最終課題(解答例) - 標準環境

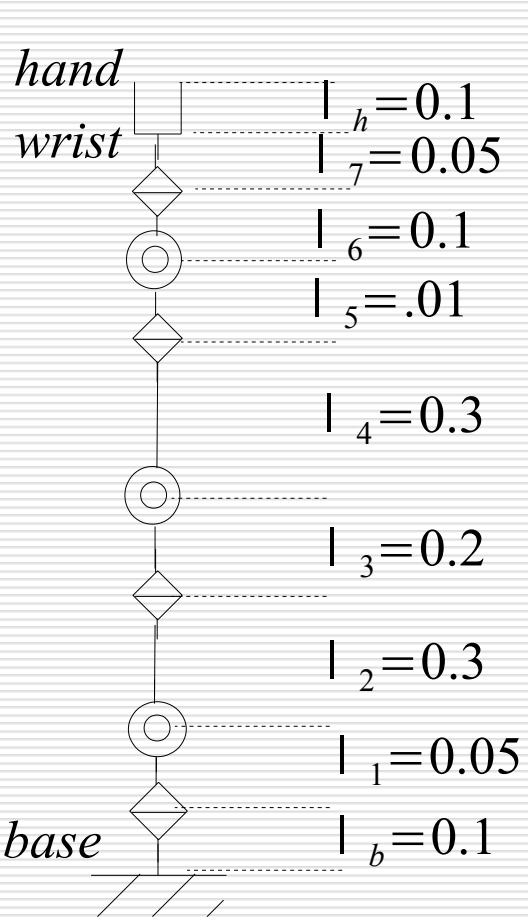
- robo_sys.py を標準環境とする。すなわち
  - room は絶対座標と一致
  - roomから見た table の座標系が、xyzabc=[0, 0, 0.5, 0, 0, pi/6]
  - その他、place_a, place_b, armなどの位置は決まっている。boxは移動する。
  - table のx, y軸はその上面にあり、box のx, y軸がその底面にある。

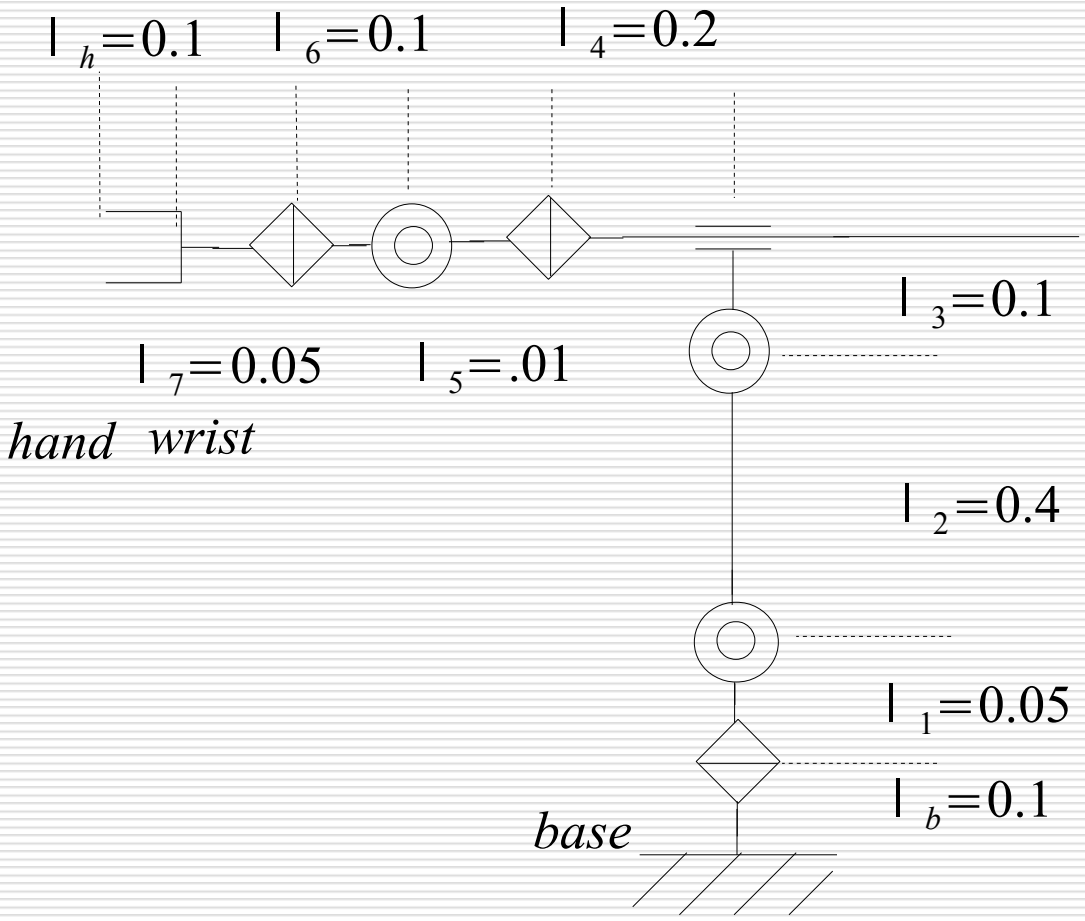
1. 自由作成アーム

- 自由作成アームの関節構造を簡単に説明せよ
- 自由作成アームを、arm学籍番号.pyとする

- robo_sys.pyに自分の作成したアームを取り込みrobo_sys学籍番号.pyとする
- 以降の課題は、robo_sys学籍番号.pyのプログラムとして提出する

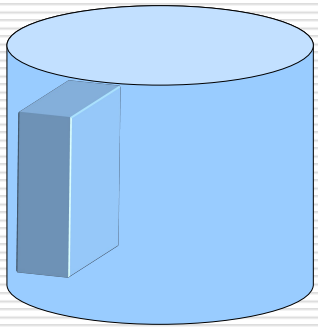
- アームの例

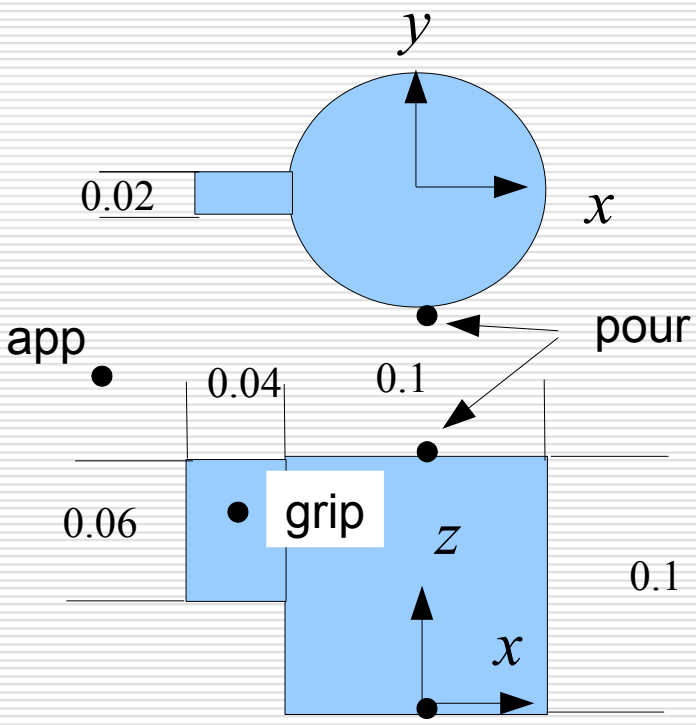

hand
wrist
$l_h=0.1$
$l_7=0.05$
$l_6=0.1$
$l_5=.01$
$l_4=0.3$
$l_3=0.2$
$l_2=0.3$
$l_1=0.05$
base
$l_b=0.1$


$l_h=0.1$
$l_6=0.1$
$l_4=0.2$
$l_3=0.1$
$l_7=0.05$
$l_5=.01$
hand
wrist
$l_2=0.4$
$l_1=0.05$
$l_b=0.1$
base

2. カップを作る

- cupの座標は底面中心
- 形状、寸法は変えてもよい
- grip, grip.app_pos を適切に設定する
- 注ぎ口 pourを設定する
- make_cup()という関数にしておくとよい





- make_cup()の例

```
def make_cup() :
    cup_body=visual.frame()
    a=visual.cylinder(radius=0.05,axis=(0,0,0.1),color=(0.5,0.5,1.0))
    b=visual.box(size=(0.04,0.02,0.06),
                          pos=(-0.07, 0, 0.07), color=(0.5, 1.0, 0.5))
    a.frame=cup_body
    b.frame=cup_body
    cup=PartsObject(vbody=cup_body)
    cup.grip.set_trans(FRAME(xyzabc=[-0.07,0,0.08,0,-pi/6,0]))
    cup.grip.width=0.02
    cup.grip.app_pos.set_trans(FRAME(xyzabc=[0,0,0.05,0,0,0]))
    cup.pour=CoordinateObject()
    cup.pour.affix(cup,FRAME(xyzabc=[0,-0.05,0.1,0,0,0]),'rigid')
    return cup
```

- オブジェクト指向でclassにする

```
class Cup(PartsObject) :
    def __init__(self) :
        cup_body=visual.frame()
        a=visual.cylinder(radius=0.05,axis=(0,0,0.1),
                          color=(0.5,0.5,1.0))
        b=visual.box(size=(0.04,0.02,0.06),
                          pos=(-0.07,0,0.07),color=(0.5,1.0,0.5))
        a.frame=cup_body
        b.frame=cup_body
        PartsObject.__init__(self, vbody=cup_body)
        self.grip.set_trans(FRAME(xyzabc=[-0.07,0,0.08,0,-pi/6,0]))
        self.grip.width=0.02
        self.grip.app_pos.set_trans(FRAME(xyzabc=[0,0,0.05,0,0,0]))
        self.pour=CoordinateObject()
        self.pour.affix(self,FRAME(xyzabc=[0,-0.05,0.1,0,0,0]),'rigid')
```

3. カップを環境に取り込む

robo_sys学籍番号.pyにて

- boxをなくす。
- cup_a, cup_bを作り、place_a, place_bに配置する。

- 最後の課題で必要ならば、place_a, place_bの位置・姿勢を変更してもよい

3のプログラム例

```
cup_a=make_cup()
cup_a.affix(table,place_a.where(table))
cup_b=make_cup()
cup_b.affix(table,place_b.where(table))
```

4. 注ぎ作業の実現

robo_sys学籍番号.pyにて

do_task(arm,cup_a,cup_b,place_a,table)

で注ぎ作業を実行させる。

4のプログラム例　pour

```
def pour(arm,a,b) :
    pour_app=FRAME(xyzabc=[0,0,0.15,0,0,pi])
    pour_start=FRAME(xyzabc=[0,0,0.01,-pi/12,0,pi])
    pour_end=pour_start*FRAME(xyzabc=[0,0,0,pi/2,0,0])
    arm.move(a.pour,pour_app,b.pour)
    arm.move(a.pour,pour_start,b.pour)
    arm.move(a.pour,pour_end,b.pour)
    arm.move(a.pour,pour_start,b.pour)
    arm.move(a.pour,pour_app,b.pour)
```

- cupの上，ぶつからな位置からアプローチ
- 僅かに傾けたところを開始姿勢とする
- pi/2までしっかり傾ける
- 0.15など，必ずしも汎用性があるとは言えない謎の定数があるのは改善の余地がある．

4のプログラム例 do_task

```
def do_task(arm, c1, c2, place, fix=None) :
    pick_up(arm,c1)
    pour(arm,c1,c2)
    place_down(arm,c1,place, fix)
```

- pourがちゃんと書けていればプログラムは簡単
- arm, c1, c2 ,,,, など，引数を変えても動くようにする

- 実際にはアームの可動範囲や障害物の問題があり，こんなに単純ではないことが多い．
- place_downで，affixする対象を指定するのは，わりと煩わしい．できれば改善したいところ．

課題のチェックポイント

- アームの関節構造が別紙の宣言どおりになっているか.
- make_cupまたはCupクラスは，汎用的にいくつもカップを生成出来るようになっているか
- do_taskはアームや対象を変えても動くように汎用的につくられているか
- 簡単なチェックとしては

  >>> do_task(arm, cup_a, cup_b, place_a, table)

  に続けて，

  >>> do_task(arm, cup_b, cup_a, place_b, table)

  としてもちゃんと動くか？
- 座標系の管理が出来ているか

  作業終了後，以下をやってみる.

  >>> table.set_pos(FRAME())

  さらに，もう一度

  >>> do_task(arm, cup_a, cup_b, place_a, table)

  としたらどうなるか？