エルゴノミクスコンピューティング実習

03 演習課題

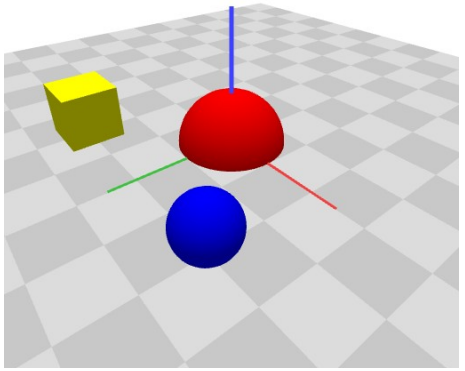
人間システム工学科 井村 誠孝
m.imura@kwansei.ac.jp

ex03_0 テンプレートのダウンロード

- カメラ位置，XY平面，ワールド座標系の座標軸を表示するテンプレート ex03_0 をダウンロードし，実行を確認する．

ex03_1 任意の位置に物体を表示

- RGB＝(255,0,0)で半径20の球体を，中心が位置(0,0,0)になるように表示する．
  - 半径20の球体は`sphere(20)`で描画できる．
- RGB＝(0,0,255)で半径10の球体を，中心が位置(40,40,10)になるように表示する．
- RGB＝(255,255,0)で一辺20の立方体を，中心が位置(-40,40,10)になるように表示する．
  - 一辺20の立方体は`box(20)`で描画できる．

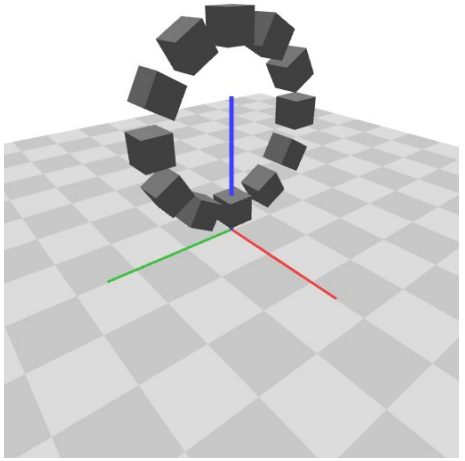


実行結果

ex03_2 並進と回転

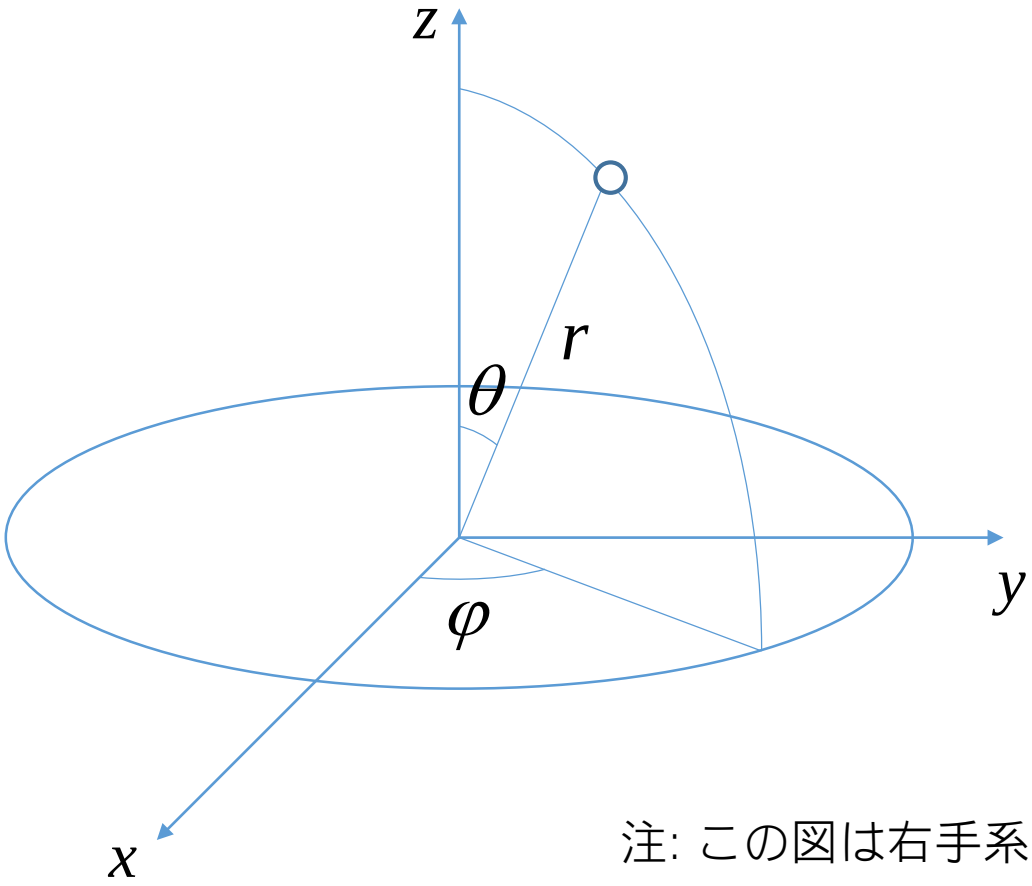
- 点(0,0,40)を中心としたYZ平面上の半径30の円を考え，その円周上を12個の立方体(一辺の長さは10，色は任意)を1周10秒で回転させる．
- バリエーションを考えて，格好よいアニメーションを目指す．
  - 色を変える．
  - 視点を変える．
  - X軸方向に数を増やす．螺旋状にする．
  - 各立方体を自転させる．
  - その他何でも．

実行結果



ex03_3 極座標

- 3次元空間における極座標とは，点の位置を動径 r と二個の偏角 θ, φ で表す方法.



注: この図は右手系

$$x = r\sin\theta\cos\varphi$$

$$y = r\sin\theta\sin\varphi$$

$$z = r\cos\theta$$

ex03_3 極座標

- 動径 rと二個の偏角 θ, φ が与えられた場合に，その位置に半径1の球を表示するプログラムを，以下の二つの方法で作成せよ.
  - r, θ, φ を直交座標系 x, y, z に変換し，`translate()`のみを用いて描画する.
  - 直交座標系への変換を行わず，`rotate?()` (?はX or Y or Z)を2回，`translate(0,0,r)`を1回，適切に用いて描画する.
- r, θ, φ を様々に変えて，適切に描画されていることを確認せよ.

draw()の重要な部分はこんな感じ (その1)

```
float r = 40;
float theta = PI / 6;
float phi = PI / 3;

float x =
float y =
float z =
translate(x, y, z);

noStroke();
fill(255, 0, 0);
sphere(1);
```

他にも適当に入れてみて確認

ここに計算式を書く

draw()の重要な部分はこんな感じ (その2)

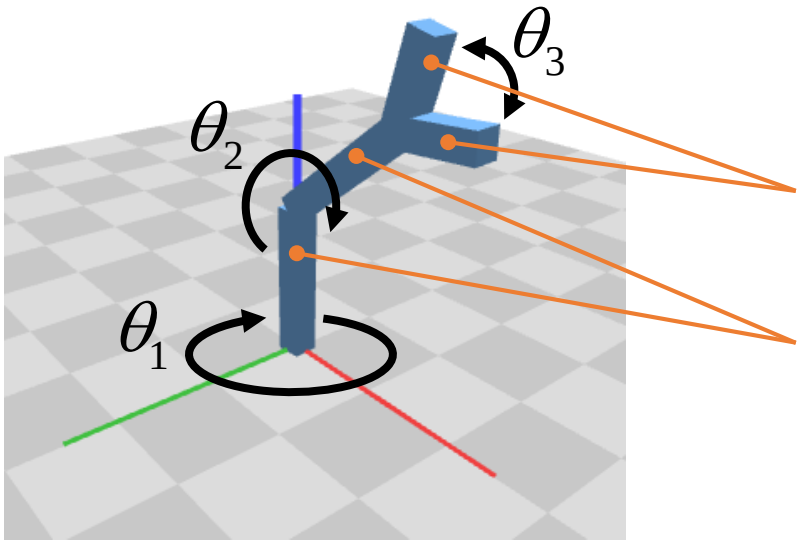
```
float r = 40;
float theta = PI / 6;
float phi = PI / 3;

translate(0, 0, r);
rotate?(phi);
rotate?(theta);

noStroke();
fill(255, 0, 0);
sphere(1);
```

他にも適当に入れてみて確認

どの軸のまわりに回転させればよいか考える.
どの順序で実行すればよいか考える.
ちなみにこの順番ではうまくいきません.

ex03_4 ロボットアーム

- 以下の仕様のロボットアームを描画するプログラムを作成せよ．角度を時間的に変化させてアニメーションさせよ．
  - アームの長さは図の通り．
  - 三つの角度で姿勢が決まる(3自由度)．回転可能な部位は図の通り．



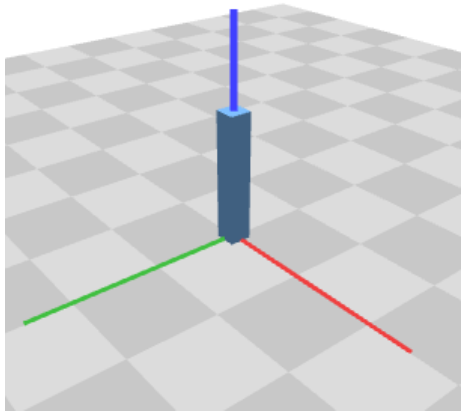
長さ15，幅5の直方体

長さ30，幅5の直方体

ex03_4 手順(1)

- 長さ30，幅5の直方体を描く関数drawArm()を作る.
  - 長辺がz軸方向になるようにする.
  - 底面中心がオブジェクト座標系の原点になるようにする.
  - 関数内でtranslate()を使った副作用が呼び出し元に残らないように，pushMatrix()とpopMatrix()を用いて呼び出されたときのカレント座標系を保存し回復する.

```
float ARM_LENGTH = 30;
float ARM_WIDTH = 5;

void drawArm() {
  pushMatrix();
  translate(0, 0, ARM_LENGTH / 2);
  box(ARM_WIDTH, ARM_WIDTH, ARM_LENGTH);
  popMatrix();
}
```

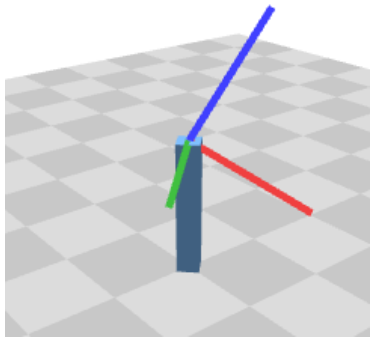


ex03_4 手順(2)

- 腕パーツ1を，z軸中心で回転させる.

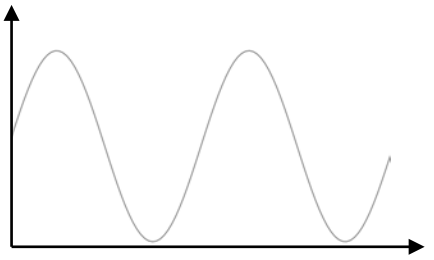
```
int m = millis();
float theta1 = TWO_PI * m / PERIOD1;

rotateZ(theta1);
drawArm();
```

- 腕パーツ1の終端までカレント座標系を移動する.
  続いて，x軸あるいはy軸中心(問題には定義がないのでいずれでもよい)に回転する.
  - 右図はy軸中心に回転した時点での
    カレント座標系.



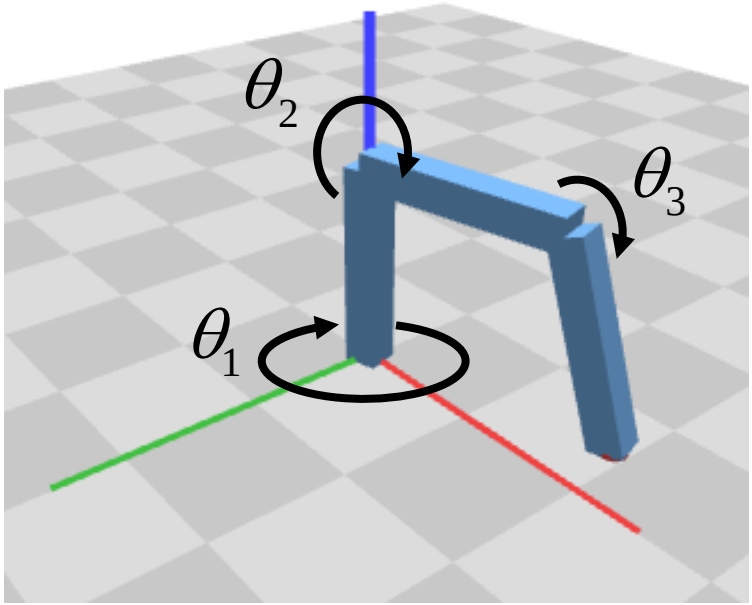
ex03_4 手順(3)

- ここで腕パーツ2を再びdrawArm()で描き，カレント座標系を腕パーツ2の末端に移動する.
- 指パーツを描く関数drawFinger()をdrawArm()を真似て作成し，2本の指を描く.
  - 途中で分岐している物体の描画については，ex02_4を参照してください.
- 関節角度について補足
  - θ_1 は時間とともに単純に増加すればよい.
  - θ_2 および θ_3 は周期的にする必要がある→



ex03_5 [発展課題] 簡単な逆運動学

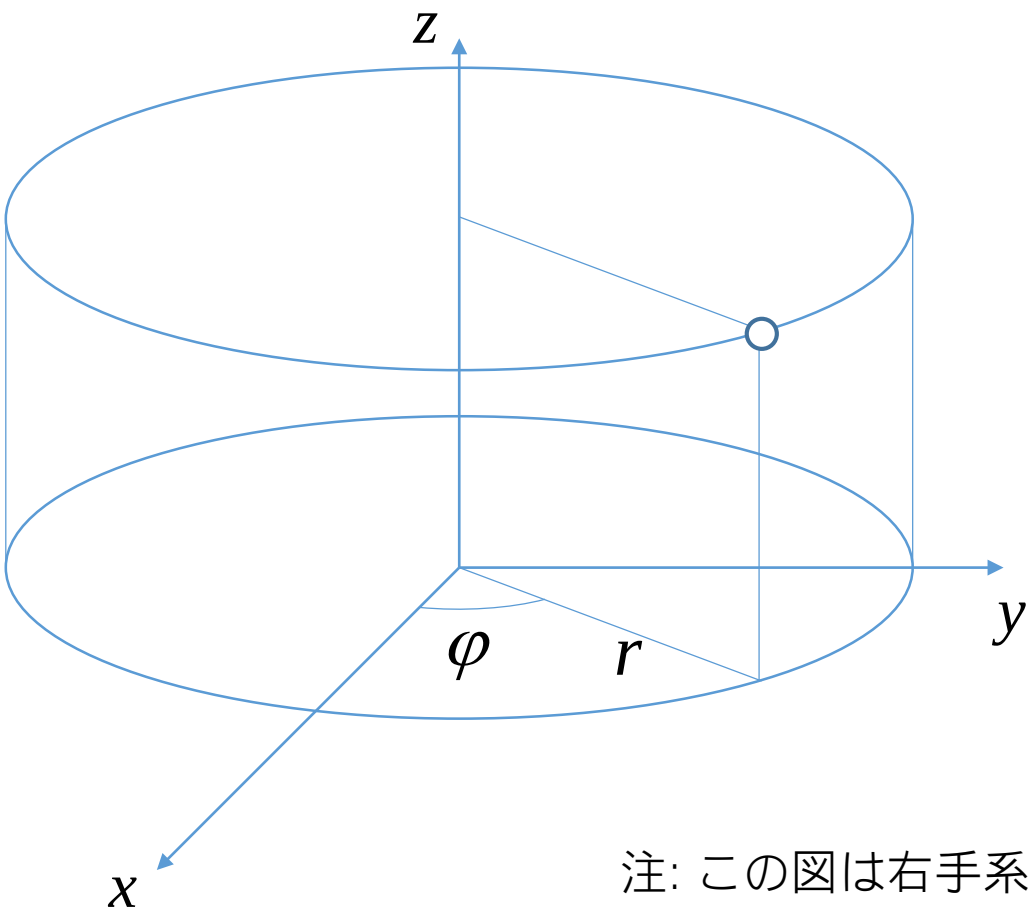
- 3自由度のロボットアームを考え，XY平面上の任意の点に置かれた物体に，アームの先端を一致させるように，関節角度をコントロールせよ.



アームのパーツの長さは
L1,L2,L3と置く.
ここではL1=L2=L3=30とする.
よってパーツは長さ30，幅5の直方体

アーム先端の座標

- 円柱座標系 r, z, φで考える.



$$x = r\cos\varphi$$

$$y = r\sin\varphi$$

$$z = z$$

注: この図は右手系

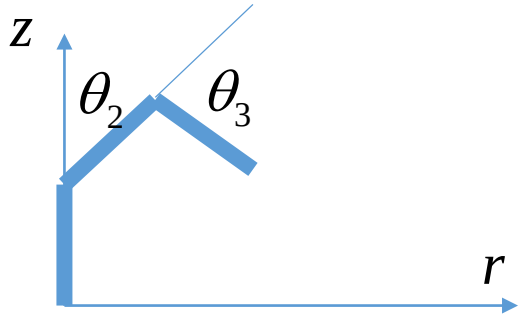
アームの角度から先端の座標を求める

- 以下の通り.

$$r = L_2 \sin\theta_2 + L_3 \sin(\theta_2 + \theta_3)$$

$$z = L_1 + L_2 \cos\theta_2 + L_3 \cos(\theta_2 + \theta_3)$$

$$\varphi = \theta_1$$ ←簡単なので以降の処理からは外して考える



- いま，ターゲットの位置がわかっているとして，r, zをその位置に近付けるにはどうすればよいか?

$$\Delta r, \Delta z \rightarrow \Delta\theta_2, \Delta\theta_3$$

全微分してみる

- 形式的には以下のようになる.

$$dr = \frac{\partial r}{\partial \theta_2} d\theta_2 + \frac{\partial r}{\partial \theta_3} d\theta_3$$

$$dz = \frac{\partial z}{\partial \theta_2} d\theta_2 + \frac{\partial z}{\partial \theta_3} d\theta_3$$

- 本問ではこうなる.

$$\begin{pmatrix} dr \\ dz \end{pmatrix} = \begin{pmatrix} L_2 \cos\theta_2 + L_3 \cos(\theta_2 + \theta_3) & L_3 \cos(\theta_2 + \theta_3) \\ -L_2 \sin\theta_2 - L_3 \sin(\theta_2 + \theta_3) & -L_3 \sin(\theta_2 + \theta_3) \end{pmatrix} \begin{pmatrix} d\theta_2 \\ d\theta_3 \end{pmatrix} \equiv J \begin{pmatrix} d\theta_2 \\ d\theta_3 \end{pmatrix}$$

- Jはヤコビアンと呼ばれる行列.

角度をどう変えればよいか?

- Jの具体的な数値は計算可能なので，逆行列を求めて左から乗じる.

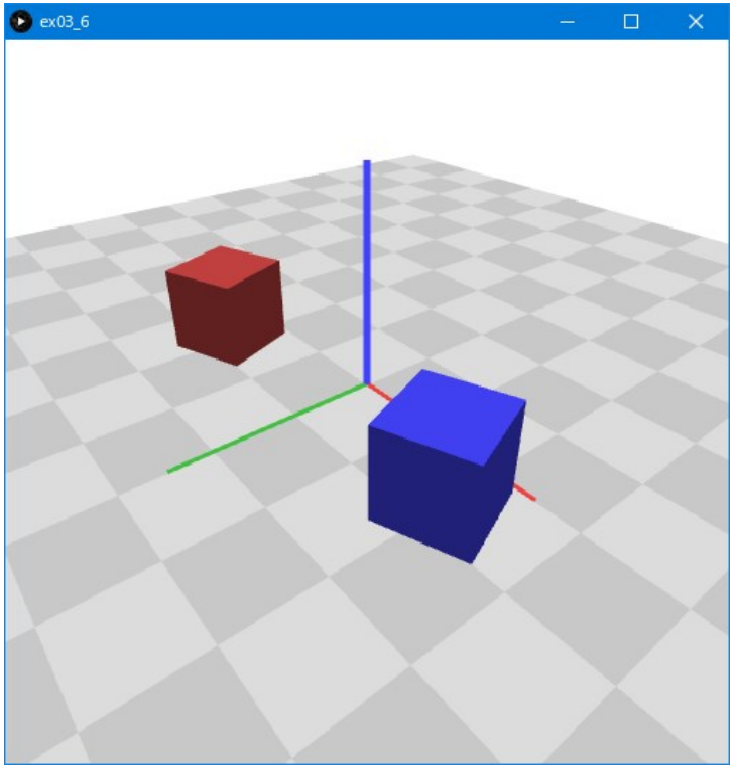
$$\begin{pmatrix} dr \\ dz \end{pmatrix} = J \begin{pmatrix} d\theta_2 \\ d\theta_3 \end{pmatrix}$$

$$\Rightarrow \begin{pmatrix} d\theta_2 \\ d\theta_3 \end{pmatrix} = J^{-1} \begin{pmatrix} dr \\ dz \end{pmatrix}$$

- 全体の手順: 以下をくりかえす
  - ロボットアームの先端の座標を求める
  - ターゲットとの差を求める
  - ヤコビアンの逆行列を計算する
  - 角度の変化量を求める

ex03_6 [発展課題] 変換行列の直接操作

- テンプレートの空所に適切な処理を記述し，青色の立方体が赤色の立方体と同じ動作をするようにせよ.



テンプレートの一部

- translate()とrotate()を用いて記述
- applyMatrix()を用いて記述

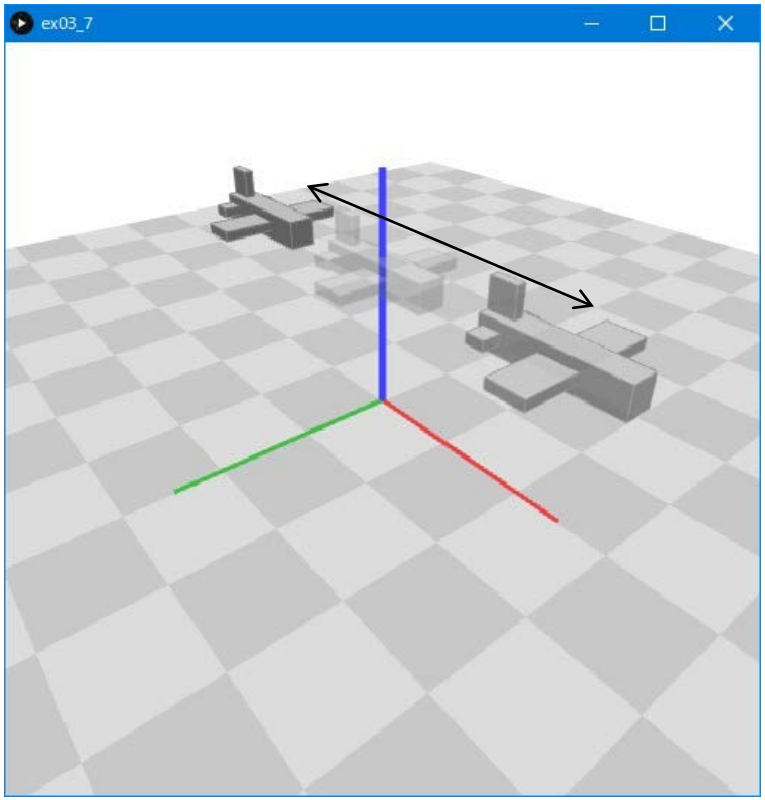
```
pushMatrix();
translate(-40, 0, 10);
translate(0, y, 0);
rotateZ(theta);
fill(192, 64, 64);
box(20);
popMatrix();
```

同様の処理を行列で記述

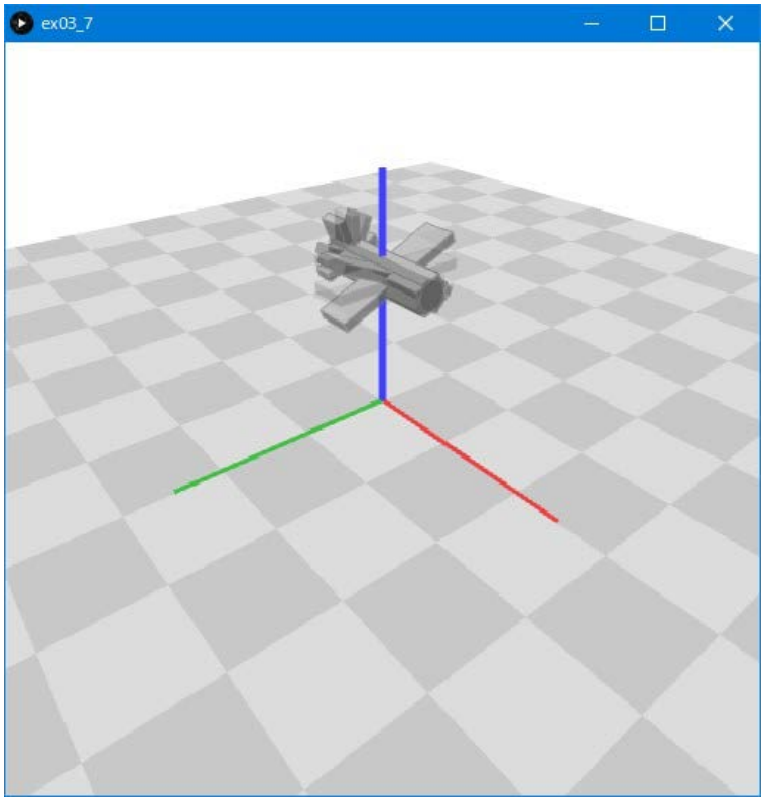
```
pushMatrix();
translate(40, 0, 10);
float s = sin(theta);
float c = cos(theta);
applyMatrix(
                    );
fill(64, 64, 240);
box(20);
popMatrix();
```

ex03_7 [発展課題] 移動物体の操作

- キーボードを用いて，3次元空間内を自由に移動できる飛行機を作成せよ．



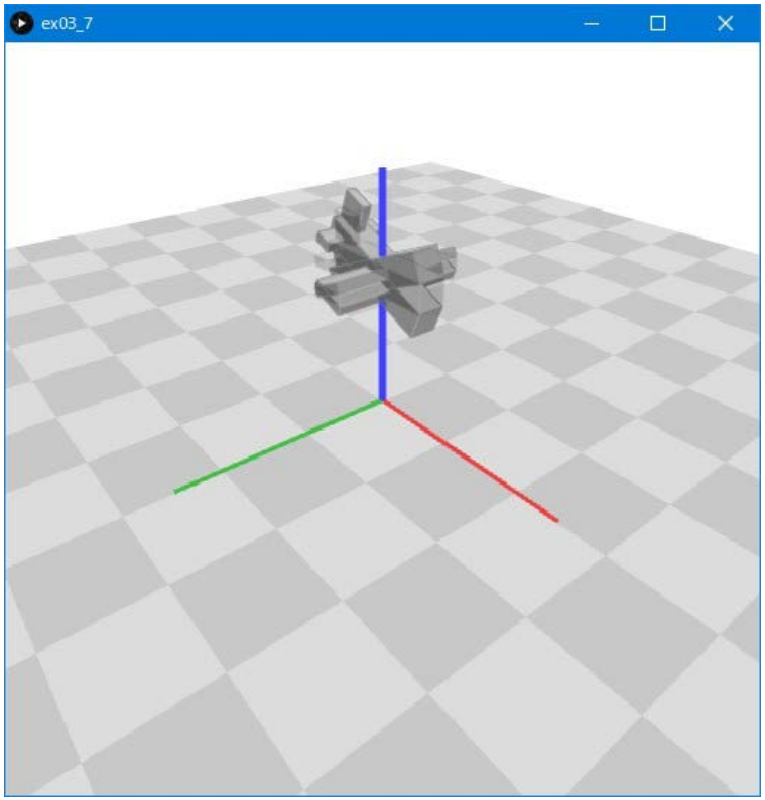
f, b で前後に移動



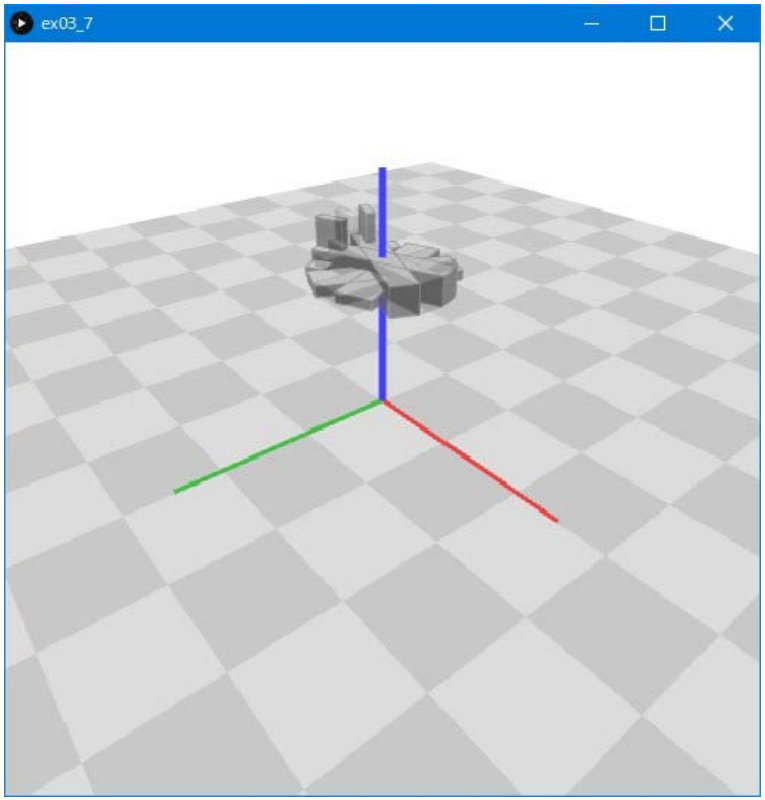
a, d でroll(X軸回転)

ex03_7 [発展課題] 移動物体の操作

- 方針: 4x4の変換行列を保持しておき，キー入力に応じた適切な行列を生成，乗算する．



w, s でpitch(Y軸回転)



q, e でyaw(Z軸回転)