

Graphics with Processing



2016-08 モデルビュー変換

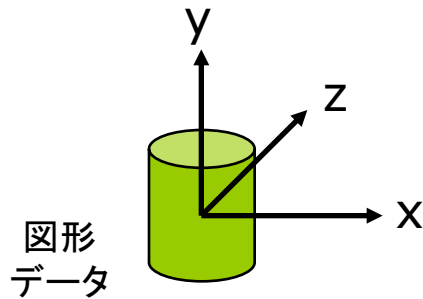
<http://vilab.org>

塩澤秀和

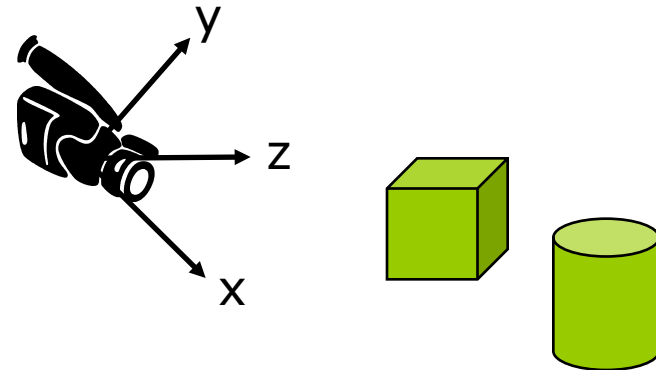
8.1 3DCGの座標系 (p.49)

□ ローカル(モデリング)座標系

■ オブジェクトの座標系

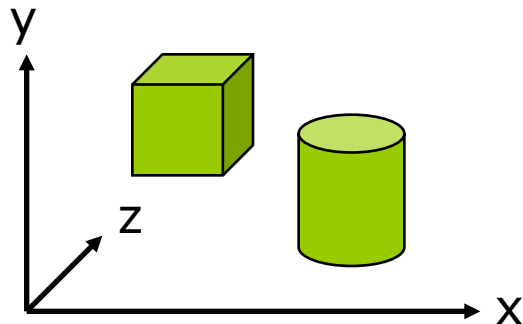


□ 視点(カメラ)座標系

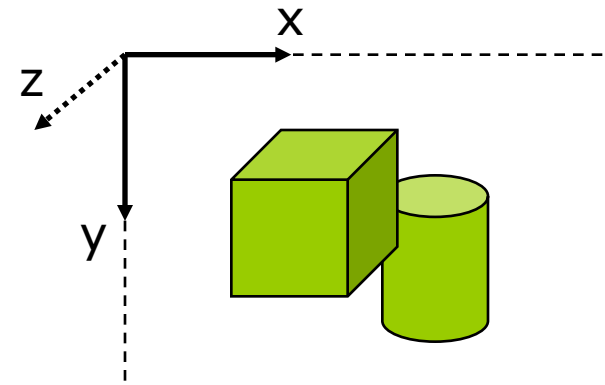


□ ワールド座標系

■ 3次元世界の座標系

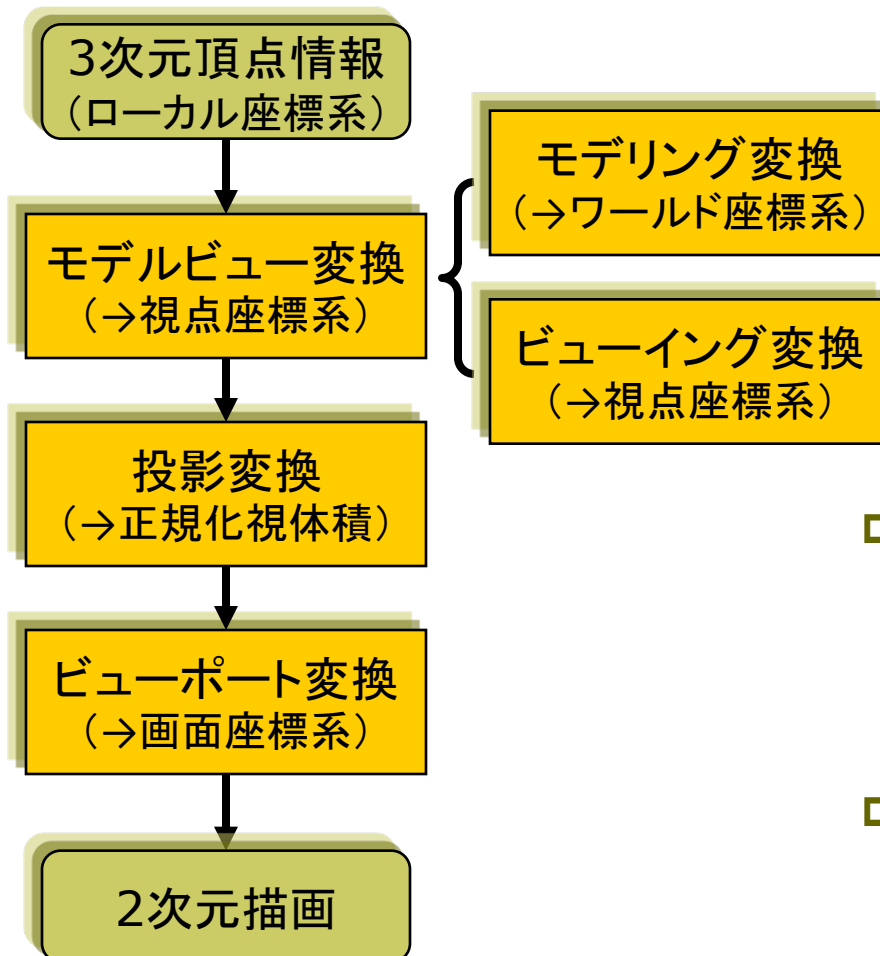


□ 画面(デバイス)座標系



8.2 3DCGの座標変換(p.49)

□ ビューイングパイプライン



□ 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(視野変換):
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)

8.3 モデルビュー変換

モデリング変換

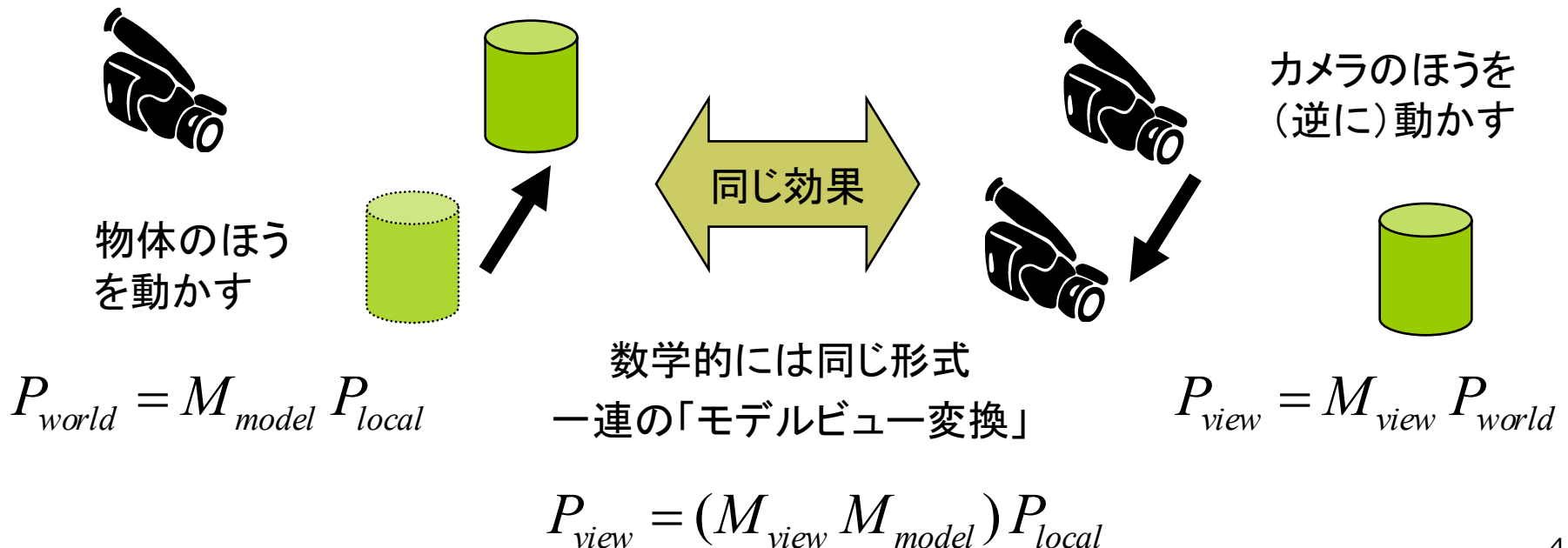
□ オブジェクトの位置設定

- 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: ローカル座標系
- 変換後: ワールド座標系

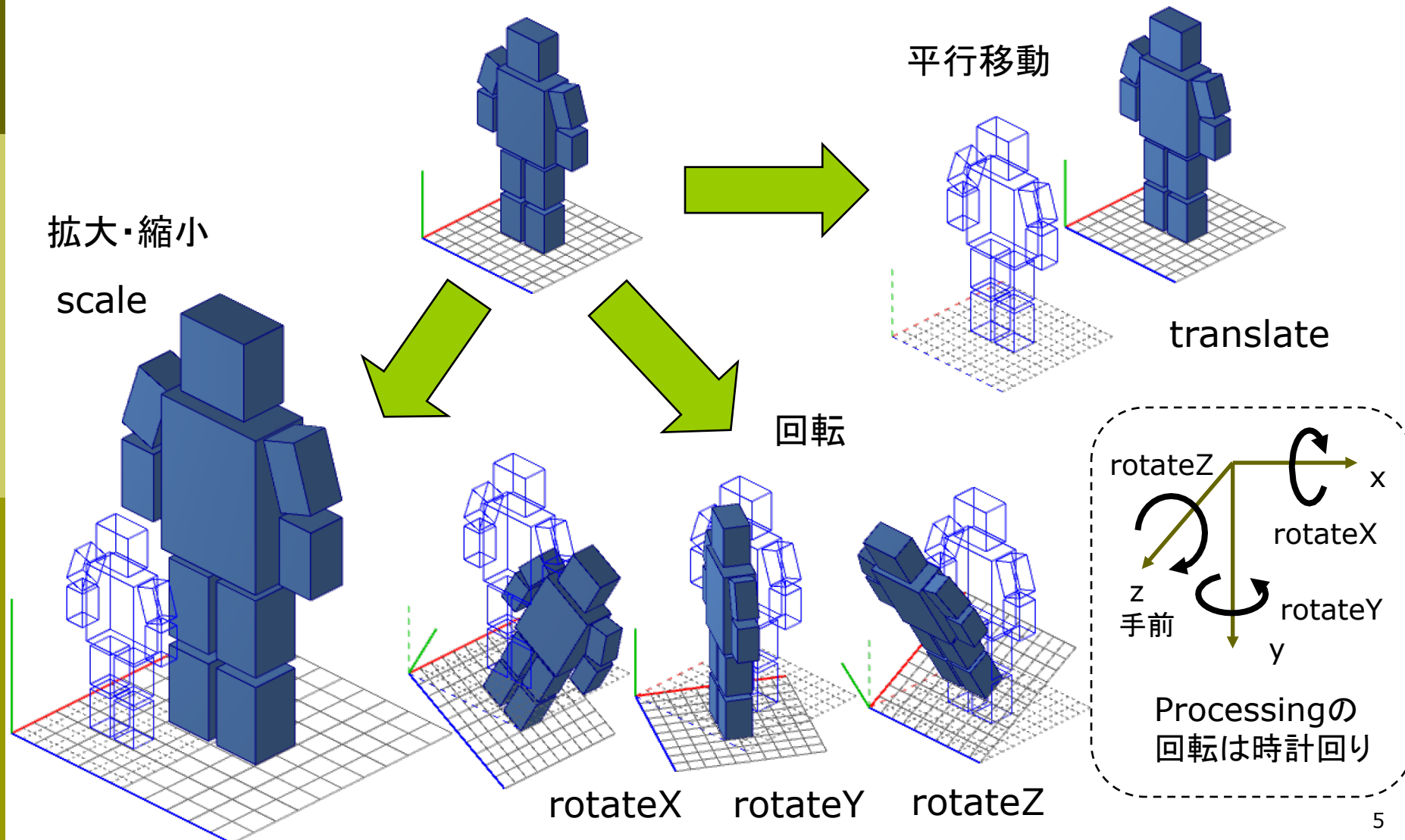
ビューイング変換(視野変換)

□ 視点(カメラ)の位置設定

- 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: ワールド座標系
- 変換後: 視点座標系



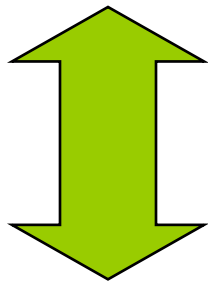
8.4 3次元幾何変換(p.34)



8.5 3次元同次座標 (p.34)

3次元同次座標表現

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



同次座標

直交座標

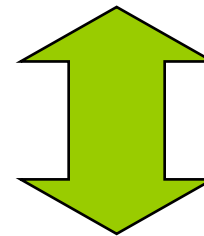
$$\left(\frac{x}{w}, \frac{y}{w}, \frac{z}{w} \right)$$

通常, $w=1$ で用いる

$$(x, y, z)$$

3次元アフィン変換

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



同次座標による表現

直交座標による表現

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

8.6 3次元幾何変換(1)

3次元幾何変換

□ 平行移動

$$x' = x + t_x$$

$$y' = y + t_y$$

$$z' = z + t_z$$

□ 拡大・縮小

$$x' = s_x x$$

$$y' = s_y y$$

$$z' = s_z z$$

同次座標を用いた表現

□ 平行移動 $T(t_x, t_y, t_z)$

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

□ 拡大・縮小 $S(s_x, s_y, s_z)$

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

8.7 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 $R_z(\theta)$

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

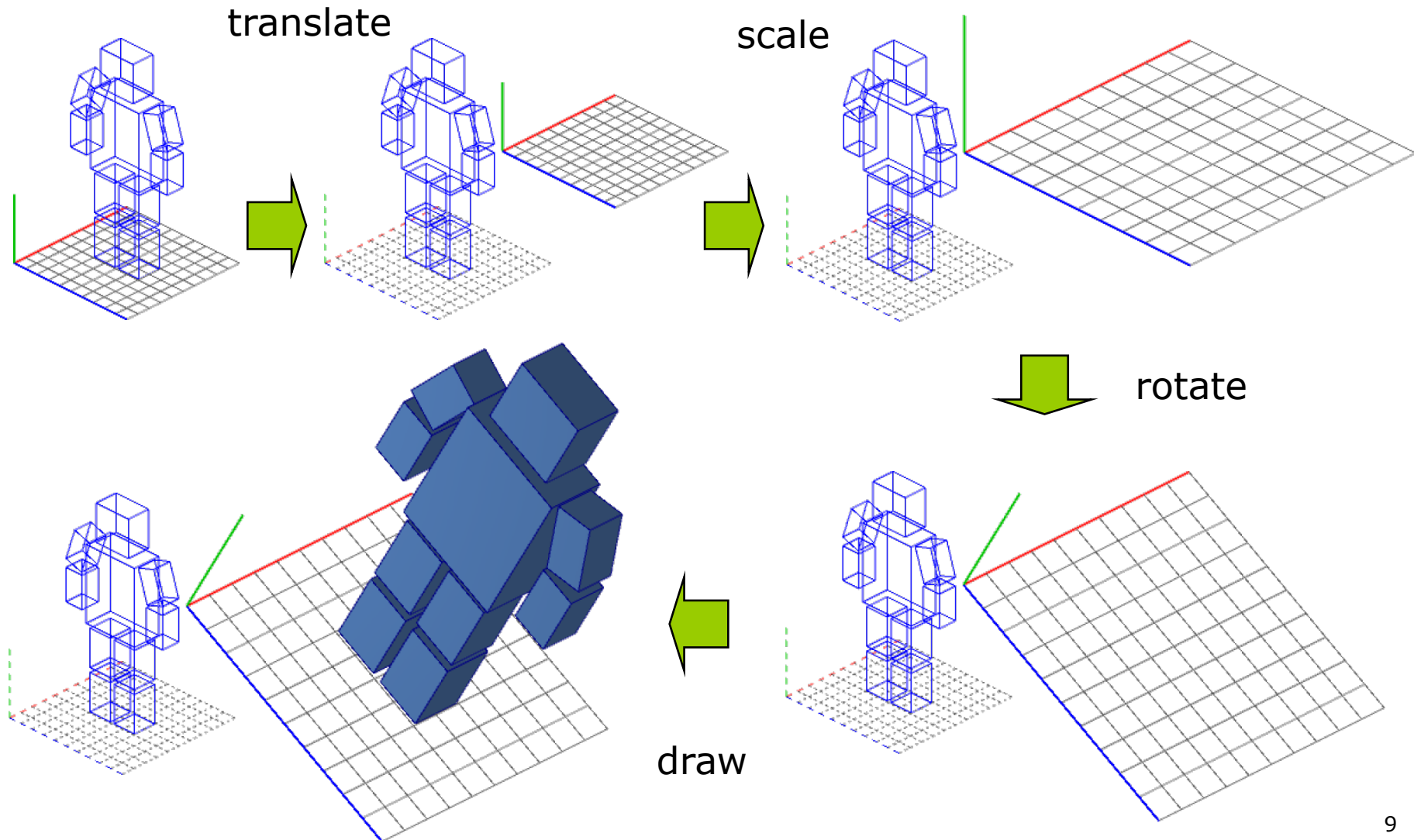
□ x軸まわりの回転 $R_x(\theta)$

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

□ y軸まわりの回転 $R_y(\theta)$

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

8.8 幾何変換の合成 (p.37)



8.9 合成変換行列

モデリング変換行列

□ 同次変換行列の積になる

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

$$M_{model} = M_1 M_2 M_3 \cdots M_n$$

□ Processingのコード例

```
translate(0, 100, 300); // M1
scale(2, 2, 2); // M2
rotateZ(PI/6); // M3
// 図形描画...
```

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

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

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

8.10 変換行列の操作 (p.54)

複数オブジェクトの配置

行列スタックの操作

□ pushMatrix()

- システム変換行列(座標系)を一時的に退避する
- 階層モデリングに利用される

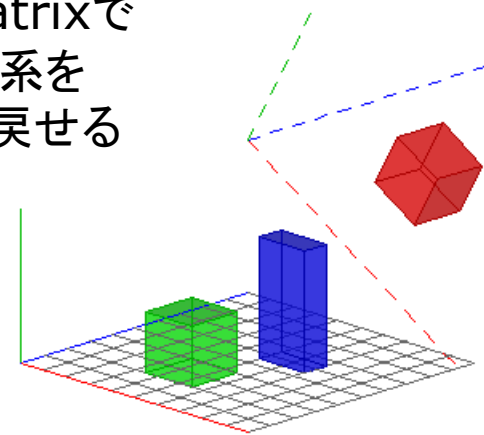
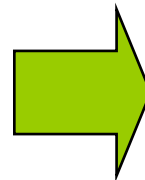
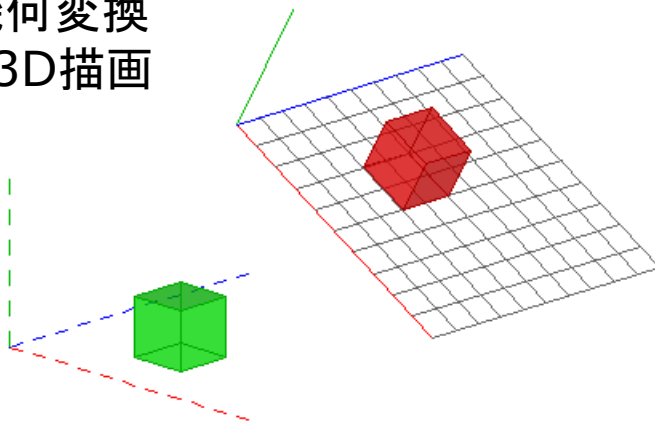
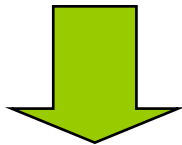
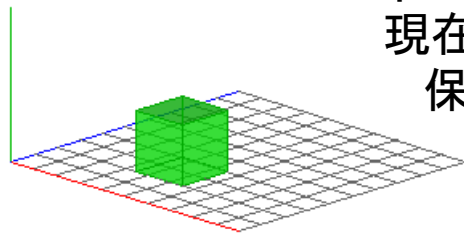
□ popMatrix()

- 最近保存した変換行列を戻す
- pushとpopは必ず対にすること

pushMatrixで
現在の座標系を
保存しておく

popMatrixで
座標系を
もとに戻せる

幾何変換
と3D描画

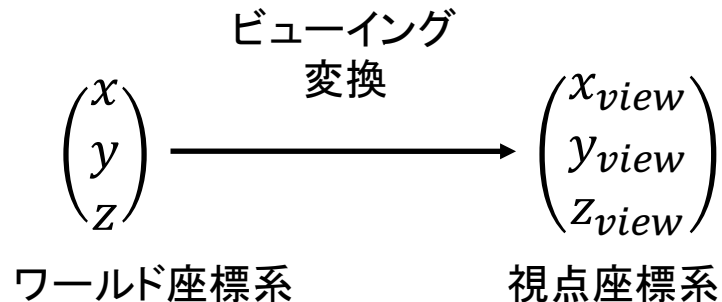


8.11 ビューイング変換 (p.50)

ビューイング変換 (視野変換)

□ 視点と視線の設定

- 同じ点を視点を変えて見る



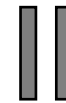
□ 幾何変換による実現

- 視線の向きがz軸の負の向きになるように座標系を“回転”
- 視点の位置が原点となるように座標系を“平行移動”

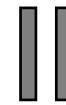
$$P_{view} = M_{view} P_{world}$$

視点移動の考え方

視点をワールド座標系の中で
原点から $E(e_x, e_y, e_z)$ に移動



世界全体(すべての物体)を
 $(-e_x, -e_y, -e_z)$ だけ平行移動



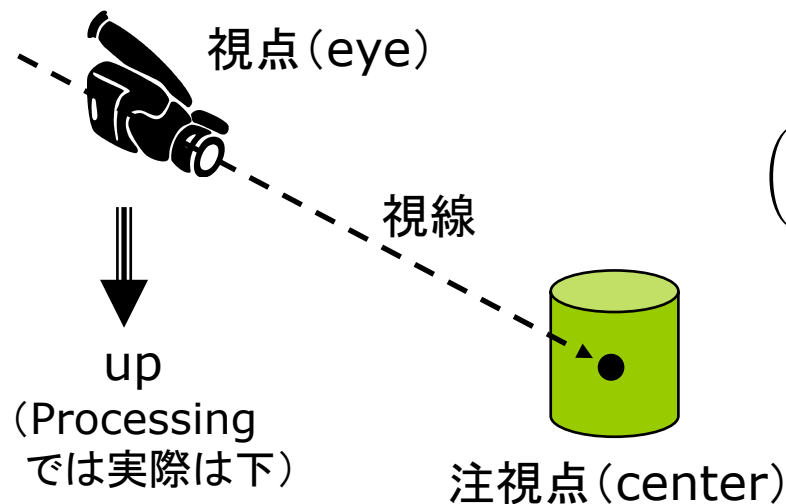
等価なモデリング変換行列

$$M_{view} = \begin{bmatrix} 1 & 0 & 0 & -e_x \\ 0 & 1 & 0 & -e_y \\ 0 & 0 & 1 & -e_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

8.12 ビューイング変換関数

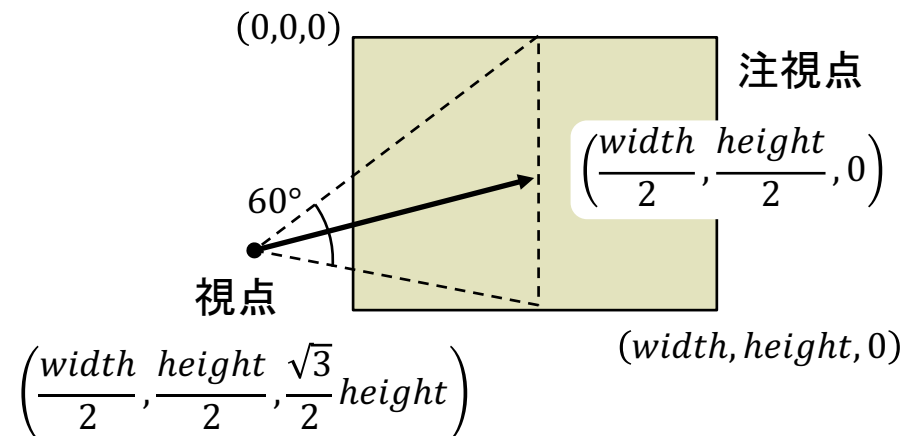
視点設定関数

- 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
 - モデリング変換より前に書くこと



□ Processingのデフォルト

- camera()
- camera(width/2, height/2, sqrt(3)*height/2, width/2, height/2, 0, 0, 1, 0)



$$M_{view} = \begin{bmatrix} 1 & 0 & 0 & -width/2 \\ 0 & 1 & 0 & -height/2 \\ 0 & 0 & 1 & -\sqrt{3}height/2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

8.13 演習課題

課題

問1) 右のプログラムを実行してから以下の問いに答えなさい

- camera関数が表すビューイング変換を行列(M_{view})で示しなさい
- さらに, ★の箇所でのモデリング変換(M_{model})を求めなさい
- 両者を合成し, モデルビュー変換 $M_{view} M_{model}$ を求め, boxの座標を視点座標系に変換しなさい
- 次回, **A4レポート用紙**で提出

問2) 複数の図形(boxなど)を使い, 立体の文字やマークなどを組み立てるプログラムを作成しなさい

- 棒1本で「I」などはダメ
- 回転など動きをつけるとよい

// 「イ」と書く例

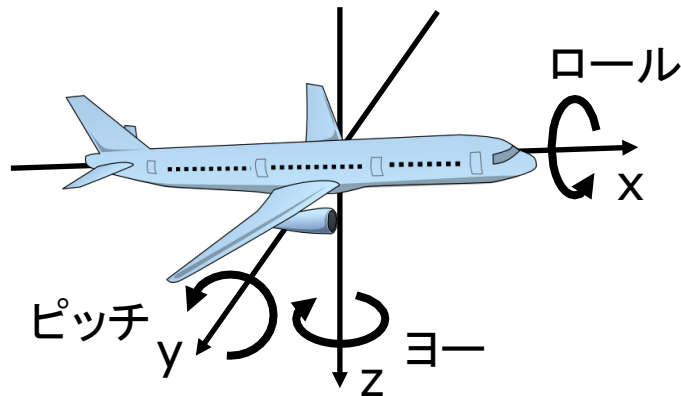
```
void setup() {
  size(500, 400, P3D);
  noLoop();
}
```

```
void draw() {
  background(0); lights();
  perspective();
  camera(0, 0, 300, 0, 0, 0, 0, 1, 0);
  printCamera(); // ← これがヒント
  rotateY(PI/6);
  fill(#ff5050); noStroke();
  pushMatrix();
    translate(-10, -30, 0);
    rotateZ(-PI/4);
    box(200, 50, 50);
  popMatrix();
  pushMatrix();
    translate(0, 50, 0);
    box(50, 150, 50); // ← ★
  popMatrix();
}
```

8.14 参考: 任意姿勢への回転

ロール・ピッチ・ヨー

移動体の姿勢回転



オイラー角

回転変換の合成

- 2軸以上の3回の回転変換の合成で任意の姿勢を指定可能
- $X \rightarrow Z \rightarrow X$: 狭義のオイラー角
- 全部で12パターン: $XYX, XZX, YXY, YZY, ZXZ, ZYZ, XYZ, XZY, YXZ, YZX, ZXY, ZYX$

任意軸周りの回転

ロドリゲスの回転公式

$\vec{n}$: 回転軸の単位ベクトル

θ : 回転量

直交座標の場合の回転行列 (3×3)

$$R_{3 \times 3} = I + (\sin \theta)N + (1 - \cos \theta)N^2$$

$$\text{ここで } N = \begin{bmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{bmatrix}$$

同次座標なら

$$R = \left[\begin{array}{ccc|c} & & & 0 \\ & R_{3 \times 3} & & 0 \\ & & & 0 \\ \hline 0 & 0 & 0 & 1 \end{array} \right]$$

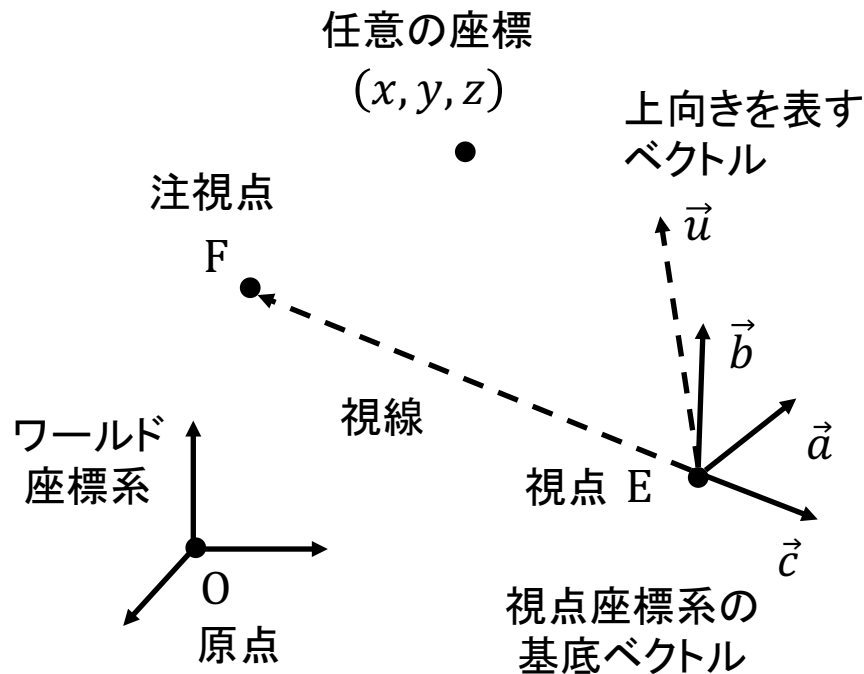
展開すると
非常に複雑

8.15 参考: ビューイング変換の導出

視点と注視点によるカメラワーク

□ camera関数の役割

- 任意の座標 (x, y, z) を視点から見た座標値に変換



まず, 視点座標系の x, y, z 軸を表す正規直交基底ベクトル $\vec{a}, \vec{b}, \vec{c}$ を求める

$$\vec{c} = \frac{-\overrightarrow{EF}}{|\overrightarrow{EF}|} \quad \vec{a} = \frac{\vec{u} \times \vec{c}}{|\vec{u} \times \vec{c}|} \quad \vec{b} = \vec{c} \times \vec{a}$$

互いに直交する単位ベクトル

視点座標系の値 $(x_{view}, y_{view}, z_{view})$ は, 視点から $\vec{a}, \vec{b}, \vec{c}$ を“1目盛り”として測った位置を示すので, 以下が成り立つ

$$P = E + x_{view} \vec{a} + y_{view} \vec{b} + z_{view} \vec{c}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} e_x \\ e_y \\ e_z \end{bmatrix} + \begin{bmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ a_z & b_z & c_z \end{bmatrix} \begin{bmatrix} x_{view} \\ y_{view} \\ z_{view} \end{bmatrix}$$

この行列を R とおく
(何らかの回転を表す)

(次のページへ)

8.16 参考:ビューイング変換の導出(続き)

E を移項した後, 両辺に左から R の行と列を入れ替えた転置行列 R^T を掛ける

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} - \begin{bmatrix} e_x \\ e_y \\ e_z \end{bmatrix} = \begin{bmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ a_z & b_z & c_z \end{bmatrix} \begin{bmatrix} x_{view} \\ y_{view} \\ z_{view} \end{bmatrix}$$

正規直交基底ベクトルを並べた
行列は「直交行列」と言われ、
実は転置行列と逆行列が等しい

$$\begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \end{bmatrix} \left(\begin{bmatrix} x \\ y \\ z \end{bmatrix} - \begin{bmatrix} e_x \\ e_y \\ e_z \end{bmatrix} \right) = \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \end{bmatrix} \begin{bmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ a_z & b_z & c_z \end{bmatrix} \begin{bmatrix} x_{view} \\ y_{view} \\ z_{view} \end{bmatrix}$$

$$\begin{matrix} \vec{a}, \vec{b}, \vec{c} \text{ は互いに直交するので} \\ \text{異なるベクトル同士の内積は0} \end{matrix} = \begin{bmatrix} \vec{a} \cdot \vec{a} & \vec{a} \cdot \vec{b} & \vec{a} \cdot \vec{c} \\ \vec{b} \cdot \vec{a} & \vec{b} \cdot \vec{b} & \vec{b} \cdot \vec{c} \\ \vec{c} \cdot \vec{a} & \vec{c} \cdot \vec{b} & \vec{c} \cdot \vec{c} \end{bmatrix} \begin{bmatrix} x_{view} \\ y_{view} \\ z_{view} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{view} \\ y_{view} \\ z_{view} \end{bmatrix} = \begin{bmatrix} x_{view} \\ y_{view} \\ z_{view} \end{bmatrix}$$

$R^{-1} = R^T$ だったことが分かる

同次座標でまとめなおすと, 行列部分が M_{view} である

$$\begin{bmatrix} x_{view} \\ y_{view} \\ z_{view} \\ 1 \end{bmatrix} = \begin{bmatrix} a_x & a_y & a_z & 0 \\ b_x & b_y & b_z & 0 \\ c_x & c_y & c_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -e_x \\ 0 & 1 & 0 & -e_y \\ 0 & 0 & 1 & -e_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} a_x & a_y & a_z & -a_x e_x - a_y e_y - a_z e_z \\ b_x & b_y & b_z & -b_x e_x - b_y e_y - b_z e_z \\ c_x & c_y & c_z & -c_x e_x - c_y e_y - c_z e_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$