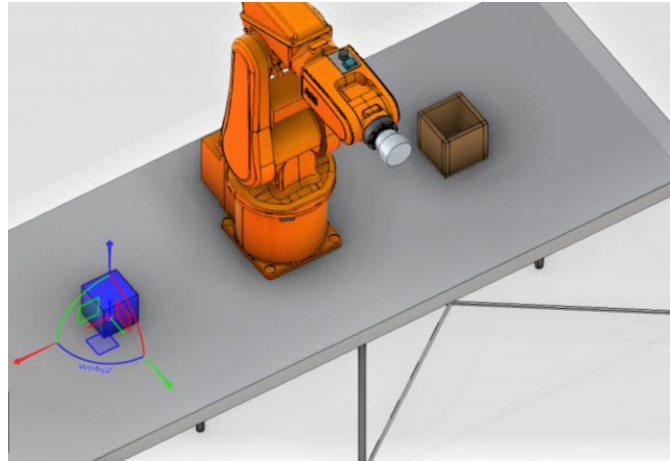
Initialization

1. Load the solution of Basic 3 and rename it for example as solution4.
2. Add a cube and a box from home/eCatalog/modelsByType/Products and containers/Visual Components
3. Move the cube to table frame 1 (remember to click the frames visible) and change the size to be 80^3
4. Move the box to table frame 2 and change the size to be (L x W x H x T)=(101 x 101 x 90 x 10)
5. Tool frame 1 should be at its initial position. If not, move the frame back to initial positions. Easiest to do this is by going to modeling tab/component graph/behavior/Tools and choose parent coordinates on the right panel and reset the values of the coordinates
6. Select the robot in home view/Top panel/Manipulation. Change the axis1 value on the right panel to 0. The initialization should look like this:



Frames initialization

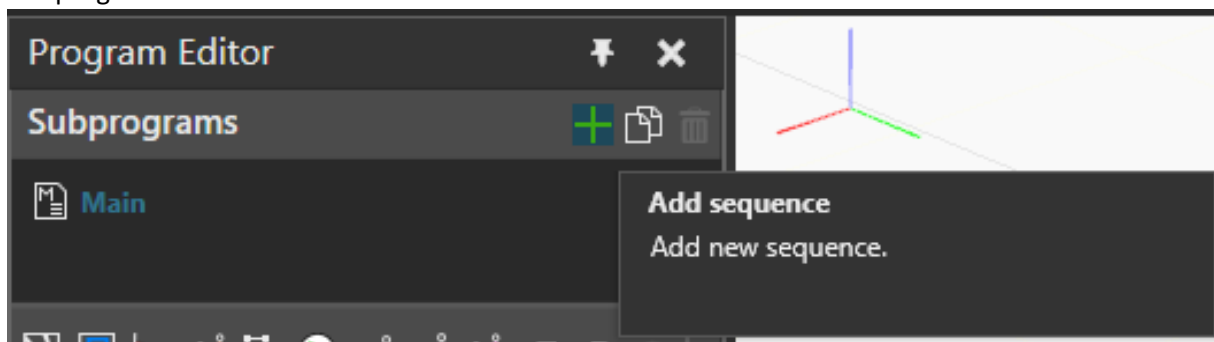
7. The frames were introduced in the Basic 3. Now we are about to use them. Choose the robot and go to modelling tab. The base frame 1 should be now connected to the table. Connect the base frame 2 to node Cube:Cube in Modelling view/Component graph/Behavior/Bases/ the panel on the right. Change all the parent coordinates of the wobj2 to zero after the change of the node.



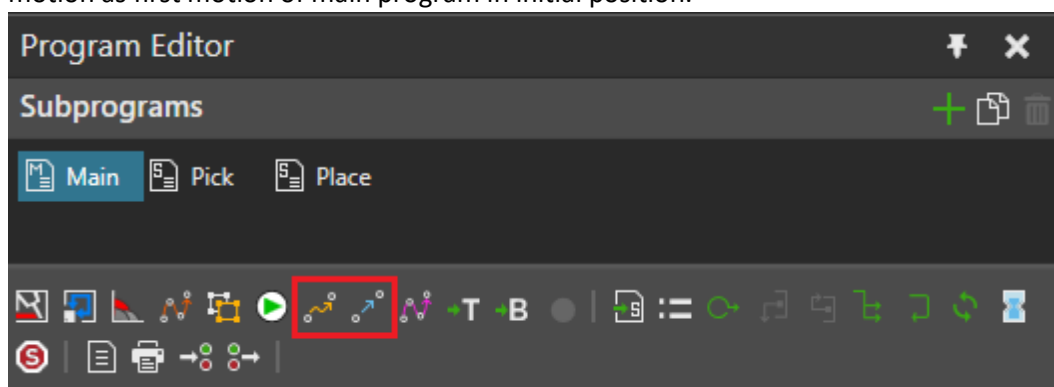
8. Connect the base frame 3 to box (choose node box:box) and set the parent coordinates to $(X,Y,Z,Rx,Ry,Rz)=(0,0,0,0,180,0)$.
9. Move the tool1 frame to parent coordinates $(X,Y,Z,Rx,Ry,Rz)=(0,0,50,0,0,0)$ and check that mount plate is the node.

Subprograms

10. Go to program view and click the Robot.
11. On the left panel is program editor. The first option is subprogram. Add two subprograms from green plus-sign and name them "pick" and "place". Name can be edited by clicking the routine and changing the name of the routine in the panel on the right. You should have now three programs in total.



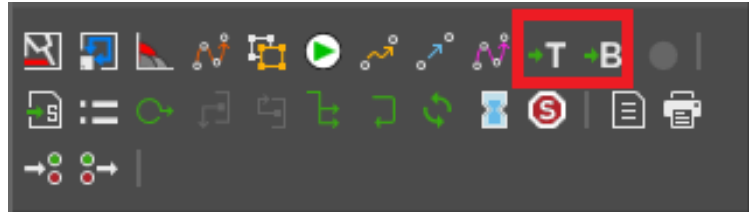
12. There are two different motion statements: Linear and PTP (Point to point). Add a PTP or a linear motion as first motion of main program in initial position.



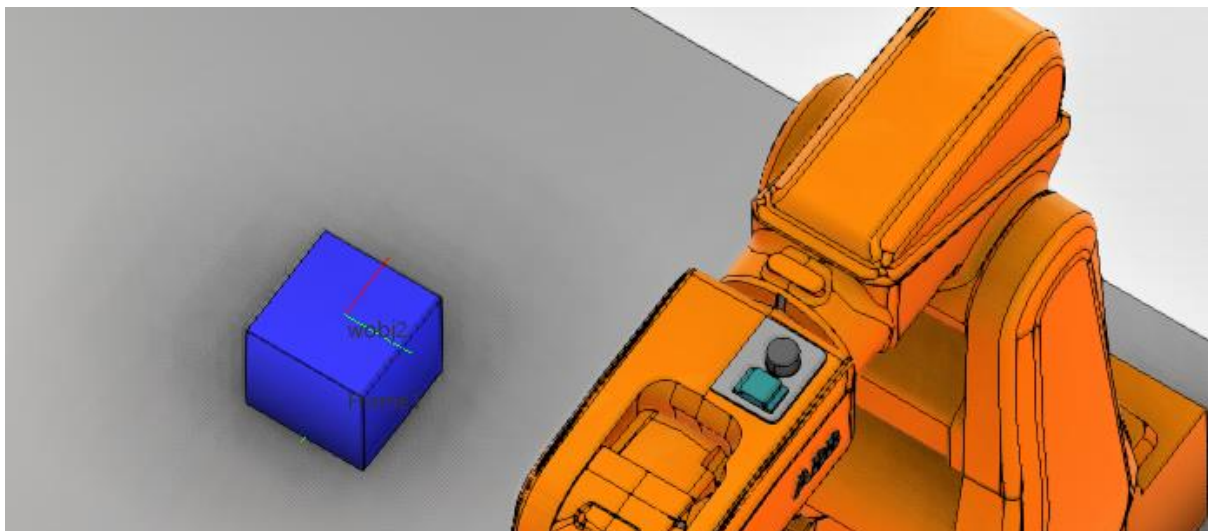
- Name advantage of initial position

Dynamical base frame set for approach point.

13. Earlier we determined the base frame 2 into origin of the cube. We can move the base frame 2 dynamically while the program is running. In program view/left panel, click the main program and from the Program/Subprograms, choose setBase (capital B). In the panel on the right, choose from drop-down menu the wobj2 and change the position to $(T_x, T_y, T_z, R_x, R_y, R_z) = (0, 0, 80, 180, 0, -90)$. Tick the "IsRelative" box. Drag the base statement to be after the initial motion.

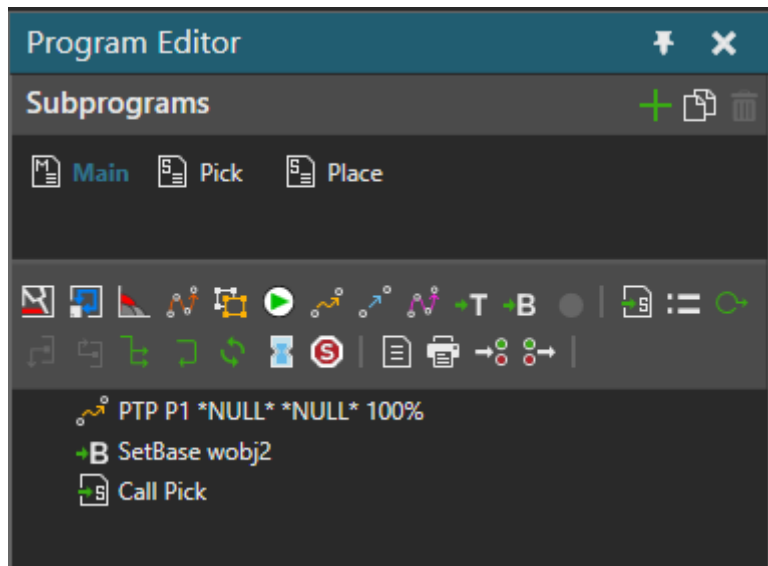


14. Set Halt- statement (looks like stop-sign) after the SetBase-statement. This stops the simulation and it can be continued by pressing the play again.
15. Run the simulation from simulation tab in the top of the workspace. You should now see the base frame 2 on the top of the cube. DO NOT RESET THE SIMULATION.

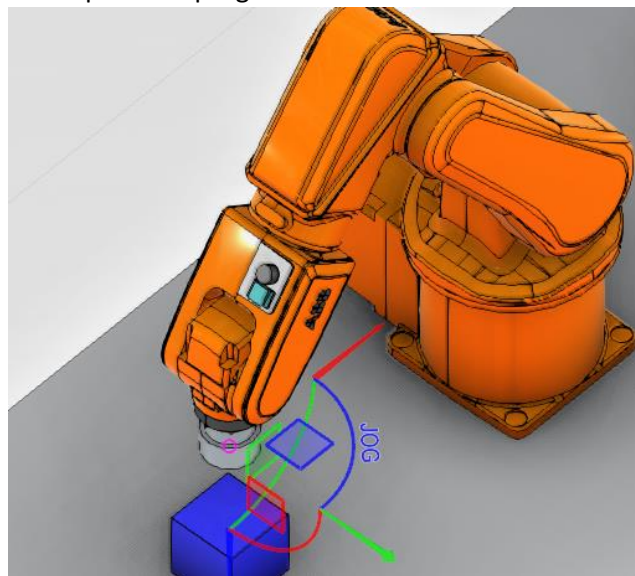


Motion statements

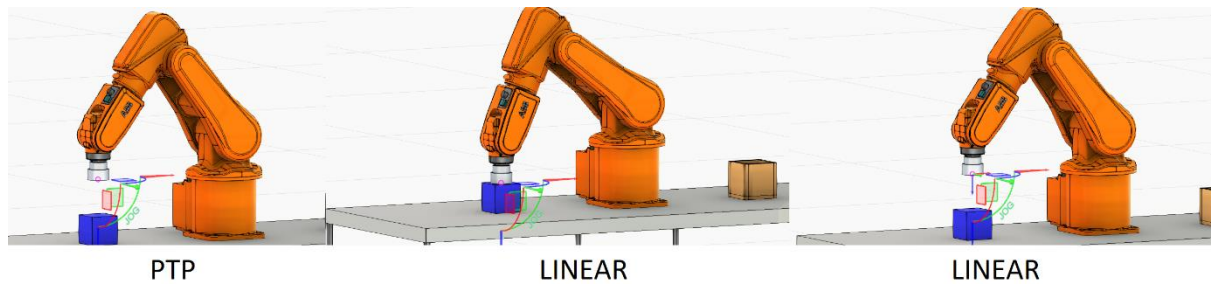
16. The description of pick subprogram is following: The tool approaches directly above the cube – lowers the tool on the surface of the cube with linear motion and comes up with linear motion. In main program "call sequence statement" (letter "s" on paper, pointed by green arrow) and from panel on the right, choose "pick". Drag the statement to be after base statement. Delete the halt statement.



17. In Program view/top panel/manipulation choose Jog. From the panel on the right, select the tool to be tool1 and base to be wobj2. Now the jogging movements are done with respect to tool and base frames.
18. Take snap tool (program/top panel/Tools) and choose “1 point mode” and “frame” as snap mode. Zoom close enough to cube and snap on top of the cube. Go to pick sub program and press linear motion.
19. Jog the robot directly upward 100 units in world coordinates – for moving, object coordinates works perfectly fine in this case (notice: do not jog the tool nor the previous statement – check that you have Robot/IRB120 on the right panel’s jog view). Check that you still have Tool1 and Wobj2. Add PTP motion to “pick” subprogram.

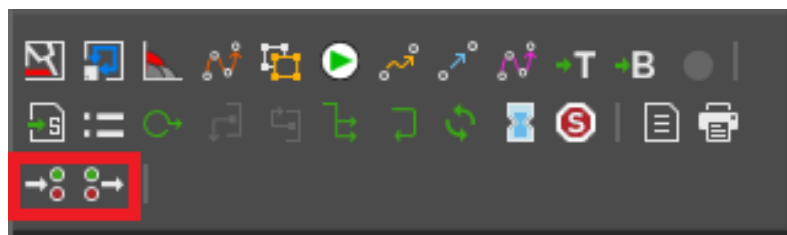


20. The “withdrawing” motion up from the box surface can be done with linear motion to same position as the approaching point was. Click the PTP you just created and add linear motion.
21. Re arrange the three motions as in figure below. Run the simulation.

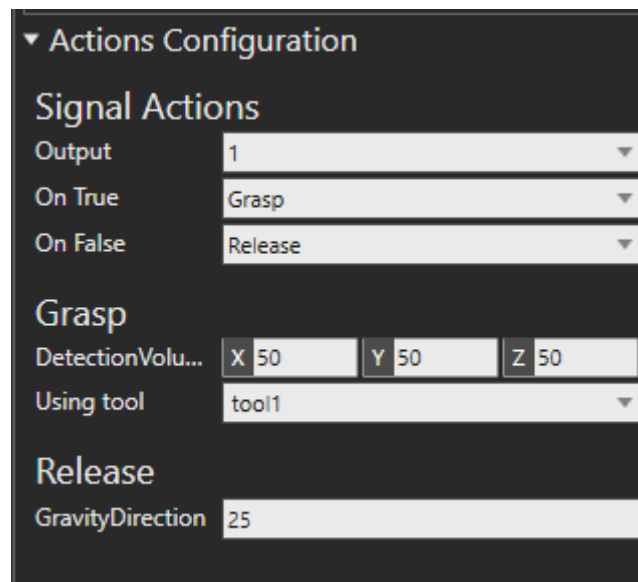


Output statement and action configuration

22. In Programming view/right panel/"pick" subprogram set binary output statement. In the panel on the right, change the OutputPort to 1 (=initialized as grasp with tool 1) and tick the OutputValue (=true).



23. Drag the statement between the two linear statements.
24. Click the robot and from the panel on the right go to Component properties and open the action configures from the bottom of the panel. You should have the view presented below. More about actions in next assignment.
25. If you change the output number from drop-down menu, you can see the preset functionalities of different outputs



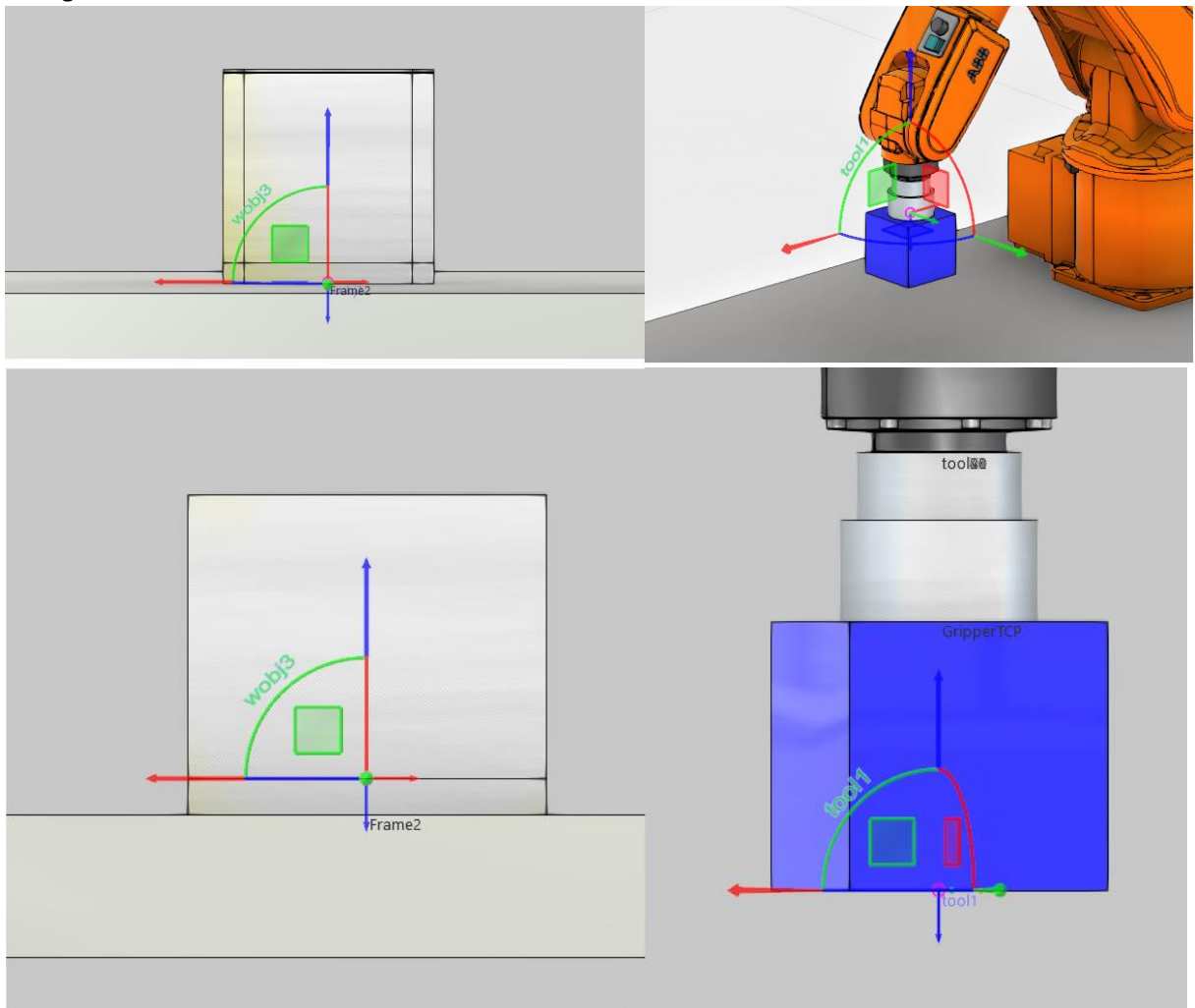
Release the base

26. When you run the simulation, there occurs a problem with reachability

- The motion is done with respect to tool and base frame. How much the difference between tool and base frame changes over time?
 - When determining the frames of the robot, what happens if the frame is connected to object that is not static?
27. Double click the PTP motion that is in approach point. On the right panel's bottom, select Jog and change the base to null. Add linear motion after the output statement. Delete the other linear motion that is after the output. Run the simulation.
 28. The robot is still behaving badly and the reason is base 2. Before the output statement, add a base statement to wobj2 that sets the base to origin position(X,Y,Z,Rx,Ry,Rz)=(0) and change node to "null".
 29. Run the simulation. The robot should now lift the cube

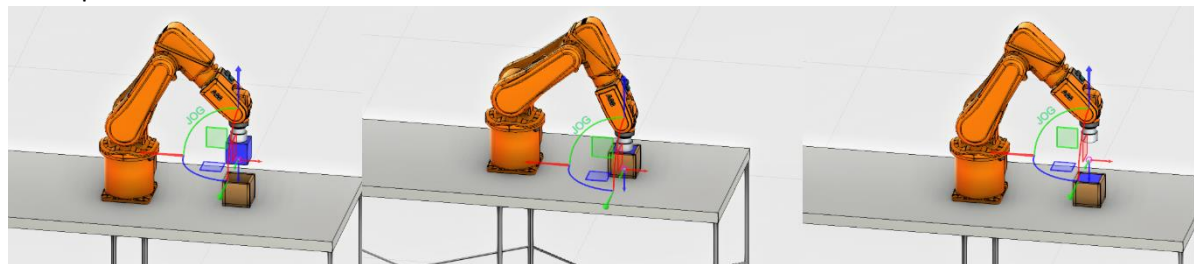
Place the cube

30. Next we want to place the cube inside the box. For this, we need to call "place" routine in main program.
31. The tool1 is located on the wrong side of the cube and base 1 is located on the bottom of the box. Set a tool and a base statement in place sub-program so that cube can be snapped properly in to the box (the cube must not go through the box/violate the boxes interiors). Use relative changes with base and tool statements.



32. In program view/top panel/manipulation/jog, Jog the cube with the snap tool inside the box. Check that you are using wobj3 and tool1. Set a linear motion inside the "place"- subprogram.

33. Lift the robot 100 units upward in world coordinates. Set linear and PTP- motion to “place”- subprogram.
34. Set output statement 1 as false to “place”-subprogram. Rearrange the statements to make the robot place the cube inside the box.



PTP

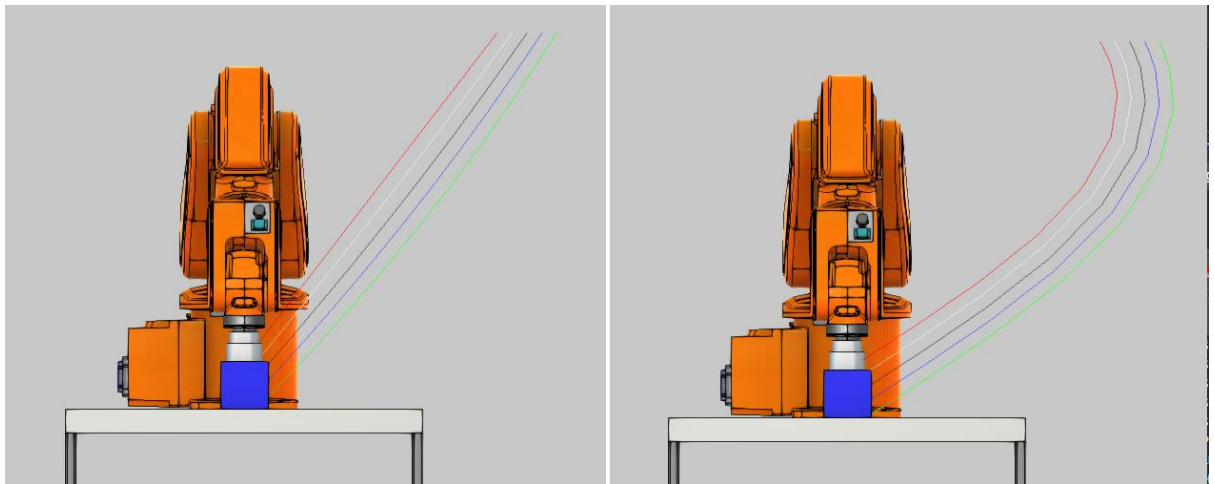
LINEAR

LINEAR

35. In main routine, double click the first motion. On the right panel, change the tab to jog. Check that tool and base are null and add a PTP- motion as final position. Run the simulation.

Analysis

- When picking up the block, we used approach point. Is it necessary? Consider the lines in figure below as force vectors



- Does it matter whether you use PTP or Linear motion in different situation? Analyze with concrete examples
- Why was the base frame rotated?

Basic 5: Tasks, signals, actions and while

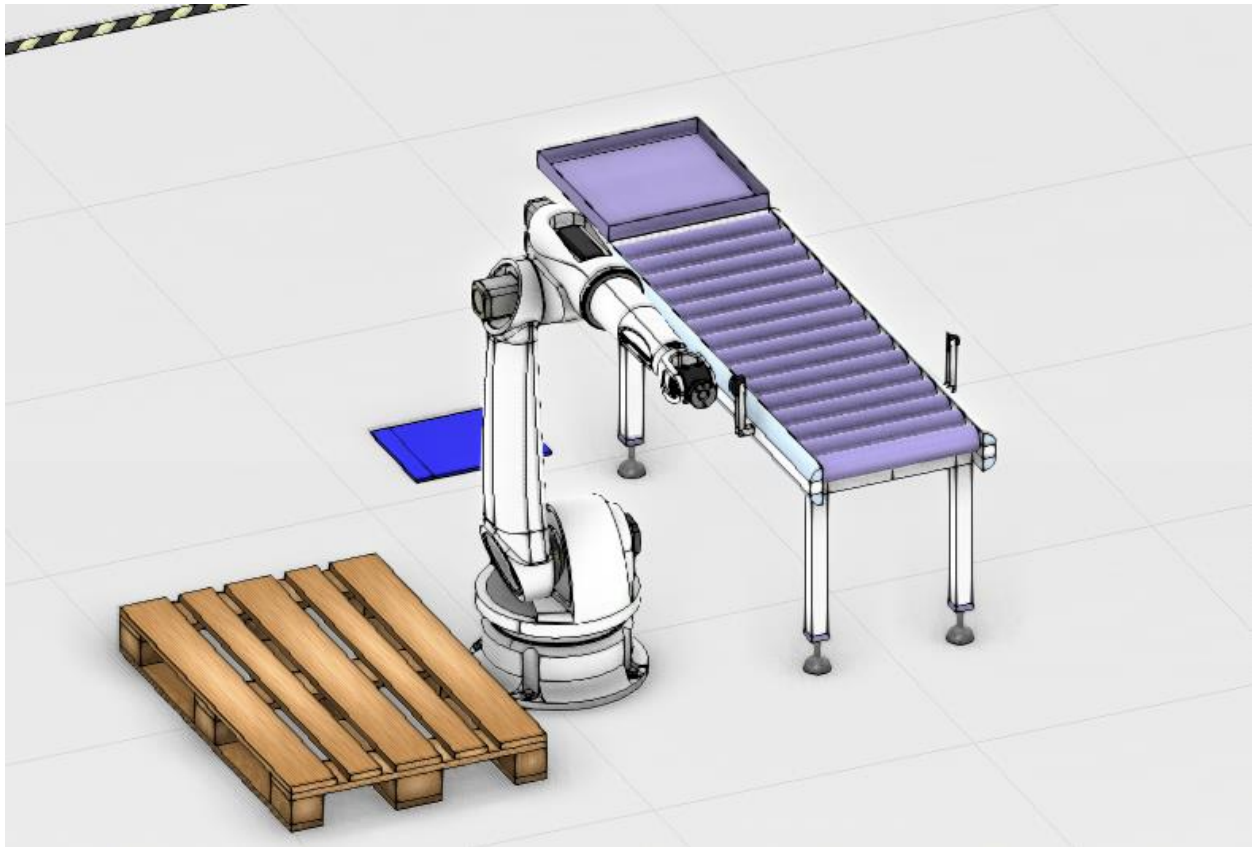
In this assignment, we are using tasks to create shapes and move them on conveyor belt. We also do some recaps. The objective of the assignment is to create two different size of planes in the beginning of the conveyor and pallet them to euro pallet. To make this possible, we introduce tasks, signals, while loop and grasp detection

Initialization

1. Add components needed

location in eCatalog	component	location in workspace (world coordinates X,Y,Z)	size
Home/eCatalog/Models By Type/Works/Visual components	Works process	PNP to conveyor start	initial
Home/eCatalog/Models By Type/Works/Visual components	Works task control	-2500,3700,0	initial
Home/eCatalog/conveyors/Visual components	conveyor	0,0,0,	initial
Home/eCatalog/Processors/Visual components	Conveyor sensor	PNP to conveyor move to x=1100	initial
Home/eCatalog/Robots/Visual components	Generic Articulated robot	1100,-900,0	initial
Home/eCatalog/Products and containers/Visual components	Euro Pallet	1100,-1600,0	initial
Home/eCatalog/Products and containers/Visual components	cube	works processors "resource location" frame [1]	(LWH)=420,320,10
Home/eCatalog/Products and containers/Visual components	cube	works processors resource frame	(LWH)=310,320,10
Home/eCatalog/Factory facilities/Visual components	Safety area	0,0,0	(LTW)=6000,60,8000

[1] From World view side buttons/frame types tick frames



2. From Home/left panel/Cell graph choose the second cube and name it in right panel as cube2
3. In modelling view connect the robots base 1 to node sensor::sensor and change the parent coordinates to $(X,Y,Z,Rx,Ry,Rz)=(0,0,0,0,180,0)$. Connect the robots base 2 to node Euro Pallet::Euro Pallet and set the parent coordinates. to $(X,Y,Z,Rx,Ry,Rz)=(0,0,144,0,180,0)$
4. In home view choose the conveyor and change in right panel/component properties the conveyor speed to be 100. Do the same for Works processor.
5. Select the sensor in home view. On the right panel, change the OnSensorAction to "StopPart" and ReactOn to "origin".

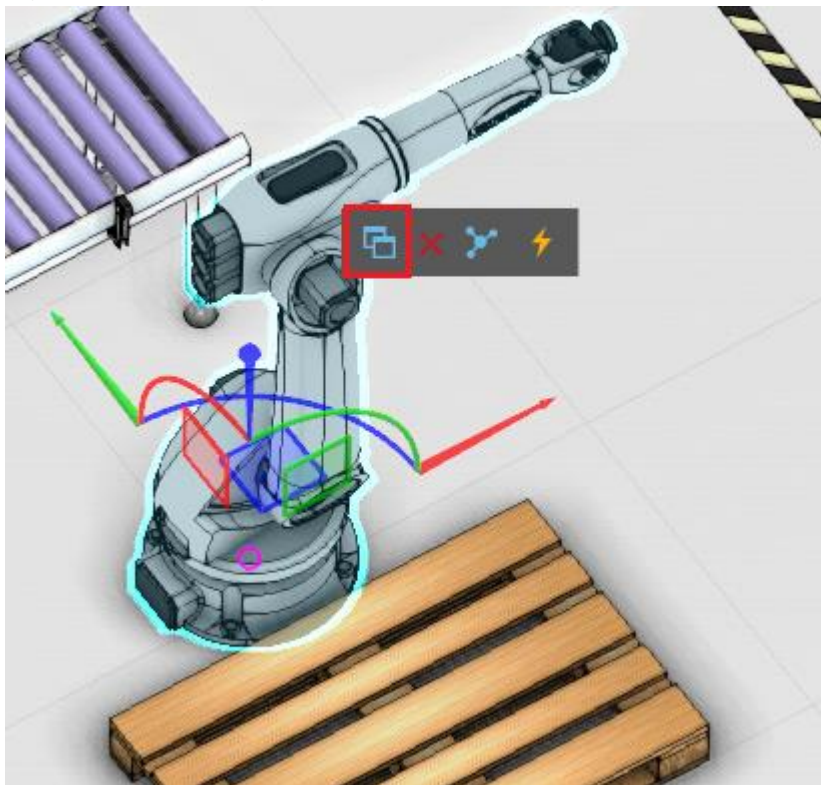
Tasks

6. Consider task as a shortcut in programming. In this case, we are about to create components and move them via conveyor.
7. First step is to select the Works process and type to panel on the right/ section ListOfProdId "Cube" and then select "create" from task drop-down list and click below it "CreateTask". There should read in InsertNewAfterLine "1:Create:Cube". Now run the simulation. You should see on the works process the cube you created.
8. Next we want to move the cube on the conveyor. Change the task to be "TransportOut" and click again CreateTask. Run the simulation. The cubes should move now on the conveyor.
9. Now select the Works Process and from panel on the right, choose Task- tab. Open the second from the top note – open in editor. It should open you the Task::Task window and you should see lines "Create: Cube: and TransportOut:Cube:False"

10. Reverse-engineer the task editor syntax and write a program that create and transport out cube and cube2 in relation 2:1, meaning that you should create 2 big cubes and one small cube in every sequence.
11. Run the simulation

Signals

12. With signals, you can connect robot to other robots and for example to conveyor. In home view's top panel/Connect choose Signals. Then select robot. Draw a line from Sensors "SensorBooleanSignal" to robot's input
13. The line goes to in[0] as initial but because some of the signals are dedicated to e.g. mounting and grasping, we need to change the port. Double click the number 0 and type 100. Run the simulation and see when input 100 is logical 1. Now the robot knows when the cube arrives to the end of the conveyor.
14. In home view, select the robot and clone the robot. Move the new robot to (X,Y,Z,Rx,Ry,Rz) = (-1500,3500,0,0,0,0).



15. In program view, select the cloned robot and in robot's main program, add Wait for binary input statement and change the port to 101 and InputValue as true. Then add a print statement and message "signal works".
16. Next click the signals and select either of the robots and draw a signal from original robot's out to clone's in and change original's output to port 100. Click the top panel/connect/signals again to deselect the signal function. Alternative way is to press "Esc".
17. In original robot's program add wait for binary input (100) and set output (101) statements and change and values to true. Now you should get a message to your output window, below the workspace.

Program Editor

Subprograms

Main pick place

While True

- Wait IN[100] == True
- Call pick
- Call place
- Set OUT[101] == True


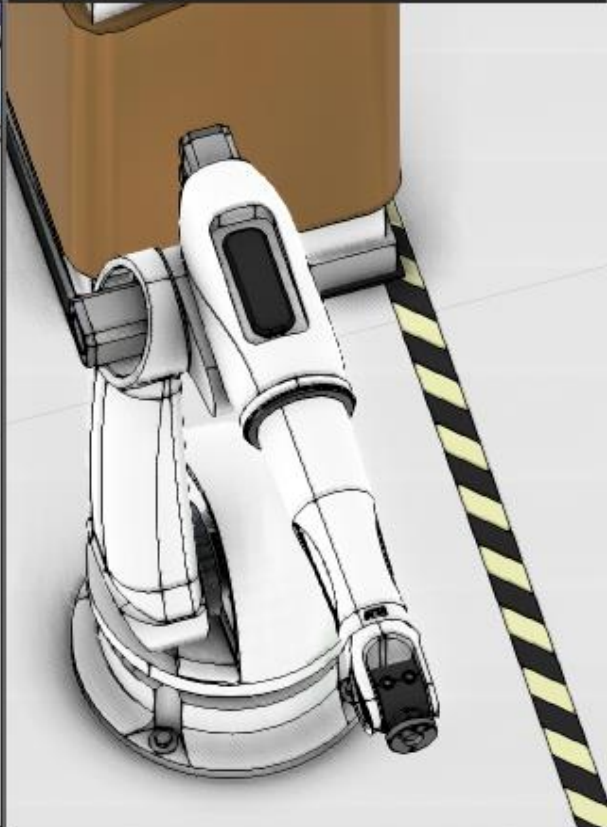
Program Editor

Subprograms

Main

Wait IN[101] == True

Print "signal works"

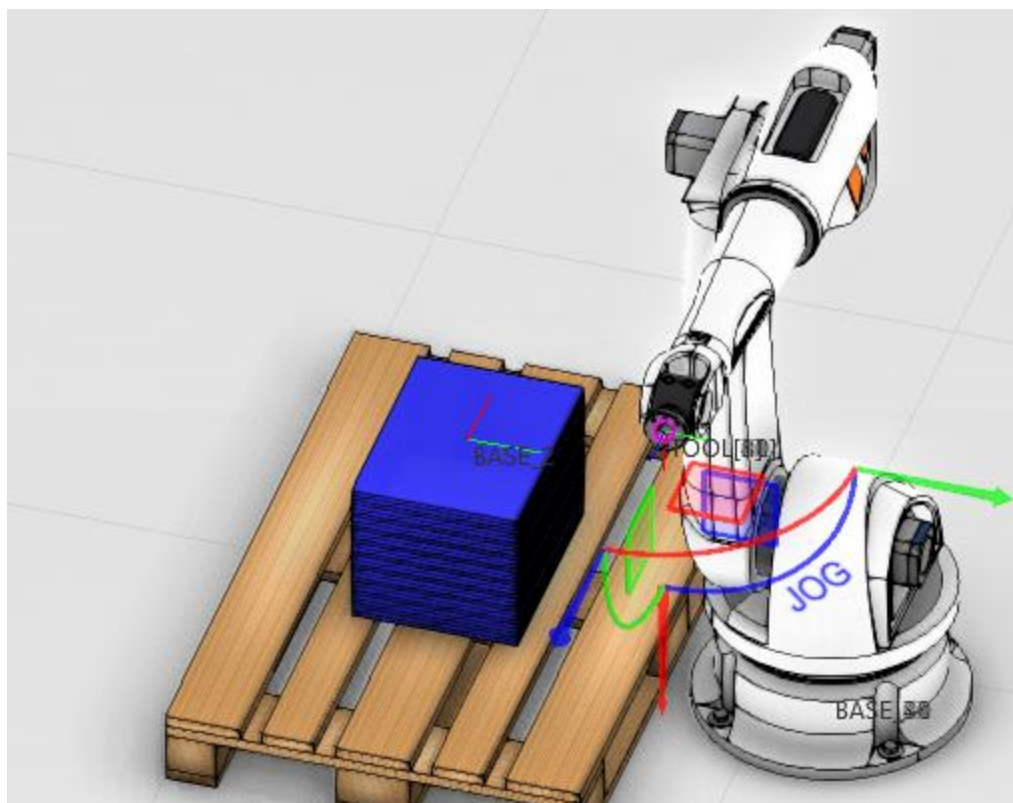
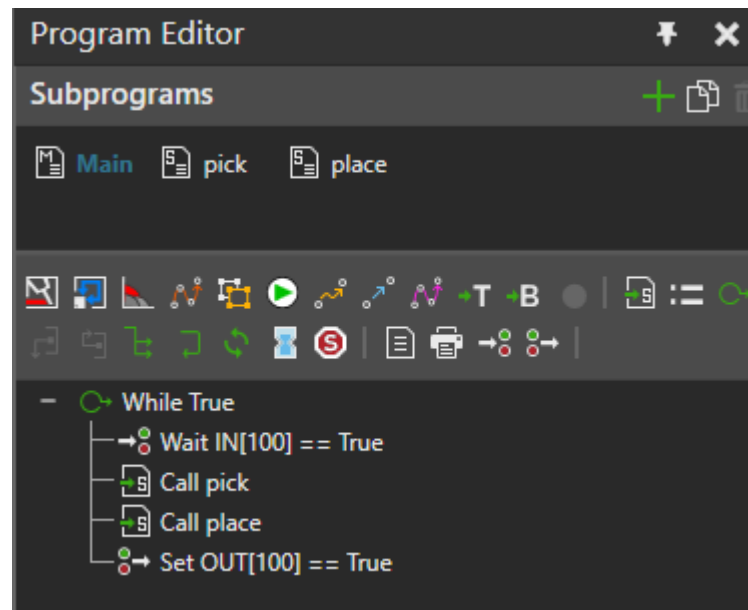
Output

GenericRobot #2: signal works

Grasp detection & while + robot program recap

18. This section is recap from basic 4. Your task is to move the cubes from conveyor to euro pallet. Use the base frames created in initialization for this purpose.
19. To make the robot continue after the first pick and place, use while statement in program editor and create your code inside of the loop.

20. To pallet the cubes, move the base 2 dynamically in the beginning of while statement. Use set base statement and in relative mode.



Analysis

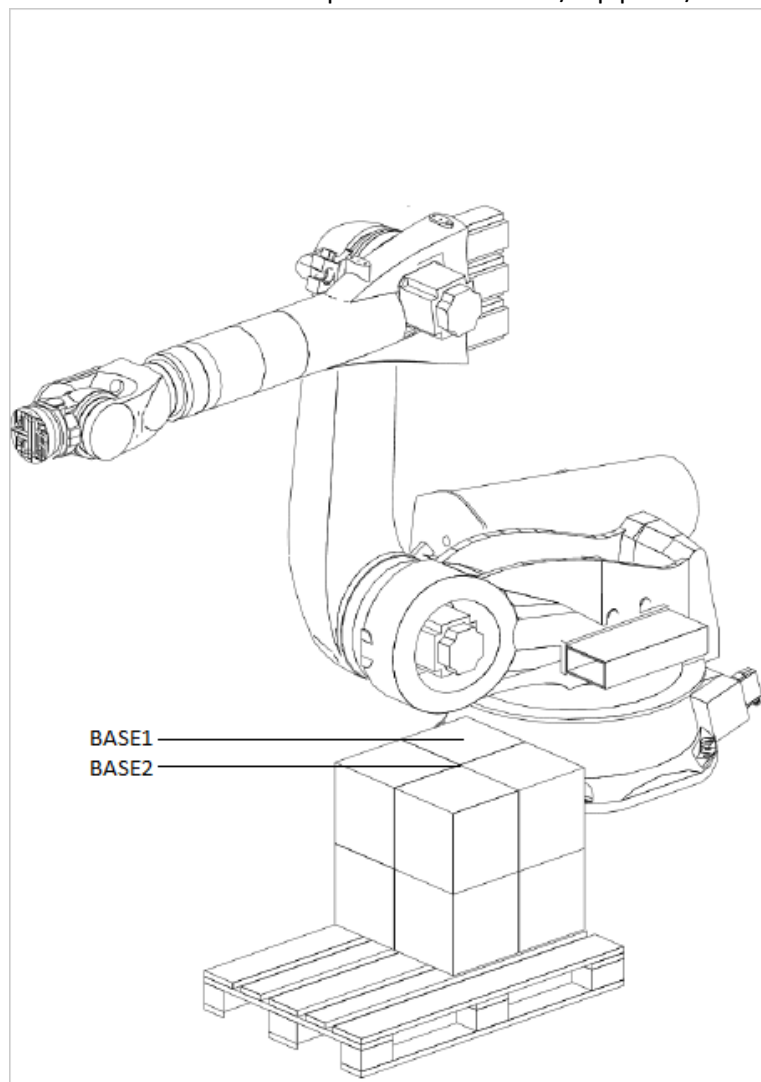
- How many times did the robot 2 say, "signal works"? If only once, why it did not say it in repeat?

Basic 6: Multi grasping

The purpose of this assignment is to show how to grasp multiple objects with a robot

Initialization

1. Load your solution of basic 2. Delete the clone robot, cube and pallet. Save your solution for example as "Solution6"
2. Move the original robot's base frame 1 on the top of the most near (top one) block of the robot and base frame 2 in the middle of four top blocks (figure below). Turn the bases z- axis to point downwards.
3. Detach the lower corner cube from euro pallet in home view/top panel/Hierarchy



Multi grasping and volume detection

4. In home view, select the robot. From panel on the right, go to action configures and make next changes to volume detection

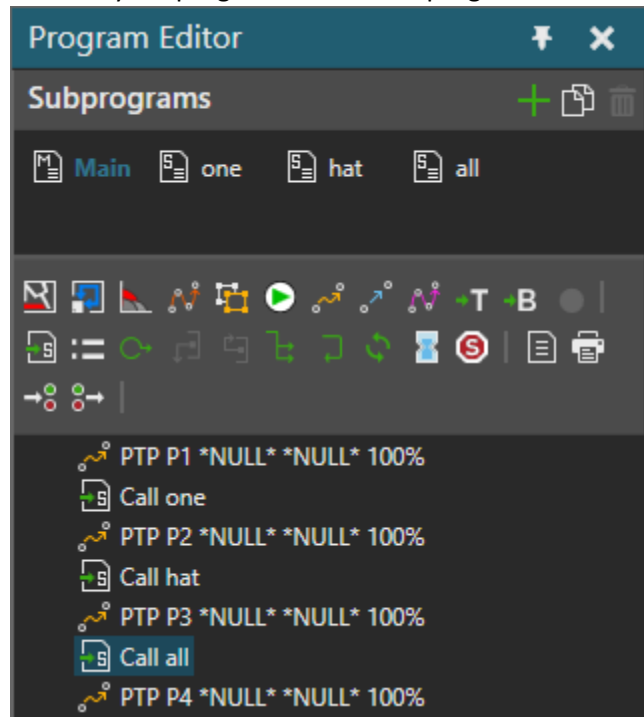
	X	Y	Z
TOOL1	1	1	1
TOOL2	1	1	600

5. Select the robot and from panel on the right, choose SignalActions tab. Tick the MultiGrasp

Program the robot (recap)

6. Now in program view/left panel - create three subprograms called “one”, “hat” and “all”.

7. Create an initial position with PTP-motion statement for the robot. Use the same PTP-statement as first, last and between every subprogram call in main program.



8. In every subprogram, you must do the same sequence:

- Approach above the base frame
- Snap on the frame
- Grasp the cube
- Lift the cube
- Lower the cube
- Release the cube
- Return to initial point via approach point

You can copy the “ hat” sub-program’s statements to “all” sub-program with minimal changes. To be able to paste the statements, there must be content before paste. Use delay statement (initialized to zero), to not affect the program. Later you can delete the delay statement.

The used tools and bases are presented below.

Subprogram	one	hat	all
used tool	1	1	2
used base	1	2	2

Analysis

- Why the detection volume has X,Y and Z components ?

Basic 7: Recap

In this assignment, we are recapping some of the learned things from previous assignments. In addition, we are adding if-else statements and PickAndPlace Pallet Inlet. The idea of this assignment is to pick boxes from one conveyor and move the boxes with robot to two different conveyors alternatively. We want to move incoming boxes to outgoing conveyor with ration 1:2.

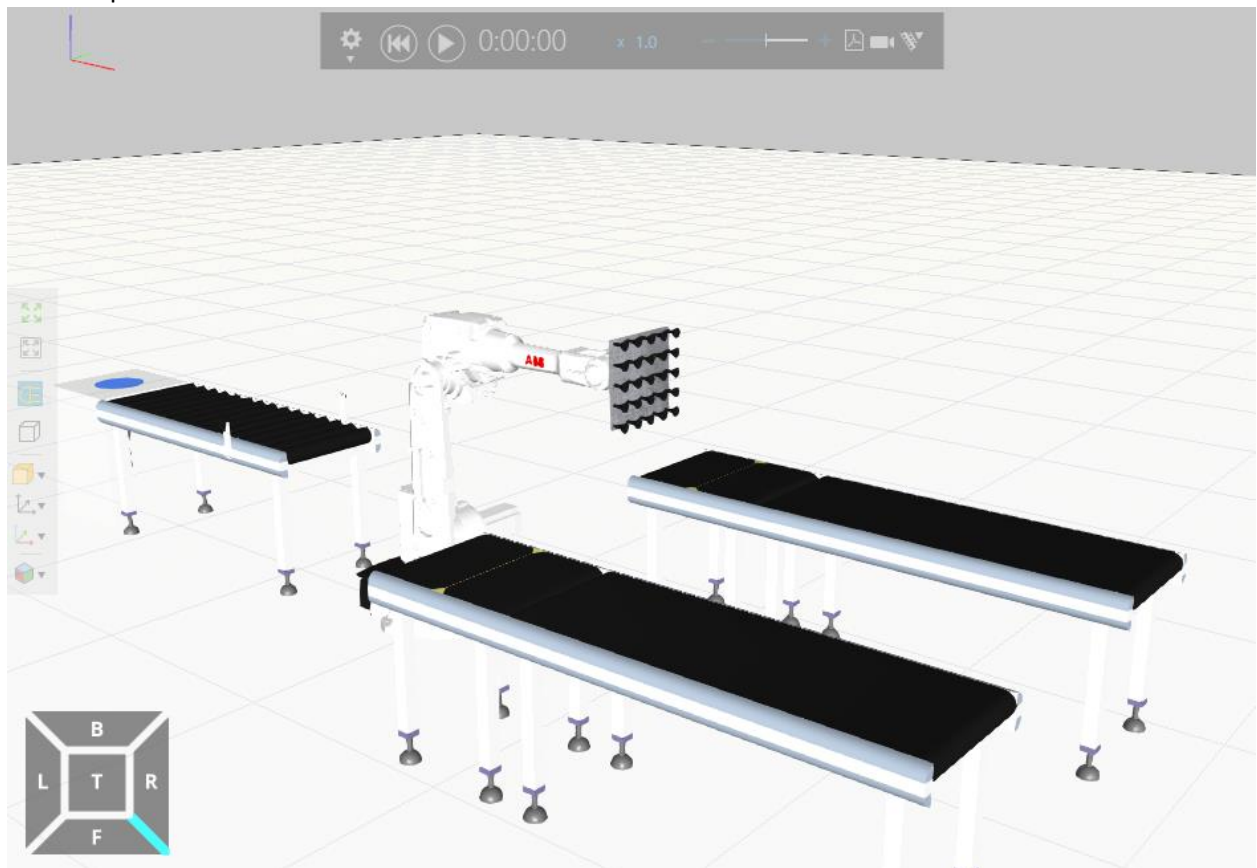
Initialization

1. Add components needed

location in eCatalog	component	location in workspace (world coordinates X,Y,Z,Rx,Ry,Rz)	setup
Home/eCatalog/Models By Type/Robots/ABB	IRB 2600-20/1.65	0,0,0,0,0,0	initial
Home/eCatalog/Models By Type/Conveyors / Visual components	Conveyor	-2200,0,0,0,0,0	Initial
Home/eCatalog/Models By Type/Conveyors / Visual components	Conveyor	1200,700,0,0,0,0	initial
Home/eCatalog/Models By Type/ Conveyors/Visual components/	Conveyor	1200,-700,0,0,0,0	initial
Home/eCatalog/Models By Type/Pick and Place/Visual Components	PickAndPlace Pallet Inlet Conveyor	PNP to one of the above conveyors (600, -700, 0) [2]	initial
Home/eCatalog/Models By Type/Pick and Place/Visual Components	PickAndPlace Pallet Inlet Conveyor	PNP to one of the above conveyors (600, 700, 0)	initial
Home/eCatalog/Processors/Visual components	Conveyor sensor	PNP to single conveyor (-1030, 0, 700)	ReactOn to "Origin" and OnSensorAction to "StopPart" (Component Properties)
Home/eCatalog/Feeders/Visual components	Shape Feeder	PNP to the single conveyor (-2200, 0, 0)	initial
Home/eCatalog/Tools/Visual components	Parametric Suction Cup Gripper	PNP to tip of the robot arm	SuctionCups_X to 5 and SuctionCups_Y to 5 (Component Properties)

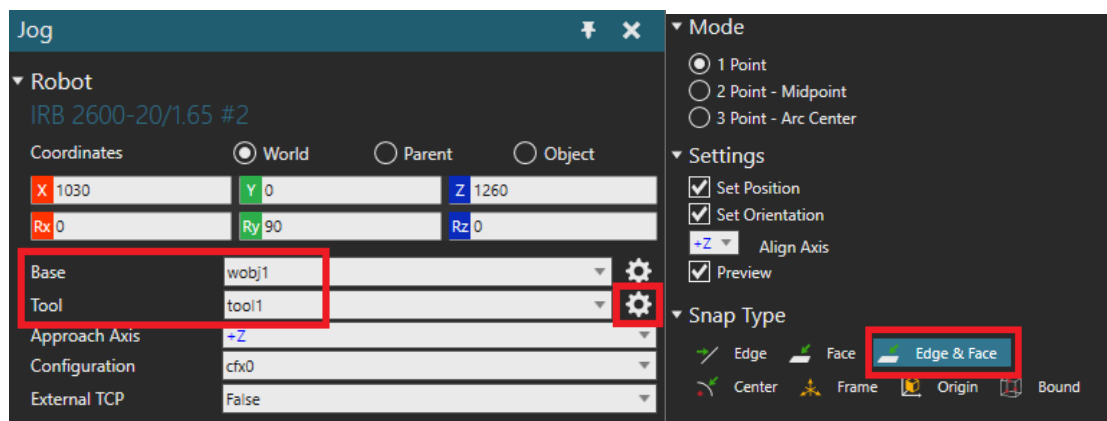
[2] The conveyors have "start and "end" If the PickAndPlace Pallet Inlet Conveyor won't PNP properly, turn the conveyers Rz = 180 and try again. If this worked, move the conveyers to locations, presented above.

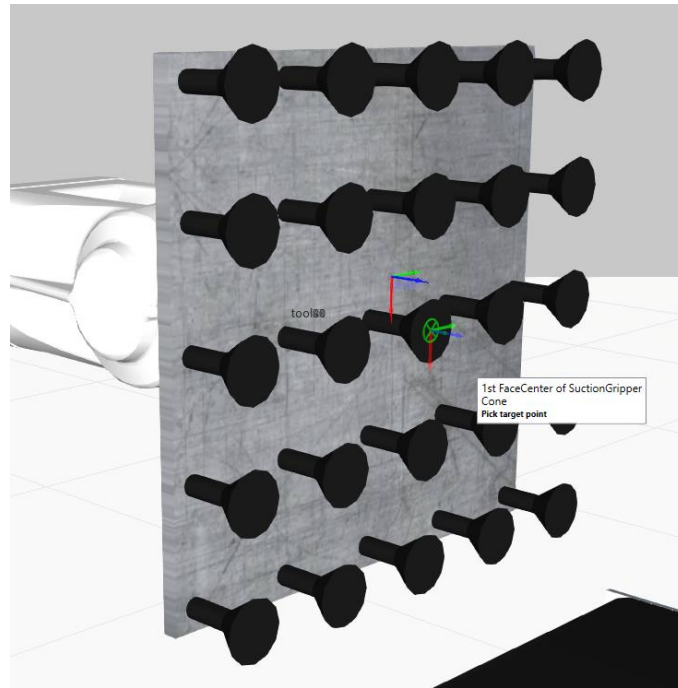
Complete initialization should look like this:



Tool frame

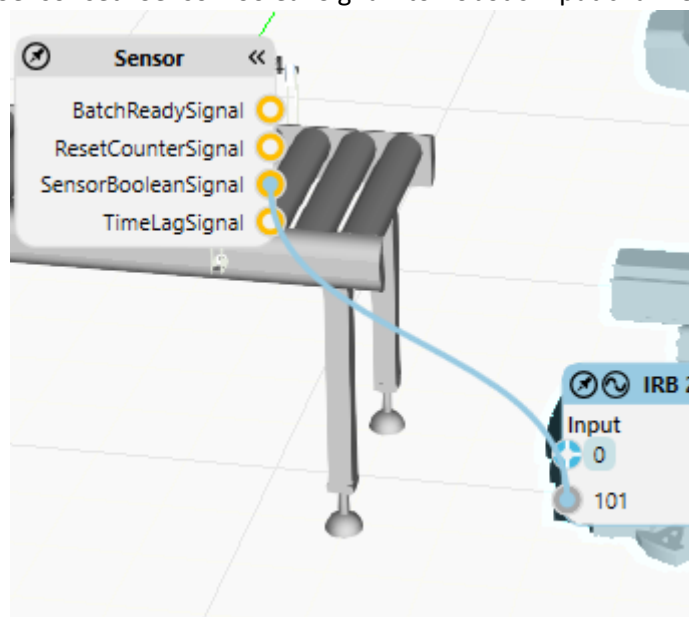
- Next set the tool frame. Go to Program view and select the robot. Select Base to wobj1 and Tool to tool1 on the right panel/robot's Jog. The tool and base frames can be moved also in programming view by selecting the gear icon in the right panel. Then select Snap and Snap type to "Edge & Face". Then click middle one of the tool cups (**check the pictures below**). Now we got tool frame set up correctly.



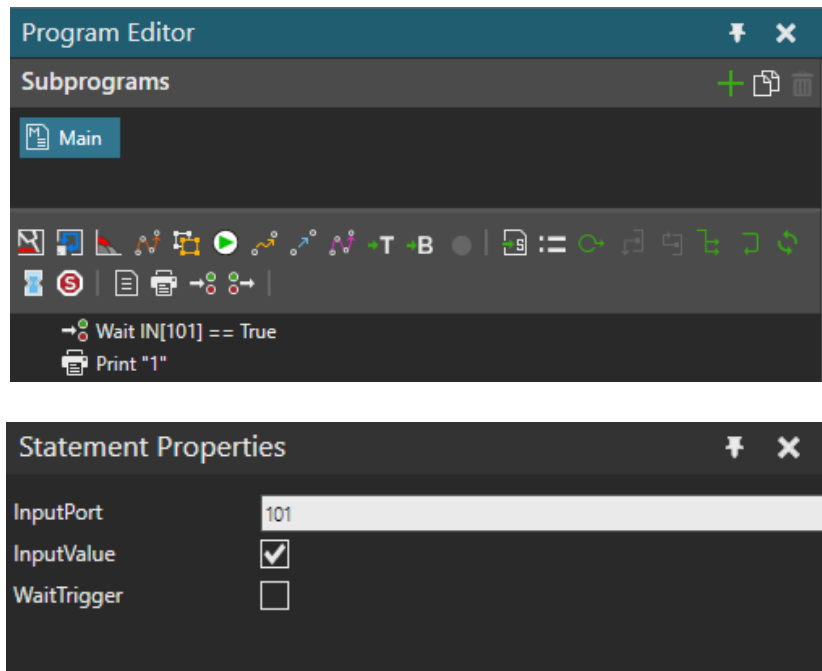


Signal setup

- Now, we need to set signal from the sensor to the robot. Basically, this is same as in the previous assignment. From Sensor set "SensorBooleanSignal" to Robot's Input channel 101.



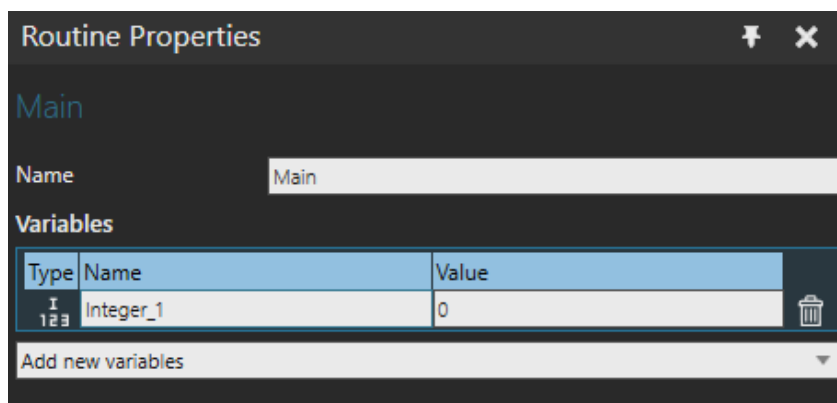
- In Program Editor, we will add statement "Wait for Binary Input Statement". Make sure to select Input port to 101 and Input value box ticked. In addition, add print command after the binary input statement and write "1" to the message box.



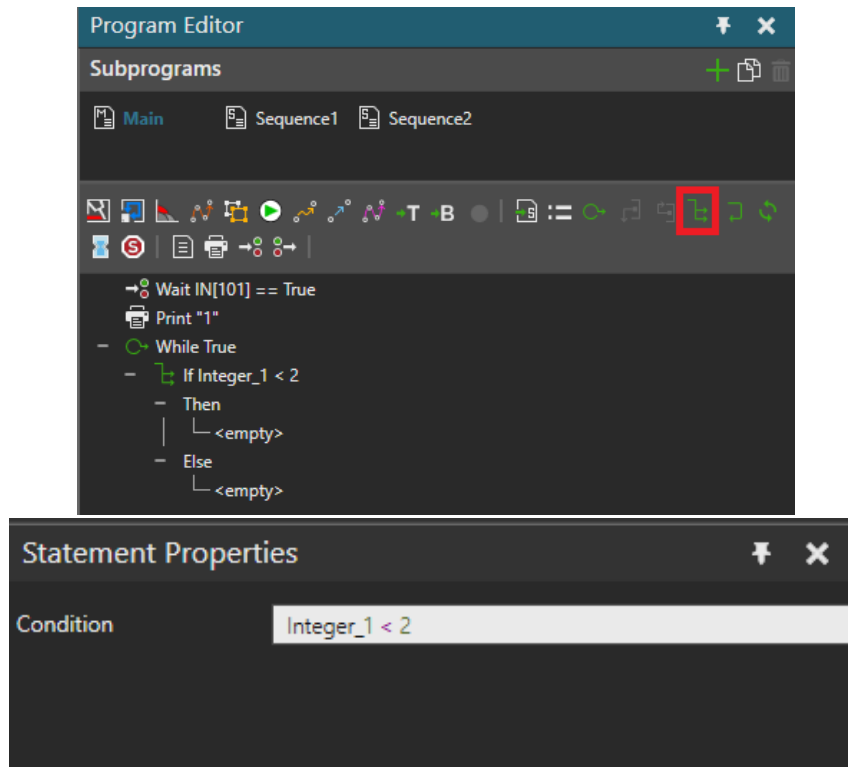
Now, if you run the simulation, robot should print “1” to the output terminal.

Program editor initialization

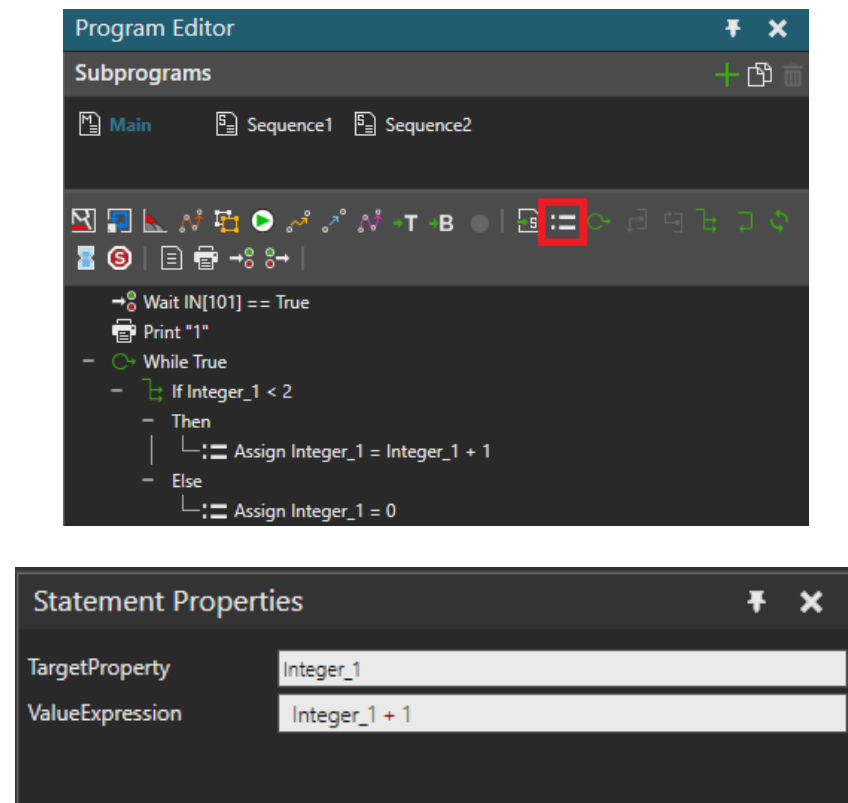
5. Create while loop to the Program Editor. Set it after the Print statement.
6. In addition, press “Main” from the Subprograms. From Routine Properties on the right panel add new Integer variable. Because we want to move boxes between the two conveyers, we need a variable to keep track on which conveyer the box we move the box next.

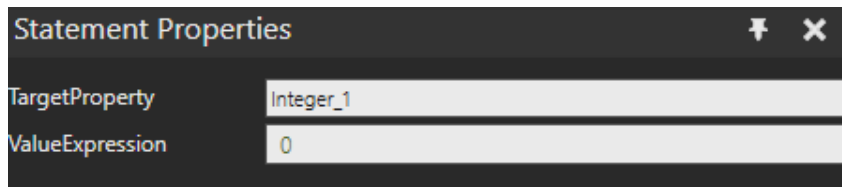


7. Add if-statement inside the while loop. And for the if-statement we use statement “If Integer_1 < 2”.

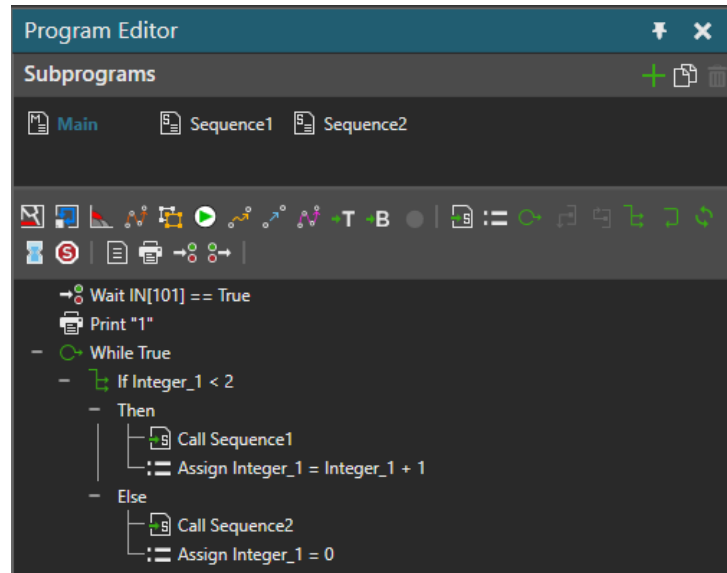


8. Add "Assign Variable Statements" after Then and Else statements. Set target property to Integer_1 and to ValueExpression set "Integer_1 + 1" and to other one set "0" in Statement Properties.





9. Create two Subprograms and add them before the variable assign statements. Use “Call Sequence Statement to do this.



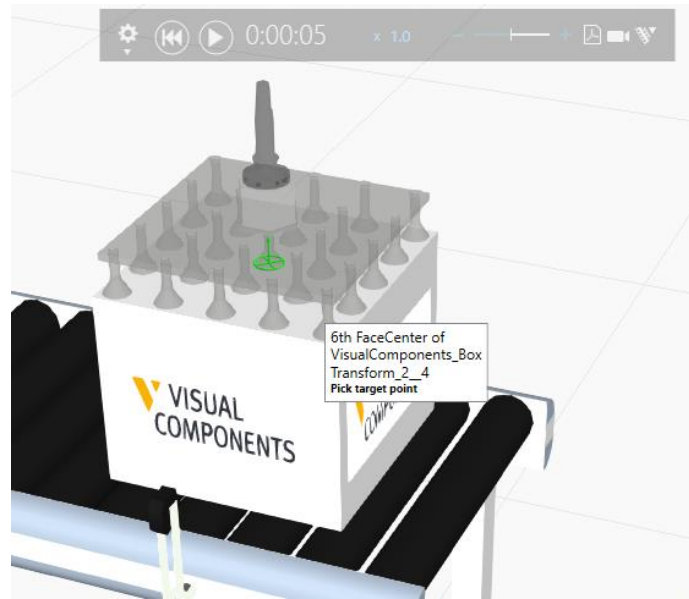
Now the if statement goes twice to “Then” and once to “Else”. After “Else”, the counter that we created will reset to 0.

Motion statements

10. Now, your task is to build motion statements to the Sequence1 and Sequence2. Basically, use statements to move boxes alternatively to the two conveyors in sequence 1:2, meaning that you bring two boxes to another conveyor and one to other. You can use PTP or Linear Motion statements, it is up to you. **See the tips below.**

Tips for the exercise:

- Play the simulation until one box stops to the sensor. You will be able to snap the tool to the box and then use Motion statements. You may also use Halt-statement to stop the simulation when needed.



- Also, make sure that you use “Set Binary Output Statement” to grasp the box and to release the box.
- Recap the usage of snap and align.
- Use the robot’s base frames if needed.

Analysis

- What will happen if we change the feeders CreationInterval?

Basic 8: Welding with Delfoi Add-on

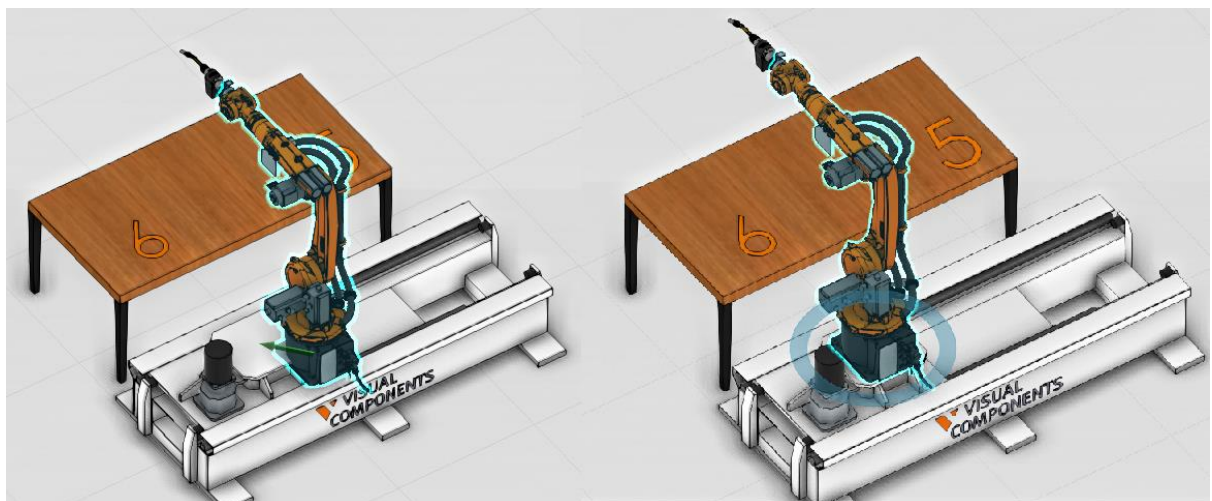
This assignment's goal is to get familiar with Delfoi Robotics Visual Components Add-on. We want to weld 2 numbers to a table. The first number is "6" and second is free of choice. In addition, we will add a Robot Floor Track to move the robot linearly from welding position to another.

Initialization

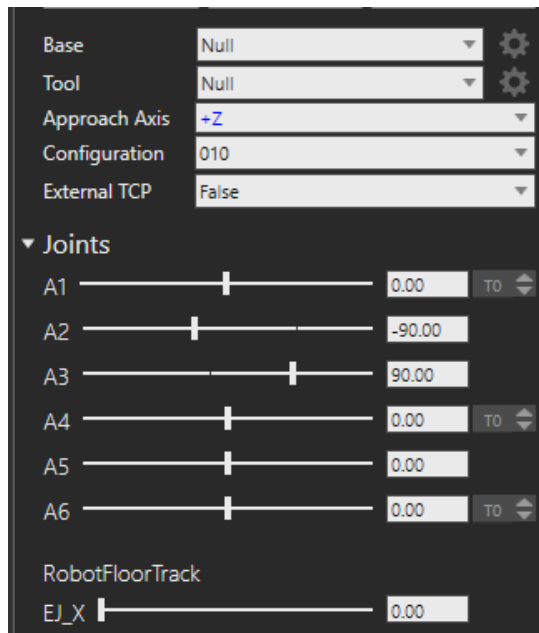
1. Add components needed:

location in eCatalog	component	location in workspace (world coordinates X,Y,Z,Rx,Ry,Rz)	Size/material
Home/eCatalog/Models By Type/Robots/KUKA	KR 22 R1610	0,0,0,0,0,0	initial
Home/eCatalog/Models By Type/Robot Positioners/Visual Components	Robot Floor Track	0,0,0,0,0,0	StrokeX = 1500
Home/eCatalog/Models By Type/Interior Facilities/Visual Components	Table A	1100,1200,0,0,0,0	(D, H, L)= (1000,750, 2000) Material = steel
Home/eCatalog/Models By Type/ Characters /Visual Components	number 6	500,850,751,0,0,0	350
Home/eCatalog/Models By Type/ Characters /Visual Components	number	On the table	Free of choice
Home/eCatalog/Models By Type/ Tools/Visual Components	Welding Torch	PNP to the tip of the robot	initial

2. Now move the robot to the Robot floor track by dragging it. Use the PNP to connect the robot on the track.

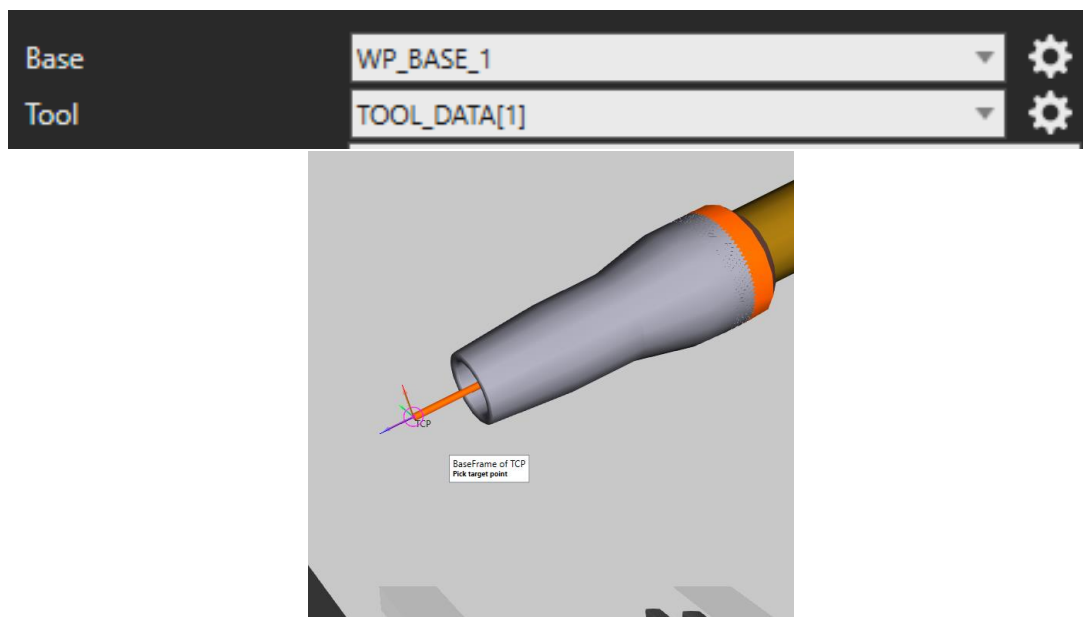


3. Create an initialization motion with following configuration



Frame initialization

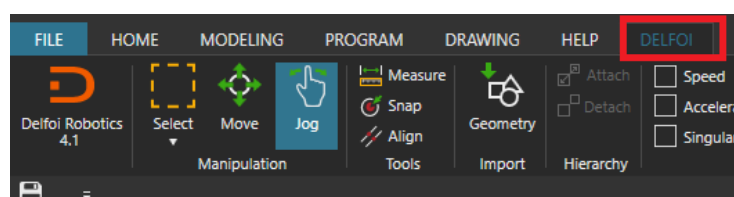
- Set the tool frame to the tip of the welding torch. Recap from previous exercise!



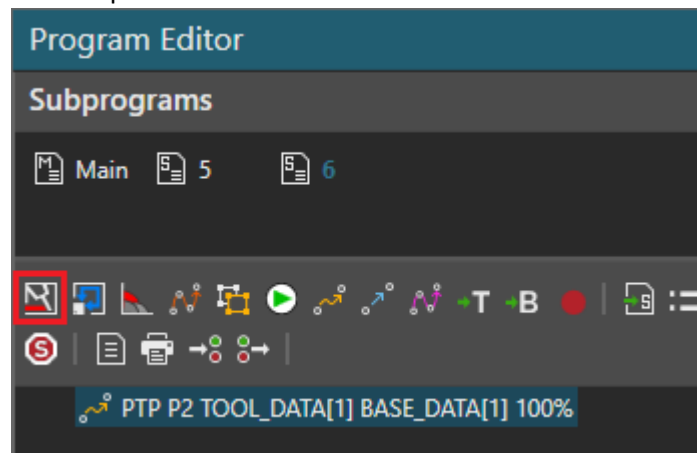
- Attach the base frames 1 and 2 to numbers and set the parent coordinates to (X, Y, Z, Rx, Ry, Rz) = (0, 150, 0, 0 180, 0)

Welding with Delfoi Add-on

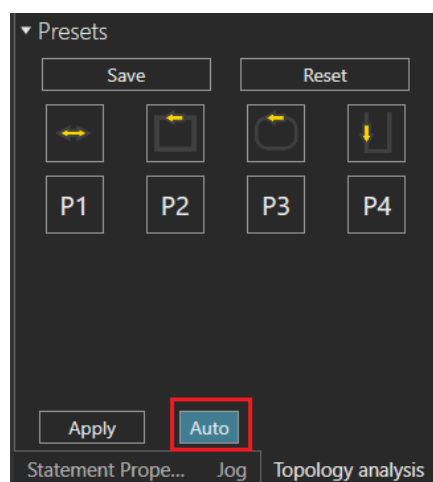
- Our next step is to use Delfoi's add-on to make Path for welding the numbers. Select Delfoi from Top panel. The panel on the right is similar to program view.



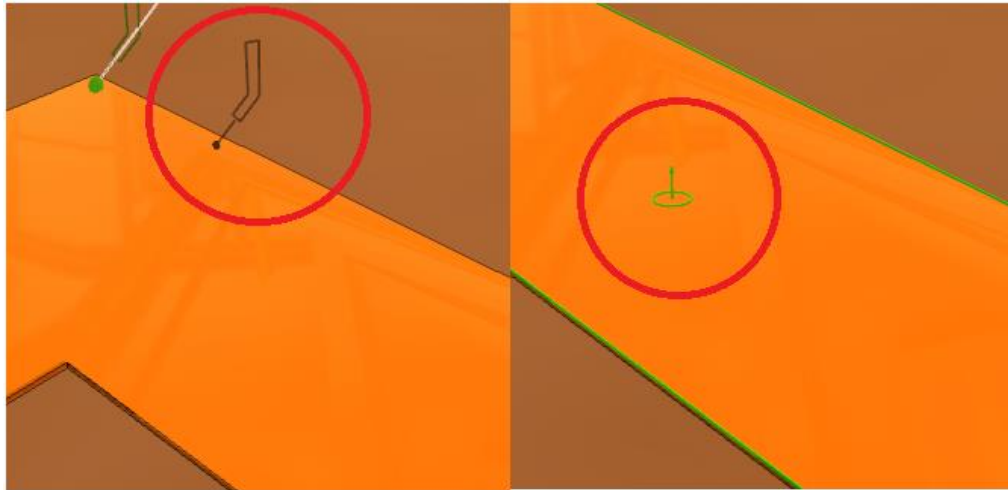
7. Create two subprograms, one for each number welding.
8. Go to first subprogram and create an approach point above the number “6” with PTP motion with respect to Tool1 and Base1.
9. From the Program Editor press “Path”.



Delfoi’s Topology analysis settings and Path setup setting will pop up to the panel on the right . Add-on offers many possible setups for choosing how the robot will weld. For this exercise, we only use the “Auto” setup. Make sure that you have “Auto” selected in the bottom of the Topology analysis panel.



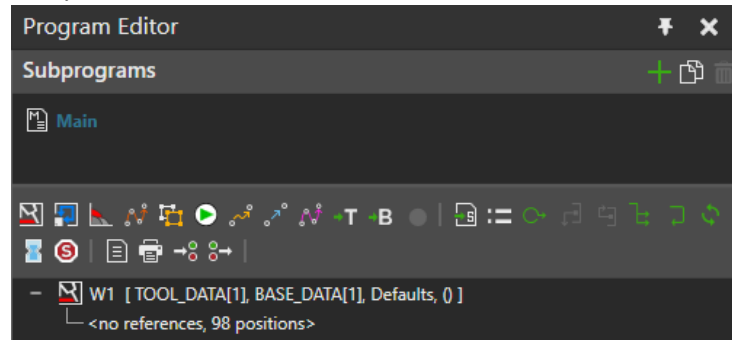
10. Navigate in the workspace **near** the outer line of number 6. In order to set the path statement, we need to press any outer line of the number. (See picture below). If you can’t see the reflection of the torch tip, you are too far away.



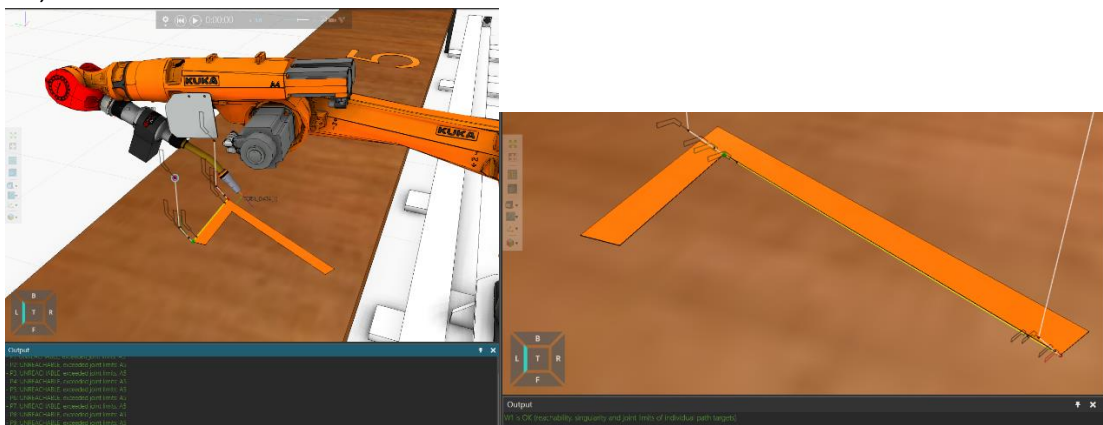
GOOD

BAD

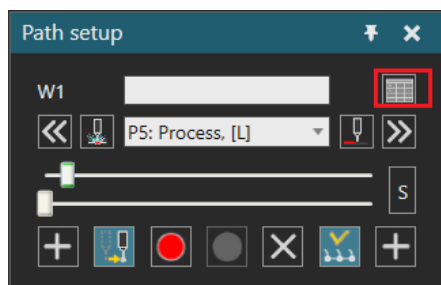
11. In the Program Editor you should see the statement that was created.



12. If the statement violates limits, it will be shown at output as is well behaving path statement (example on right figure below). The Delfoi Add- on suggests two orientation per line. In the case of bad behaving statement, try to find the other orientation and try it out. If both orientations have problems, you have to either move the robot on the track or change the parameters of table, number or floor track.

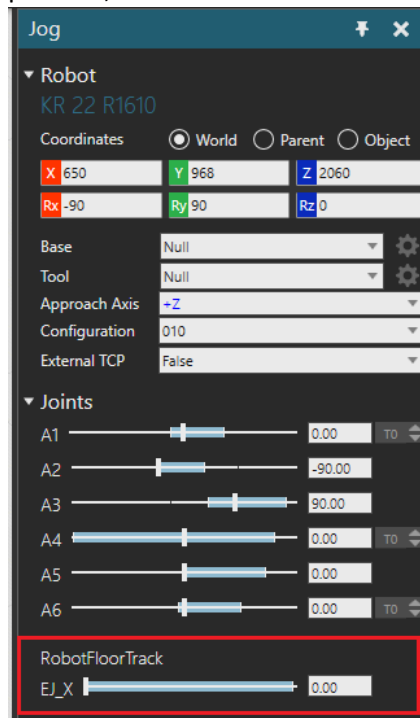


13. Select the path statement you managed to create without violations. On the right panel, you should have Path setup. On the top-right corner of the path setup is a table, where from you can determine the velocity of the welding. If you select multiple points and change one's velocity, it changes the velocity of all selected.

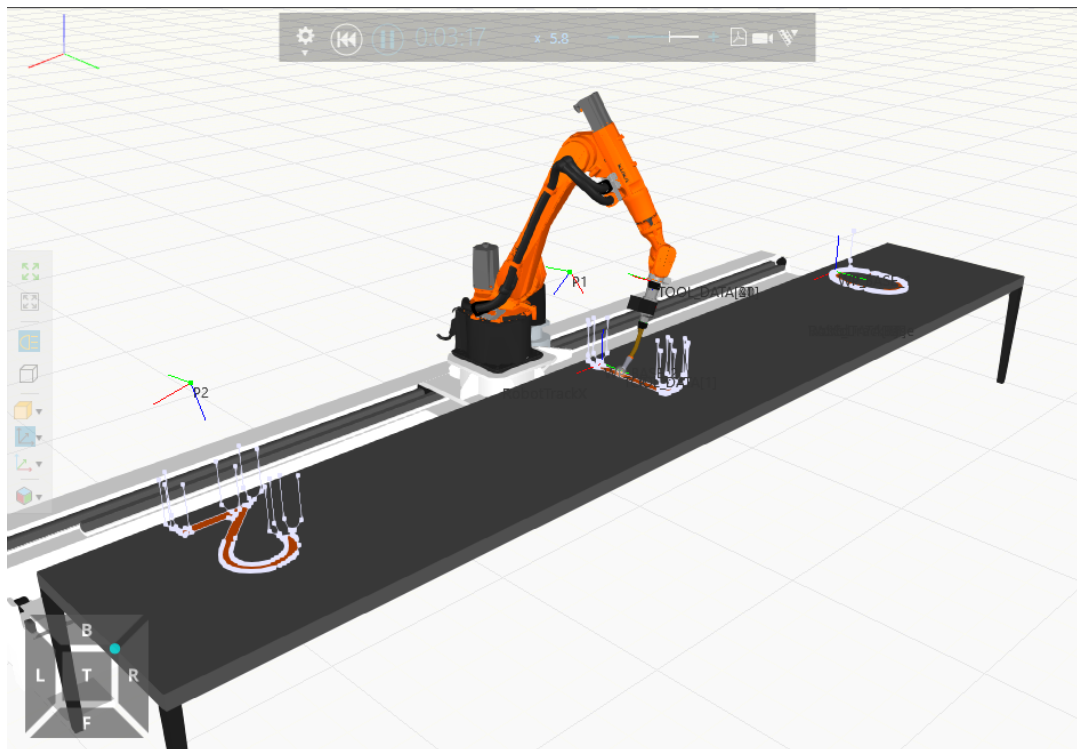


Point	Kind	Motion	Config	Search	Turn J4	Turn J6	Aux 1	Speed	WPS
P1	Via	Joint	010		-1	0	0	100	-
P2	Approach	Linear	010		-1	0	0	1000	-
P3	Near	Linear	010		-1	0	0	400	-
P4	Start	Linear	010		-1	0	0	100	-
P5	Process	Linear	010		-1	0	0	50	-
P6	Process	Linear	010		-1	0	0	10	-
P7	Process	Circ 1 path	010		-1	0	0	10	-
P8	Process	Circ 2 path	010		-1	0	0	10	-
P9	Process	Circ 1 path	010		-1	0	0	10	-
P10	Process	Circ 2 path	010		-1	0	0	10	-
P11	Process	Circ 1 path	010		-1	0	0	10	-
P12	Process	Circ 2 path	010		-1	0	0	10	-

14. When the first number is welded, we need to move the robot to next one. Double click the initialization statement and add a PTP- motion after the first subprogram. From panel on the right, go to tab jog and move the robot with slider of RobotFloorTrack. When moved to good position, add PTP- motion



15. Jog the robot with the help of RobotFloorTrack to the next position and make a linear motion statement.
16. Weld the next number fully. Below is presented an example of three number welding task (you need to do only two!)



Analysis

- Name at least three ways to solve the problem of exceeded joint limits and unreachable positions

Basic 9: Singularity and collision detection

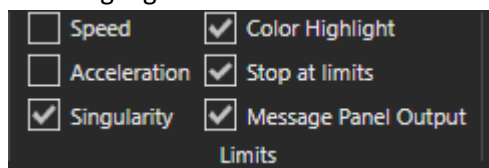
This basic assignment is really short. It is only an example of how detectors work

Initialization

1. take a generic robot $(X,Y,Z)=(0,0,0)$ and car tyre $(X,Y,Z)=(750,0,0)$
2. in modelling tab connect base1 to car tyre and move it to parent coordinates $(X,Y,Z,Rx,Ry,Rz)=(0,0,135,180,0,90)$
3. create initial and final state with PTP-motion in program view/ left panel/main
4. Jog the tool 1 to base 1 and make a PTP- motion -> your robot should bow

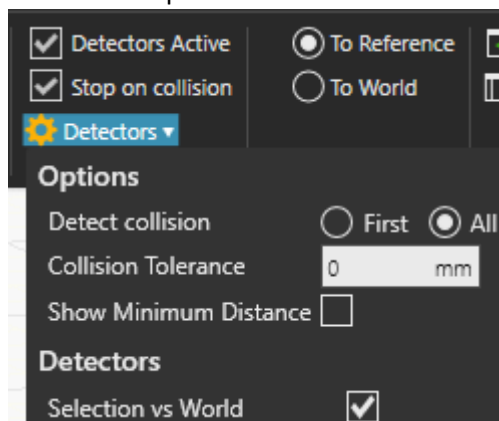
Singularity

5. In home view on top panel select from limits Singularity, Stop at limits Message Panel output, Color highlight

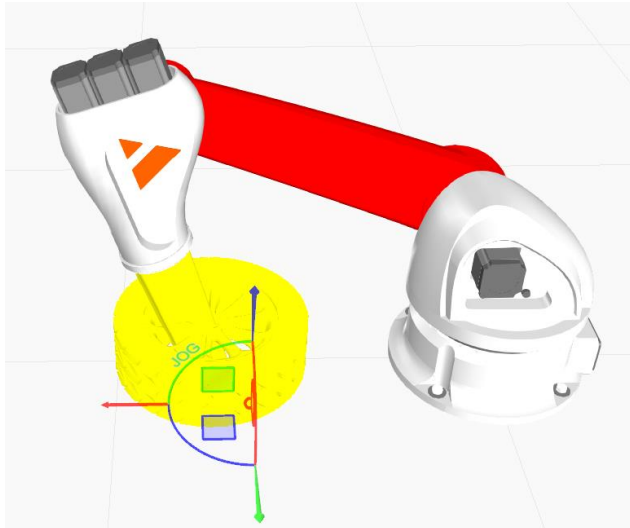


Collision detector

6. In program view **select the robot** and then from top panel/Collision detection tick both options and from drop-down menu select detectors/Selection vs. world.

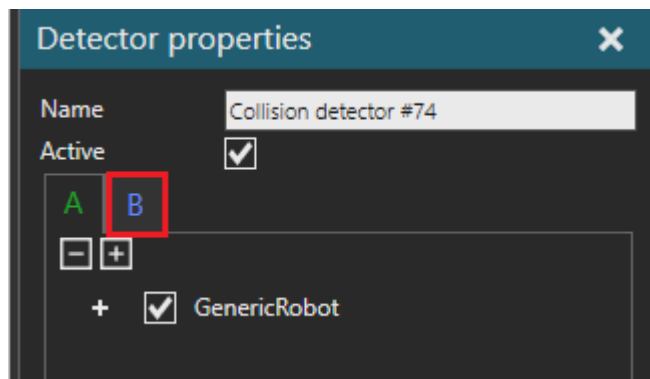


7. Change the base of second PTP- statement to base 2. Robot should hit to the tyre.
8. Reset the simulation and untick the "Stop on collision". Run the simulation. The robot should break the Joint limit in addition to collision. Check from right panel/jog the joint values



Alternative collision detector

9. In bigger projects, you might want to have more detailed collision detections. Go to program/collision detection/Detectors/detectors/create detector
10. In right panel, you have detector properties. Select the robot and in right panel, select “add selection”. Then on top of the right panel, select the blue capital B, select the tire and select “add selection”.



11. Check that you have selected “detectors activated” and from drop-down menu the detector you just created. Run the simulation

Analysis

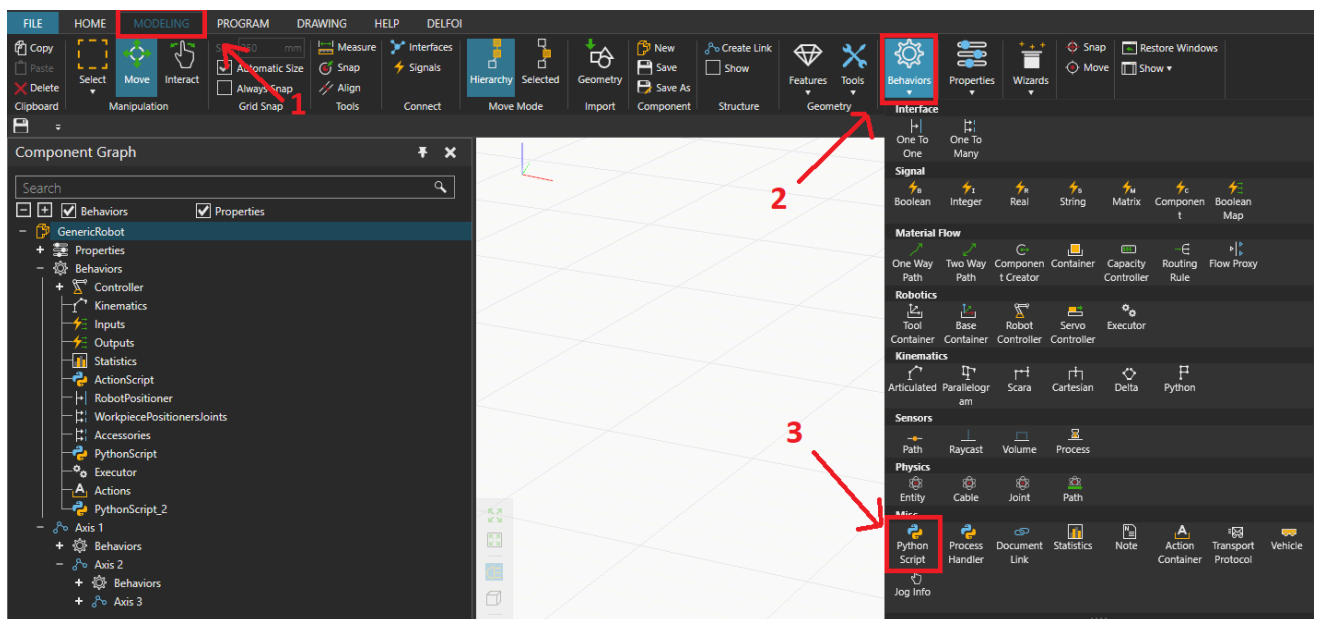
- What happens if both, the robot and the tyre are both in A list? (in perspective of collision detection)
- In what cases you would like to have objects on the same list.

Basic 10: Robot programming with Python

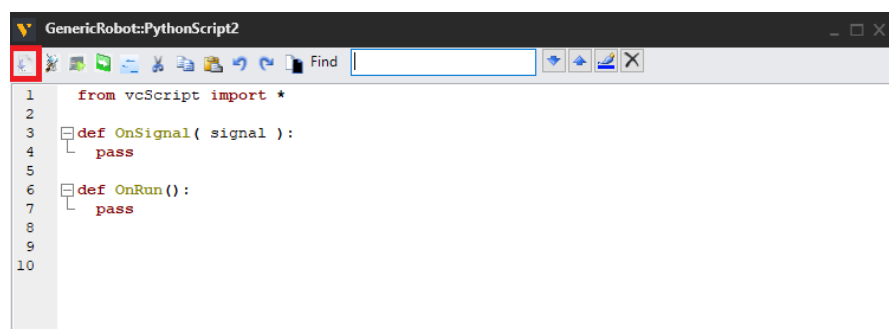
The goal of this assignment is to learn programming robot with Python. We are basically creating a Python script, which is controlling the Program Editor. Your task is to compile the code given in the Python example (In appendices) and understand how to program visual components with Python.

Accessing to Python script editor

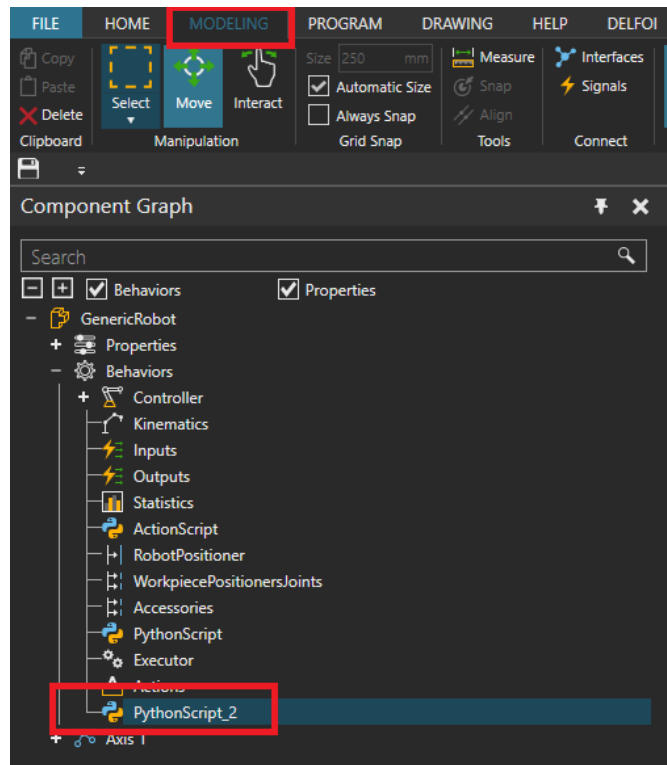
1. Bring Generic Articulated Robot to the workspace from eCatalog.
2. To access the Python script editor, press the robot and select Modeling tab (1). After that navigate to Behaviors (2) and Python Script (3).



3. Script editor will pop up to the window. Code can be built from the button Compile. Which is in the left-up corner. Compiling here means that the written code will be compiled to the selected robot.



4. Close the editor. You will find the script that you wrote under Component Graph in modelling view. Robot has to be selected. Our python script is named as PythonScript_2.



5. By double clicking this you can modify the script.

Initialization

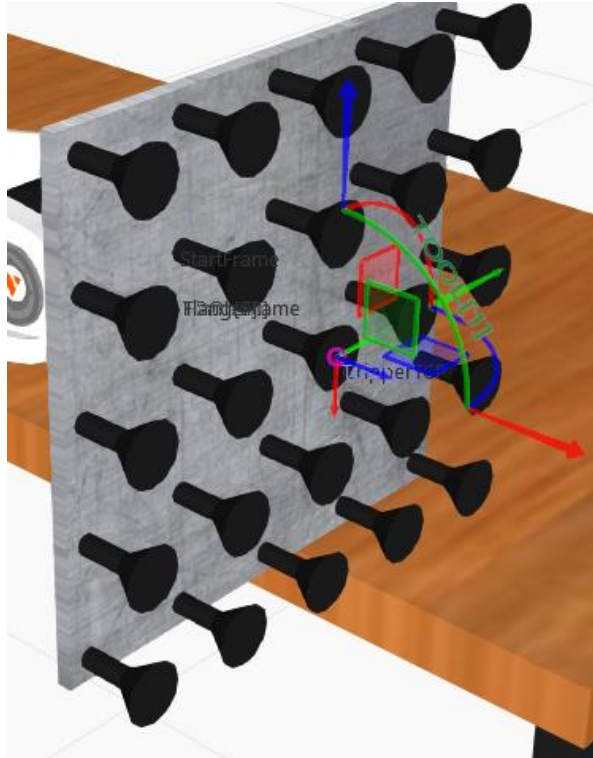
6. Now we need to bring some more components to the workspace:

location in eCatalog	component	location in workspace (world coordinates X,Y,Z,Rx,Ry,Rz)	Size
Home/eCatalog/Robots/Visual components	Generic Articulated robot	0,0,0,0,0,0	initial [3]
Home/eCatalog/Models By Type/Tools/Visual Components	Parametric Suction Cup Gripper	PNP to the tip of the robot	initial
Home/eCatalog/Models By Type/Basic Shapes/Visual Components	Block Geo	900,0,0,0,0,0	initial (you can change the material, if you want)
Home/eCatalog/Models By Type/Interior Facilities/Visual Components	Table A	0,1100,0,0,0	Change Length to 2000

[3] should be already in workspace

Tool Frame

Set the tool1 frame as in basic 7 exercises:



Python script

Copy and compile the code. You can modify it to test out how the Python works with the robot.

Analysis

- Read the script, change the parameters and try to understand the syntax
- Compare the workload between writing the python script and programming with visual components' graphical interface.

Main assignment

The next assignment is compulsory in the courses ELEC-C1320/ ELEC-D1320 robotics. The maximum points for the assignment is 20. The minimum points to pass the assignment is 10. In addition if you get >60% right in MyCourses quiz, you get +2 points.

The object for the assignment is to use the tools presented in basic assignments. As final output, the object is to automate the assembly of Finnish sauna water heater that is connected to sauna stove. The assembly line needs to assembly **only one** heater.

The drawings of the Heater is presented appendices. The cube in the model is expected to be hollow and the needed holes for the pipes and other objects such as tap are **not** considered as part of this assignment.

In the file there are all the measures for the Heater except the position of the identify numbers. The identify number and its location will be calculated from the sum of your group members student numbers. The alphabets are referred as zero. If the sum is over 1 000 000, the least significant number will be erased.

The identify number is the two least significant numbers of the sum of your group members student numbers (later referred as student numbers) The numbers will be located to the big side (side is sized 380 x 450) of the heater tank. Scale the numbers to be reasonable. The position is determined by the student number(s) so that position of (x,y)=(0,0) is the center point of the big side.

	x	y
+ sign	The fourth most significant number of student numbers is odd	The third most significant number of student numbers is odd
- sign	The fourth most significant number of student numbers is even	The third most significant number of student numbers is even
location from center of the heater's top	three most significant number of student numbers/5	three least significant number of student numbers/6

As example students have student numbers 132456, 95215N and 987512

The sum of student numbers is $132456 + 952150 + 987512 = 207211(8)$

The number to be on the big side of the tank is 11

The signs of x- value is '−' and y-value is '+'

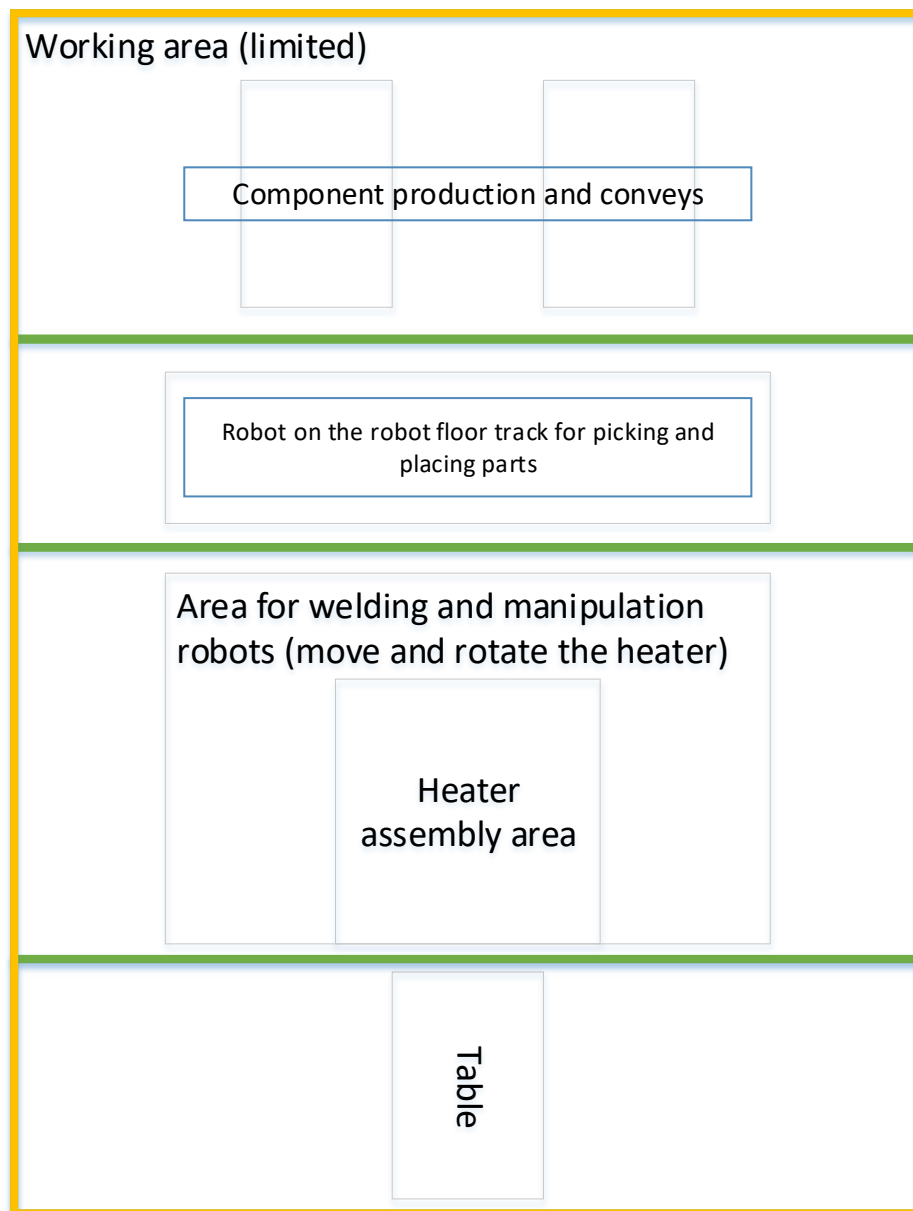
The x- coordinate is $207/5=41$

The y-coordinate is $211/6=35$

The sequence of assembly line is following.

1. The parts, presented in WaterHeaterModel.pdf are produced and delivered via conveyor(s)
2. A robot is taking parts from the conveyor(s) in correct order and moving those to heater assembly area or area where from the parts are moved to assembly area
3. Robots manipulate and weld parts to correct positions
4. Assembled heater moved to table.

The planned assembly line sequence is presented below **and the maximum size of total working area is $24m^2$** . You may choose the location of different phases freely.



In the table below the points to gain from the task in chronological order are explained. You are allowed to leave certain parts off from your solution (with loss of points naturally) and replace by your own, easier solution. **There are also few restrictions that needs to be taken into consideration, so please – read the following table with extra care.**

Part	Points max	penalty	wrong answers
Component production and conveys	2	50%	There are more than two conveyors
		50%	There are no conveyors but only parts
Robot on the robot floor track for picking and placing parts	6	33%	The robot is not on floor track
		66%	The robot on the floor track is Generic robot by Visual Components
Heater assembly	8	12,5%	Joint limits are broken frequently
		25%	Collison between robots and surroundings (including other robots) frequently/collision is highly detectable
		25%	The heater is not welded with Delfoi arc
		50%	The robot assembling the heater is a Generic robot by Visual Components
Ready heater	2	100%	The heater is not even near as in WaterHeaterModel.pdf
		50%	Identity numbers missing
Assembled heater is moved to separate table	2	50%	Parts are missing
		50%	The robot moving the heater is a Generic robot by Visual Components
Other restrictions		-1 point	The working area is not marked
		-2 points	- 0,5 points/extra square meter
		-2 points	The laws of physics such as gravitation are violated
		-20 points	The solution is copied from other group
Total	20		

Return your solution as Video and simulation model. To create a video, watch <https://www.youtube.com/watch?v=wyLlws9140g> . Approximately first three minutes are enough to understand how to export a video.

Tips

- Make a “detailed draft” – a plan of at least next issues
 - signals (how to configure different robots to co-operate together)
 - process flow (what happens in what order and by which robot)
 - locations of base/tool frames
- If you did not do a “detailed draft”, think twice and do the plan
- ctrl + z does not affect to changes done in program i.e. if you delete a statement, it is gone.
- It is not always easiest to start creating the line in chronological order. It might be easier to create the simple tasks first and then assemble the production line.
- If you do it in groups, make sure that both know how software works and how you are using bases and tools etc.
 - If one uses tools, bases and subprograms and other programs everything with respect to world coordinates in one big main program, it might be difficult to debug.
- Remember to select the “include components” when saving
- You are allowed to change the parameters of the gripper (e.g. size)
- **READ BEFOREHAND THE FIX FOR THE PROBLEMS list “e.g. The robot started welding to arbitrary location/ weld itself when I used Delphi arc” –shown below**

Known problems and possible fixes

My software crashed and I didn't save

We recommend to save the program every once in a while to a new file.

The search can't find the object form eCatalog, even it is presented in assignment instructions

You probably have a filter on your eCatalog. Click the All Models folder and do the search again. Check that you have spelled the word correctly.

Some of my object were lost after I saved and opened the software again

Did you check the "include components" when you saved your solution? You may have a possibility to get the objects back if you log in to same computer where you did your solution previously.

The robot's initial state is wrong

The problem description is usually following: "I started to manipulate the robot and did PTP or linear movement and played simulation. Then I did something and now if I reset my simulation the robot won't return to its initial position even if I tell it to start the program from initial position"

This happens, because you have probably pressed play after the simulation has ended. You can fix the problem by setting the wanted initial position as last, then play the simulation, wait the simulation to stop and press play again. Reset the simulation and delete the final "initial position" from your program.

The robot started welding to arbitrary location/ weld itself when I used Delfoi arc

The delfoi add-on has difficulties to work properly when there are works (such as Works task control with works processor) due to ever changing child-parent relationships. You will face a situation with delfoi arc that when you run the simulation, the robot starts to weld in arbitrary place (most likely it will weld itself).

To avoid the problem, you need to add a ready model (of the heater) to welding location (the location where you weld the parts together) and determine the welding for the preset model (heater in main assignment). Change the preset model as "glass" from the properties to "hide" the model. If you hide the model by changing the "visible" status from the properties, you can't see the welding curve when done. Set the parts coming from the conveyor to exact same locations as the preset model is.



Appendices

Python example

```
from vcScript import *

# Access to the owner component of the script.
comp = GetComponent()

# Get the application. With this we can access for example to the different
# components of the workspace
app = GetApplication()

# Now let's find the block and table from the workspace
table = app.findComponent('Table1')
block = app.findComponent('Block')

# Next we want to get access to the main routine, which is familiar from the
# Program Editor
robotExecutor = comp.findBehavioursByType("rRobotExecutor")[0]
robotProgram = robotExecutor.Program
mainRoutine = robotProgram.MainRoutine

# We can clear the main routine with this command
mainRoutine.clear()

# Now lets make our first statement to the main routine in Program Editor
# We want to pick the box with one PTP motion and then with linear motion

# 1) Add PTP statement
statement_PTP = mainRoutine.addStatement(VC_STATEMENT_PTPMOTION)

# 2) Get position, set position matrix and get table center
position = statement_PTP.Positions[0]
pos_matrix = block.PositionMatrix
block_center = block.BoundCenter

# 3) Modify the position and rotate robot tip to correct position
# We can make the approaching point by modifying the Z coordinate
pos_matrix.translateRel( block_center.X, block_center.Y, block_center.Z*2+400)
pos_matrix.rotateRelY(180)

# 4) Set new position
position.PositionInWorld=pos_matrix

# 5) Add Linear motion statement
statement_LIN = mainRoutine.addStatement(VC_STATEMENT_LINMOTION)

# 6) Set correct base and tool
statement_LIN.Base = 'BASE_1'
statement_LIN.Tool = 'TOOL[1]'

# 7) Get position, set position matrix and get table center
position = statement_LIN.Positions[0]
pos_matrix = block.PositionMatrix
block_center = block.BoundCenter

# 8) Modify the position and rotate robot tip to correct position
pos_matrix.translateRel( block_center.X, block_center.Y, block_center.Z*2)
pos_matrix.rotateRelY(180)
```

```

# 9) Set new position
position.PositionInWorld=pos_matrix

#####

# Now we want to give robot the grasping command with binary output

# 1) Binary output statement
statement_bino = mainRoutine.addStatement(VC_STATEMENT_SETBIN)

# 2) Set Output port and value. Grasp value for input port 1 is True
statement_bino.OutputPort = 1
statement_bino.OutputValue = True

#####

# Next we want to move the box with one linear motion to up and then with
# PTP motion to up of the table and after that to the table.

# 1) Add Linear motion statement
statement_LIN2 = mainRoutine.addStatement(VC_STATEMENT_LINMOTION)

# 2) Get position, set position matrix and get table center
position = statement_LIN2.Positions[0]
pos_matrix = block.PositionMatrix

# 3) Modify the position and rotate robot tip to correct position
pos_matrix.translateRel( block_center.X, block_center.Y, block_center.Z*2+900)
pos_matrix.rotateRelY(180)

# 4) Set new position
position.PositionInWorld=pos_matrix

# 5) Add PTP statement
statement_PTP = mainRoutine.addStatement(VC_STATEMENT_PTPMOTION)

# 6) Get position, set position matrix and get table center
position = statement_PTP.Positions[0]
pos_matrix = table.PositionMatrix
table_center = table.BoundCenter

# 7) Modify the position and rotate robot tip to correct position
# We can make the approaching point by modifying the Z coordinate
pos_matrix.translateRel( table_center.X-400, table_center.Y, table_center.Z*2+600)
pos_matrix.rotateRelY(180)

# 8) Set new position
position.PositionInWorld=pos_matrix

# 9) Add Linear motion statement
statement_LIN2 = mainRoutine.addStatement(VC_STATEMENT_LINMOTION)

# 10) Set correct base and tool
statement_LIN2.Base = 'BASE_1'
statement_LIN2.Tool = 'TOOL[1]'

# 11) Get position, set position matrix and get table center
position = statement_LIN2.Positions[0]
pos_matrix = table.PositionMatrix

```

```

# 12) Modify the position and rotate robot tip to correct position
pos_matrix.translateRel( table_center.X-400, table_center.Y,table_center.Z*2+400)
pos_matrix.rotateRelY(180)

# 13) Set new position
position.PositionInWorld=pos_matrix

#####

# Binary output, grasping to false
# 1) Binary output statement
statement_bino = mainRoutine.addStatement(VC_STATEMENT_SETBIN)

# 2) Set Output port and value. Grasp value for input port 1 is True
statement_bino.OutputPort = 1
statement_bino.OutputValue = False

#####

# Unmount the tool to the table with linear, PTP motions and binary output

# 1) Add Linear motion statement
statement_LIN2 = mainRoutine.addStatement(VC_STATEMENT_LINMOTION)

# 2) Get position, set position matrix and get table center
position = statement_LIN2.Positions[0]
pos_matrix = table.PositionMatrix

# 3) Modify the position and rotate robot tip to correct position
pos_matrix.translateRel( table_center.X-400, table_center.Y,table_center.Z*2+600)
pos_matrix.rotateRelY(180)

# 4) Set new position
position.PositionInWorld=pos_matrix

# 5) Add PTP statement
statement_PTP = mainRoutine.addStatement(VC_STATEMENT_PTPMOTION)

# 6) Get position, set position matrix and get table center
position = statement_PTP.Positions[0]
pos_matrix = table.PositionMatrix
table_center = table.BoundCenter

# 7) Modify the position and rotate robot tip to correct position
# We can make the approaching point by modifying the Z coordinate
pos_matrix.translateRel( table_center.X+400, table_center.Y, table_center.Z*2+600)
pos_matrix.rotateRelY(180)

# 8) Set new position
position.PositionInWorld=pos_matrix

# 9) Add Linear motion statement
statement_LIN2 = mainRoutine.addStatement(VC_STATEMENT_LINMOTION)

# 10) Set correct base and tool
statement_LIN2.Base = 'BASE_1'
statement_LIN2.Tool = 'TOOL[1]'

```

```

# 11) Get position, set position matrix and get table center
position = statement_LIN2.Positions[0]
pos_matrix = table.PositionMatrix

# 12) Modify the position and rotate robot tip to correct position
pos_matrix.translateRel( table_center.X+400, table_center.Y,table_center.Z*2)
pos_matrix.rotateRelY(180)

# 13) Set new position
position.PositionInWorld=pos_matrix

# 14) Unmount the tool with binary output channel 33
statement_bino = mainRoutine.addStatement(VC_STATEMENT_SETBIN)

# 15) Set Output port and value. Grasp value for input port 1 is True
statement_bino.OutputPort = 33
statement_bino.OutputValue = False

# 16) Last linear statement
statement_LIN2 = mainRoutine.addStatement(VC_STATEMENT_LINMOTION)

# 17) Get position, set position matrix and get table center
position = statement_LIN2.Positions[0]
pos_matrix = table.PositionMatrix

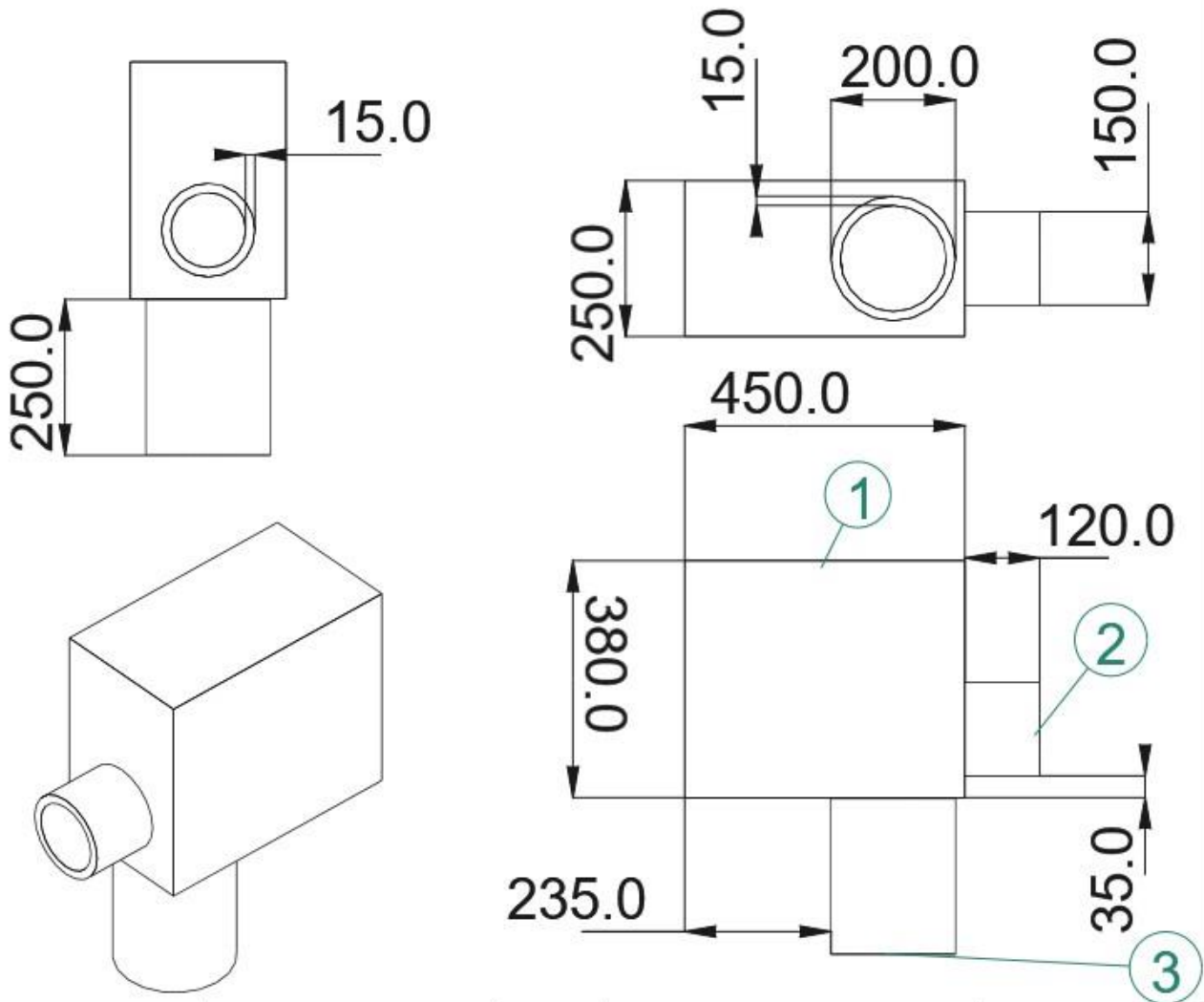
# 18) Modify the position and rotate robot tip to correct position
pos_matrix.translateRel( table_center.X+400, table_center.Y,table_center.Z*2+400)
pos_matrix.rotateRelY(180)

# 19) Set new position
position.PositionInWorld=pos_matrix

```

Water heater model

ITEM	QTY	NAME
1	1	Tank (Block Geo)
2	1	OutPipe (Tube Geometry)
3	1	InPipe (Tube Geometry)



UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH TOLERANCES: LINEAR: ANGULAR:							FINISH:		DEBUR AND BREAK SHARP EDGES		DO NOT SCALE DRAWING			REVISION 1				
											TITLE: <h1>Water Heater</h1>							
DRAWN																		
CHK'D																		
APP'VD																		
MFG																		
Q.A									MATERIAL: <h2>Steel</h2>		DWG NO.					A4		
									WEIGHT: 10 kg					SCALE: 1:5			SHEET 1 OF 1	