Graphics with Processing

2023-07 3DCGとモデリングの基礎

https://vilab.org

塩澤秀和

7.1 3D図形の描画

3D基本設定

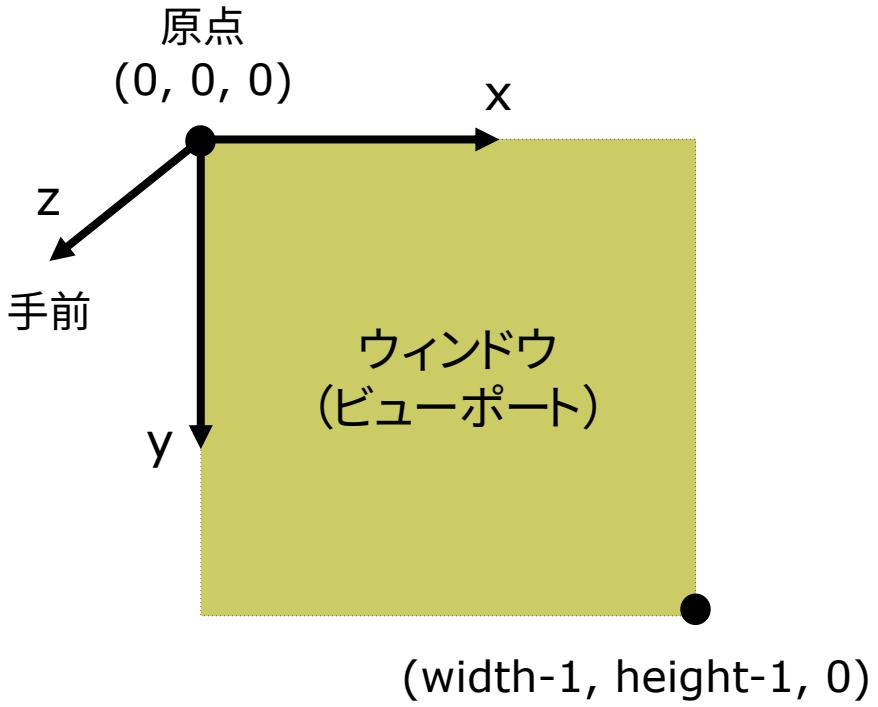
- size(幅, 高さ, P3D)
  - ウィンドウを3D用で開く
- lights()
  - 標準の照明を設定
  - draw()のなかで最初に書く
- perspective()
  - 透視投影に設定（第9回）

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

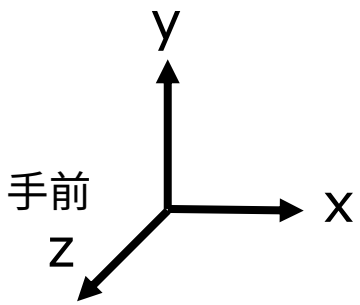
3次元座標系（無指定時）

- Processingではz軸は手前方向
  - 左手系：x,y,z軸の向きが、左手の親指、人さし指、中指に対応

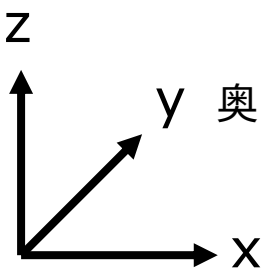


7.2* 座標系のとり方(p.32)

- 右手系
  - 数学・物理学・工学で一般的

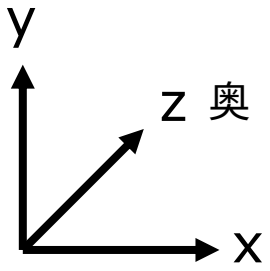


OpenGL,Maya

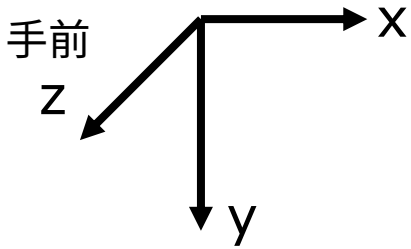


建築座標系,Blender

- 左手系
  - CG・ゲームで多くの例



DirectX,Unity



Processing

- 縦ベクトル表記
  - 数学・工学で一般的

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ a_3 & b_2 & c_2 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- 横ベクトル表記
  - 成分は転置、掛け算の順序は逆

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1] \begin{bmatrix} a_1 & a_2 & a_3 & 0 \\ b_1 & b_2 & b_3 & 0 \\ c_1 & c_2 & c_3 & 0 \\ d_1 & d_2 & d_3 & 1 \end{bmatrix}$$

7.3 3Dでの位置設定

3Dでの位置設定

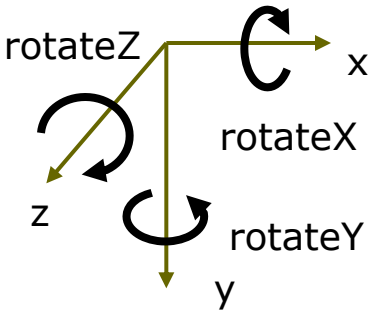
- 座標変換を駆使せよ
  - 3DCGでは、幾何変換で図形を配置する考え方が必須!!
  - boxもsphereもそのときの描画座標系の原点付近に図形を描く
  - 「原点」と「拡大率」を常に意識!

行列スタックの操作

- pushMatrix(), push()
  - システム変換行列(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(), pop()
  - 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(tx, ty, tz)
  - 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(sx, sy, sz)
  - 座標系の拡大・縮小
  - 原点を中心に全体が拡大
- rotateX(θx)
  - x軸まわりの回転
- rotateY(θy)
  - y軸まわりの回転
- rotateZ(θz)
  - z軸まわりの回転
  - 2次元のrotate(θz) と同じ



7.4 3D描画の例

```
void setup() {
  // P3Dモードでウィンドウを開く
  size(400, 400, P3D);
  frameRate(30);
}

void draw() {
  background(0);
  float a = radians(frameCount);

  lights(); // 標準の照明
  perspective(); // 透視投影

  // 原点を画面中心に移動
  translate(width/2, height/2, 0);
  stroke(255, 0, 0);
  fill(255, 255, 0 );

  pushMatrix();
    translate(-100, 0, 0);
    rotateY(a);
    box(100);
  popMatrix();

  pushMatrix();
    translate(100, 0, 0);
    rotateX(a);
    // sphereDetail(10);
    sphere(70);
  popMatrix();
}
```

noFillも試してみよう

球の細かさ

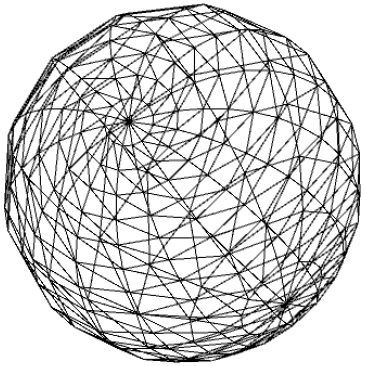
7.5* モデリングの基礎

モデリング

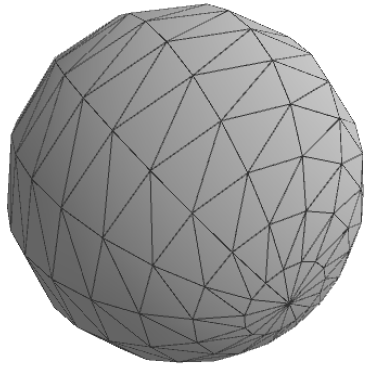
- モデリングとは（p.33）
  - 3Dオブジェクト（物体）の形状を数値データの集合で表すこと

形状モデル（p.60）

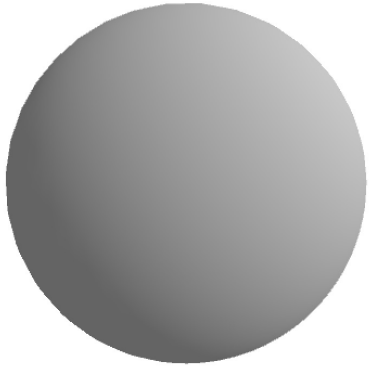
- ワイヤーフレームモデル
  - 線の集合で物体を表現する
- サーフェスモデル
  - 物体の表面（だけ）を表す
  - 通常はポリゴン（多角形）の集合
- ソリッドモデル
  - 物体の内外を示す情報もあり、中身が詰まっているモデル
- ポイントクラウド（点群）
  - 点（＋色）の集合によるデータ



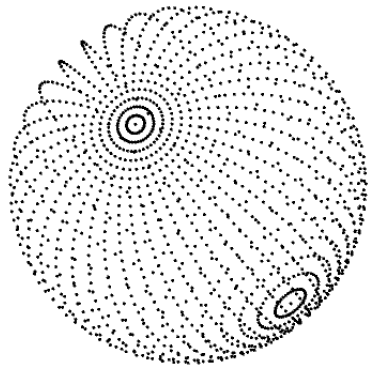
ワイヤーフレームモデル



サーフェスモデル



ソリッドモデル
（中身が詰まっている）



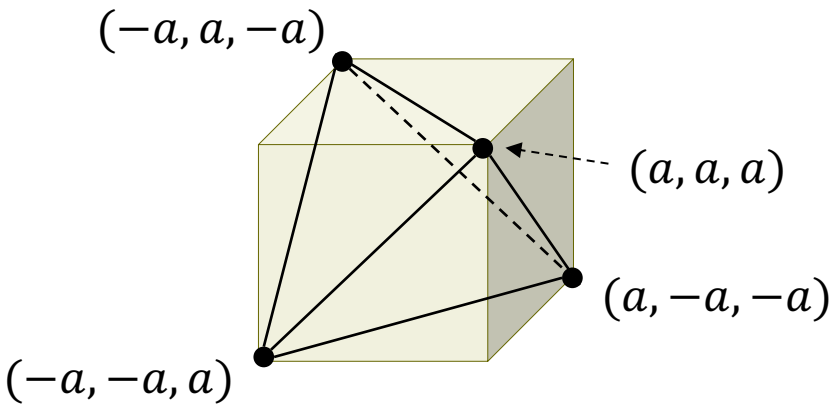
ポイントクラウド

7.6 ポリゴンモデルの描画例

7.4に
追加せよ

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）正四面体（tetrahedron）
  - 立方体（正六面体）の8個の頂点から1つおきに4個選び、対角線を結ぶように辺を作るのが容易




```
// 正四面体の描画
void tetrahedron(float a) {
  beginShape(TRIANGLES);
  vertex(a, a, a);
  vertex(-a, a, -a);
  vertex(a, -a, -a);          面1
  vertex(a, a, a);
  vertex(a, -a, -a);
  vertex(-a, -a, a);          面2
  vertex(a, a, a);
  vertex(-a, -a, a);
  vertex(-a, a, -a);          面3
  vertex(a, -a, -a);
  vertex(-a, -a, a);
  vertex(-a, a, -a);          面4
  endShape();
}
```

7.7 3Dモデルデータの利用

3Dモデルデータ

- PShape型
  - P3DではOBJデータが使える
- loadShape("ファイル名")
  - モデルデータの読み込み
  - setup()内で準備する
- createShape(図形, サイズ)
  - SPHERE, BOXのモデルの生成
  - テクスチャを貼りたいときに便利
- shape(model)
- shape(model, x, y, z)
  - モデルの表示
- 描画色の変更
  - model.setFill(色or真偽値)
  - model.setTexture(画像)

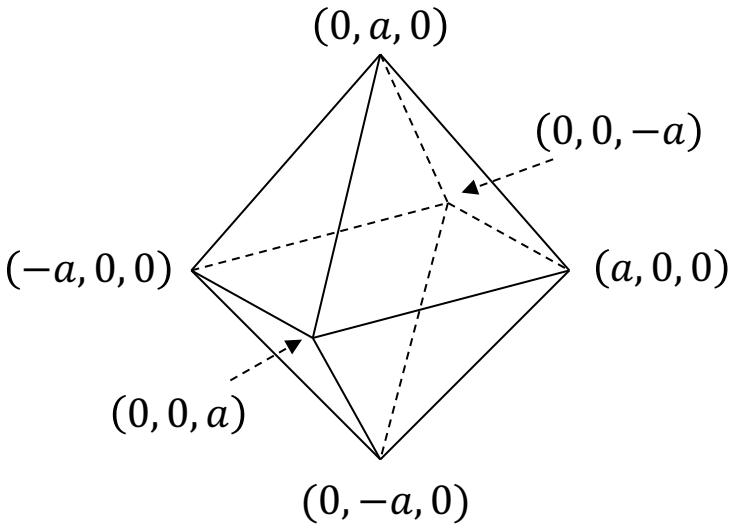
```
PShape model;

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

  // beethoven.{obj,mtl,png}の3つの
  // ファイルをdataフォルダに入れておく
  model = loadShape("beethoven.obj");
  //model.setTexture(null);
}

void draw() {
  background(0, 0, 100);
  lights();
  perspective();
  translate(width/2, height/2, 0);
  rotateX(PI);
  rotateY(radians(frameCount));
  scale(200);
  shape(model);
}
```

7.8 演習課題

課題

問1) 正八面体(octahedron)が1つ以上と他の3D図形(3Dモデル)を使った3Dシーンを表示するプログラムを作成しなさい

- 正八面体は、8枚の正三角形で構成する
- 回転などで裏側も確認すること



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

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

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用紙に記入して提出

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、表示される画面座標は何か?
3. 合成変換行列BAを計算して、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

7.9 参考：点群による球の描画

```
PImage img;  // 球の表面画像

void setup() {
  size(600, 600, P3D);
  img = loadImage("earth.jpg");
  img.loadPixels();
}

void draw() {
  background(0);
  lights(); perspective();
  translate(width/2, height/2);
  pushMatrix();
    rotateX(radians(frameCount)/2);
    rotateY(radians(frameCount));
    pointSphere(200, 2, img);
  popMatrix();
}
```

```
// 点群による球面の描画例
void pointSphere(float r, int d, PImage g) {
  strokeWeight(d * 4);

  // 緯度(lat)と経度(lng)による2重ループ
  for (int lat = 90 - d; lat > -90; lat -= d) {
    float a = radians(lat);
    int e = (int)(d / cos(a));  // 緯度で間隔調整
    for (int lng = -180; lng < 180; lng += e) {

      // 表面画像の対応点から色を抽出
      int u = (lng + 180) * g.width / 360;
      int v = (-lat + 90) * g.height / 180;
      stroke(g.pixels[u + v * g.width]);

      // 緯度・経度から球面上の3D座標を計算
      float b = radians(lng);
      point(r * cos(a) * cos(b), r * sin(a),
            r * cos(a) * sin(b));
    }
  }
}
```

7.10 参考：背景表示とベクトルクラス

3Dシーンの背景表示

- スカイボックス / スカイドーム
  - シーン全体を包み込む立方体や球に背景のテクスチャを貼る

```
PShape bg;
PImage tex;

void setup() {
  size(800, 600, P3D);
  tex = loadImage("park360.jpg");
  bg = createShape(SPHERE, 800);
  bg.setTexture(tex);
  bg.setStroke(false);
}

void draw() {
  perspective();
  translate(width/2, height/2, 0);
  rotateY(radians(frameCount)/4);
  noLights(); shape(bg);
}
```

テクスチャを貼った球を作る

ベクトルクラス（一部）

成分から生成	v = new PVector(x, y, z)
角度から生成	v = PVector.fromAngle(a)
ランダムに生成	v = PVector.random3D()
複製	u = v.copy()
ベクトルの和	v.add(u) w = PVector.add(v, u)
ベクトルの差	v.sub(u) w = PVector.sub(v, u)
スカラー倍	v.mult(s) w = PVector.mult(v, s)
内積	s = v.dot(u)
外積	w = v.cosss(u)
大きさ（ノルム）	s = v.mag()
ノルムの2乗	s = v.magSq()
正規化	v.normalize()
ベクトルの角度	v.heading()

7.11 参考：3DCGソフトウェア紹介

3DCGソフトウェア

- MagicaVoxel
  - ephtracy.github.io
  - Minecraftのようにボクセル（立方体）の集合でモデリング
- Blender
  - www.blender.org
  - 高機能でフリー&オープンソース
- Maya / 3ds Max など
  - Autodesk社のプロ向け製品
  - 学生は無償で利用可能
  - www.autodesk.co.jp/education
- SketchUp for Web
  - www.sketchup.com/ja/products/sketchup-for-web
  - 建物・人工物のモデリングに向く
- ScupltGL (Web)
  - stephaneginier.com/sculptgl/
  - 粘土・彫刻のようにモデリング
- 3DF Zephyr
  - www.3dflow.net/3df-zephyr-free/
  - 多数の写真から3Dモデルを構築
- MeshLab
  - www.meshlab.net
  - 3Dモデルデータの表示・変換
- 3Dモデルの取得
  - market.pmnd.rs
  - free3d.com
  - www.freepik.com/3d-models
  - archive3d.net
  - www.cgtrader.com
  - www.turbosquid.com