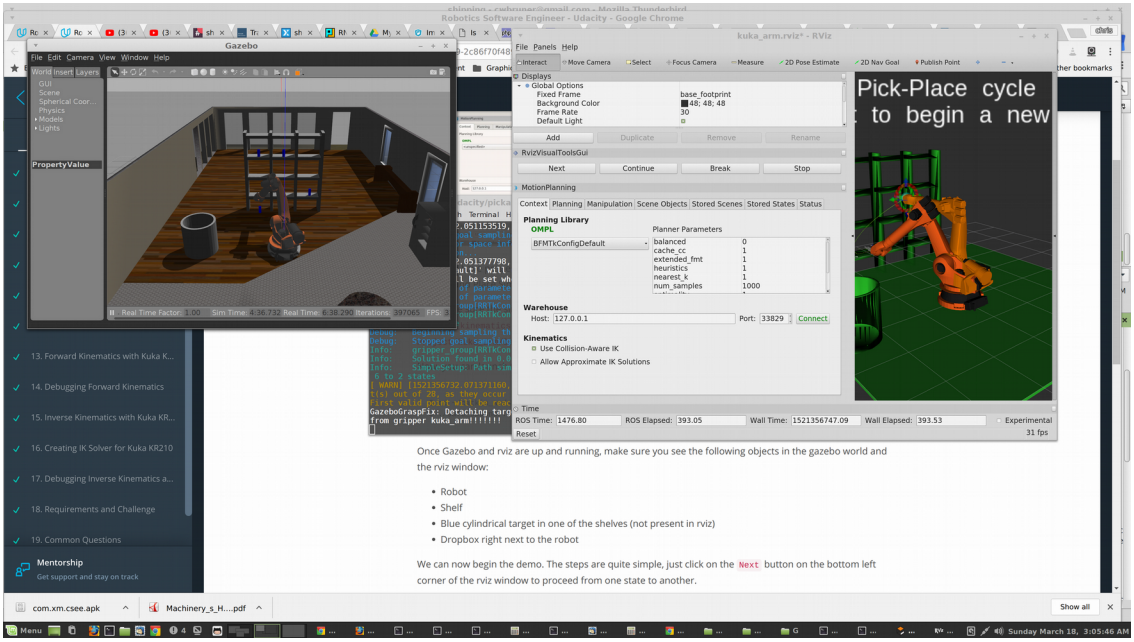
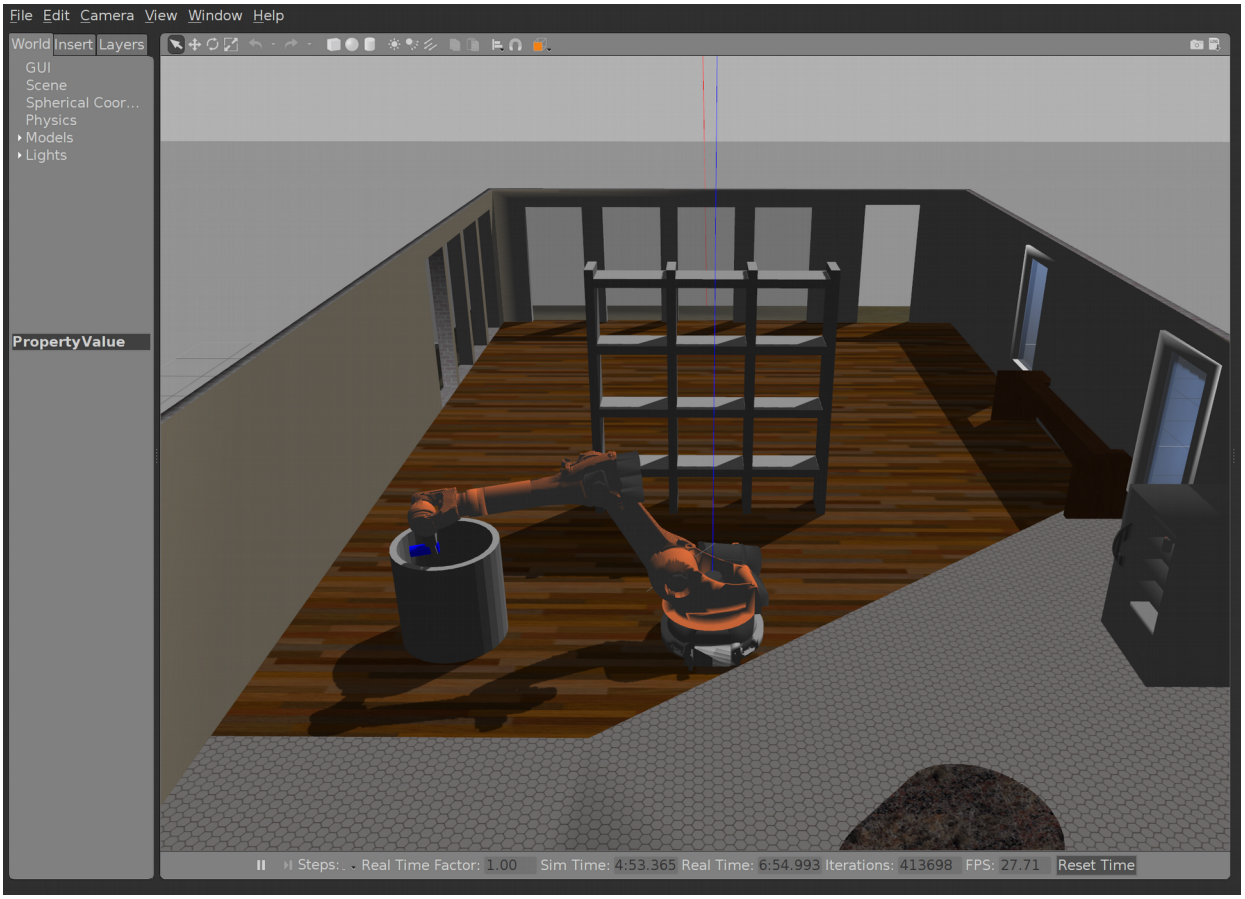
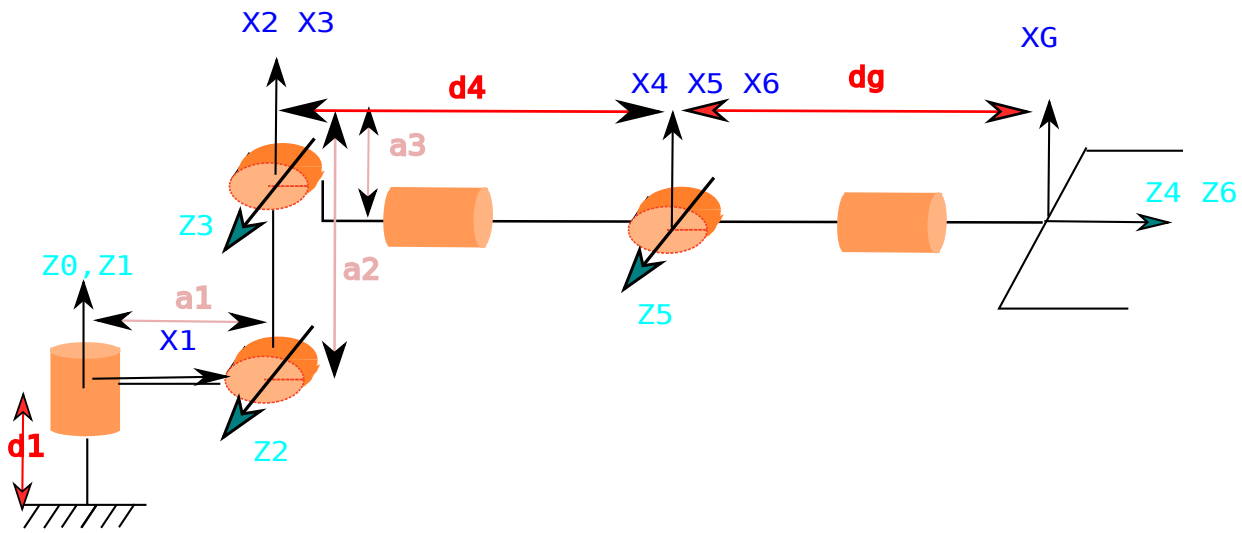
Pick and Place project.

I believe I was successful in implementing this project (finally!). In the end I was able to get most targets into the barrel, with the only misses being where the target hit the robot arm as it was being dropped. I've uploaded two videos at https://youtu.be/HvxHfF8D-4o and https://youtu.be/V6wXcIFHWUg. Unfortunately Rvis is now acting up, and the next button stopped working after completing the first pick and place. Both videos have the same result, although earlier in the day, it had no problems (with out any code changes between the times).







Joint	a(joint-1)	a(joint-1)	d(joint)	theta(joint)
1	0	0	d1	theta1
2	-90	a1	0	Theta2 - 90
3	0	a2	0	Theta 3
4	-90	a3	d4	theta4
5	90	0	0	theta5
6	-90	0	0	theta6
7	0	0	dg	0

Joint	a(joint-1)	alpha(joint-1)	d(joint)	theta(joint)
1	0	0	.75=.33+.42	theta1
2	-90	.35	0	Theta2 - 90
3	0	1.25	0	Theta 3
4	-90	-0.054	1.5 =	theta4
5	90	0	0	theta5
6	-90	0	0	theta6
7	0	0	.303	0

a(i) is the distance from Joint(i-1) to Joint(i) measured along X(i-1)
alpha(i) is the angle from Joint(i-1) to Joint(i) measured along X(i-1)
d(i) is the distance from Joint(i-1) to Joint(i) measured along Z(i-1)
theta(i) is the angle from Joint(i-1) to Joint(i) measured along Z(i-1)

The joint values shown in the kr210.urdf.xcro are:

joint_1	0,	0,	0.33
joint_2	0.35,	0,	0.42
joint_3	0,	0,	1.25
joint_4	0.96,	0,	-0.054
joint_5	0.54,	0,	0
joint_6	0.193,	0,	0

```
        # Create Modified DH parameters
DH_Table = { linkTwist0:    0,          linkLength0: 0,      linkOffset1: 0.75,   joint1:   joint1,
             linkTwist1:    -pi/2.,     linkLength1: 0.35,   linkOffset2: 0,      joint2: -pi/2. + joint2,
             linkTwist2:    0,          linkLength2: 1.25,   linkOffset3: 0,      joint3:   joint3,
             linkTwist3:    -pi/2.0,    linkLength3: -0.054, linkOffset4: 1.5,    joint4:   joint4,
             linkTwist4:    pi/2.0,     linkLength4: 0,      linkOffset5: 0,      joint5:   joint5,
             linkTwist5:    -pi/2.0,    linkLength5: 0,      linkOffset6: 0,      joint6:   joint6,
             linkTwist6:    0,          linkLength6: 0,      linkOffset7: 0.303,  joint7:   0.0 }
```

```
def TF_Matrix(twist,length,Offset,q):
        TF = Matrix([[cos(q), -sin(q),    0.0,  length],
                     [sin(q) * cos(twist),cos(q) * cos(twist), -sin(twist), -sin(twist) * Offset],
                     [sin(q) * sin(twist),cos(q) * sin(twist), cos(twist),     cos(twist) * Offset],
                     [0.0,                0.0, 0.0,            1.0]])
        return TF
```

```
    T0_1 = TF_Matrix(linkTwist0, linkLength0, linkOffset1,joint1).subs(DH_Table)
    T1_2 = TF_Matrix(linkTwist1, linkLength1, linkOffset2,joint2).subs(DH_Table)
    T2_3 = TF_Matrix(linkTwist2, linkLength2, linkOffset3,joint3).subs(DH_Table)
    T3_4 = TF_Matrix(linkTwist3, linkLength3, linkOffset4,joint4).subs(DH_Table)
    T4_5 = TF_Matrix(linkTwist4, linkLength4, linkOffset5,joint5).subs(DH_Table)
    T5_6 = TF_Matrix(linkTwist5, linkLength5, linkOffset6,joint6).subs(DH_Table)
    T6_EE =TF_Matrix(linkTwist6, linkLength6, linkOffset7,joint7).subs(DH_Table)
    T0_EE = T0_1 * T1_2 * T2_3 * T3_4 * T4_5 * T5_6 * T6_EE;
```

$$
{}_0T^1 = \begin{bmatrix} \cos(joint\,1) & -\sin(joint\,1) & 0.0 & 0.0 \\ \sin(joint\,1) & \cos(joint\,1) & 0 & 0 \\ 0 & 0 & 1 & 0.750000000000000 \\ 0.0 & 0.0 & 0.0 & 1.00000000000000 \end{bmatrix}
$$

$$
{}_1T^2 = \begin{bmatrix} \cos(joint\,1) & -\sin(joint\,1) & 0.0 & 0.0 \\ \sin(joint\,1) & \cos(joint\,1) & 0 & 0 \\ 0 & 0 & 1 & 0.750000000000000 \\ 0.0 & 0.0 & 0.0 & 1.00000000000000 \end{bmatrix}
$$

$$
{}_2T^3 = \begin{bmatrix} \cos(joint\,3) & -\sin(joint\,3) & 0.0 & 1.250 \\ \sin(joint\,3) & \cos(joint\,3) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}
$$

$$
{}_3T^4 = \begin{bmatrix} \cos(joint\,4) & -\sin(joint\,4) & 0.0 & -0.0540 \\ 0 & 0 & 1 & 1.50 \\ -\sin(joint\,4) & -\cos(joint\,4) & 0 & 0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}
$$

$$
{}_4T^5 = \begin{bmatrix} \cos(joint\,5) & -\sin(joint\,5) & 0.0 & 0.0 \\ 0 & 0 & -1 & 0 \\ \sin(joint\,5) & \cos(joint\,5) & 0 & 0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}
$$

$$
{}_5T^6 = \begin{bmatrix} \cos(joint\,6) & -\sin(joint\,6) & 0.0 & 0.0 \\ 0 & 0 & 1 & 0 \\ -\sin(joint\,6) & -\cos(joint\,6) & 0 & 0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}
$$

$$
{}_6T^7 = \begin{bmatrix} 1 & 0 & 0.0 & 0.0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.3030 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}
$$

Inverse Kinematics

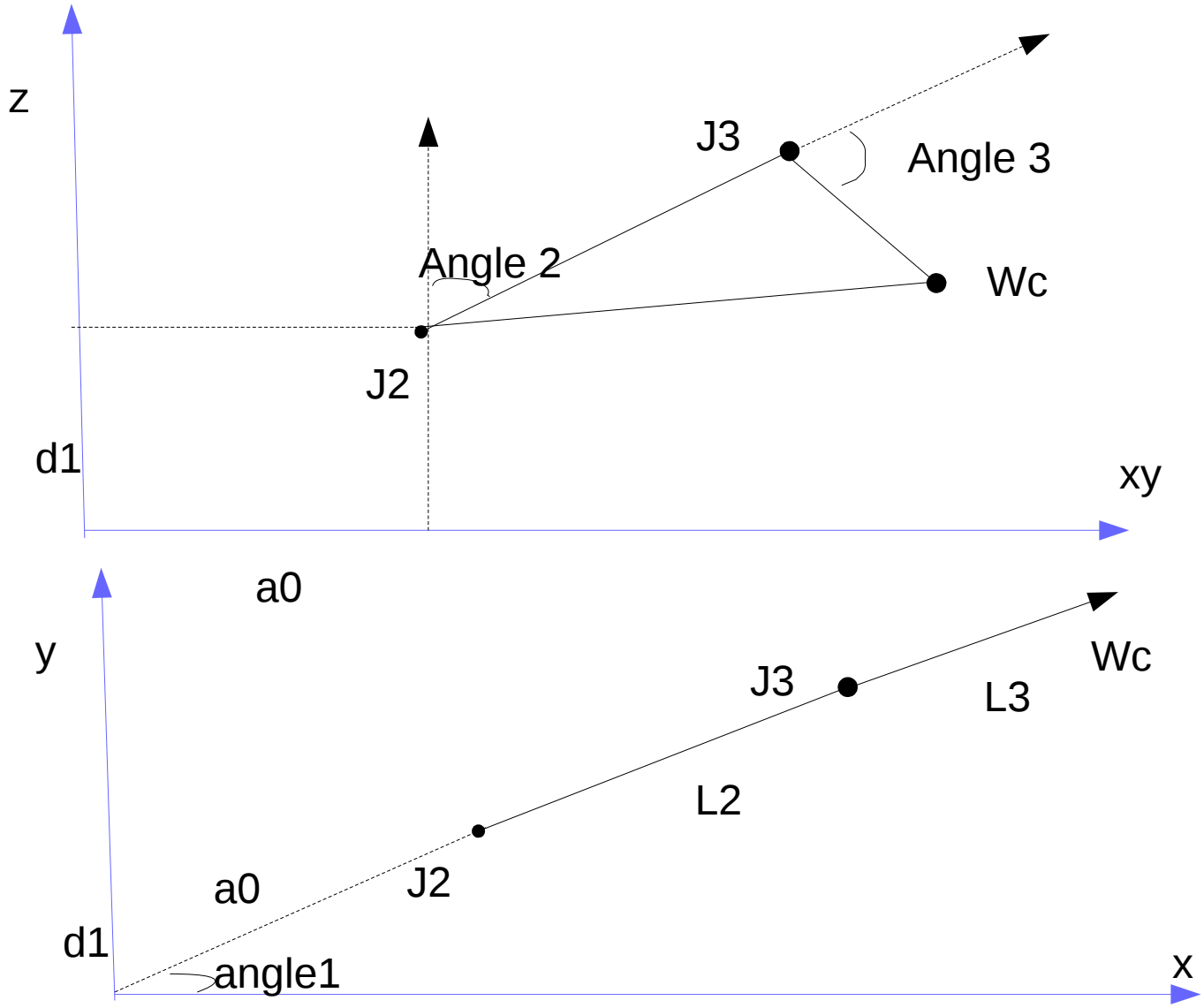
The problem is broken into two parts, one for the revolute arm joints, another for a spherical arm. The revolute arm would return the joint angles for joints 1 through 3 and the spherical arm for joints 4 through 6. The reason for doing this is because the Joint 7 end effector, has a common point of intersection (wrist center).

theta4 = atan2(r36[2, 2], -r36[0, 2])

theta5 = atan2(sqrt(r36[0, 2]^2 + r36[2, 2]^2), r36[1, 2])

theta6 = atan2(-r36[1, 1], r36[1, 0])

So for the revolute part we would have:



For angle 1, it only lies on the xy plane so is atan2(y,x)

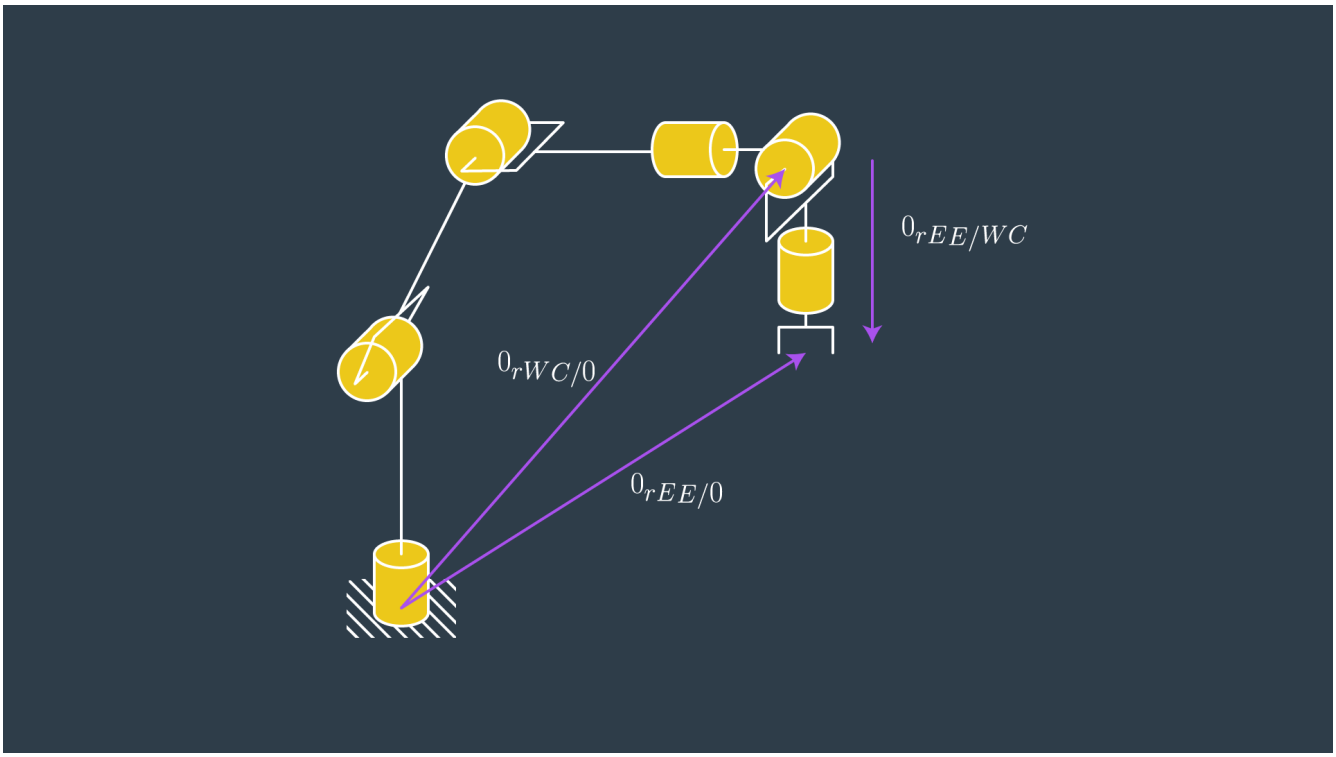
For the 2nd and 3rd angles, on the z plane use the cosine rule to obtain the angle first for angle 3, then calculating angle 2.

$$l2 = angle2$$
$$l3 = \sqrt{(a3^2 + d4^2)}$$
$$Angle3 = 180 - \theta 3$$
$$xy = \sqrt{(Wc_x^2 + Wc_y^2)} - a0$$

using the cosine rule

$$D^2 = xy^2 + z^2 = l2^2 + l3^2 - 2(l2)(l3)\cos(Angle3)$$
$$\cos(angle3) = (xy \wedge 2 + z^2 - l3^2 - l2^2)/(2 * l3 * l2) = r$$
$$angle3 = atan2(\sqrt{(1 - r^2)}, r), where \sqrt{(1 - r^2)} = \sin(angle3)$$
$$angle3 = atan2(-\sqrt{(1 - r^2)}, r)$$
$$angle2 = y - a0 = atan2(xy, z) - atan2(l3 * \sin(angle3), l2 + l3 * \cos(angle3)) angle3 = angle3 - pi/2$$

Angle 3 has to be -90 degrees as the starting point is horizontally right not vertically up.



Angle for joint 4,5,6

The overall roll, pitch, yaw for the end effector relative to the base link is as follows

$$\frac{a}{b} R_{zyx} = R_Z(\alpha) R_Y(\beta) R_X(\gamma) = \begin{bmatrix} r11 & r12 & r13 \\ r21 & r22 & r23 \\ r31 & r32 & r33 \end{bmatrix}$$

The complete transform is calculated by multiplying all the individual transforms

R0_6 = R0_1 * R1_2 * R2_3*R3_4*R4_5*R5_6

Correction of angle between gripper and DH

Rot_Error = ROT_z.subs(y, radians(180)) * ROT_y.subs(p,radians(-90))

ROT_EE = ROT_EE * Rot_Error

Results

I had very mixed results from this. Often the simulator wouldn't work, other times, things worked well, other times, the arm just knocked targets around with no changes in the source code. Sometimes the arm would pick up some objects and dump them correctly, but not grasped. Sometimes on this very first target, more then one target would appear. It turns out, that using the continue is NOT the same as using next with pauses. In the end I was able to get most targets into the barrel, with the only misses being where the target hit the robot arm as it was being dropped.

The graphics on the system do seem to be updating slowly as though they are not going through opengl's gpu system. This slows the whole system down, and on my system flashes in a very irritating manner.