知能システム論1 (8)

2012.6.6

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

10. 描画モデルと座標系モデルの統合

visual.frameとCoordinateObjectの統合を行う(object_model_v.py)。

- 形状を座標系で管理するために、形状全体を管理するframeを対応するCoordinateObject に持たせる。
- 親子関係はそれぞれ一致させる。
- 原点はもそれぞれ一致させる。
- frameの姿勢に関してはCoordinateObjectで管理する。

上記の整合性を常に注意する。

- 表示用座標軸を加える

```
class CoordinateObject :
    def __init__(self) :
        self.parent = None
        self.rel = None
        self.rel_trans = FRAME()
        self.vframe=AxesXYZ()

    def where(self,wrt=None) :
        if self.parent :
            my_trans= self.parent.where() * self.rel_trans
        else :
            my_trans= self.rel_trans
        if wrt :
            wrt_trans=wrt.where()
            my_trans =(- wrt_trans) * my_trans
        return my_trans
```

- 描画モデルのframeの連結

```
def affix(self, mama, rel_trans=None, rel=None) :
    if self.parent :
        print "error! no more parent."
    else :
        if rel_trans:
            self.set_trans(rel_trans)
        else :
            self.set_trans(self.where(mama))
        self.parent = mama
        self.vframe.frame = mama.vframe
        self.rel = rel
def unfix(self) :
    tmp = self.where()
    self.parent = None
    self.vframe.frame = None
    self.rel = None
    self.set_trans(tmp)
```

- 描画frameの移動と座標変換

```
def set_pos(self, pos) :
    if self.rel :
        print "error! cannot set_pos."
    elif self.parent :
        tmp =( - self.parent.where()) * pos
        self.set_trans(tmp)
    else :
        self.set_trans(pos)

def set_trans(self, trans) :
        tmp = trans
        tmp_f = (- self.rel_trans) * tmp
        tmp_axis=tmp_f.mat.rot_axis()
        tmp_axis[1]=self.rel_trans.mat * tmp_axis[1]
        self.vframe.rotate(axis=tmp_axis[1],angle=tmp_axis[0])
        self.vframe.pos=visual.vector(tmp.vec)
        self.rel_trans = tmp
```

- これは変更なし

```
def ancestor(self, coord) :
    if self == coord :
        return True
    elif not self.parent :
        return False
    elif (coord == self.parent) :
        return True
    else :
        return self.parent.ancestor(coord)
```

- 物体の形状の導入

```
class PartsObejct(CoordinateObject) :
    def __init__(self, vbody=None) :
        CoordinateObject__init__(self)
        self.vbody = vbody
        self.vbody.frame = self.vframe
```

- ArmWithHand

```
class ArmWithHand :
    def __init__(self, base_pos=None,hand=None,init_pos=None) :
        self.base = CoordinateObject()
        if base_pos :
            self.base.set_pos(base_pos)
        if not init_pos :
            init_pos=FRAME()
        self.init_pos=init_pos
        self.wrist = CoordinateObject()
        self.wrist.affix(self.base,init_pos,'rigid')
        self.tools={}
        self.hand=hand
        if hand :
            self.tools['hand']=hand
            hand.affix(self.wrist,hand.rel_trans,'rigid')
```

- ArmWithHand：sleepを入れる

```
def move(self, smthng, smwhr, wrt=None) :
    if smthng.__class__ == str :
       smthng = self.tools[smthng]
    if smthng.ancestor(self.wrist) :
       if smwhr.__class__ == CoordinateObject :
          target=smwhr.where(self.base)
       elif smwhr.__class__ == FRAME :
          if wrt :
             target = wrt.where(self.base) * smwhr
          else :
             target = (- self.base.where()) * smwhr
       else :
          print "error: wrong place expression"
          return False
       self.wrist.set_trans(target*(- smthng.where(self.wrist)))
       time.sleep(1)
    else :
      print "error: cannot move unfixed object"
      return False
```

- Handにもsleepを入れる

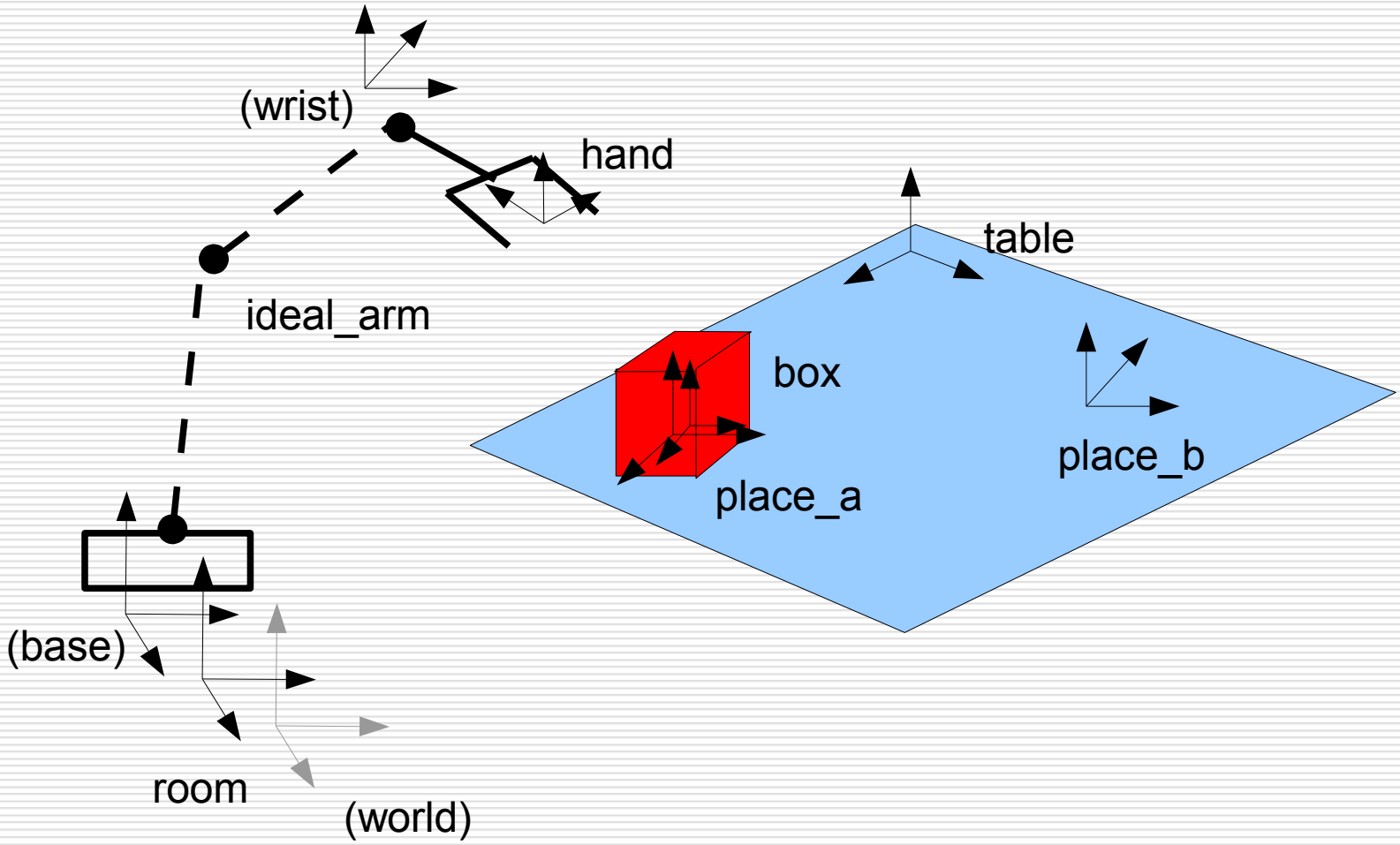
```
class Hand(PartsObject) :
    def __init__(self, trans=None) :
        if not trans :
            trans=FRAME()
        PartsObject.__init__(self)
        if trans :
            self.set_pos(trans)
        self.width=0.0
    def open(self, width) :
        self.widht = width
        time.sleep(1)
        print "open"
    def close(self, width) :
        self.widht = width
        time.sleep(1)
        print "close"
    def width(self) :
        return self.width
```

- 環境構築例

env_model_v_draft1.py

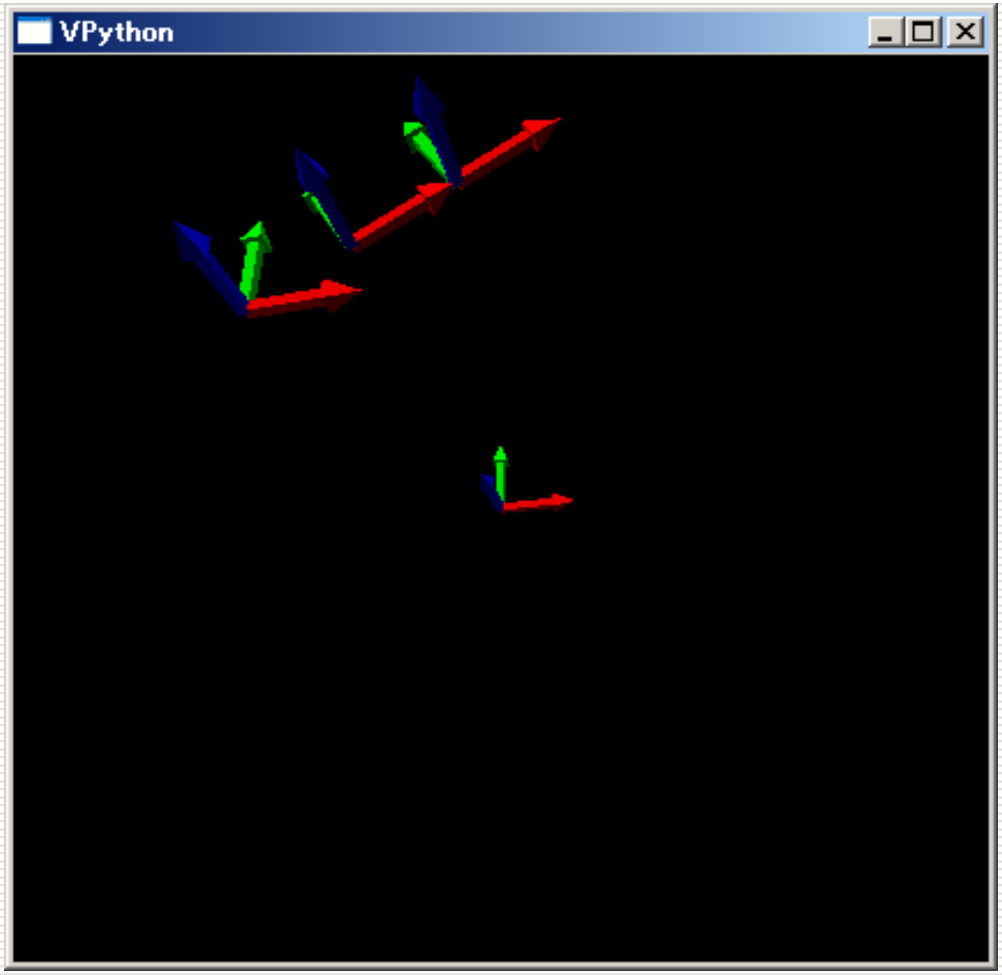
```
from object_model_v import *
room = CoordinateObject()
table = PartsObject()
table.affix(room,FRAME(xyzabc=[0,0,0.5,0,0,pi/6]))
place_a = CoordinateObject()
place_a.affix(table,FRAME(xyzabc=[0.1,0,0,0,0,0]))
place_b = CoordinateObject()
place_b.affix(table,FRAME(xyzabc=[-0.1,0,0,0,0,-pi/6]))
box = PartsObject()
box.affix(table,place_a.where(table))
hand = Hand()
ideal_arm = ArmWithHand(None, hand)
```

- 構築される環境


(wrist)
hand
ideal_arm
table
box
place_b
place_a
(base)
room
(world)

- 環境構築結果

- よくわからない。



- 環境に物体形状を付ける

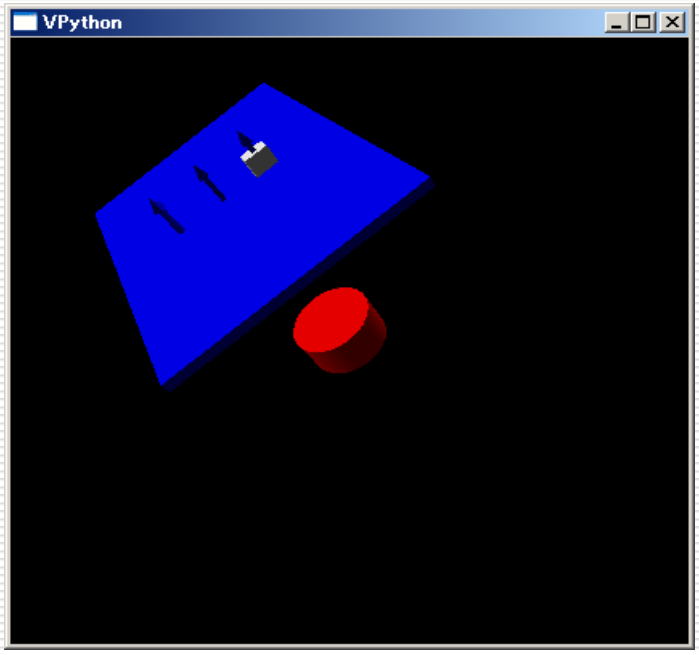
env_model_v_draft2.py

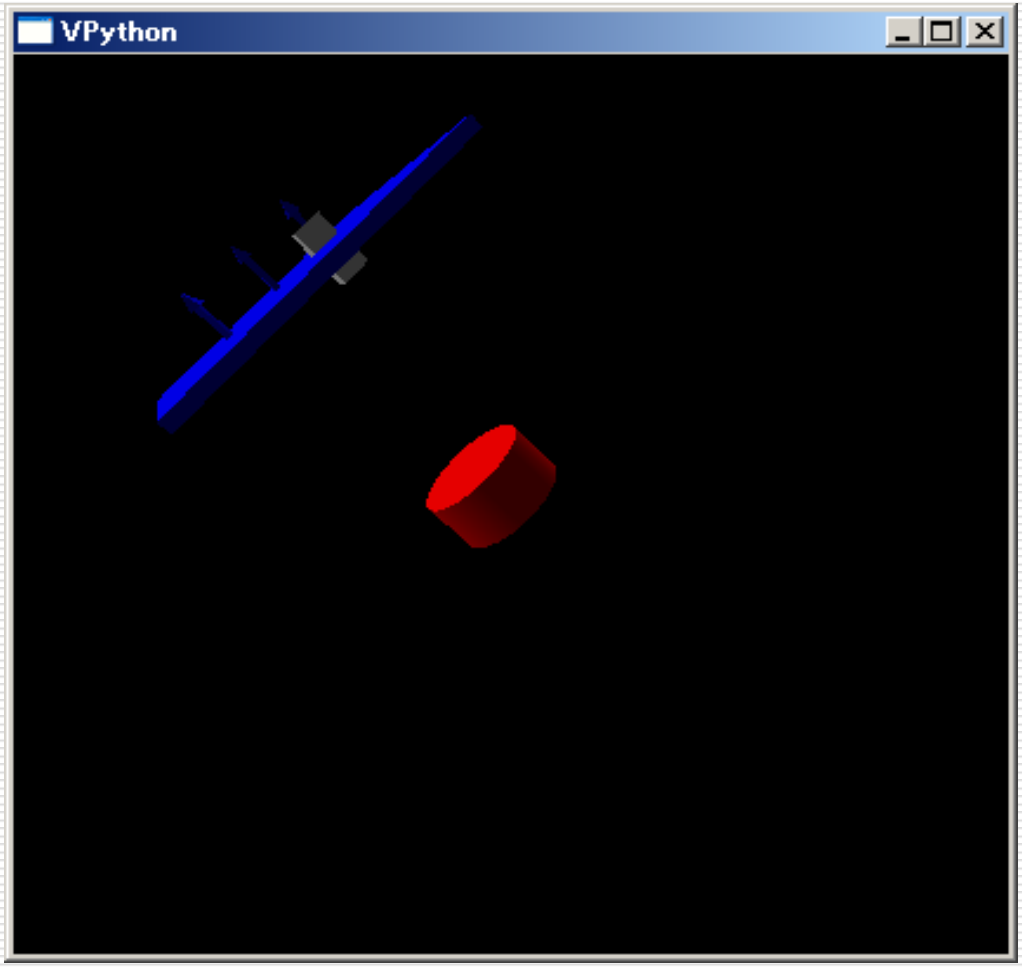
```
from object_model_v import *
room = CoordinateObject()
table_body=visual.box(color=visual.color.blue,width=0.02,length=0.5,
height=0.7)
table = PartsObject(vbody=table_body)
table.affix(room,FRAME(xyzabc=[0,0,0.5,0,0,pi/6]))

box_body=visual.box(width=0.1, length=0.05, height=0.03)
box = PartsObject(vbody=box_body)
box.affix(table,place_a.where(table))
hand_body=visual.sphere(radius=0.03, color=visual.color.green)
hand = Hand()
hand.set_vbody(hand_body)
base_body=visual.cylinder(axis=(0,0,0.1),radius=0.1,color=visual.colo
r.red)
ideal_arm = ArmWithHand(None, hand)
ideal_arm.base.set_vbody(base_body)
```

- 形状付き環境構築結果

- なんか変？
- 位置が微妙におかしい。





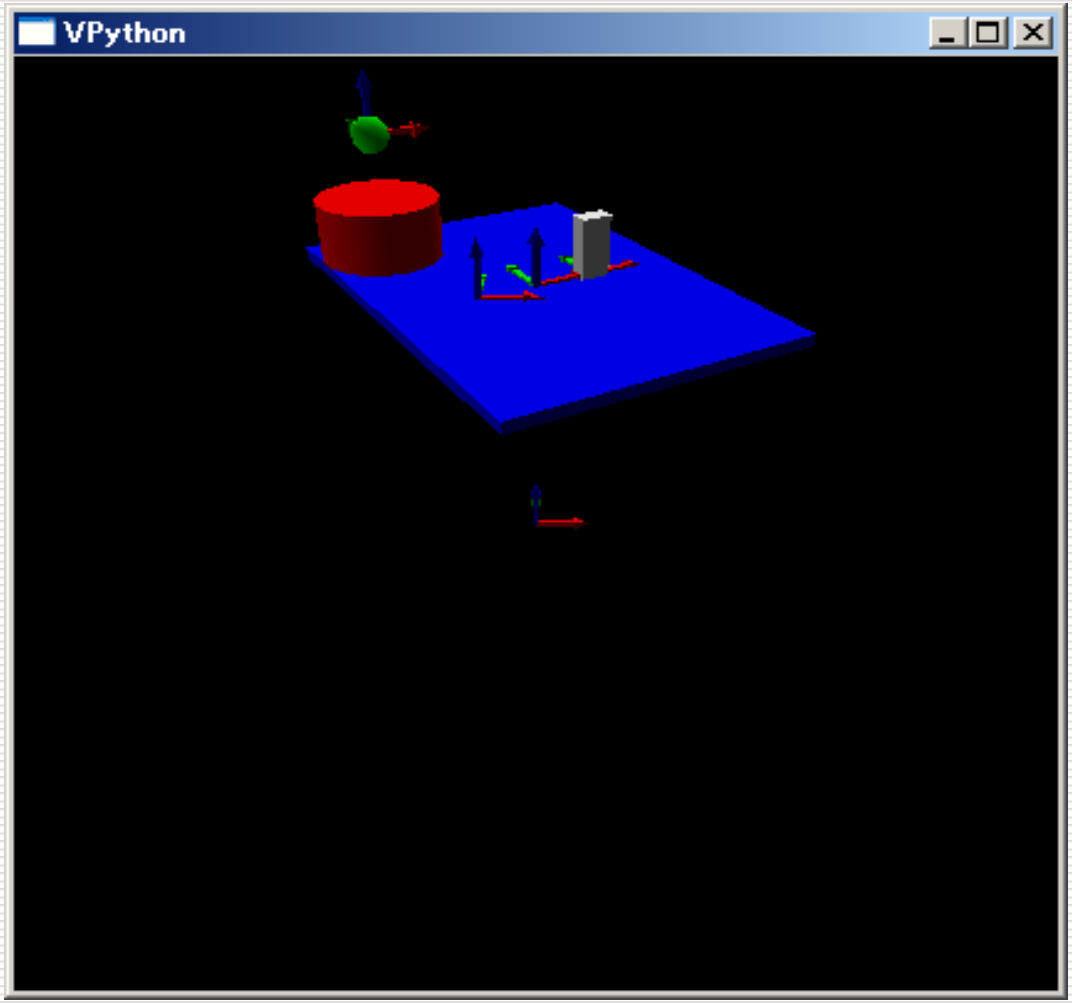
- 物体形状の位置を正しくする(1)

env_model_v.py

```
def create_env() :
    global room, table, place_a, place_b, box, hand, ideal_arm,
up200
    room = CoordinateObject()
    table_body=visual.box(color=visual.color.blue,
width=0.02,length=0.5,height=0.7)
    table_body.pos=(0,0,-0.01)
    table = PartsObject(vbody=table_body)
                :
    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.affix(table,place_a.where(table))
                :
```

- 形状位置の修正結果

- OK
-



- 作業プログラムを作ってみる

env_model_v.py

```
def do_task(arm) :
    arm.move('hand',up200,box)
    arm.hand.open(0.1)
    arm.move('hand',box)
    arm.hand.close(0)
    box.unfix()
    box.affix(arm.hand)
    arm.move(box,up200,box)
    arm.move(box,up200,place_b)
    arm.move(box,place_b)
    arm.hand.open(0.1)
    box.unfix()
    box.affix(table)
    arm.move('hand',up200,place_b)
```

- 環境をリセットする

env_model_v.py

```
def reset_env() :
    table.unfix()
    table.affix(room,FRAME(xyzabc=[0,0,0.5,0,0,pi/6]))
    box.unfix()
    box.affix(table,place_a.where(table))
```

- 場所を次々指していく

env_model_v.py

```
def point_places(arm, place_list):
    for pp in place_list:
        arm.move('hand',up200,pp)
        print pp
```

- 使ってみる

```
>>> create_env()
>>> do_task(ideal_arm)
open
close
open
>>> reset_env()
>>> do_task(ideal_arm)
open
close
open
>>> point_places(ideal_arm, [box, place_a, place_b, table])
<object_model_v.PartsObject instance at 0x0344A2D8>
<object_model_v.CoordinateObject instance at 0x03442D78>
<object_model_v.CoordinateObject instance at 0x0344A1E8>
<object_model_v.PartsObject instance at 0x03442D00>
>>>
```

- いくつかの問題点

- 持つところが変？
  - 物体ごとに持つ場所を持たせる
  - 持つところを自動計算する => 把持計画
- いちいちunfix, affixが煩わしい
  - grasp, ungrasp 命令でまとめる

- ロボットアームの本体がない
  - これからの講義で

- 次回以降の予告

- アーム本体の座標計算
- 手先はどこにあるか？
- 目標の場所に持っていくにはどうしたらよいか？