知能システム論1（15）

2012.7.18

情報システム学研究科
情報メディアシステム学専攻
知能システム学講座
末廣尚士

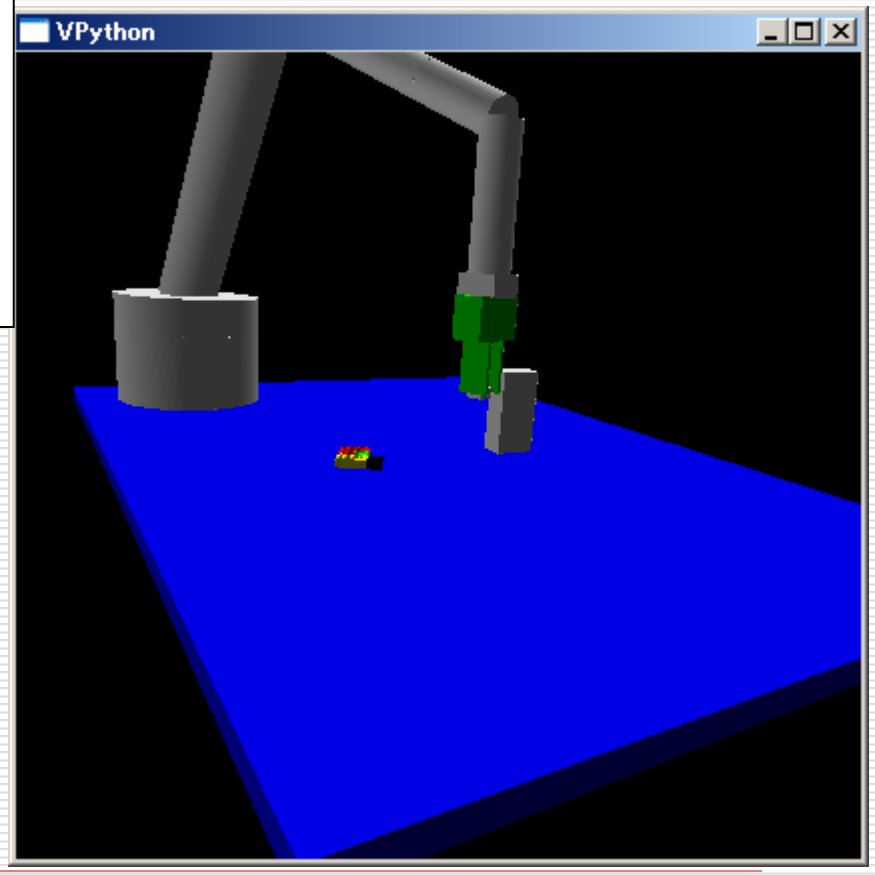
- 作業プログラムの作成

- PartsObjectへの属性の追加
  - Gripオブジェクト(把持点情報):
    把持点、指幅、アプローチ点
- 作業、作業動作のモデル化
  - 作業の分解
  - 作業の記述
- 作業スキル

- 作業プログラムの作成(実行例)

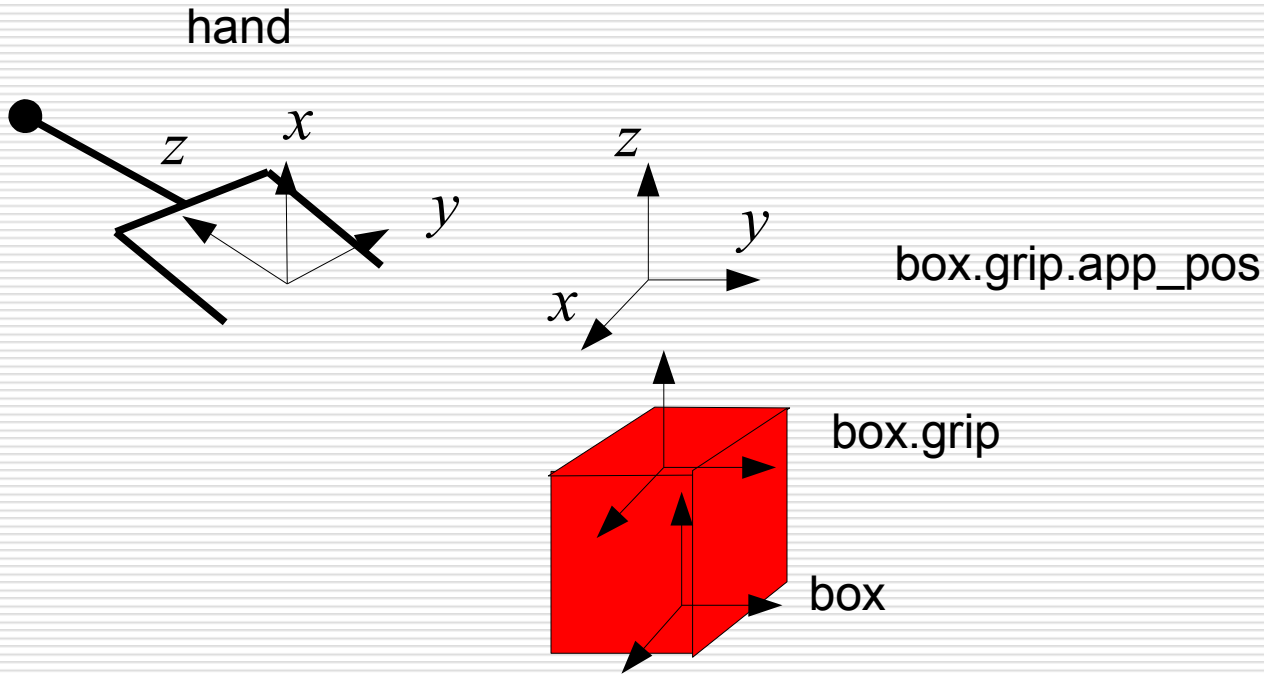
- lego ブロック遊び(robo_sys.py)

```
>>> create_env()
>>> a=make_pyramid_data(3)
>>> build_lego_obj(arm,place_b,feeder,a)
>>>
```



- grip：把持点

- ハンドが持つ場所、アプローチする場所を決めておく。



- 作業プログラムの作成

- PartsObject

```
class PartsObject(CoordinateObject) :
    def __init__(self, vbody=None) :
        CoordinateObject.__init__(self)
        if vbody :
            self.set_vbody(vbody)
        self.grip=Grip(self)
    def set_vbody(self, vbody):
        self.vbody = vbody
        self.vbody.frame = self.vframe
```

- 作業プログラムの作成

- Grip

```
class Grip(CoordinateObject) :
    def __init__(self, mama, trans=None) :
        CoordinateObject.__init__(self)
        self.app_pos=CoordinateObject()
        self.app_pos.affix(self,FRAME(xyzabc=[0,0,0.05,0,0,0]))   #default
        self.width = 0.0
        if not trans :
            trans = FRAME()
        self.affix(mama,trans)
```

- 作業プログラムの作成

- create_env

```
box_body=visual.box(width=0.1, length=0.05, height=0.03)
box_body.pos=(0,0,0.05)
box = PartsObject(vbody=box_body)
box.grip.set_trans(FRAME(xyzabc=[0,0,0.08,0,0,0]))
box.grip.width = 0.03
box.grip.app_pos.set_trans(FRAME(xyzabc=[0,0,0.1,0,0,0]))
box.affix(table,place_a.where(table))
```

- 作業プログラムの分解

- pick_up

```
def pick_up(arm,obj) :
    arm.move("hand",up100,obj.grip.app_pos)
    arm.move("hand", obj.grip.app_pos)
    arm.hand.open(obj.grip.width+0.01)
    arm.move("hand", obj.grip)
    arm.hand.close(obj.grip.width)
    obj.unfix()
    obj.affix(arm.hand)
    arm.move(obj,up200,obj)
```

- 作業プログラムの分解

- place_down

```
def place_down(arm,obj,place,fix=None) :
    arm.move(obj,up200,place)
    arm.move(obj,place)
    arm.hand.open(obj.grip.width+0.01)
    obj.unfix()
    if fix :
        obj.affix(fix)
    arm.move("hand", obj.grip.app_pos)
    arm.hand.close(0)
    arm.move("hand",up100,obj.grip.app_pos)
```

- 作業プログラムの例

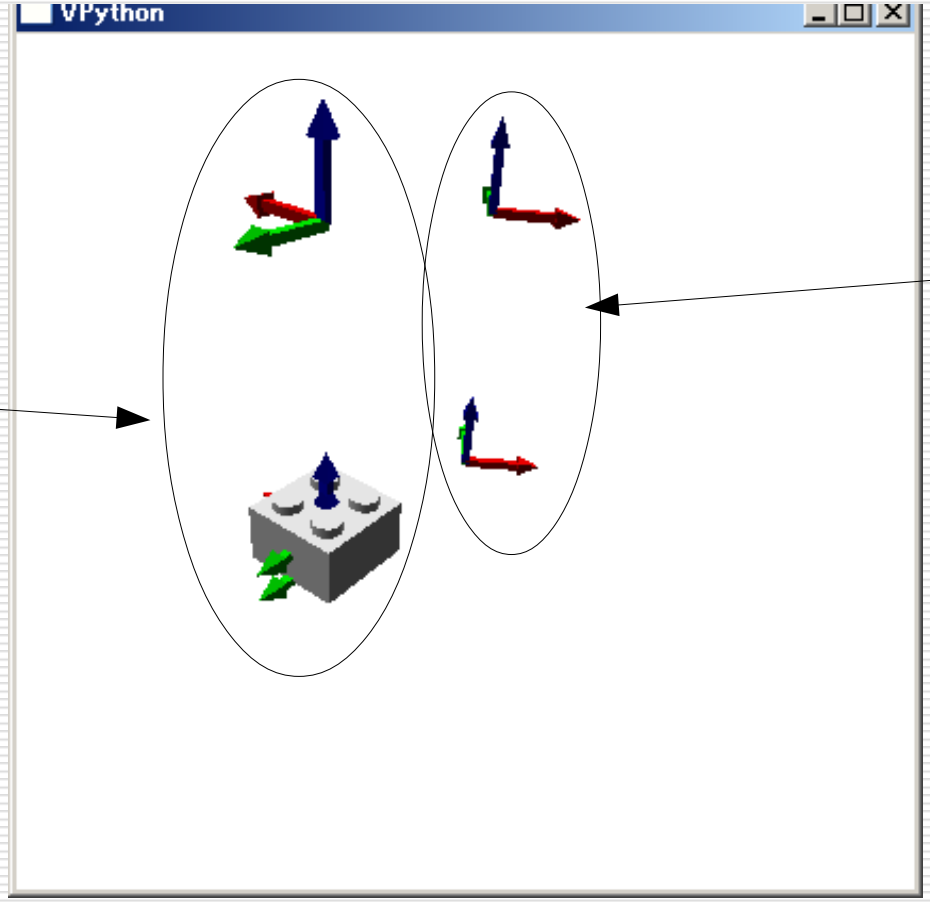
- do_task()

```
def do_task(arm, obj, pl, fix=None) :
    pick_up(arm,obj)
    place_down(arm,obj,pl,fix)
```

```
>>> do_task(arm, box, place_b, table)
```

- Lego (l_block.py)

- LegoBlockとBlockFeeder



LegoBlock

BlockFeeder

- Lego (l_block.py)

- LegoBlock

```
class LegoBlock(PartsObject) :
    unit = 0.0254/16 # 1/16 inch
    def __init__(self, size=[2,2,1], color=None) :
        PartsObject.__init__(self)
        self.vframe.resize(LegoBlock.unit*2)
        tmpf=visual.frame()
        self.size=size[:]
        for i in range(3) :
            self.size[i]=int(self.size[i])
        b_size=[self.size[0]*LegoBlock.unit*5, self.size[1]*LegoBlock.unit*5,
                    self.size[2]*LegoBlock.unit*6]
        if color :
            self.color = color
        else :
            self.color = (1,1,1)
```

- Lego (l_block.py)

- LegoBlock

```
box=visual.box(size=b_size,color=self.color,z=b_size[2]/2)
box.frame=tmpf
for i in range(self.size[0]) :
    for j in range(self.size[1]) :
        tmp = visual.cylinder(axis=(0,0,LegoBlock.unit),
              radius=LegoBlock.unit*1.5, color=self.color)
        tmp.z=b_size[2]
        tmp.x=(5*i+2.5)*LegoBlock.unit-b_size[0]/2
        tmp.y=(5*j+2.5)*LegoBlock.unit-b_size[1]/2
        tmp.frame=tmpf
self.set_vbody(tmpf)
self.grip.width = b_size[1]
self.grip.set_trans(FRAME(xyzabc=[0,0,b_size[2]-0.005,0,0,0]))
```

- Lego (l_block.py)

- BlockFeeder

```
class BlockFeeder(PartsObject) :
    color=[visual.color.black, visual.color.white, visual.color.red,
           visual.color.green, visual.color.blue, visual.color.yellow]
    def __init__(self) :
        PartsObject.__init__(self)
        self.area =0.05
    def feed(self,col=None) :
        pos_x = self.area*random.uniform(-1,1)
        pos_y = self.area*random.uniform(-1,1)
        ori = pi*random.uniform(-1,1)
        r_frame=FRAME(xyzabc=[pos_x, pos_y, 0, 0, 0, ori])
        if not col :
            col_len=len(self.color)
            col=BlockFeeder.color[int(col_len*random.random())]
        blk = LegoBlock(color=col)
        blk.affix(self, r_frame)
        return blk
```

- feederの設定

- create_env()

```
feeder=BlockFeeder()
feeder.affix(table, FRAME(xyzabc=[0,-0.25,0,0,0,0]))
```

- レゴ組み立てデータの作成

- レゴブロックの組み立てデータ
  - 組み立てられるブロックの位置を[x,y,z]のlistで表現。
  - x,yは5[lego_unit]、zは6[lego_unit]

```
>>> a=make_pyramid_data(3)
>>> a
[[-2, -2, 0], [-2, 0, 0], [-2, 2, 0], [0, -2, 0], [0, 0, 0], [0, 2, 0], [2, -2, 0],
[2, 0, 0], [2, 2, 0], [-1, -1, 1], [-1, 1, 1], [1, -1, 1], [1, 1, 1], [0, 0, 2]]
```

- ピラミッドのデータ作成

- make_pyramid_data()

```
def make_pyramid_data(n) :
    if n == 1 :
        return [[0,0,0]]
    else :
        data =[]
        for i in range(n) :
            for j in range(n) :
                data.append([2*i +1-n, 2*j+1-n,0])
        rest= make_pyramid_data(n-1)
        for dd in rest :
            dd[2]+=1
        return data + rest
```

- レゴ組み立てプログラム

- build_lego_obj()
- レゴ組み立てデータを、実際の場所に対応させる。
- レゴブロックをフィーダーから1つずつ取り出す。
- pick_upとplace_downを

- レゴ組み立てプログラム

- build_lego_obj()

```
def build_lego_obj(arm, place,feeder, data) :
    d_pos=CoordinateObject()
    d_pos.affix(place)
    for dd in data :
        d_pos.set_trans(FRAME(vec=[dd[0]*LegoBlock.unit*5,
                dd[1]*LegoBlock.unit*5,dd[2]*LegoBlock.unit*6]))
        bb = feeder.feed()
        pick_up(arm,bb)
        place_down(arm, bb, d_pos, place)
```

- 表示の微調整

- create_env()

```
AxesXYZ.visible_all(False)
visual.scene.center=table.where().vec
visual.scene.autoscale=False
visual.scene.scale=(2.0,2.0,2.0)
arm.ready()
```

- 例題5

- legoブロックを使って自分なりに何か作ってみる。

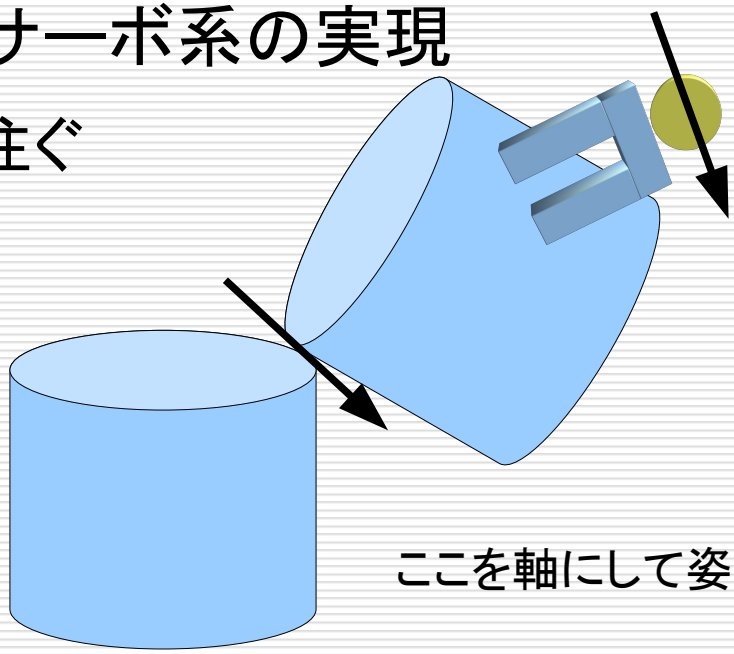
-例題5のヒント

- pyramidでは縦横10unit、高さ6ユニットのブロックを並べることを考えて、縦横5unit、高さ6unitを基準にして、置くべきブロックの位置を[x,y,z]のリストで表現している。
- build_lego_objでは、上記リストデータを与えるとpyramidをくみ上げる。
- データだけ作って、build_lego_objを使っても良い。
- feederでは、色を指定することが出来る。
- feederを使わずに、LegoBlockを直接作れば、ブロックの大きさも自由に出来る。

- もっと器用な作業をする(スキル)

- 作業座標系
  - 作業動作を記述するための座標系
  - 力制御や動作の座標系を作業に応じて分かりやすく表現する
  - サーボ系の実現

例:水を注ぐ



ビデオ

- もっと器用な作業をする(スキル)

- 力制御
  - 指定された力で押し付ける
  - 接触の維持
- たとえば、

  press('z', -3.0)

ビデオ

- もっと器用な作業をする(スキル)

- 突き当て
  - 指定された方向で点接触を実現する
- たとえば、

  move_to_touch('x', 5.0)

ビデオ

- もっと器用な作業をする(スキル)

- 辺合わせ
  - 指定された軸まわりで辺接触を実現する
- たとえば、

  rotate_to_level('y', 0.05, 'z', -3.0)

ビデオ

- もっと器用な作業をする(スキル)

- はめ合い
  - 指定された軸まわりで回転させ、はめ合いを実現する
- たとえば、

  rotate_to_inset('x', 0.05, 'y', -3.0)

ビデオ

- もっと器用な作業をする(スキル)

ビデオ

- 四角柱のはめ合い
  - スキルの組み合わせ
    - touch('z', -10.0)
    - press('z', -5.0)
    - move_to_touch('x', -10.0)
    - press('x', -5.0)
    - rotate_to_level('z', 0.05, 'x', -5.0)
    - move_to_touch('y', -10.0)
    - press('y', -5.0)
    - rotate_to_insert('x', 0.05, 'z', -5.0)
    - rotate_to_insert('y', 0.05, 'z', -5.0)

- 次回予告

- Rtミドルウェア
- 最終課題解答解説