ROBOTICA

CINEMATICA DIRETTA IN PYTHON

(Prof. Fischetti Pietro)

Il programma di simulazione 'Roby' per la cinematica diretta e inversa sviluppato in Python 2.7 ispirato al testo [1], e' costituito da:

Roby.py – Programma 'main' che crea le finestre e dialog di visualizzazione

La cartella myG2D - contiene funzioni wrapper (sulla libreria matplotlib) per il disegno 2D

La cartella myG3D - contiene funzioni wrapper (sulla libreria matplotlib) per il disegno 3D

La cartella myMatrix - contiene funzioni per l'algebra delle Matrici e Matrici di Denavit-Hartenberg.

La cartella Robots - contiene il file Robots.py dove viene definita la classe base Robot incaricata di inizializzare la grafica (2D/3D) al fine di disegnare giunti,bracci, assi etc., e il calcolo dei parametri D-H.

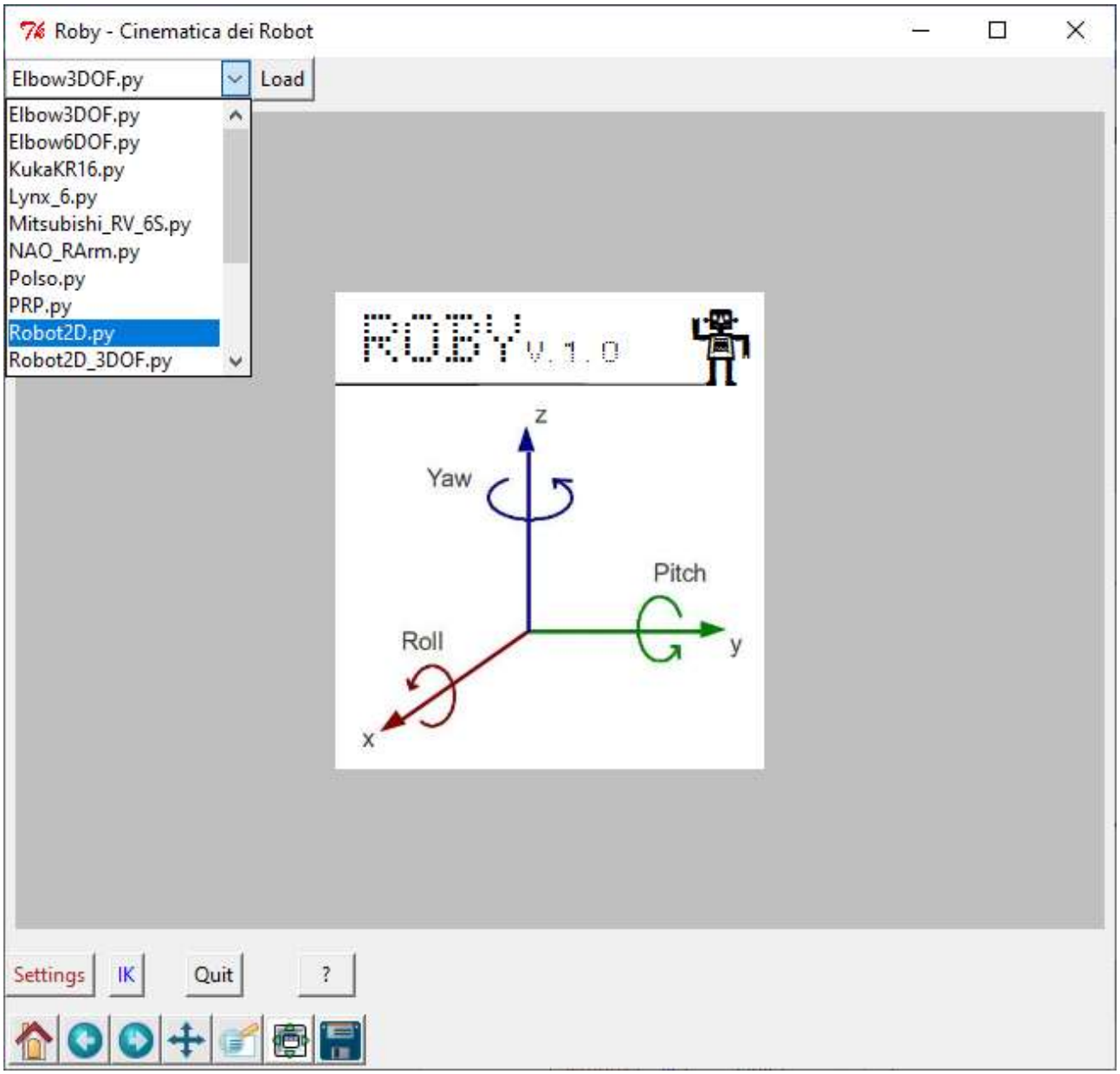
La cartella Robots/myRobots – contiene gli script dei vari Robot che devono seguire le specifiche seguenti:

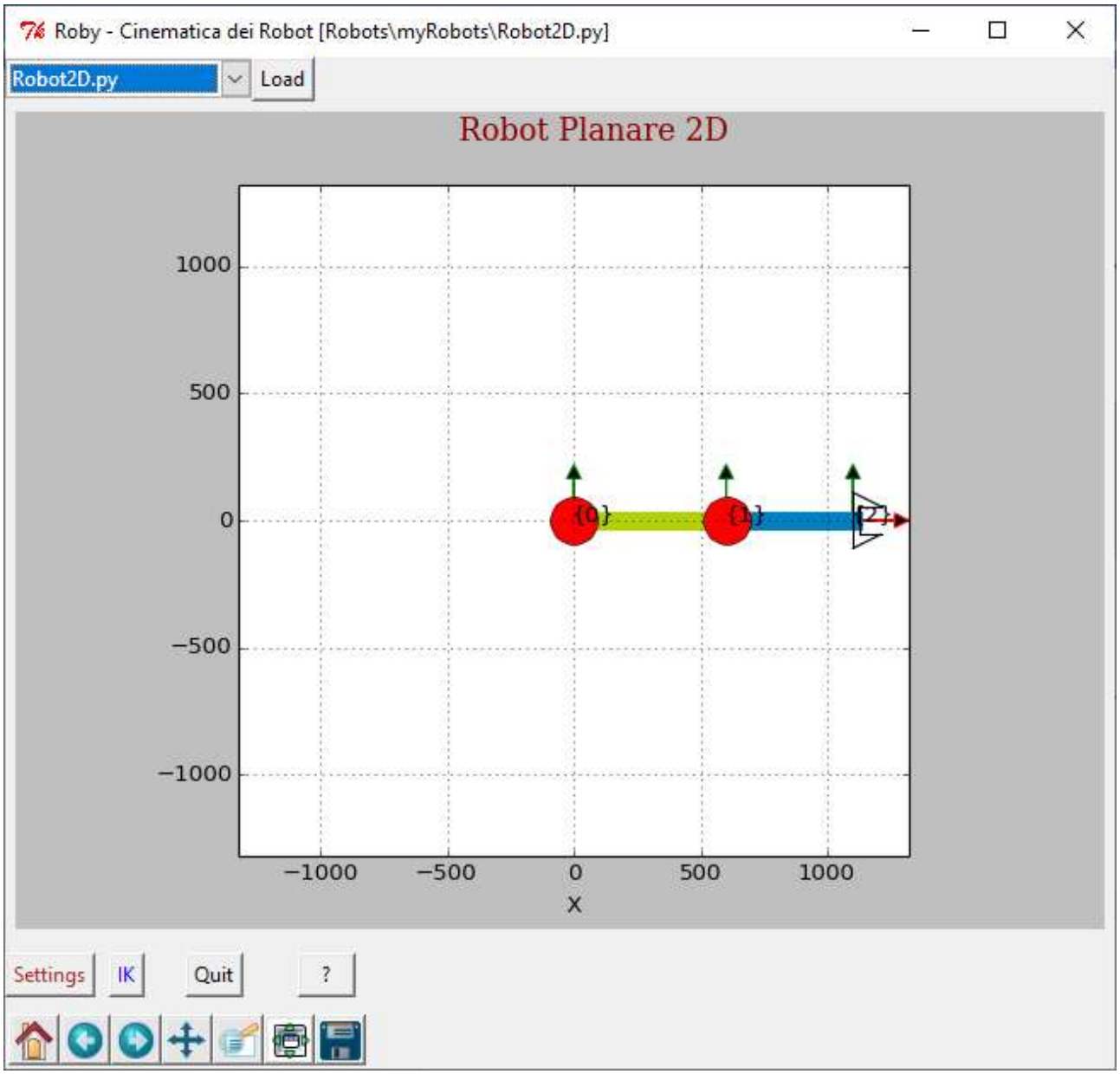
deve essere presente un classe che deriva dalla classe base Robots.Robot e con nome identico al file Python che la contiene. Es file:

Robot2D.py

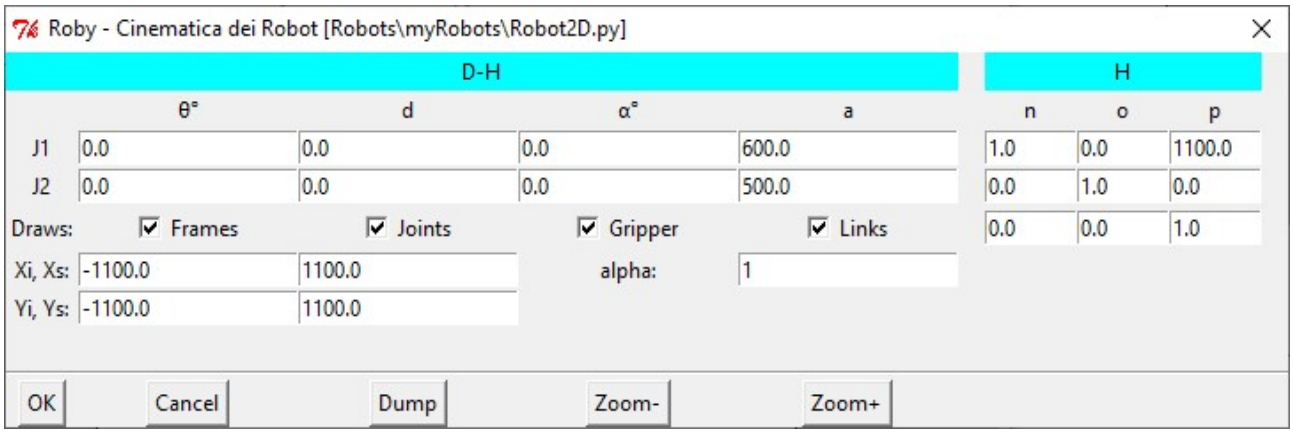
```
import sys
import math
sys.path.append('..')
import Robots
#NB!!!:Il nome della classe deve essere uguale al nome del File .py che la contiene.
class Robot2D(Robots.Robot):##Classe Base
   """
   Robot Planare 2D
   """
   DSpace=2    ##Definisce il tipo ambiente grafico del robot: 2 Planare 2D, 3 Spazio 3D(e' il default)
   def __init__(self):
      Robots.Robot.__init__(self)
      ## Definizione Parametri del robot
      l1=600.0
      l2=500.0

      th1=0;       d1=0;       k1=0;       a1=l1
      th2=0;       d2=0;       k2=0;       a2=l2
      ##Impostazione Matrice D-H
      self.DH=[[th1,d1,k1,a1],
            [th2,d2,k2,a2]]

     ##Esecuzione: Calcoli e Visualizzazione grafica
      self.calcDH()

   ## Impostazione dei tipi di giunti:ROLL,PITCH,YAW,PRYSM
   def DrawJoints(self):
      d=dict()
      d[0]=self.objGD.RJoint.ROLL
```

```
    d[1]=self.objGD.RJoint.ROLL
    super(type(self),self).DrawJoints(dJointsType=d)
```





Il bottone ‘settings’ visualizza la finestra con dati e impostazioni del robot:



Sotto la Label ‘D-H’ viene riportata la Matrice dei Parametri D-H del Robot mentre sotto la Label ‘H’ la matrice di trasformazione Finale:

La riga identificata da ‘Draws’ da la possibilita’ di visualizzare o meno i sistemi di riferimento, i giunti, la pinza e i bracci. Il parametro ‘alpha’ (da 0 a 1) indica la trasparenza degli oggetti disegnati. ‘Xi,Xs’ e ‘Yi,Ys’ indicano i limiti fisici della finestra di visualizzazione. Con il bottone ‘DUMP’ viene visualizzato in dettaglio un riepilogo dei dati per il robot quali le matrici coinvolte piu’ la visualizzazione in formato simbolico della matrice di trasformazione utile per il calcolo della cinematica inversa.

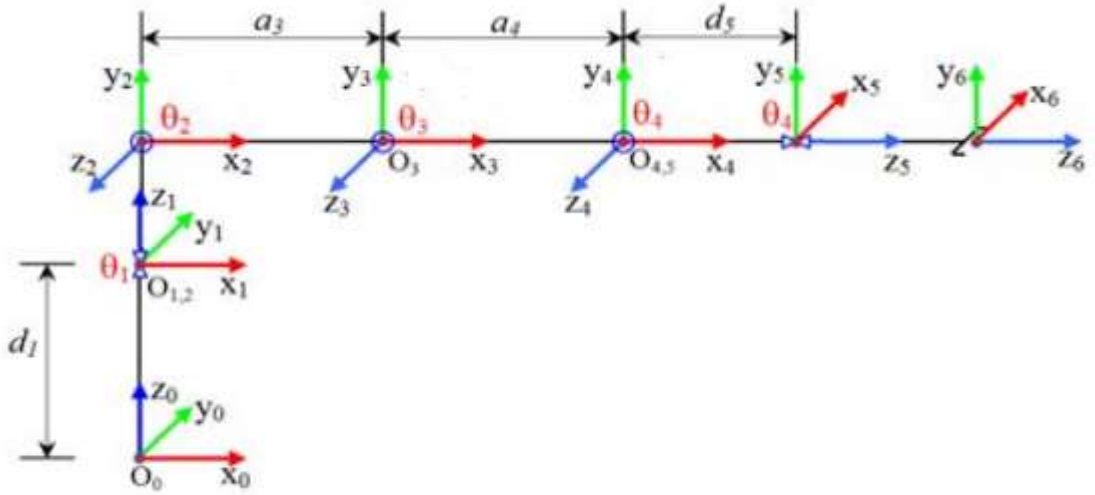
‘Dump’:

```
Robot Planare 2D

*** TRASFORM MATRIX ***
H0:
[1.00 0.00 0.00 0.00
 0.00 1.00 0.00 0.00
 0.00 0.00 1.00 0.00
 0.00 0.00 0.00 1.00 ]
A1:
[1.00 -0.00  0.00 600.00
 0.00  1.00 -0.00   0.00
 0.00  0.00  1.00   0.00
 0.00  0.00  0.00   1.00 ]
A2:
[1.00 0.00 0.00 500.00
 0.00 1.00 0.00   0.00
 0.00 0.00 1.00   0.00
 0.00 0.00 0.00   1.00 ]
H:
[1.00 0.00 1100.00
 0.00 1.00    0.00
 0.00 0.00    1.00 ]
*** SYMBOLIC TRASFORM MATRIX ***
A1:
[   C1   -S1  0.00 600.0*C1
    S1    C1  0.00 600.0*S1
  0.00  0.00  1.00     0.00
  0.00  0.00  0.00     1.00 ]
A2:
[   C2   -S2  0.00 500.0*C2
    S2    C2  0.00 500.0*S2
  0.00  0.00  1.00     0.00
  0.00  0.00  0.00     1.00 ]

H:
[nx ox az pz
 ny oy ay py
 nz oz az pz
 0  0  0  1  ]
nx=-S1*S2 + C1*C2
ny=S1*C2 + S2*C1
nz=0
ox=-S1*C2 - S2*C1
oy=-S1*S2 + C1*C2
oz=0
ax=0
ay=0
```

az=1.00000000000000
px=-500.0*S1*S2 + 500.0*C1*C2 + 600.0*C1
py=500.0*S1*C2 + 600.0*S1 + 500.0*S2*C1
pz=0

Robot Tinkerkit Braccio (5 dof)

Frames Robot Tinkerkit braccio



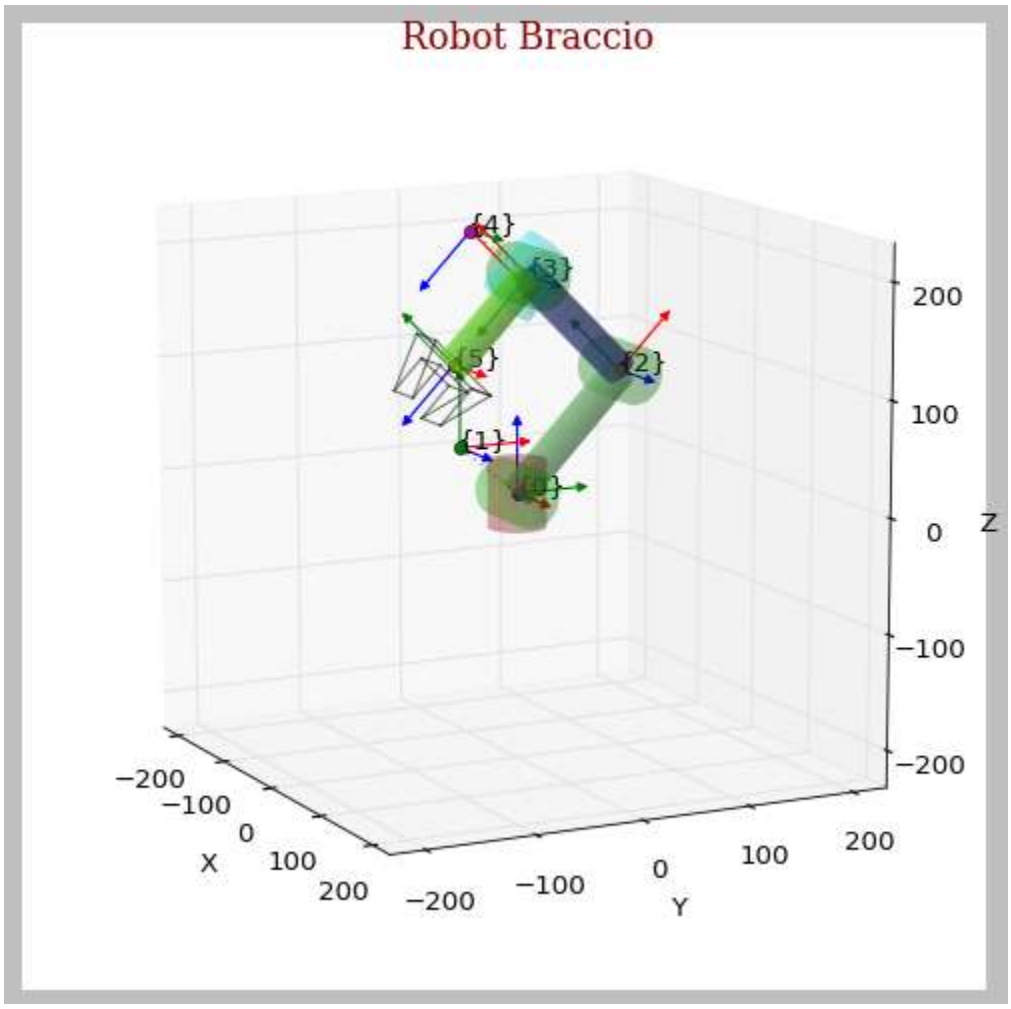
Matrice D-H			
θ	**d**	**α**	**a**
θ_1	d1	90°	0
θ_2	0	0	a2
θ_3–90°	0	0	a3
θ_4	0	90°	0
θ_5	d5	0	0

REALIZZAZIONE IN PYTHON

Dato che la tabella DH contiene una cella con un ‘espressione :-90°+ θ_3 occorre specificarlo nello script racchiudendola dalle doppie virgolette e mettendo l’offset tra parentesi tonde:

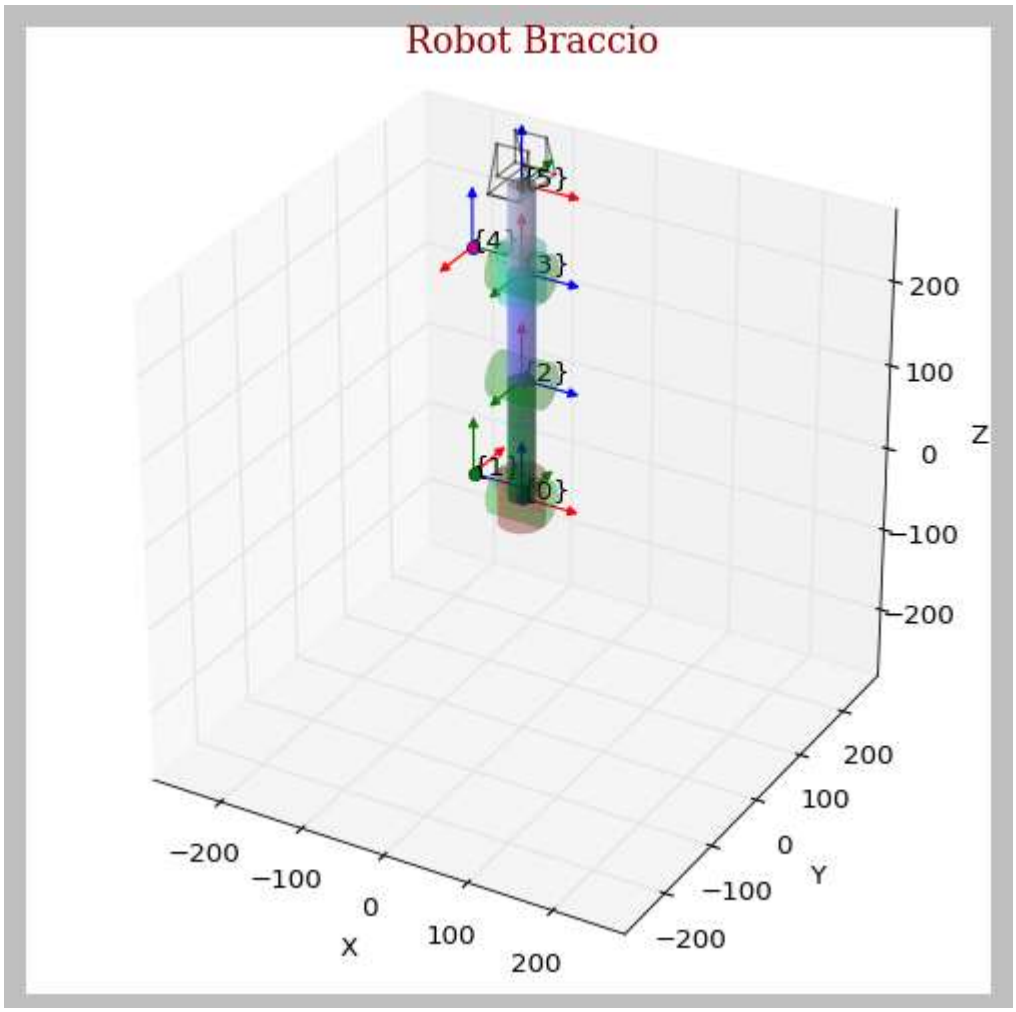
```
def __init__(self):
```

```
    Robots.Robot.__init__(self)
    ###Parametri DH
    d1=60*0
    d5=100
    a2=140
    a3=125
    #safety
    th1 =  90;                    d1= d1;    k1=  90;     a1=0
    th2 =  45;                    d2=  0;    k2=   0;     a2=a2
    th3 =  "(-90)+180";           d3=  0;    k3=   0;     a3=a3
    th4 =  180;                   d4=  0;    k4=  90;     a4=0
    th5 =  90;                    d5=d5;     k5=   0;     a5=0

    self.DH=[[th1,d1,k1,a1],
                    [th2,d2,k2,a2],
                     [th3,d3,k3,a3],
                     [th4,d4,k4,a4],
                     [th5,d5,k5,a5]]

    self.calcDH()

def DrawJoints(self):
    d=dict()
    d[0]=self.objGD.RJoint.ROLL
    d[1]=self.objGD.RJoint.PITCH
    d[2]=self.objGD.RJoint.PITCH
    d[3]=self.objGD.RJoint.PITCH
    d[4]=self.objGD.RJoint.YAW
    super(type(self),self).DrawJoints(dJointsType=d,alpha=0.2)
```



Test 90:

th1 = 90; d1= d1; k1= 90; a1=0
th2 = 90; d2= 0; k2= 0; a2=a2
th3 = "(-90)+90"; d3= 0; k3= 0; a3=a3
th4 = 90; d4= 0; k4= 90; a4=0
th5 = 90; d5=d5; k5= 0; a5=0



RIFERIMENTI

[1] Remo Sala: 'Automazione Industriale e Robotica', Edizioni Cupido 1991

[2] Ghafil,Mohammed,Hadi, "A Virtual Environment for 5-DOF Robot Manipulator based on XNA Framework"