Graphics with Processing

2008-12 モデリング

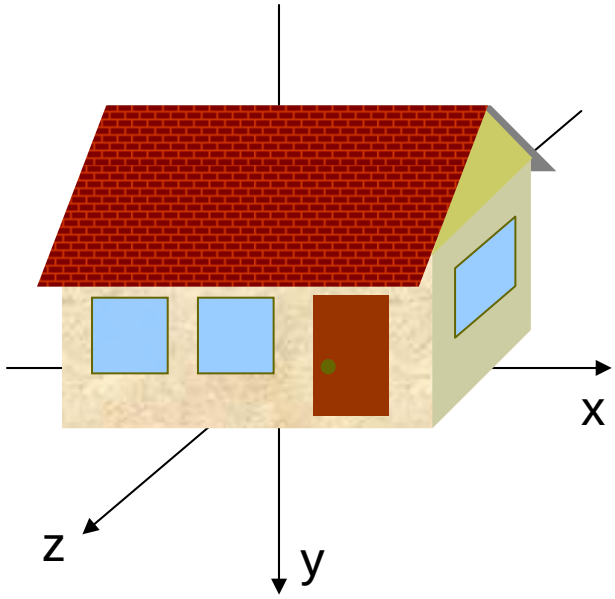
http://vilab.org

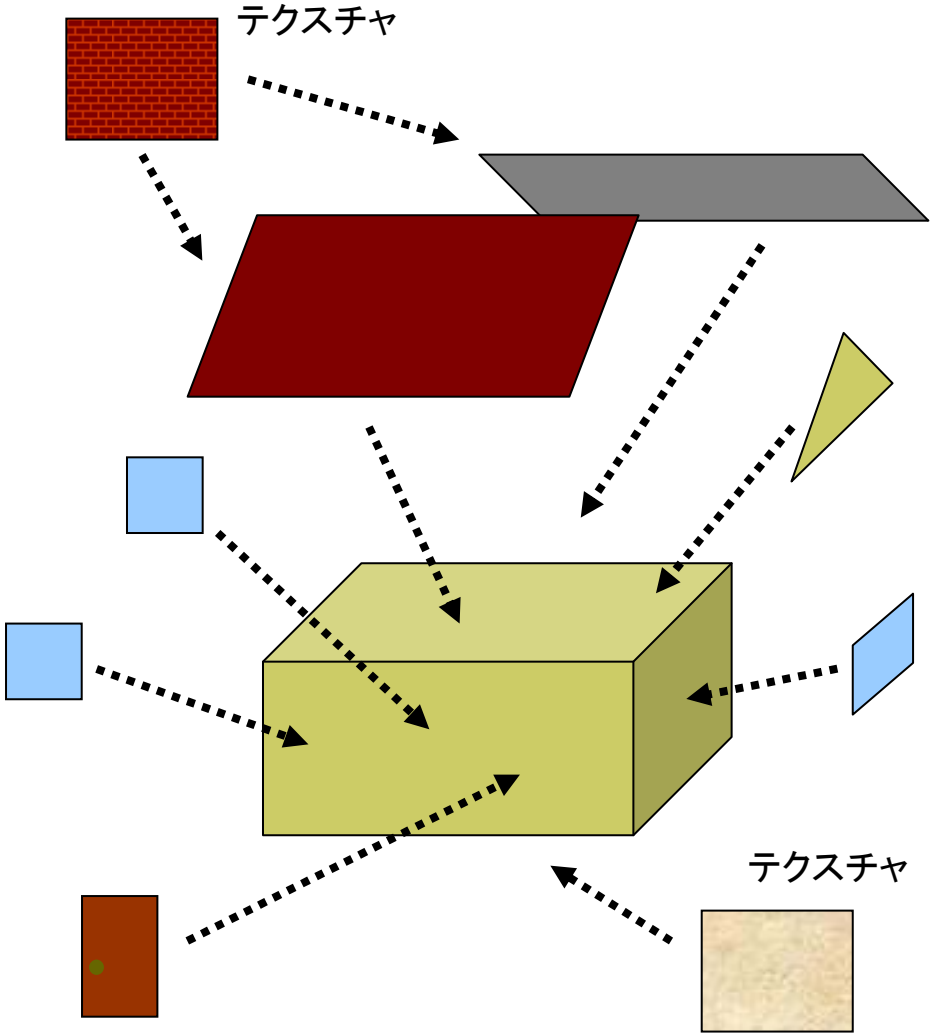
塩澤秀和

12.1 3Dモデリング

モデリング

- 3Dモデルを作り上げること
- 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





12.2 オブジェクトの関数例

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();
}
```

12.3 少し複雑なモデリング

```
// 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();
}
```

12.4 ソフトウェアを利用したモデリング

Art of Illusion

- ホームページ
  - http://www.artofillusion.org
  - 3DモデルをOBJ形式で保存し，Processingで利用できる
  - レンダリングやアニメーション作成もできるフリーソフトウェア
- インストールと実行
  - ArtOfIllusion???-Windows.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CG生成
- 色とテクスチャ
  - 単色：タイプ[Uniform]
  - 画像：タイプ[Image Mapped]
- OBJ形式での保存
  - 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(mtl)変換での注意点**
  - 色の対応がおかしい(バグ？)
  - 拡散反射色 → 環境反射色(Ka)
  - 発光色 → 拡散反射色(Kd)

12.5 モデルデータの利用

モデルデータの読み込み

- .OBJ Loader
  - Processingの拡張機能
  - OBJ形式のモデルを表示できる（dataフォルダに入れておく）
  - http://code.google.com/p/saitoobjloader/
- インストール
  - まずobjloader???.zipを展開
  - objeloaderというフォルダを見つけて，Processingフォルダの下のlibrariesのなかにコピー
- 利用方法
  - プログラム冒頭で読み込む
  - import saito.objloader.*;

モデルデータの描画

- OBJModel型
  - まず，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 か POLYGON
- 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

12.6 .OBJ Loader の使用例

```
// 準備:モデルデータ(beethoven.obj,
// beethoven.mtl, beethoven1.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);
  rotateX(radians(200));
  rotateY(radians(frameCount));
  scale(150);
  noStroke();
  model.enableTexture();
  model.drawMode(TRIANGLES);
  model.draw();
  popMatrix();

  pushMatrix();
  translate(width*0.7, height/2, 0);
  rotateX(radians(200));
  rotateY(radians(frameCount));
  scale(150);
  stroke(#ffffff);
  model.drawMode(LINES);
  model.draw();
  popMatrix();
}
```

12.7 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();
}
```

12.8 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（12.7参照）