Graphics with Processing

2007-07 3次元描画の基礎

http://vilab.org

塩澤秀和

7.1 3D図形の描画

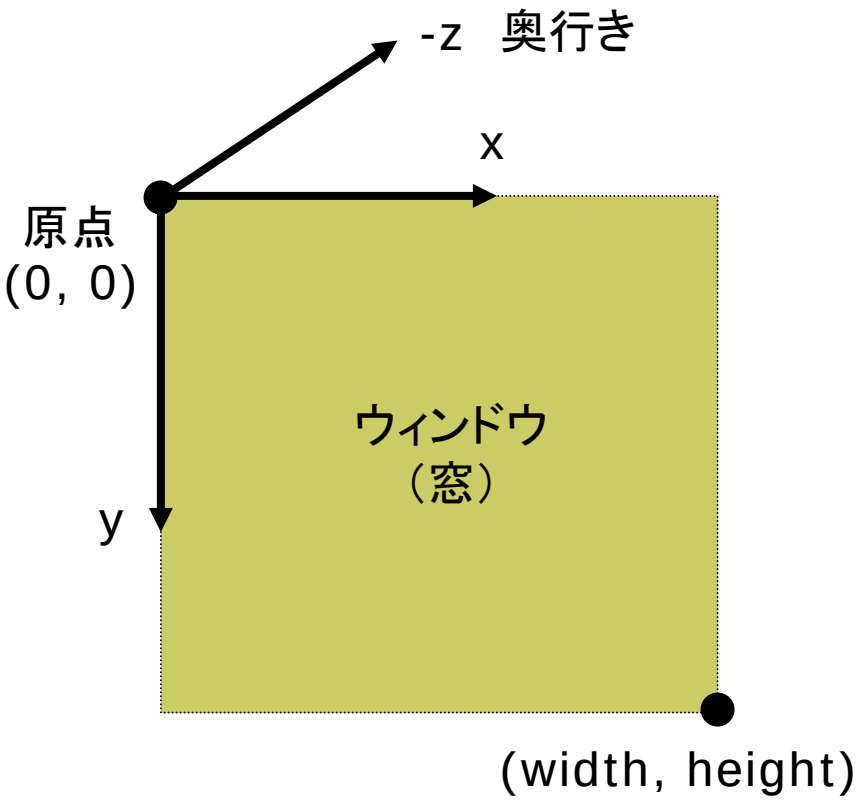
3D設定

- size(幅, 高さ, P3D)
  - 3D(と2D)が使えるウィンドウ
- lights()
  - 標準の照明を設定
  - draw()内に書いたほうがいい

3D基本形状

- box(辺の長さ)
- box(幅, 高さ, 奥行き)
  - 原点に立方体・直方体を描画
- sphere(半径)
  - 原点に球を描画
- サンプル
  - 3D and OpenGL → Forms → Primitives3D

3次元座標系(無指定時)



7.2 3Dでの位置指定

3次元幾何変換

- translate(tx, ty, tz)
  - 3次元平行移動
- scale(sx, sy, sz)
  - 3次元拡大・縮小
- rotateX(θx)
  - x軸まわりの回転
  - x軸を回転軸とした回転
- rotateY(θy)
  - y軸まわりの回転
  - y軸を回転軸とした回転
- rotateZ(θz)
  - z軸まわりの回転
  - 2次元のrotate(θz) と同じ

変換行列の操作

- pushMatrix()
  - 現在の変換行列を一時保存
  - スタックの一番上に積む
- popMatrix()
  - 最近保存した変換行列を戻す
  - スタックの上から取り出す
  - pushMatrixと必ず対にする
- 描画例

```
pushMatrix();
  translate(150, 100, -100);
  rotateY(radians(30));
  box(150, 50, 100);
popMatrix();
```

7.3 投影関数

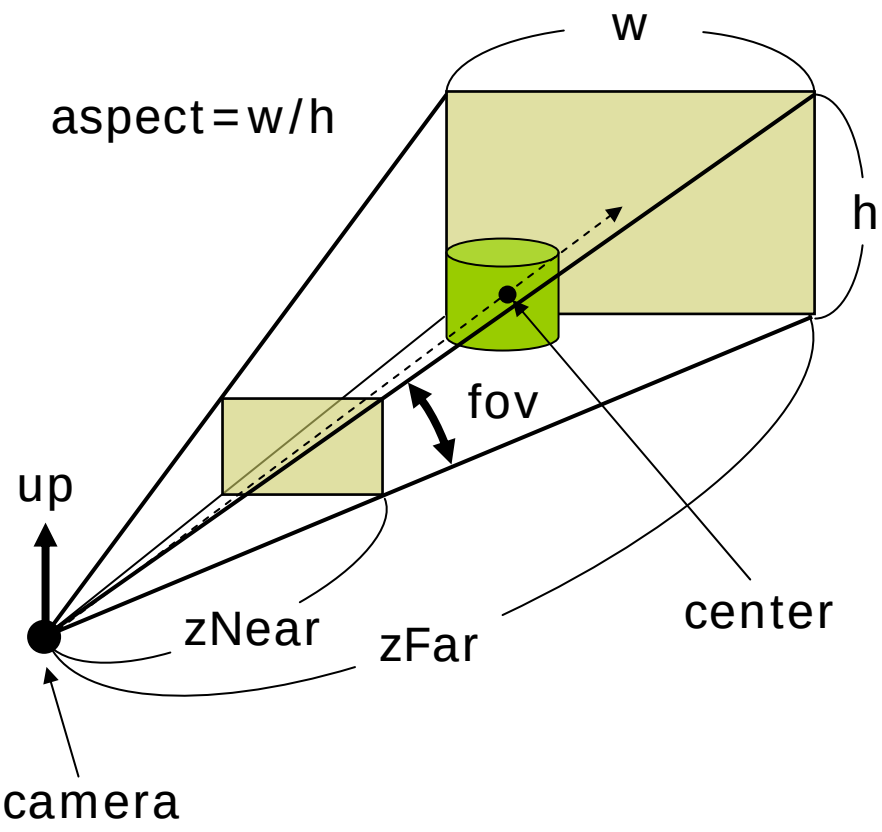
平行投影(直交投影)

- ortho(x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max})
  - 遠近感をつけない投影
  - x_{min}～ x_{max}: x座標の範囲
  - y_{min}～ y_{max}: y座標の範囲
  - z_{min}～ z_{max}: z座標の範囲

透視投影(透視図法)

- perspective(fov, aspect, zNear, zFar)
  - 遠くのものを小さく描く遠近法
  - fov: 視野角(ラジアン)
  - aspect: 視体積の縦横比
  - zNear, zFar: クリッピング範囲
  - 無指定でも適当な設定がされる

視体積(view volume)



7.4 視点位置と演習課題

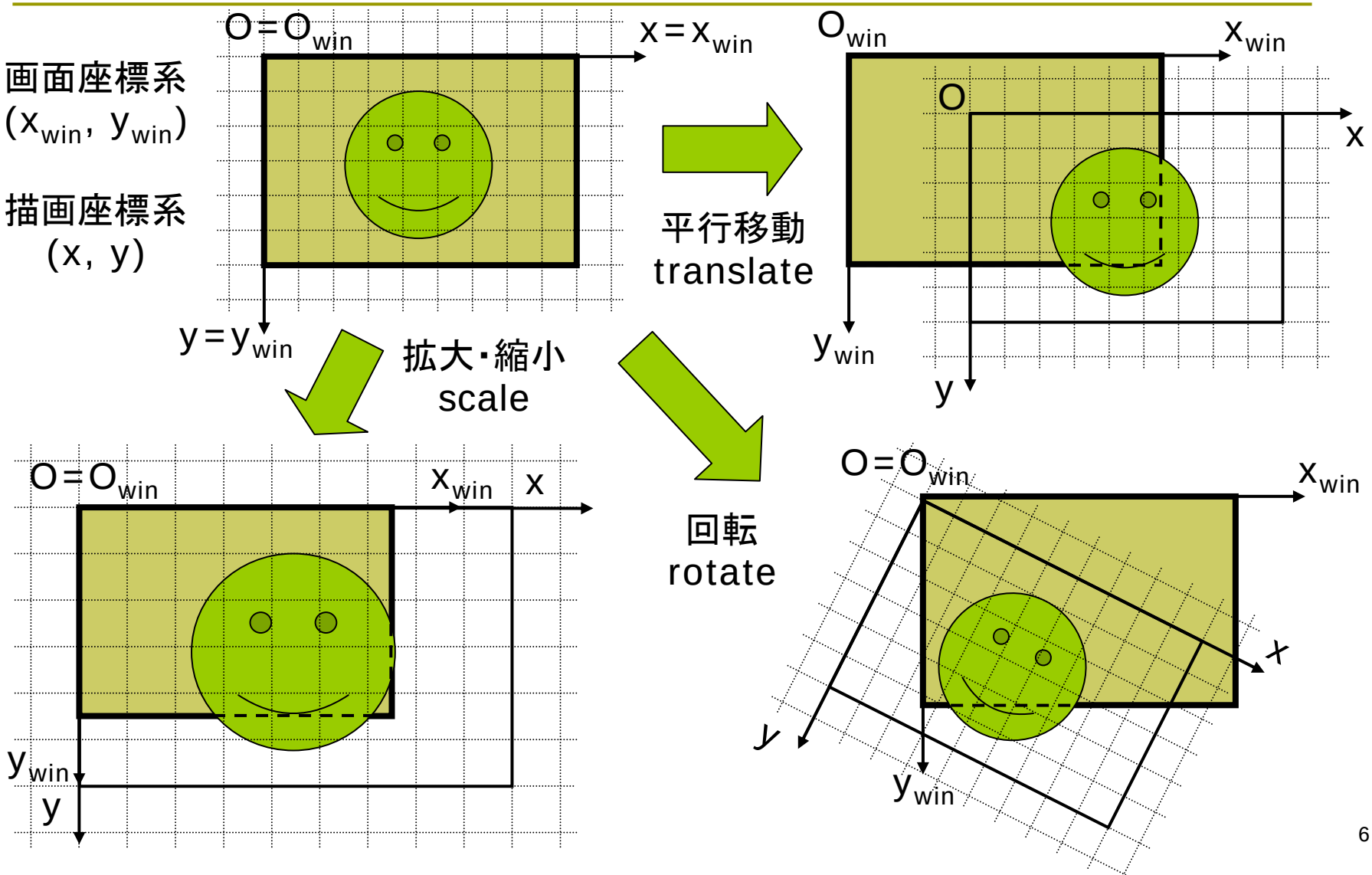
視点

- 幾何変換による設定
  - 視点の移動・回転＝描画図形の逆方向への移動・回転
  - 視点位置を設定するかわりに，図形を見える位置に移動
- camera(eyeX, eyeY, eyeZ, centerX, centerY, centerZ, upX, upY, upZ)
  - 視点の設定関数
  - 無指定時は，7.1の図のように見える適当な値が設定される
  - eye：カメラ（視点）の座標
  - center：カメラで狙う座標
  - up：上方向を示すベクトル（通常は各要素は，0か±1）

課題

- 1つの3D図形をウィンドウの真ん中に見えるように描画するプログラムを作成しなさい
  - できれば，平行投影と透視投影を切り替えられるようにしなさい
- 注意
  - 未提出の課題のある人は，必ず提出すること
  - プログラムは，ただ「動けばいい」ものじゃありません。分かりやすくきれいに書かないといけません
  - 会社に入ったら，1つのソフトで1000行ぐらいは1人で分担して，しかもちゃんと他人が見ても分かるように書かないといけません

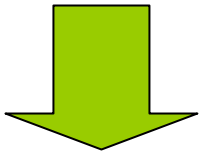
4.2 幾何変換の効果


画面座標系
(x_win, y_win)
描画座標系
(x, y)
O=O_win
x=x_win
y=y_win
平行移動
translate
O_win
x_win
O
x
y_win
y
拡大・縮小
scale
O=O_win
x_win
x
y_win
y
回転
rotate
O=O_win
x_win
x
y
y_win

4.4 同次座標系

同次行列

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}$$



数学的に
より簡便な表記

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & e \\ c & d & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

行列1つですべての座標変換を表せる

同次座標系による表現

- 平行移動

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

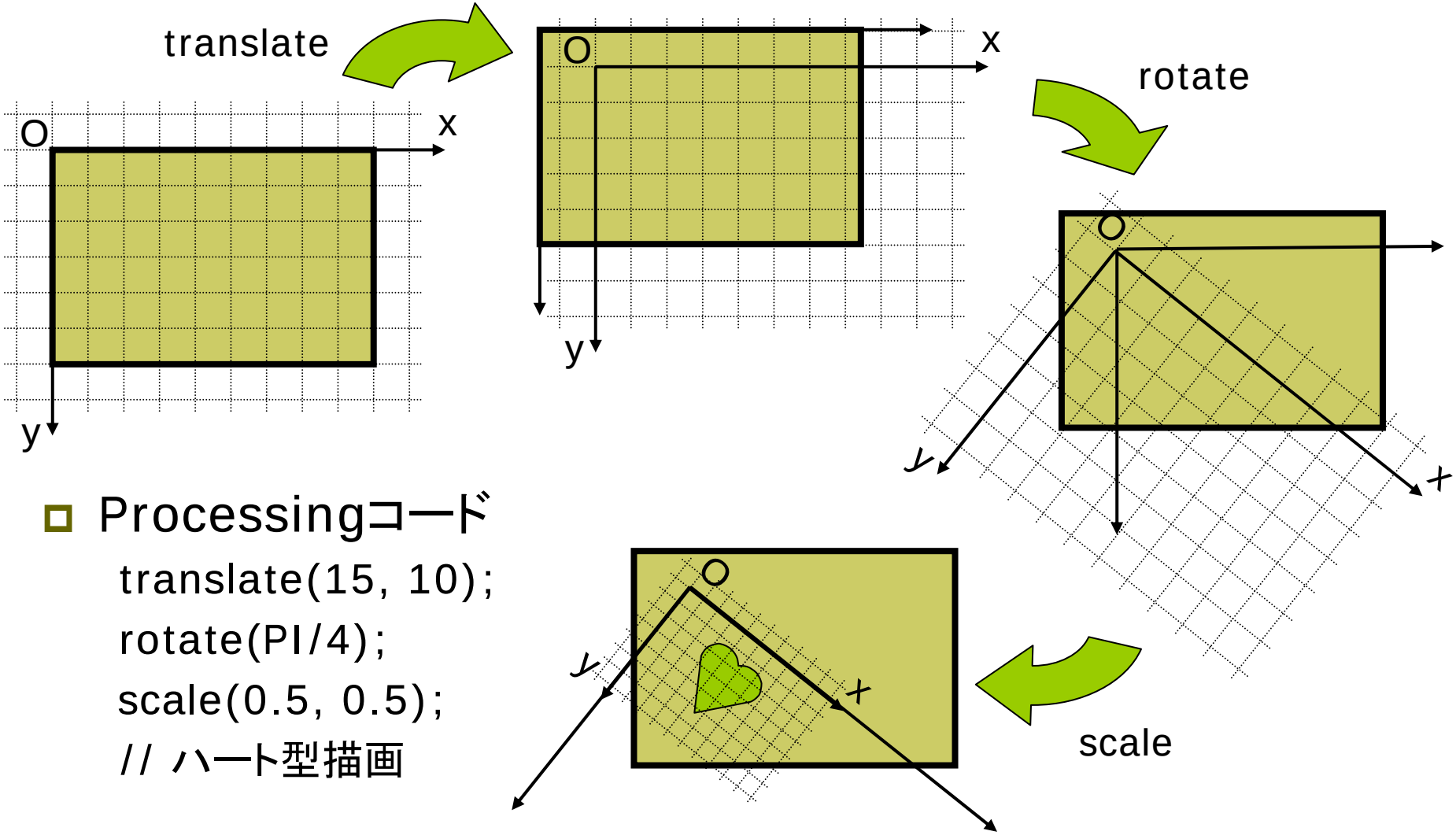
- 拡大縮小

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- 回転

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

4.5 幾何変換の合成



- Processingコード

```
translate(15, 10);
rotate(PI/4);
scale(0.5, 0.5);
// ハート型描画
```

4.6 合成変換行列

合成変換の数学表現

- 同次変換行列の積になる

$$P_{win} = M_1 M_2 M_3 \cdots M_n P$$

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

- Processingコード（4.3の例）

```
translate(15, 10);  // 変換 M1
rotate(PI/4);       // 変換 M2
scale(0.5, 0.5);    // 変換 M3
// 図形描画...
```

- 右上の例の行列表現

$$\begin{bmatrix} x_{win} \\ y_{win} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 15 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\pi/4) & -\sin(\pi/4) & 0 \\ \sin(\pi/4) & \cos(\pi/4) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x_{win} \\ y_{win} \\ 1 \end{bmatrix} = \begin{bmatrix} \sqrt{2}/4 & -\sqrt{2}/4 & 15 \\ \sqrt{2}/4 & \sqrt{2}/4 & 10 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad \therefore M = \begin{bmatrix} \sqrt{2}/4 & -\sqrt{2}/4 & 15 \\ \sqrt{2}/4 & \sqrt{2}/4 & 10 \\ 0 & 0 & 1 \end{bmatrix}$$