TASKBOARD IN GAZEBO

This document captures the basic implementation and problems that arose porting the Solidworks STL export into Gazebo.

## Off-center origin Problem

The first and foremost problem was that a centroid origin not being "in the middle at the bottom" of a part. That is, the part that gives a pose as 0,0,0 should be centered around the xy origins and have z minimum equal to 0. The problem symptom was the origin being offset from the origin. Figure 1 shows the peg array with the expected centroid in the middle, and the off-center centroid.



Figure Peg array with off-center centroid

This corresponded to Gazebo placing an object with centroid given by the pose as off-center as shown below with round peg and peg array both having the same centroid pose but only the round peg is centered:

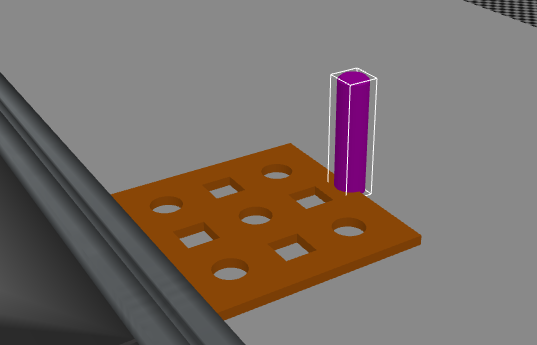


Figure 2 Peg board and peg with same centroid pose

It was determined that this was due to the STL being rotated upon Gazebo creation – that is creation of the model in Gazebo moved the origin in accordance with the rotation of 90o around the X axis as shown below. For example, this pose

<pose frame="">0 0 0 1.57079 0 0</pose>

used when creating a taskboard\_base model, led to the part model pose being moved. To fix this the STL was "prerotated" by 90o (or -90o depending on the STL orientation) and then the STL was centered and translated so that Z min was set to 90. This lead to the Gazebo SDF definition having the pose not contain any rotation to fix the part orientation.

<pose frame="">0 0 0 0 0 0</pose>

This fixed the "bad centroid" problem. Thus, the centroid of a peg array is hole 5 (middle one) and all the offsets can be calculated from this – it saves one transform from the off-center origin to the centroid. However, for the square peg this off-center centroid would be a major headache as it could have many different orientations.

## Normalizing the STL Model

I used the Python STL program I wrote originally to fix the APRS STL gear images so they would be centered around the origin, with minimum z being 0 so that when you placed the object given its centroid into a slot, it would go where you expected. NOTE, it appears as if the gears origin is 0,0,0 which corresponds to how the centroid is given – which corresponds to the bottom of a gear. The following is the bash script that used the Python STL library to modify the STL orientations of the taskboard: (note rotate -rx 90 is counterclockwise, -rx -90 is clockwise)

$python38 AnalyzeMeshes.py -rx -90 -i ".\STL\taskboard\_arraybase-1.STL"

$python38 AnalyzeMeshes.py -a -c -i ".\STL\taskboard\_arraybase-1\_Rotatex.stl"

$python38 AnalyzeMeshes.py -a -minz -i ".\STL\taskboard\_arraybase-1\_Rotatex\_Centered.stl"

$python38 AnalyzeMeshes.py -rx -90 -i ".\STL\taskboard-arrayback-1.STL"

$python38 AnalyzeMeshes.py -a -c -i ".\STL\taskboard-arrayback-1\_Rotatex.stl"

$python38 AnalyzeMeshes.py -a -minz -i ".\STL\taskboard-arrayback-1\_Rotatex\_Centered.stl"

$python38 AnalyzeMeshes.py -rx -90 -i ".\STL\taskboard-pegarray1-1.STL"

$python38 AnalyzeMeshes.py -a -c -i ".\STL\taskboard-pegarray1-1\_Rotatex.stl"

$python38 AnalyzeMeshes.py -a -minz -i ".\STL\taskboard-pegarray1-1\_Rotatex\_Centered.stl"

$python38 AnalyzeMeshes.py -rx 90 -i ".\STL\taskboard\_square\_peg-1.STL"

$python38 AnalyzeMeshes.py -a -c -i ".\STL\taskboard\_square\_peg-1\_Rotatex.stl"

$python38 AnalyzeMeshes.py -a -minz -i ".\STL\taskboard\_square\_peg-1\_Rotatex\_Centered.stl"

$python38 AnalyzeMeshes.py -c -i ".\STL\taskboard\_round\_peg-1.STL"

$python38 AnalyzeMeshes.py -a -minz -i ".\STL\taskboard\_round\_peg-1\_Centered.stl"

$python38 AnalyzeMeshes.py -rx -90 -i ".\STL\taskboard-pegarray2-1.STL"

$python38 AnalyzeMeshes.py -a -c -i ".\STL\taskboard-pegarray2-1\_Rotatex.stl"

$python38 AnalyzeMeshes.py -a -minz -i ".\STL\taskboard-pegarray2-1\_Rotatex\_Centered.stl"

This generated the following reoriented STL files for use in Gazebo so that the centroid is in the middle of the part:

taskboard\_arraybase-1\_Rotatex\_Centered\_ZeroZmin.stl

taskboard\_square\_peg-1\_Rotatex\_Centered\_ZeroZmin.stl

taskboard-arrayback-1\_Rotatex\_Centered\_ZeroZmin.stl

taskboard-pegarray1-1\_Rotatex\_Centered\_ZeroZmin.stl

taskboard-pegarray2-1\_Rotatex\_Centered\_ZeroZmin.stl

The code for AnalyzeMeshes.py is found in the Python folder.

## Importing the STL Model into Gazebo

Gazebo defines models in SDF that can be part of the world. A model, such as taskboard\_base, is defined with a folder name taskboard\_base containing a model.config that references a model.sdf that defines the properties of the model: name, pose, inertia, mass, color, visual, collision bounding box, etc.

model.sdf

model.config

taskboard\_arraybase-1\_Rotatex\_Centered\_ZeroZmin.stl

Here is a sample sdf file for taskboard\_base, which will not be explained:

<?xml version="1.0" encoding="UTF-8"?>

<sdf version="1.6">

<model name="taskboard\_base">

<static>1</static>

<link name="link1">

<pose frame="">0 0 0 0 0 0</pose>

<velocity\_decay>

<angular>

.0005

</angular>

</velocity\_decay>

<collision name="collision1">

<pose frame="">0 0 0 0 0 0</pose>

<geometry>

<mesh>

<scale>0.002 0.002 0.002</scale>

<uri>model://taskboard-base/taskboard\_arraybase-1\_Rotatex\_Centered\_ZeroZmin.stl</uri>

</mesh>

</geometry>

<max\_contacts>10</max\_contacts>

</collision>

<visual name="visual1">

<pose frame="">0 0 0 0 0 0</pose>

<geometry>

<mesh>

<scale>0.002 0.002 0.002</scale>

<uri>model://taskboard-base/taskboard\_arraybase-1\_Rotatex\_Centered\_ZeroZmin.stl</uri>

</mesh>

</geometry>

<material>

<script>

<name>Gazebo/Orange</name>

<uri>\_\_default\_\_</uri>

</script>

</material>

</visual>

<inertial>

<mass>0.0385</mass>

<inertia>

<ixx>4.78233711e+00</ixx>

<ixy>2.06443170e-09</ixy>

<ixz>3.14256508e-00</ixz>

<iyy>4.78233728e+00</iyy>

<iyz>1.83990112e-08</iyz>

<izz>2.44284722e-01</izz>

</inertia>

</inertial>

<self\_collide>0</self\_collide>

<gravity>0</gravity>

</link>

</model>

</sdf>

The visual is defined by :

<mesh>

<scale>0.002 0.002 0.002</scale>

<uri>model://taskboard-base/taskboard\_arraybase-1\_Rotatex\_Centered\_ZeroZmin.stl</uri>

</mesh>

Which says the use the taskboard-base model folder to find the STL file taskboard\_arraybase-1\_Rotatex\_Centered\_ZeroZmin.stl which provides the visual mesh for the taskboard\_base. Note, the STL is defined in millimeters and Gazebo uses meters for linear measurements, so the scale is actually twice as big as the STL file. This does not effect the centered origin.

The Gazebo world file then uses this model to define the taskboard\_base 3 times, once for feeder tray for Motoman, once for a supply tray for Motoman robot and the last as a Fanuc feeder container base.

<model name="sku\_taskboard\_base">

    <include>

<static>true</static>

<uri>model://taskboard-base</uri>

    </include>

    <pose frame="">0.258805 -0.627349 0.910184 0 0 3.14</pose>

</model>

<model name="sku\_taskfeeder\_base">

    <include>

<static>true</static>

<uri>model://taskboard-base</uri>

    </include>

    <pose frame="">-0.098805 -0.627349 0.910184 0 0 3.14</pose>

</model>

<model name="sku\_taskfeeder\_base\_fanuc1">

    <include>  
 <static>true</static>

<uri>model://taskboard-base</uri>

    </include>

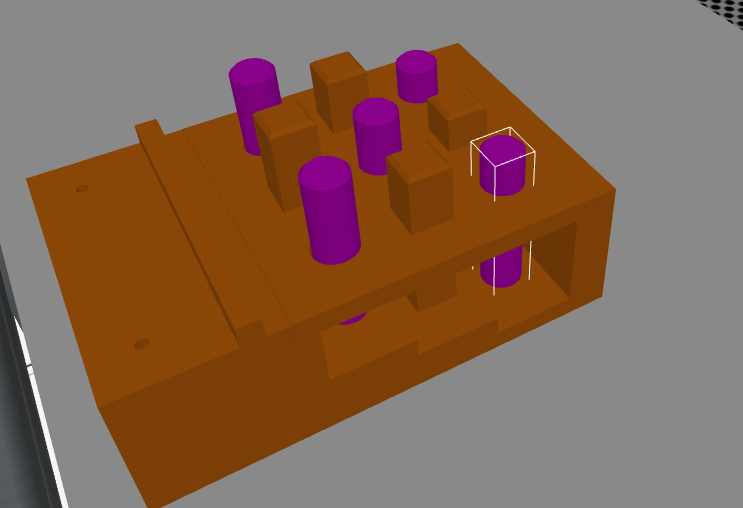
    <pose frame=""> 0.183254 -1.053015 0.915990 0 0 1.57079632679</pose>

</model>

Note, the <pose> defines the centroid of the Gazebo model (taskboard-base). Note for the motoman, the taskboard-base is rotated 180o around the Z axis in effect, causing the base to face the Motoman. While for the fanuc robot it is flipped 90o to face the robot.

Below code is found in Excel that by changing the XYZ value - in this case (-0.09888 -0.5891 0.9351) – will change the XYZ offsets for each hole in a peg array. Below are xyz numbers modified to reflect the offset from the hole 5 centroid location. Note, changes in Z offset are due to the steps in the task board base piece. Using the modified Z, the peg will sit flush against the task board base piece.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Hole | X Offset | Y offset | Z offset | X | Y | Z |
| Hole | X offset | Y offset | Z offset | -0.09888 | -0.5891 | 0.9351 |
| 5 - rnd | 0.0000 | 0.0000 | 0.0000 | -0.0989 | -0.5891 | 0.9351 |
| 1- rnd | 0.0508 | -0.0508 | 0.0143 | 0.0508 | -0.0508 | 0.9494 |
| 2- sq | 0.0508 | 0.0000 | 0.0000 | -0.0481 | -0.5891 | 0.9351 |
| 3- rnd | 0.0508 | 0.0508 | -0.0129 | -0.0481 | -0.5383 | 0.9222 |
| 4 - sq | 0.0000 | -0.0508 | 0.0143 | -0.0989 | -0.6399 | 0.9494 |
| 6 - sq | 0.0508 | 0.0000 | -0.0129 | -0.0481 | -0.5383 | 0.9222 |
| 7- rnd | -0.0508 | -0.0508 | 0.0143 | -0.1497 | -0.6399 | 0.9494 |
| 8- sq | -0.0508 | 0.0000 | 0.0000 | -0.1497 | -0.5891 | 0.9351 |
| 9- rnd | -0.0508 | 0.0508 | -0.0129 | -0.1497 | -0.5383 | 0.9222 |



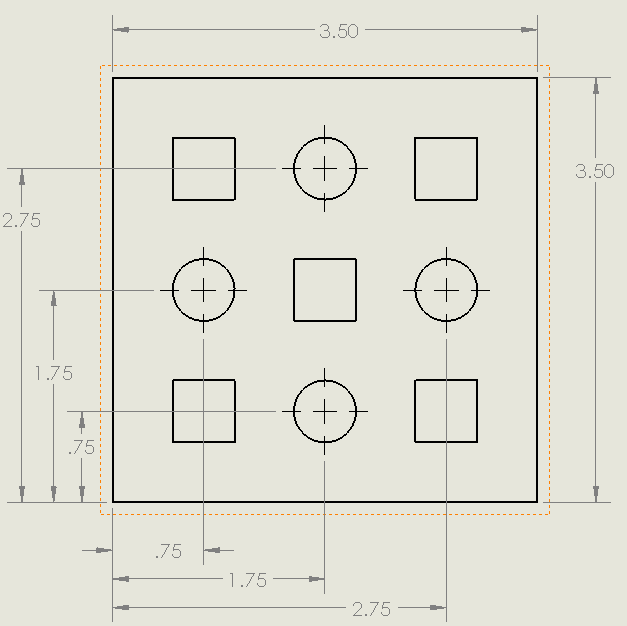
The peg board 2 is identical in dimensions and centroids, but round and square pegs are reversed.

Why did I measure the gear tray offsets from the centroid? Given a centroid and an orientation, you can compute the location of any hole.

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | Meters | | |
| Round Peg | R=0.006328197 H=0.0126999969 | | |
| Square Peg |  |  |  |
|  |  |  |  |

NOTE1: In gazebo the image is magnified x 2 so all offset numbers x 2.  
NOTE 2: This is with zero rotation, but in gazebo to face Motoman, must rotate around Z by 180o

Peg Array 1:



Peg Array 2:

