Graphics with Processing

2016-13 モデリング

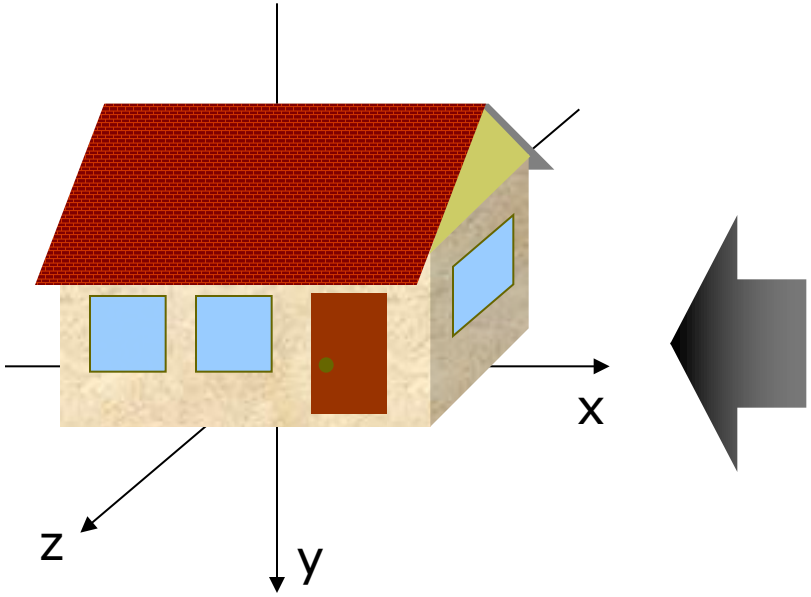
http://vilab.org

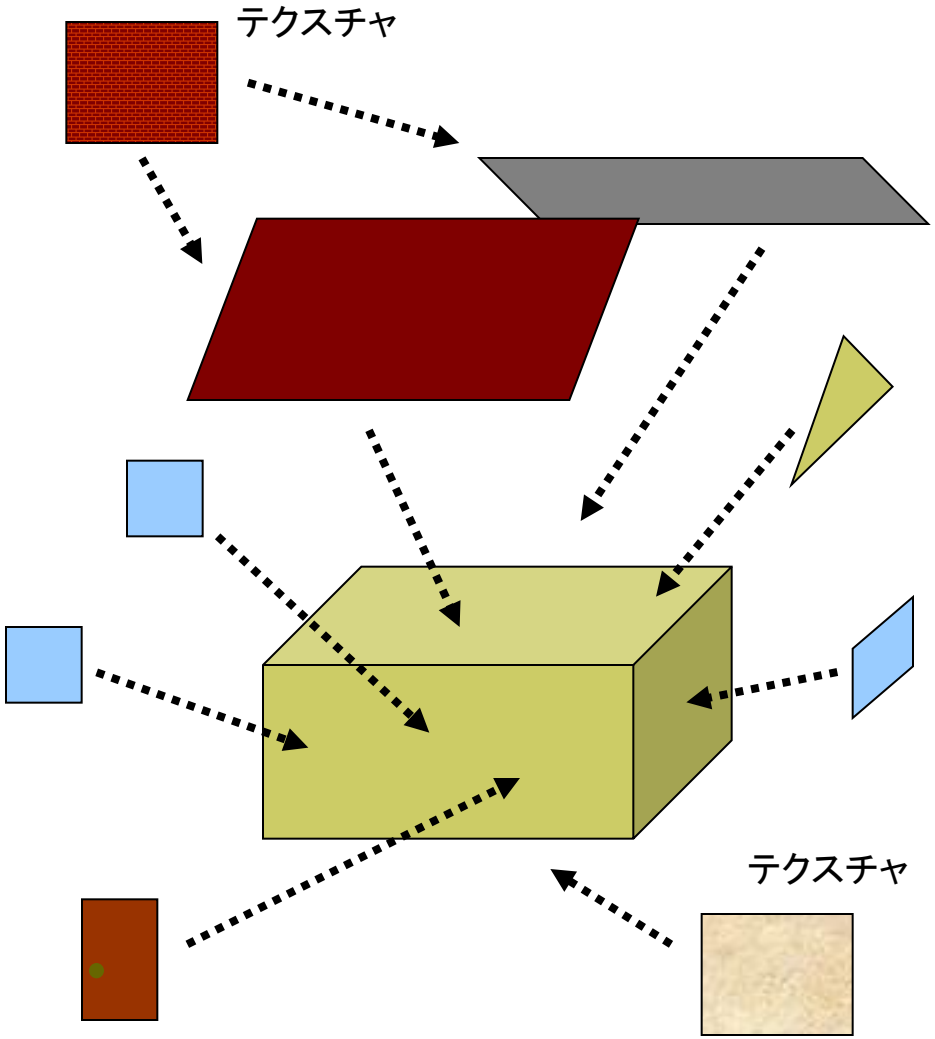
塩澤秀和

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





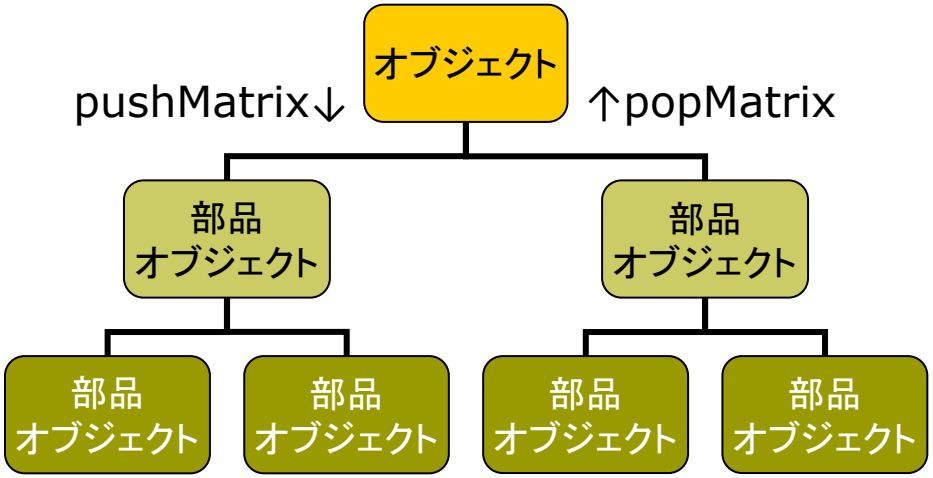
13.2 3Dモデルの描画例

```
// 3Dモデルを描画する手順を
// 関数として作成する例
void house(PImaeg roof)
{
  // 壁
  pushMatrix();
  translate(0, -25, 0);
  fill(#ffffaa);
  box(100, 50, 70);
  popMatrix();

  // 屋根裏の壁
  beginShape(TRIANGLES);
  vertex(50, -50, 35);
  vertex(50, -85, 0);
  vertex(50, -50, -35);
  vertex(-50, -50, 35);
  vertex(-50, -85, 0);
  vertex(-50, -50, -35);
  endShape();

  // 屋根
  beginShape(QUAD_STRIP);
  fill(#ffffff);
  texture(roof);
  textureMode(NORMAL);
  vertex(-55, -41, 45, 0, 1);
  vertex(55, -41, 45, 1, 1);
  vertex(-55, -86, 0, 0, 0);
  vertex(55, -86, 0, 1, 0);
  vertex(-55, -41, -45, 0, 1);
  vertex(55, -41, -45, 1, 1);
  endShape();

  // 煙突
  fill(#880000);
  pushMatrix();
  translate(-25, -70, -25);
  box(10, 50, 10);
  popMatrix();

  beginShape(QUADS);
  // 窓
  fill(#4444ff);
  vertex(-40, -35, 36);
  vertex(-40, -15, 36);
  vertex(-20, -15, 36);
  vertex(-20, -35, 36);
  vertex(-10, -35, 36);
  vertex(-10, -15, 36);
  vertex(10, -15, 36);
  vertex(10, -35, 36);

  // ドア
  fill(#883333);
  vertex(20, -40, 36);
  vertex(20, -5, 36);
  vertex(40, -5, 36);
  vertex(40, -40, 36);
  endShape();
}
```

13.3 階層モデリング

階層モデリング（p.54）

- 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
  - 動きの基準点（関節など）を原点として可動部を部品化する
  - 行列スタックを利用して描画（pushMatrix / popMatrix）




```
void cone() {  // 円錐
  pushMatrix();
  beginShape(TRIANGLE_FAN);
  vertex(0, -100, 0);
  for (int a = 0; a <= 360; a += 10) {
    float x = 100 * cos(radians(a));
    float z = 100 * sin(radians(a));
    vertex(x, 0, z);
  }
  endShape();
  popMatrix();
}

void tree() {  // 円錐を重ねた木
  noStroke();
  pushMatrix();
    translate(0, -30, 0);
    scale(0.2 , 0.7, 0.2);
    fill(0, 255, 0);  cone();  // 円錐1
  popMatrix();
  pushMatrix();
    scale(0.1, 1, 0.1);
    fill(100, 0, 0);  cone();  // 円錐2
  popMatrix();
}
```

13.4 モデルデータの構築と描画

```
// 13.3と同様のモデルをデータとして構築する例
PShape tree;

vod setup() {
   size(600, 600, P3D);
   tree = makeTree();
}

PShape makeCone() {
   // 空の図形を作成してポリゴンを追加していく
   PShape s = createShape();
   s.beginShape(TRIANGLE_FAN);
   s.vertex(0, -100, 0);
   for (int a = 0; a <= 360; a += 10) {
      float x = 100 * cos(radians(a));
      float z = 100 * sin(radians(a));
      s.vertex(x, 0, z);
   }
   s.endShape();
   return s;
}
```

```
PShape makeTree() {
   // 図形を階層的にグループ化する方法
   PShape g = createShape(GROUP);
   PShape c1 = makeCone();  // 円錐1
   c1.scale(0.2, 0.7, 0.2);
   c1.translate(0, -30, 0);
   c1.setFill(color(0, 255, 0));
   g.addChild(c1);  // 親図形に追加
   PShape c2 = makeCone();  // 円錐2
   c2.scale(0.1, 1, 0.1);
   c2.setFill(color(0, 100, 0));
   g.addChild(c2);  // 親図形に追加
   g.setStroke(false);
   return g;
}

void draw() {
  background(0); lights();
  translate(width/2, height/2);
  shape(tree);  // 構築したモデルの描画
}
```

13.5 複雑な形状の表現

曲面や自然形状

- パラメトリック曲面（p.87）
  - 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.94）
  - 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.109）
  - 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

```
// フラクタルによる地形生成の例（14.5へ続く）
final int N = 256;
float [][] h = new float[N+1][N+1];
int w = N;  // wは計算済みの要素の間隔

void setup() {
  size(800, 800, P3D);
  frameRate(30);
  randomSeed(millis());
  // 計算の起点になる4隅の高度を0とする
  h[0][0] = h[0][N] = h[N][N] = h[N][0] = 0.0;
}

// クリックで1段階ずつ細かくなる
void mouseClicked() {
  generate();
}

// 補間点の高度に加えるランダム量
float rnd() {
  return random(-0.2, +0.2) * w;
}
```

13.6 地形生成の例（続き）

```
void draw() {
  background(50, 50, 150);
  lights();
  translate(width/2, height/2);
  rotateX(PI/4);
  rotateZ(radians(frameCount));
  noStroke(); fill(180, 150, 50);

  // 間隔wの要素を使って地形を描画
  scale(2.0);  translate(-N/2, -N/2, 0);
  beginShape(QUADS);
  for (int x = 0; x < N; x += w) {
    for (int y = 0; y < N; y += w) {
      vertex(x, y, h[x][y]);
      vertex(x, y+w, h[x][y+w]);
      vertex(x+w, y+w, h[x+w][y+w]);
      vertex(x+w, y, h[x+w][y]);
    }
  }
  endShape();
}
```

```
// 地形を1段階細かくする
void generate() {
  if (w == 1) return;

  for (int x = 0; x < N; x += w) {
    for (int y = 0; y < N; y += w) {
      // 中点の高度を補間し，適当な乱数を加える
      h[x+w/2][y] = (h[x][y] + h[x+w][y]) / 2 + rnd();
      h[x][y+w/2] = (h[x][y] + h[x][y+w]) / 2 + rnd();
      // 4点の中央の高度も同様の計算で求める
      h[x+w/2][y+w/2] = (h[x][y] + h[x+w][y]
          + h[x+w][y+w] + h[x][y+w]) / 4 + rnd();
    }
  }
  for (int i = 0; i < N; i += w) {
    h[i+w/2][N] = (h[i][N] + h[i+w][N]) / 2 + rnd();
    h[N][i+w/2] = (h[N][i] + h[N][i+w]) / 2 + rnd();
  }
  // 計算済みの要素の間隔は1/2になる
  w /= 2;
}
```

13.7 3DCGソフトウェア(1)

Art of Illusion

- 概要
  - www.artofillusion.org
  - ArtOfIllusion???-Windows.exe
  - 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)
  - Java&フリー&オープンソース
- 使い方の参考(日本語)
  - yunzu.qee.jp/artofillusion/docs/AoIManual29_J2/layout.html
  - ei-www.hyogo-dai.ac.jp/~masahiko/moin.cgi/AOI
- Processingとの連携
  - OBJ形式でエクスポート
  - shape関数で描画(第10回)
  - 可能な限りポリゴン数を減らす

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形式での出力
  - [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出力での注意点
  - AoIの発光色(Ke)は，OBJでは環境反射色(Ka)に変換される

13.8 3DCGソフトウェア(2)

- SketchUp Make
  - www.sketchup.com
  - 建物・人工物のモデリングに向く
  - OBJ出力できるプラグイン sketchup-onigiri.jimdo.com/sketchup-plugins/su2objmtl/
- Blender
  - www.blender.org
  - 高機能でフリー&オープンソース
- MagicaVoxel
  - ephtracy.github.io
  - Minecraftのようにボクセル(立方体)でモデリング ←おすすめ
- Sculptris
  - pixologic.com/sculptris/
  - 粘土・彫刻のようにモデリング
- 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
- Vue Pioneer
  - www.e-onsoftware.com
  - 自然景観生成(非商用フリー版)
- DAZStudio
  - www.daz3d.com/get_studio
  - 人体ポーズ&アニメーション作成
- UnityからOBJ書き出し
  - www.assetstore.unity3d.com/en/#!/content/22250
- ブラウザソフトウェア
  - www.tinkercad.com
  - stephaneginier.com/sculptgl/