Graphics with Processing

2013-07 3DCGとモデリングの基礎

http://vilab.org

塩澤秀和

7.1 3D図形の描画

3D基本設定

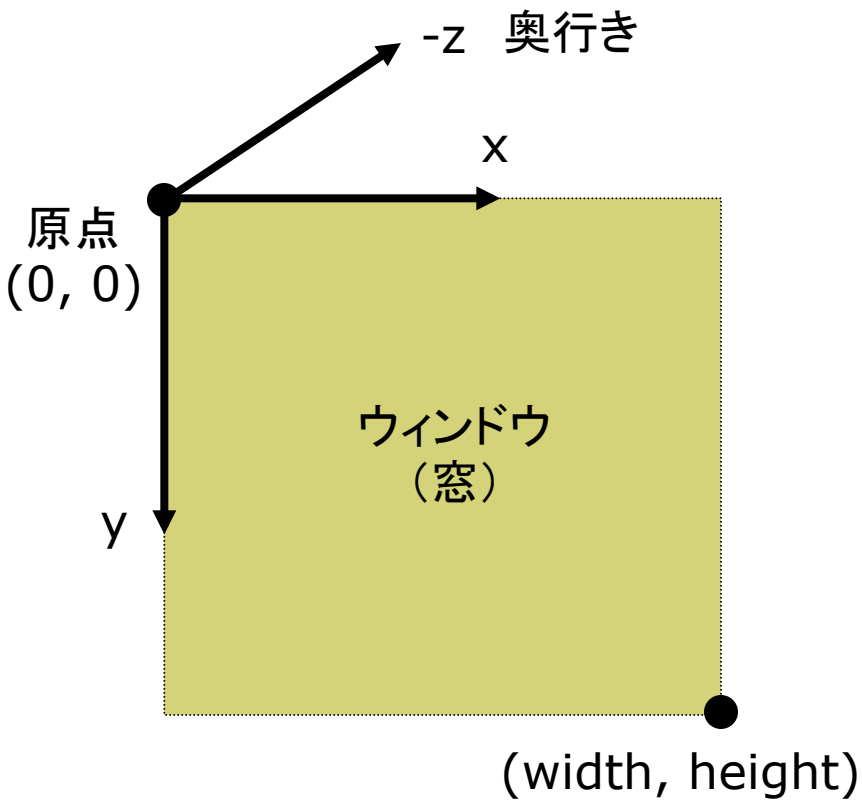
- size(幅, 高さ, P3D)
  - ウィンドウを3D用で開く
  - 自分のPCでうまく動かない場合は，バージョン1.5をインストールする（PCのOpenGL対応の問題）
- lights()
  - 標準の照明を設定
  - draw()のなかで最初に書く

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
  - 通常は noStroke() で描く

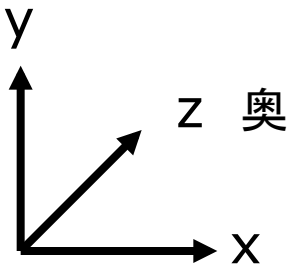
3次元座標系（無指定時）

- Processingではz軸は手前方向

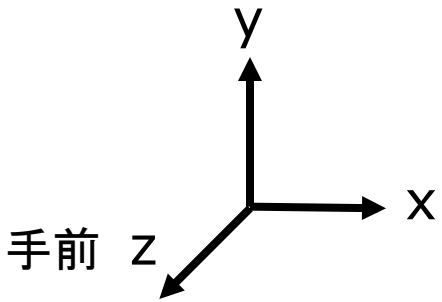


7.2 座標系のとり方(p.26)

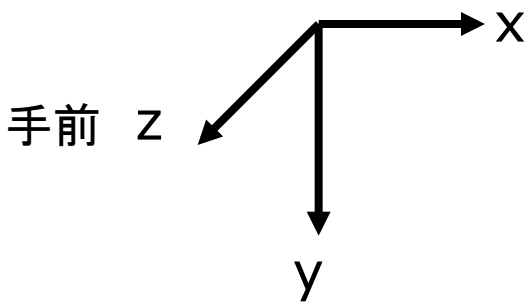
- 左手系
  - 視点座標系・CGゲーム
  - DirectX



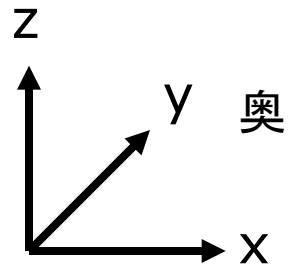
- 右手系
  - CG理論・数学・工学分野
  - OpenGL



- 左手系
  - Processing



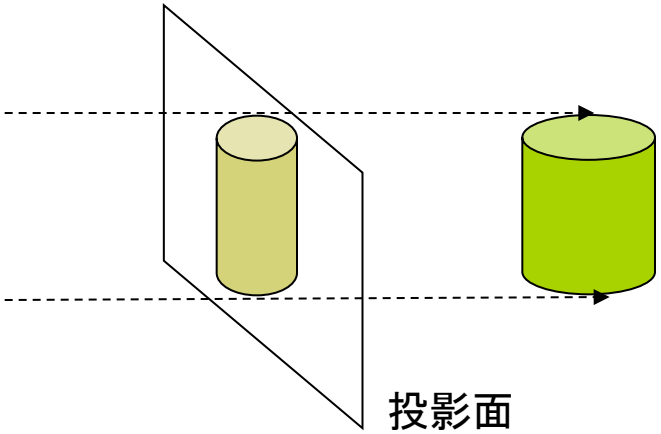
- 右手系
  - 建築座標系



7.3 平行投影と透視投影(p.32)

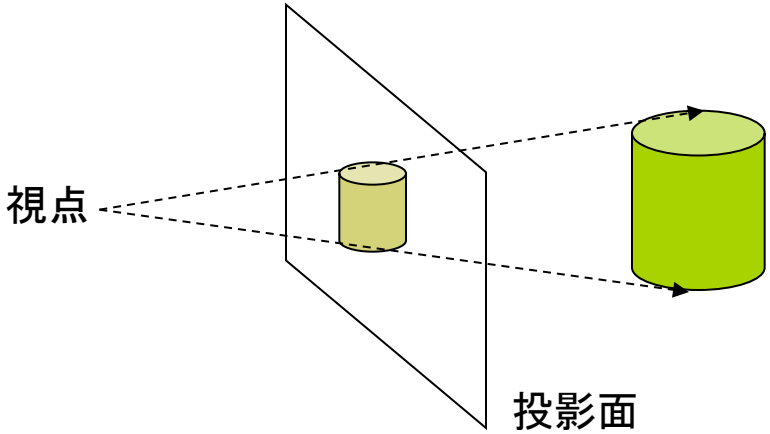
平行投影(直交投影)

- ortho(x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max})
  - 遠近感をつけない投影方法
  - 画面に表示するx, y, z座標の範囲(視体積)を設定
- サンプル
  - Basics (3D) → Camera



透視投影(透視図法)

- pserspective()
  - 近くのものを大きく，遠くのものを小さく，遠近法を使って描画する
- perspective(fov, aspect, zNear, zFar)
  - 視野角(画角)などを指定できる
  - 詳しくは第9回で説明



7.4 3Dでの位置設定

3Dでの位置設定

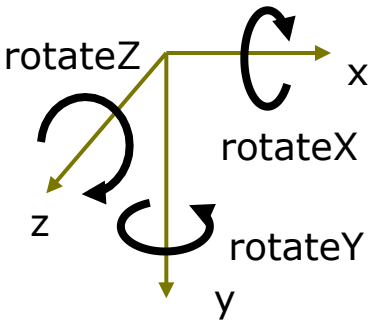
- 座標変換を駆使せよ
  - 3DCGでは，幾何変換で図形を配置する考え方が必須!!
  - boxもsphereもそのときの描画座標系の原点付近に図形を描く

行列スタックの操作

- pushMatrix()
  - 変換行列（論理座標系）を一時的に退避する
  - 使い方は，2次元と同じ
- popMatrix()
  - 最近保存した論理座標系を戻す
  - pushとpopは必ず対にすること

3次元幾何変換

- translate(t_x, t_y, t_z)
  - 座標系の平行移動
  - 最初に (width/2, height/2, 0) に原点をもってくると分かりやすい
- scale(s_x, s_y, s_z)
  - 座標系の拡大・縮小
  - 原点が中心に全体が拡大
- rotateX(θ_x)
  - x軸まわりの回転
- rotateY(θ_y)
  - y軸まわりの回転
- rotateZ(θ_z)
  - z軸まわりの回転
  - 2次元のrotate(θ_z) と同じ



7.5 3D描画の例

```
// P3Dモードの例
void setup() {
  size(400, 400, P3D);
  noLoop();
}

void draw() {
  background(0);
  // 標準の照明
  lights();
  // 透視投影
  perspective();
  // 原点を移動
  translate(width/2, height/2, 0);
  noStroke();
  fill(255, 200, 200);
  // 原点に半径100の球を描画
  sphere(100);
}
```

```
// バージョン1.xのOPENGLモードの例
// 2.0からはP3DでもOPENGLモード
import processing.opengl.*;

float rot = 0.0;

void setup() {
  size(400, 400, OPENGL);
}

void draw() {
  background(70);
  lights(); perspective();
  translate(width/2, height/2, 0);
  pushMatrix();
     rotateY(radians(rot++));
     stroke(255, 0, 0);
     fill(255, 255, 0);
     box(100);
  popMatrix();
}
```

OpenGL のとき必要

7.6 モデリングの基礎(p.27)

モデリング

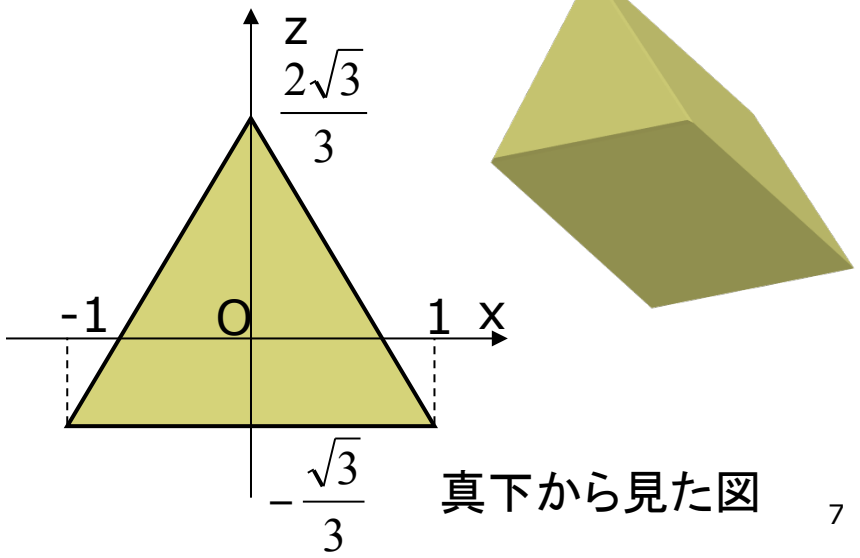
- モデリングとは
  - 3Dオブジェクト(物体)の形状を数値データの集合で表すこと
  - 複雑なモデリングは専用のソフトウェアを使う(第14回に説明)

形状モデル(p.48)

- ワイヤーフレームモデル
  - 線の集合で物体を表現する
- サーフェスモデル
  - ポリゴン(多角形)の集合で物体の表面(だけ)を表す
- ソリッドモデル
  - 物体の内外を示す情報もあり、中身が詰まっているモデル

簡単なモデリング

- ポリゴンの描画
  - ポリゴンpolygon ＝ 多角形
  - 物体表面のポリゴンを描画する(beginShape～endShape)
- 例)三角柱
  - 幅＝2(-1≦x≦1), 原点に重心, 高さ＝2(-1≦y≦1)



真下から見た図

7.7 ポリゴンの描画例

prism（プリズム）は角柱という意味

```
// 回転する三角柱を表示する
float rot = 0.0;

void setup() {
  size(400, 400, P3D);
}

void draw() {
  background(0);
  lights(); perspective();
  translate(width/2, height/2);
  pushMatrix();
    rotateX(radians(rot++));
    fill(255, 255, 0);
    // scaleで適当な大きさに拡大
    scale(30, 60, 30);
    prism3();
  popMatrix();
}
```

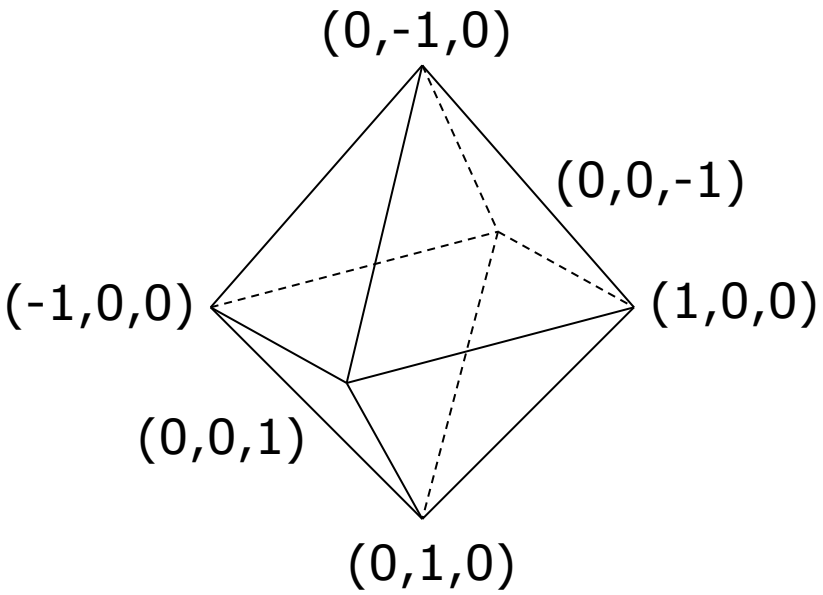
```
void prism3() {
  float g = sqrt(3) / 3.0;
  // 側面の3枚の長方形
  beginShape(QUAD_STRIP);
    vertex(1, -1, -g); vertex(1, 1, -g);
    vertex(0, -1, g*2); vertex(0, 1, g*2);
    vertex(-1, -1, -g); vertex(-1, 1, -g);
    vertex(1, -1, -g); vertex(1, 1, -g);
  endShape();
  // 底面と上面の三角形
  beginShape(TRIANGLES);
    vertex(1, -1, -g);
    vertex(0, -1, g*2);
    vertex(-1, -1, -g);
    vertex(1, 1, -g);
    vertex(0, 1, g*2);
    vertex(-1, 1, -g);
  endShape();
}
```

7.8 演習課題

課題

問1) 正八面体を描画するプログラムを作成しなさい

- 8枚の正三角形を描画する
- beginShapeでTRIANGLESかTRIANGLE_FANを用いる
- もっと凝った図形をやってもよい



$$A = \begin{bmatrix} 3.0 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 0 & 40 \\ 0 & 1 & 20 \\ 0 & 0 & 1 \end{bmatrix}$$

$$C = \begin{bmatrix} \cos 45^\circ & -\sin 45^\circ & 0 \\ \sin 45^\circ & \cos 45^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

問2)《前回の復習》

2次元幾何変換A～Cについて以下の問いに答え，**A4用紙**で提出しなさい

1. 合成変換行列ABを計算しなさい
2. 変換ABの後に座標 (20, 60) に点を打つと，画面のどこに表示されるか？
3. 合成変換行列BAを計算し，ABとの意味の違いを説明しなさい
4. 行列Cに対応するProcessingの命令を示しなさい（定数PIを用いてもよい）
5. 合成変換行列C^2=CCを計算し，どのような変換か説明しなさい