

Graphics with Processing



2009-14 モデリング

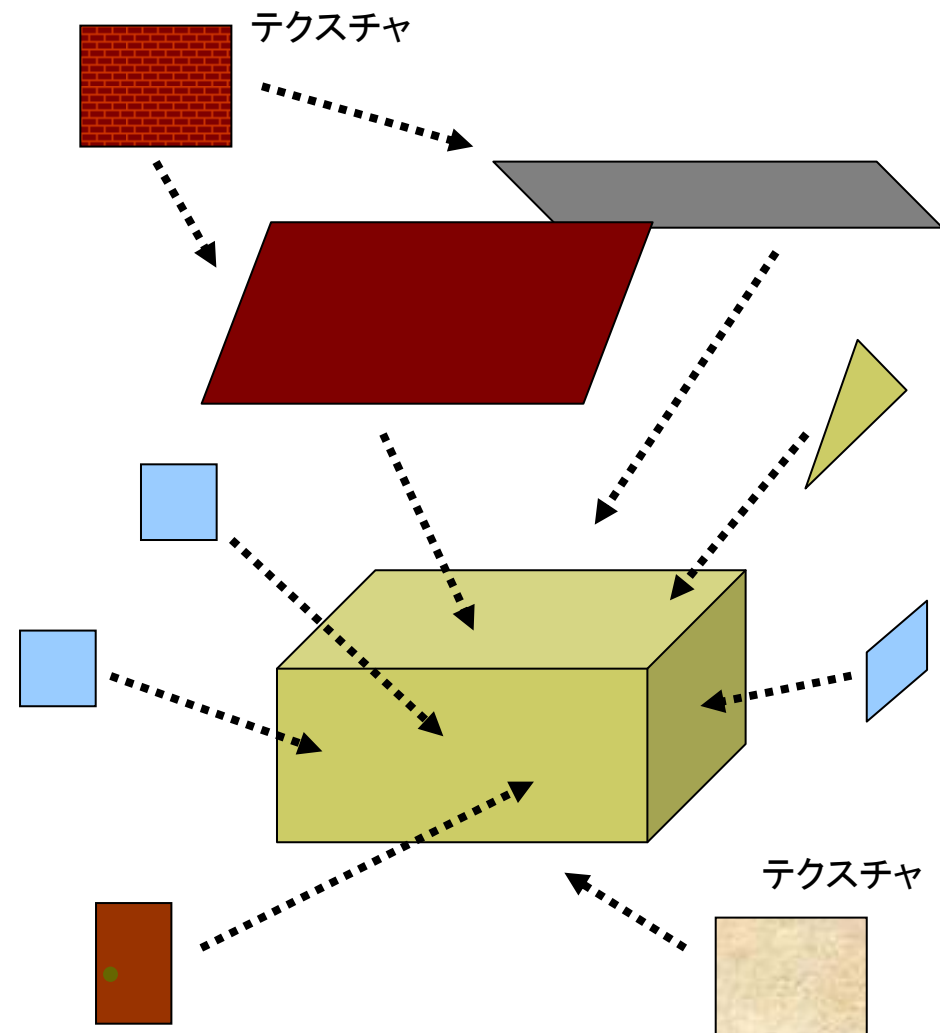
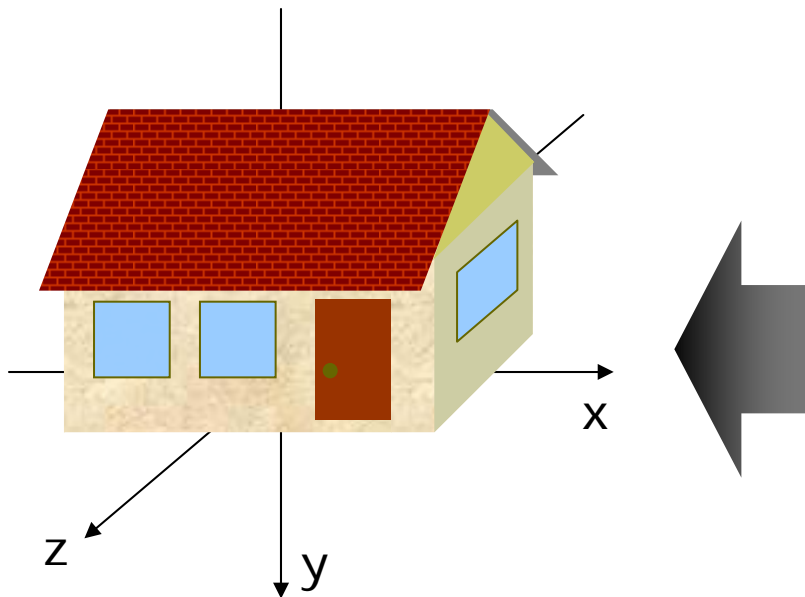
<http://vilab.org>

塩澤秀和

14.1 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



14.2 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、関数にしておく和利用しやすい

雪だるま

円錐(底なし)

木(のようなもの)

```
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();  
}
```

```
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();  
}
```

```
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();  
}
```

14.3 少し複雑なモデリング例

```

// OPENGGLのほうが正確
// size(幅, 高さ, OPENGGL);
// 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();

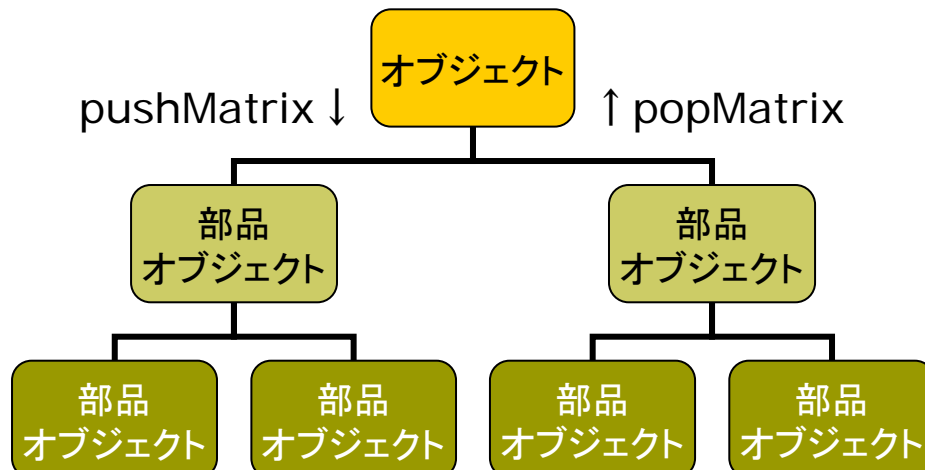
    beginShape(QUADS);
    // 窓
    fill(#4444ff);
    float z = 0.701;
    vertex(-0.8, -0.7, z);
    vertex(-0.8, -0.3, z);
    vertex(-0.4, -0.3, z);
    vertex(-0.4, -0.7, z);
    vertex(-0.2, -0.7, z);
    vertex(-0.2, -0.3, z);
    vertex(0.2, -0.3, z);
    vertex(0.2, -0.7, z);
    // ドア
    fill(#883333);
    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();
}

```

14.4 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リ、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

14.5 モデルデータの利用

モデルデータの読み込み

- .OBJ Loader
 - OBJ形式の3Dモデルを表示できるProcessingの拡張機能
 - <http://code.google.com/p/saitoobjloader/>
- インストール
 - まずobjloader???.zipを展開
 - objloader というフォルダを見つけて、Processingのフォルダの下のlibrariesのなかにコピー
- 利用方法
 - モデルは画像同様「Add File」でdataフォルダに入れておく
 - プログラム冒頭に次の行が必要
`import saito.objloader.*;`

モデルデータの描画

- OBJModel型
 - setupでデータ用の変数を用意
 - `OBJModel m = new OBJModel(this);`
- `m.load("ファイル名.obj")`
 - データファイルの読み込み
- `m.drawMode(描画モード)`
 - 描画モードの設定
 - TRIANGLES または QUADS
- `m.enableTexture()`
- `m.disableTexture()`
 - テクスチャの有効化と無効化
- `m.draw()`
 - モデルの描画

14.6 .OBJ Loader の使用例

```
// 準備:モデルデータ(beethoven.obj,  
// beethoven.mtl, beethoven.jpg  
// の3つのファイル)をダウンロードし,  
// スケッチのdataフォルダに入れておく  
// (メニューで Sketch → Add File...)
```

```
import saito.objloader.*;
```

```
OBJModel model;
```

```
void setup() {  
  size(400, 400, P3D);  
  model = new OBJModel(this);  
  model.load("beethoven.obj");  
}
```

```
void draw() {  
  background(0, 0, 100);  
  lights();
```

```
  pushMatrix();  
  translate(width*0.3, height/2, 0);  
  rotateY(radians(frameCount));  
  scale(150);  
  noStroke();  
  model.enableTexture();  
  model.drawMode(TRIANGLES);  
  model.draw();  
  popMatrix();
```

```
  pushMatrix();  
  translate(width*0.7, height/2, 0);  
  rotateY(radians(frameCount));  
  scale(150);  
  stroke(#ffffff);  
  model.drawMode(LINES);  
  model.draw();  
  popMatrix();  
}
```

14.7 3DCGソフトウェア(1)

Art of Illusion

- 3DCGフリーソフトウェア
 - 基本機能をサポート(モデリング, レンダリング, アニメーション)
 - <http://www.artofillusion.org>
 - Processingで使えるOBJ形式の3Dモデルを作成可能
- インストールと実行
 - ArtOfIllusion???.exe
 - (英語で)ライセンスへの承諾を求められるので, [Yes]を選択
 - スタートメニューの[Start Art of Illusion]から起動
- 使い方の参考(日本語)
 - <http://ei-www.hyogo-dai.ac.jp/~masahiko/aoi/index.html>

使い方のポイント

- 基本描画
 - 左のツールボタンから選択
 - 図形の配置, 移動, 回転など...
 - [シーン]→[レンダー]でレイトレーシングのCGも生成できる
- 色とテクスチャ
 - 単色: タイプ[Uniform]
 - 画像: タイプ[Image Mapped]
- OBJ形式への変換
 - [ファイル]→[データ書き出し]→[Wavefront(.obj)]
 - [テクスチャをmtlで書き出し]
- OBJ変換での注意点
 - AoIの発光色(Ke)は, OBJでは環境反射色(Ka)に変換される

14.8 3DCGソフトウェア(2)

Google SketchUp

□ 概要

- 人工物のモデリングに適する
- Google Earthに建物のモデルをアップロードして設置できる
- <http://sketchup.google.com>

□ OBJ形式への変換

- 商品版(Pro)だけの機能だが...
- フリーのプラグイン(拡張機能)を使えば, 無料版でもOBJ形式で書き出すことができる
- <http://onigiriburning.so.land.to>

□ 参考サイト

- <http://www.atmarkit.co.jp/fwcr/rensai2/3dcurl01/01.html>
- <http://sketchup.google.com/3dwarehouse/>

223演習室で使えるソフトウェア

□ Shade

- 総合3DCGソフトウェア(商品)
- <http://shade.e-frontier.co.jp>
- モデリング, 写實的レンダリング, アニメーション映像の作成など

□ Terragen

- 3D景観生成ソフトウェア
- <http://www.planetside.co.uk>
- 大自然の映像をCGで生成できる
- 実際の映画やCMで使われている

プロ向けのハイエンド製品

□ 3大CGソフト(Autodesk社)

- 3ds Max, Maya, SoftImage
- <http://www.autodesk.co.jp>