Graphics with Processing

2010-13 モデリング（1）

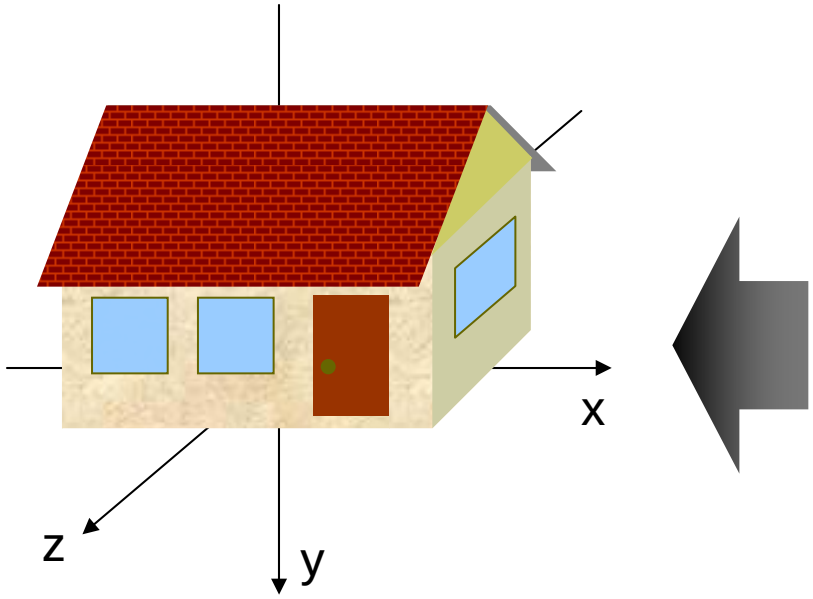
http://vilab.org

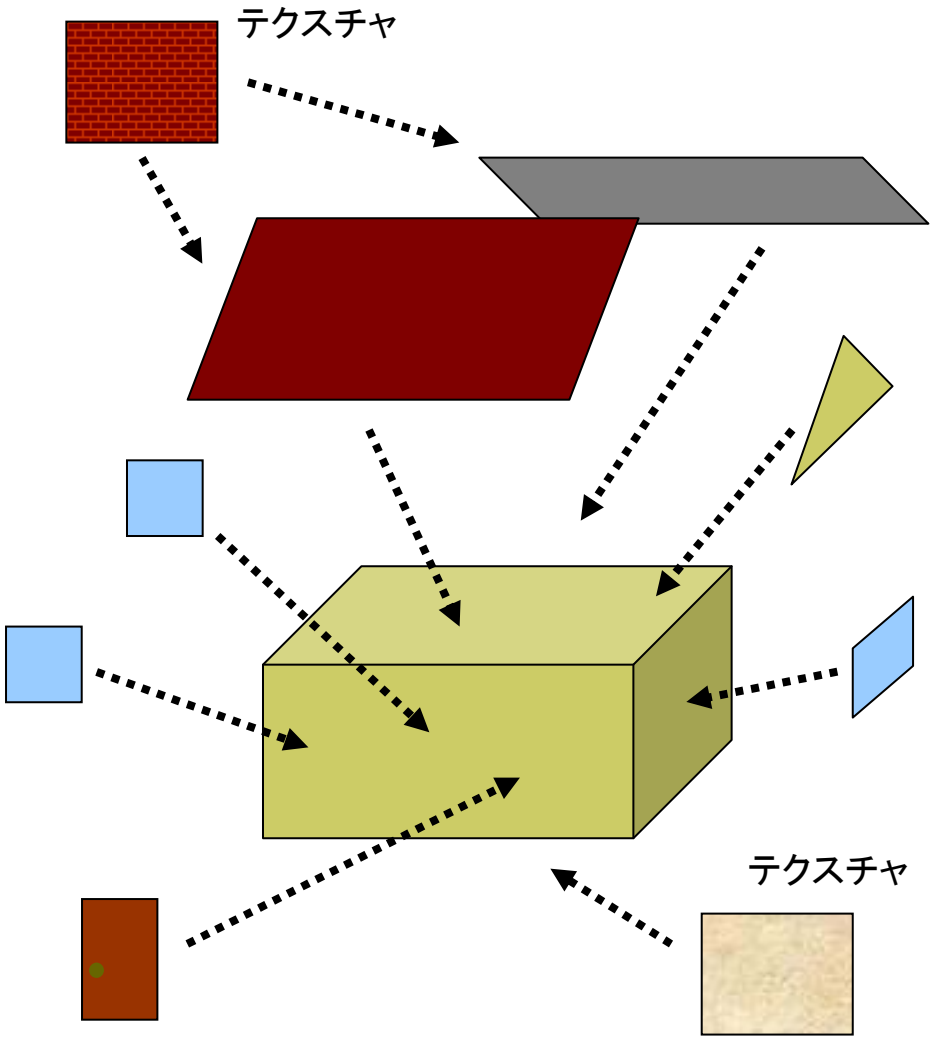
塩澤秀和

13.1 3Dモデリング

モデリング

- 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
- オブジェクト座標系で基本図形やポリゴンを組み合わせる





13.2 オブジェクトの関数化

複雑なオブジェクトは，大きさ1を目安としてモデリングし，関数にしておくと利用しやすい

雪だるま

```
void snowman() {
  fill(255, 255, 255);
  noStroke();
  pushMatrix();
  translate(0, -0.7);
  sphere(0.2);
  popMatrix();
  pushMatrix();
  translate(0, -0.3);
  sphere(0.3);
  popMatrix();
}
```

円錐（底なし）

```
void cone() {
  pushMatrix();
  beginShape(TRIANGLE_FAN);
  vertex(0, -1, 0);
  for (int th = 0; th <= 360;
       th += 10) {
    float x = cos(radians(th));
    float z = sin(radians(th));
    vertex(x, 0, z);
  }
  endShape();
  popMatrix();
}
```

木（のようなもの）

```
void tree() {
  pushMatrix();
  fill(0, 255, 0);
  translate(0, -0.3, 0);
  scale(0.2, 0.7, 0.2);
  cone();
  popMatrix();
  pushMatrix();
  fill(100, 0, 0);
  scale(0.1, 1, 0.1);
  cone();
  popMatrix();
}
```

13.3 少し複雑なモデリング例

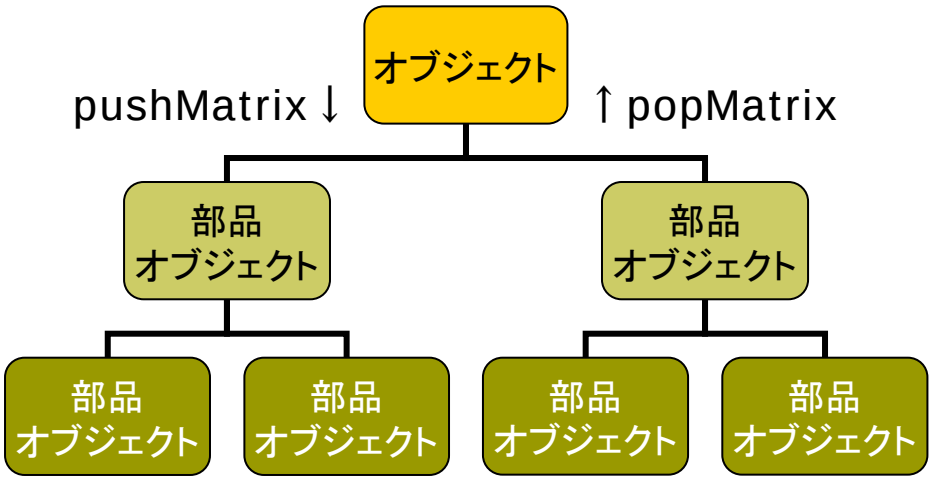
```
// OPENGLのほうが正確
// size(幅, 高さ, OPENGL);
// P3Dだとテクスチャが歪む

void house()
{
  // 壁
  pushMatrix();
  translate(0, -0.5, 0);
  fill(#ffffaa);
  box(2, 1, 1.4);
  popMatrix();
  // 屋根の下
  beginShape(TRIANGLES);
  vertex(1, -1, 0.7);
  vertex(1, -1.7, 0);
  vertex(1, -1, -0.7);
  vertex(-1, -1, 0.7);
  vertex(-1, -1.7, 0);
  vertex(-1, -1, -0.7);
  endShape();
  // 屋根
  beginShape(QUAD_STRIP);
  fill(#ffffff);
  // テクスチャはsetup()の中で
  // roof = loadImage("roof.jpg");
  // として読み込んでおく
  texture(roof);
  textureMode(NORMALIZED);
  vertex(-1.1, -0.8, 0.9, 0, 1);
  vertex(1.1, -0.8, 0.9, 1, 1);
  vertex(-1.1, -1.7, 0, 0, 0);
  vertex(1.1, -1.7, 0, 1, 0);
  vertex(-1.1, -0.8, -0.9, 0, 1);
  vertex(1.1, -0.8, -0.9, 1, 1);
  endShape();
  // 煙突
  fill(#880000);
  pushMatrix();
  translate(-0.5, -1.4, -0.5);
  box(0.2, 1, 0.2);
  popMatrix();
  beginShape(QUADS);
  // 窓
  fill(#4444ff);
  float z = 0.701;
  vertex(-0.8, -0.7, z);
  vertex(-0.8, -0.3, z);
  vertex(-0.4, -0.3, z);
  vertex(-0.4, -0.7, z);
  vertex(-0.2, -0.7, z);
  vertex(-0.2, -0.3, z);
  vertex(0.2, -0.3, z);
  vertex(0.2, -0.7, z);
  // ドア
  fill(#883333);
  vertex(0.4, -0.8, z);
  vertex(0.4, -0.1, z);
  vertex(0.8, -0.1, z);
  vertex(0.8, -0.8, z);
  endShape();
}
```

13.4 モデリング技術

階層モデリング（p.45）

- ローカル座標系の階層化
  - 部品はそれぞれの座標系で作り，階層的に大きな部品に組み立てていくようにモデリングする
  - 可動部は，動きの基準点（関節など）を原点として部品化
  - 描画では行列スタックを使う（pushMatrix / popMatrix）



曲面や自然形状の表現

- パラメトリック曲面（p.73）
  - パラメータ方程式による曲面
  - ベジエ曲面やNURBS曲面など
  - レンダリング時にポリゴンに変換する方式としない方式がある
- ポリゴン曲面の操作（p.78）
  - 細分割曲面：ポリゴンを再帰的に分割し，滑らかな面を生成
  - 詳細度制御：視点から遠い曲面のポリゴン数を削減して簡略化
- フラクタル（p.86）
  - 自然界によく見られる再帰的な形状（※）のモデリングに適する

※ 海岸線や木の枝など，一部分が全体の縮小のような形状のもの

13.5 参考:タイポグラフィ(文字表示)

```
// 描画用フォントの変数(PFont型)
PFont font1, font2;

void setup()  {
  size(300, 300, P3D);  // ※ 注意

  // 1. Processing専用フォントの利用例
  //(Tools→Create Font...で作っておく)
  font1 = loadFont("Impact-48.vlw");

  // 2. システムフォント(Java+OS)の利用例
  hint(ENABLE_NATIVE_FONTS);
  font2 = createFont("Century", 48);
}

void draw() {
  background(255);
  translate(width/2, height/2);
  rotateX(radians(frameCount));
  // 座標モードとxy方向の位置あわせ方法
  textMode(MODEL);
  textAlign(CENTER, TOP);

  textFont(font1, 32);    // フォントとサイズ
  fill(128, 0, 0);        // 色
  text("impact", 0, 20);  // 文字列と座標

  fill(0, 0, 128);
  textFont(font2, 64);
  text("century", 0, 80);
}
```

※ 注意:3Dモード(P3DまたはOPENGL)では
日本語の文字列が描画できない。
対策として,オフスクリーンレンダリングを
利用し,2Dの隠し画面に日本語を書いから
それをテクスチャマッピングで3Dモデルの
表面に貼るという方法がある(12.10参照)

13.6 参考:オフスクリーンレンダリング

```
import processing.opengl.*;

PGraphics pg;  // 隠し画面用変数

void setup() {
  size(400, 300, OPENGL);
  // 隠し画面を開く
  // 3つの引数の意味はsize関数と同じ
  pg = createGraphics(100, 100,
         JAVA2D);
}

void draw() {
  // 隠し画面上での描画処理
  pg.beginDraw();  // 開始
  pg.background(255);
  pg.translate(50, 50);
  pg.fill(240, 180, 180);
  pg.rotate(radians(frameCount));
  pg.rect(-100, -3, 200, 6);
  pg.endDraw();    // 終了

  // 表示画面での処理
  background(255);
  lights();
  translate(width / 2, height / 2, 0);
  rotateX(radians(frameCount) / 8);

  scale(90);
  beginShape(QUADS);
  texture(pg);  // 隠し画面を画像として使う
  textureMode(IMAGE);

  vertex(-1, 1, 1,   0,   0);
  vertex( 0, 1, 0,  50,   0);
  vertex( 0,-1, 0,  50, 100);
  vertex(-1,-1, 1,   0, 100);
  vertex( 0, 1, 0,  50,   0);
  vertex( 1, 1, 1, 100,   0);
  vertex( 1,-1, 1, 100, 100);
  vertex( 0,-1, 0,  50, 100);
  endShape();
}
```

13.7 参考:影付けの例

```
// これは難しいのであくまでも参考です
// (Processingではここらへんが限界)
import processing.opengl.*;

PGraphics3D pg;
PImage tex;

float fw =256;
float fh = 256;

float lightX = -100;
float lightY = -200;
float lightZ = -100;

void setup() {
  size(400, 400, OPENGL);
  tex = loadImage("davinci.jpg");
  pg = (PGraphics3D)
    createGraphics(256, 256, P3D);
}
```

```
void draw() {
  pg.beginDraw();
  pg.background(255, 255, 255, 0);
  pg.copy(tex,
     0, 0, tex.width, tex.height,
     0, 0, pg.width, pg.height);

  pg.camera(lightX, lightY, lightZ,
                lightX, 0, lightZ, 0, 0, 1);
  pg.frustum((-pg.width/2  - lightX) * 0.1,
               ( pg.width/2  - lightX) * 0.1,
               (-pg.height/2 - lightZ) * 0.1,
               ( pg.height/2 - lightZ) * 0.1,
               -lightY/10, -lightY);
  pg.noLights();
  pg.scale(pg.width/fw, 1.0 , pg.height/fh);
  pg.stroke(0, 0, 0, 50);
  pg.fill(0, 0, 0, 128);
  drawScene(pg);
  pg.endDraw();
```

13.8 参考:影付けの例(続き)

```
background(0);

camera(0, -300, -300,
       0, 0, 0,  0, 1, 0);
rotateY(radians(frameCount));

pointLight(128, 128, 128,
           lightX, lightY, lightZ);
ambientLight(128, 128, 128);

stroke(0);
fill(255, 0, 0);
drawScene(this.g);

pushMatrix();
translate(lightX, lightY, lightZ);
noStroke();
fill(255, 255, 0);
sphere(5);
popMatrix();
```

```
  textureMode(NORMAL);
  fill(255);
  beginShape(QUADS);
  texture(pg);
  vertex(-fw/2, 0, -fh/2,   0,   0);
  vertex( fw/2, 0, -fh/2,   1,   0);
  vertex( fw/2, 0,  fh/2,   1,   1);
  vertex(-fw/2, 0,  fh/2,   0,   1);
  endShape();
}

void drawScene(PGraphics3D g) {
  g.pushMatrix();
  g.translate(0, -50, 0);
  g.rotateX(radians(frameCount));
  g.rotateZ(radians(frameCount));
  g.box(30, 20, 40);
  g.popMatrix();
}
```