Graphics with Processing

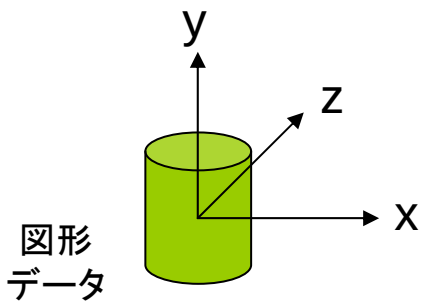
2007-08 モデルビュー変換

http://vilab.org

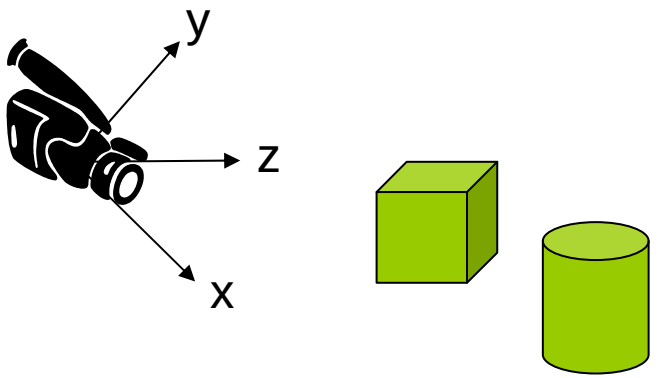
塩澤秀和

8.1 3DCGの座標系

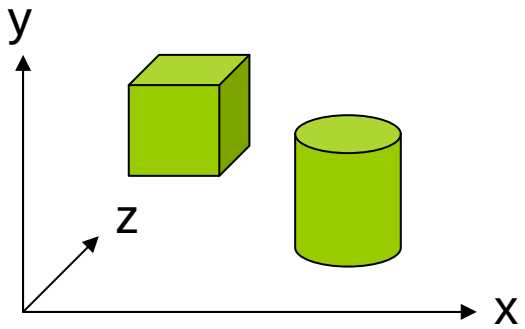
- ローカル座標系
  - オブジェクトの座標系



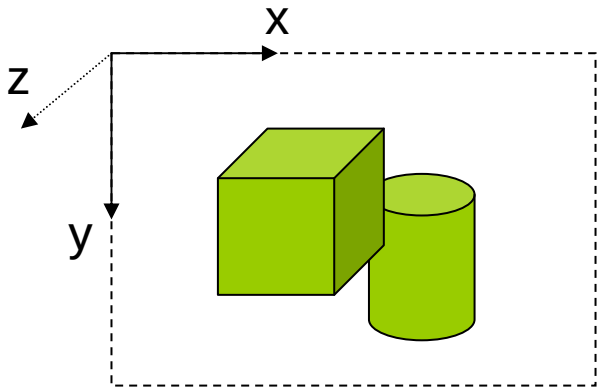
- 視点（カメラ）座標系



- ワールド座標系
  - 3次元世界の座標系

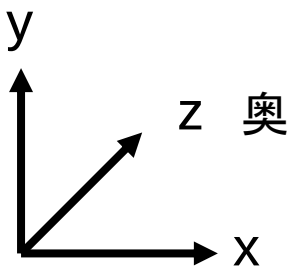


- スクリーン（画面）座標系

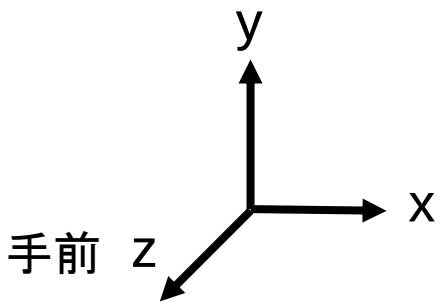


8.2 左手系と右手系

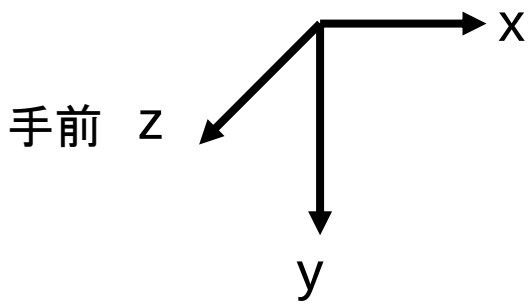
- 左手系
  - 視点座標系・CGゲーム
  - DirectX



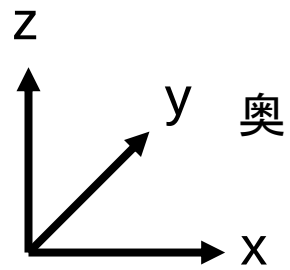
- 右手系
  - CG理論・数学・工学分野
  - OpenGL



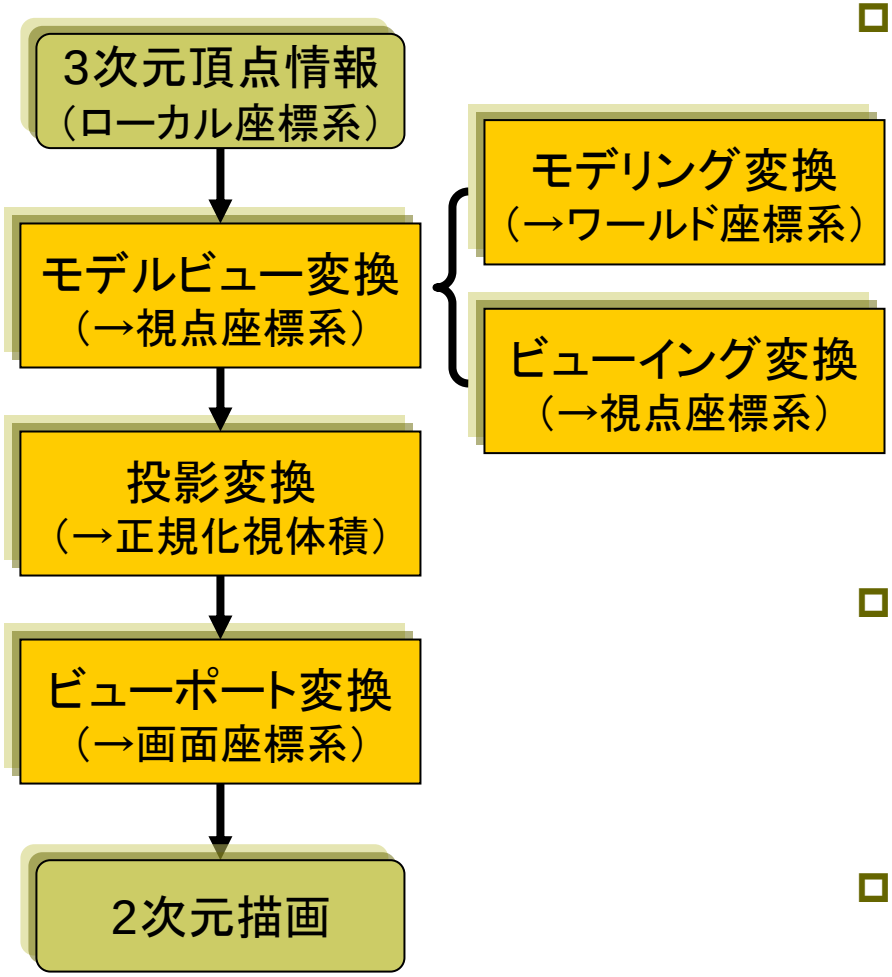
- 左手系
  - Processing



- 右手系
  - 建築座標系



8.3 3DCGの座標変換

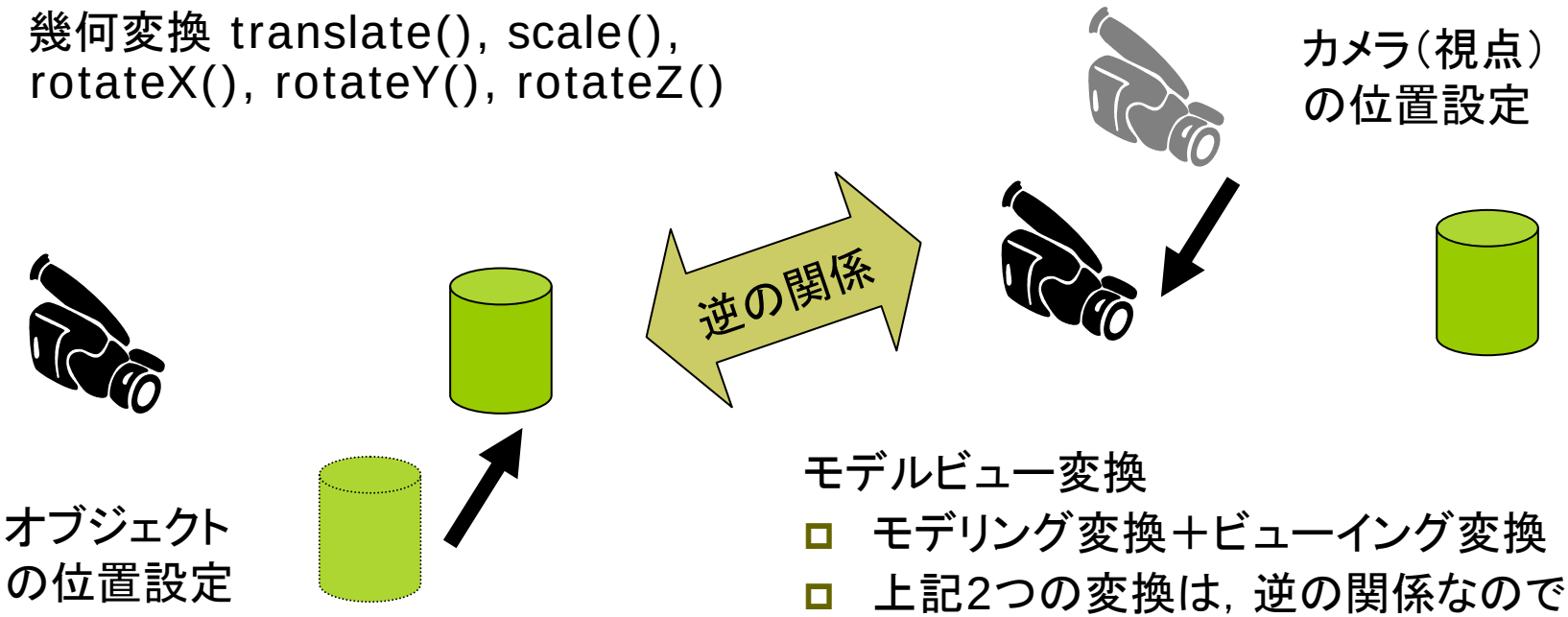


- モデルビュー変換
  - オブジェクト（図形・物体）と視点（カメラ）の位置関係の設定
  - モデリング変換：
    オブジェクトの配置
  - ビューイング変換（視界変換）：
    視点の位置設定
  - translate(), scale(), rotate{X,Y,Z}(), camera()
- 投影変換（次回）
  - 投影面へ（正規化視体積へ）
  - 平行投影：ortho()
  - 透視投影：perspective()
- ビューポート変換
  - 正規化視体積から画面座標へ（自動）

8.4 モデルビュー変換

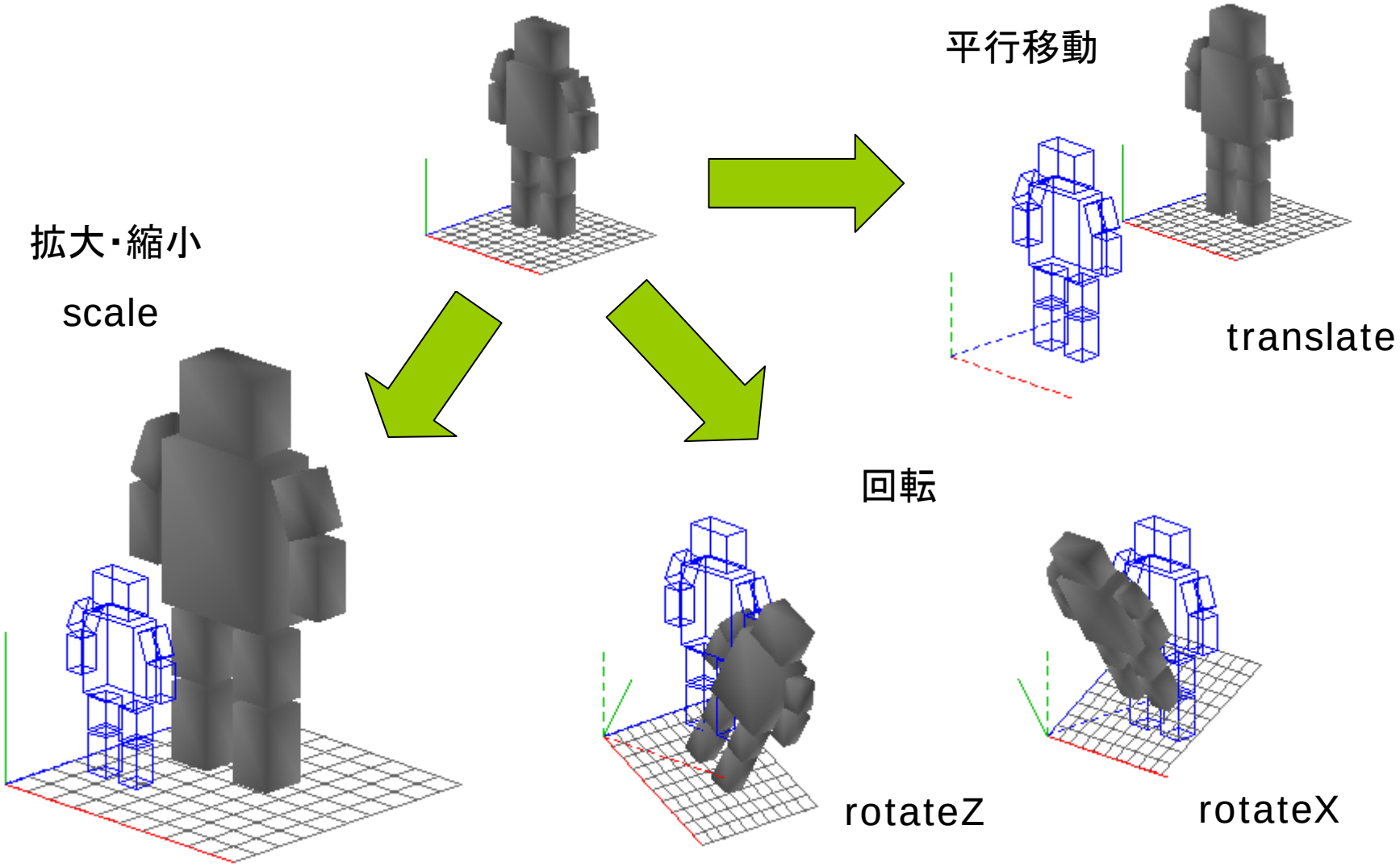
モデリング変換
- 目的: ワールド座標系に個々の3Dモデルを配置する
- 変換前の座標系: ローカル座標系（オブジェクトごとの座標系）
- 変換後の座標系: ワールド座標系
- 幾何変換 translate(), scale(), rotateX(), rotateY(), rotateZ()

ビューイング変換（視界変換）
- 目的: 投影計算のために，座標の原点を視点に移動する
- 変換前の座標系: ワールド座標系
- 変換後の座標系: 視点座標系



モデルビュー変換
- モデリング変換＋ビューイング変換
- 上記2つの変換は，逆の関係なので1つの変換行列で表現できる

8.5 3次元幾何変換(参考:4.3)

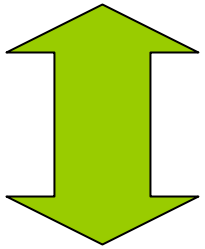


8.6 3次元同次座標

3次元同次座標

$$(x,\ y,\ z,\ w)$$

wは任意の定数



同次座標

直交座標

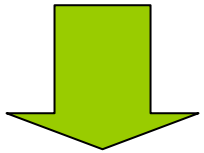
$$\left(\frac{x}{w},\ \frac{y}{w},\ \frac{z}{w}\right)$$

通常，w＝1で用いる

$$(x,\ y,\ z)$$

同次座標によるアフィン変換

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$



数学的に
より簡便な表記

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & t_x \\ a_{21} & a_{22} & a_{23} & t_y \\ a_{31} & a_{32} & a_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

8.7 3次元幾何変換(1)

3次元アフィン変換(1)

- 平行移動

$$x' = x + t_x$$
$$y' = y + t_y$$
$$z' = z + t_z$$

- 拡大縮小

$$x' = s_x\, x$$
$$y' = s_y\, y$$
$$z' = s_z\, z$$

同次座標系を用いた表現

- 平行移動

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- 拡大縮小

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

8.8 3次元幾何変換(2)

- z軸まわりの回転

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$
$$z' = z$$

- x軸まわりの回転

$$x' = x$$
$$y' = y\cos\theta - z\sin\theta$$
$$z' = y\sin\theta + z\cos\theta$$

- y軸まわりの回転

$$x' = z\sin\theta + x\cos\theta$$
$$y' = y$$
$$z' = z\cos\theta - x\sin\theta$$

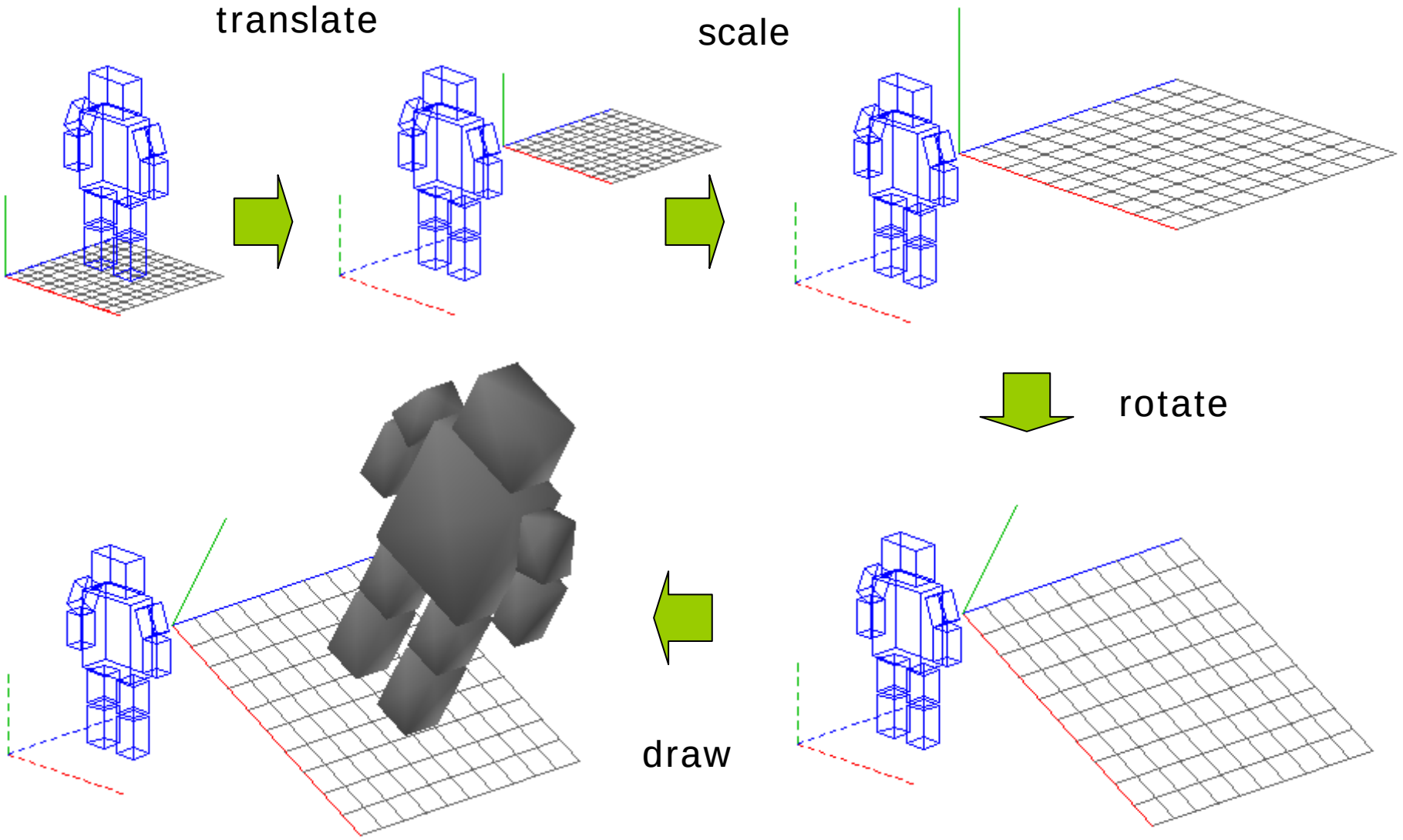
- z軸まわりの回転

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- x軸まわりの回転

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

8.9 幾何変換の合成(参考:4.5)



8.10 3次元合成変換行列(参考:4.6)

合成変換の数学表現

- 同次変換行列の積になる

$$P_{world} = M_1 M_2 M_3 \cdots M_n P_{local}$$

$$M = M_1 M_2 M_3 \cdots M_n$$

- Processingコード(8.9)

```
translate(0, 100, 300); // ← M1
scale(2, 2, 2);         // ← M2
rotateZ(PI/6);          // ← M3
/* このあと描画 */
```

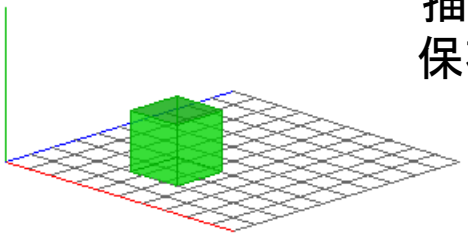
$$\begin{bmatrix} x_{world} \\ y_{world} \\ z_{world} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 100 \\ 0 & 0 & 1 & 300 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\pi/6) & -\sin(\pi/6) & 0 & 0 \\ \sin(\pi/6) & \cos(\pi/6) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{local} \\ y_{local} \\ z_{local} \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x_{world} \\ y_{world} \\ z_{world} \\ 1 \end{bmatrix} = \begin{bmatrix} \sqrt{3} & -1 & 0 & 0 \\ 1 & \sqrt{3} & 0 & 100 \\ 0 & 0 & 2 & 300 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{local} \\ y_{local} \\ z_{local} \\ 1 \end{bmatrix}$$

$$\therefore M = \begin{bmatrix} \sqrt{3} & -1 & 0 & 0 \\ 1 & \sqrt{3} & 0 & 100 \\ 0 & 0 & 2 & 300 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

8.11 変換行列の操作(参考:4.7)

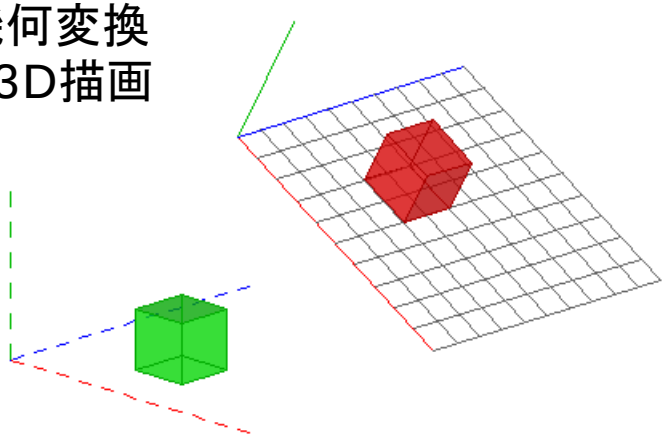
pushMatrixで
描画座標系
保存しておく



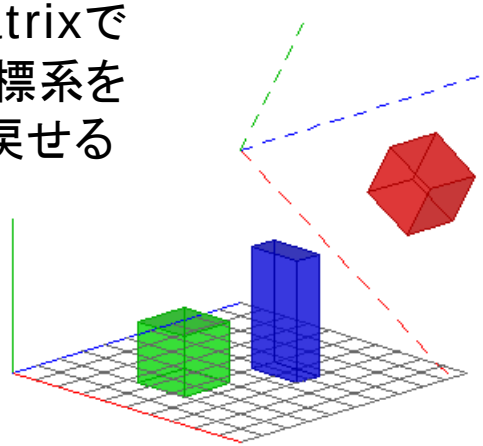
行列操作関数

- pushMatrix()
  - システム変換行列を一時待避
  - 変換行列＝現在の描画座標系
- popMatrix()
  - 最近保存した変換行列を戻す
- resetMatrix()
  - 変換行列をリセットする

幾何変換
と3D描画



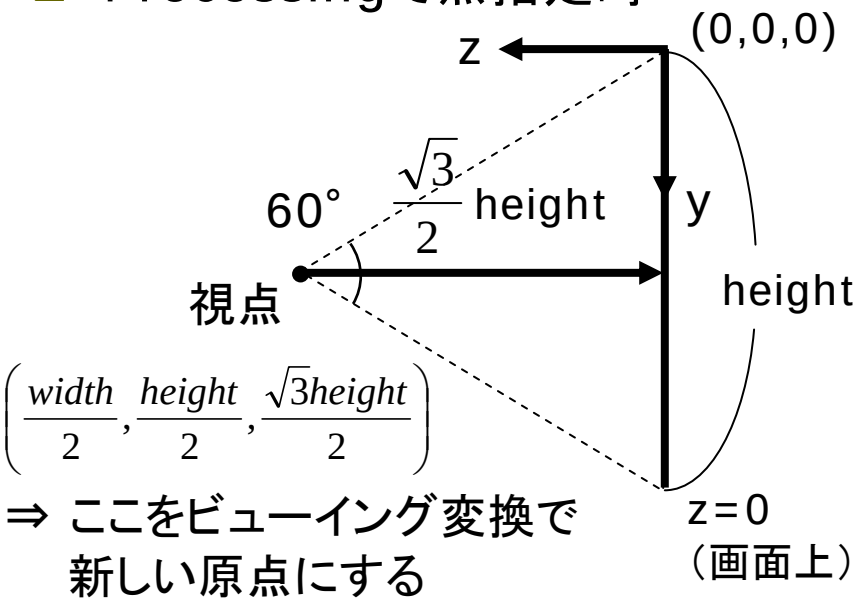
popMatrixで
描画座標系を
もとに戻せる



8.12 ビューイング変換(7.3参照)

ビューイング(視界)変換

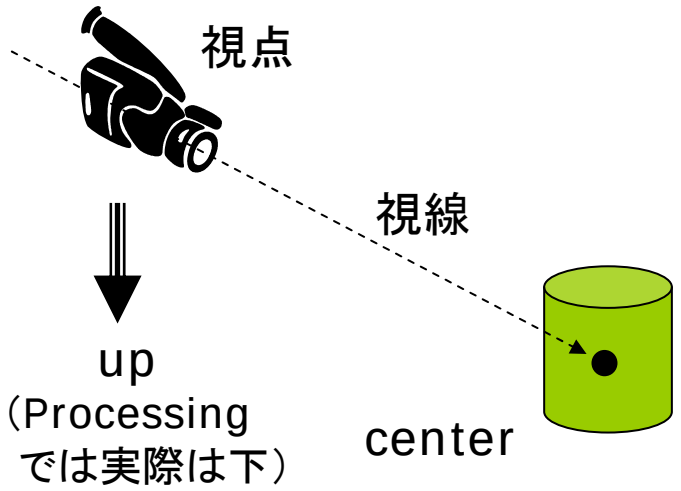
- 視点と視線の設定
  - 原点を視点に移動する変換
    (ワールド座標系→視点座標系)
  - 全オブジェクトを視点と逆方向に
    移動するモデリング変換と同じ
- Processingで無指定時



⇒ ここをビューイング変換で
新しい原点にする

視点設定関数

- camera(eyeX, eyeY, eyeZ,
  centerX, centerY, centerZ,
  upX, upY, upZ)
  - eye: カメラ(視点)の座標
  - center: カメラで狙う座標
  - up: 上下方向を示すベクトル
    (通常は各要素は, 0か±1)



8.13 サンプルと演習課題

サンプル(8.11の一部)

```
void draw() {
  lights();
  ortho(-width/2, width/2,
        -height/2, height/2,
        -100, 100);
  translate(20, height - 100, 0);
  rotateX(PI*0.9); rotateY(PI/4);

  pushMatrix();  // 1つめの箱
    translate(90, 20, 60);
    box(40);
  popMatrix();
  pushMatrix();  // 2つめの箱
    translate(0, 100, 200);
    rotateZ(-PI/6);
    translate(80, 20, 80);
    box(40);
  popMatrix();
  pushMatrix();  // 3つめの箱
    translate(120, 40, 120);
    box(40, 80, 20);
  popMatrix();
}
```

課題

- ウィンドウの中央あたりに立体図形を何個か描画し，それらが回転軸の周りを，メリーゴーランドのように回転するプログラムを作成しなさい
- 全体を斜め上から見下ろすように表示すること(真上から見たのはダメ)
- サンプルプログラムを利用してもよい