

Graphics with Processing



2019-13 モデリング

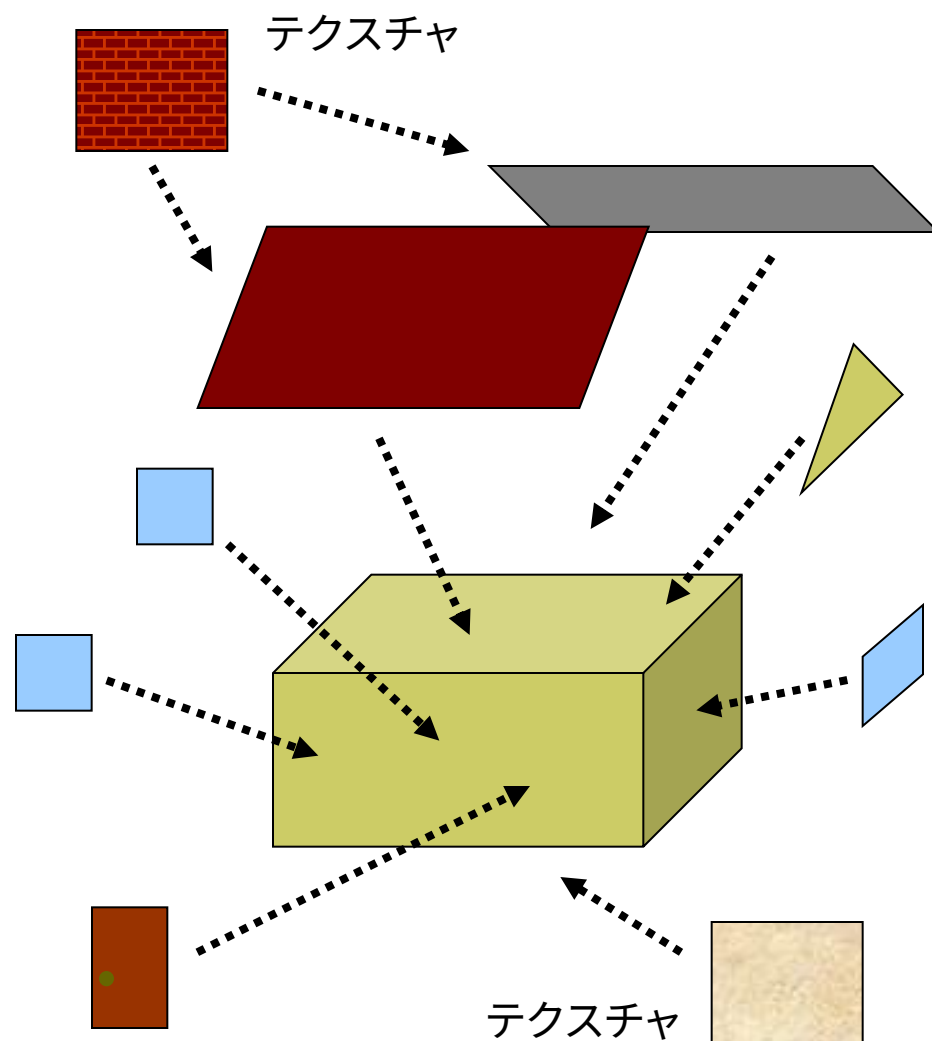
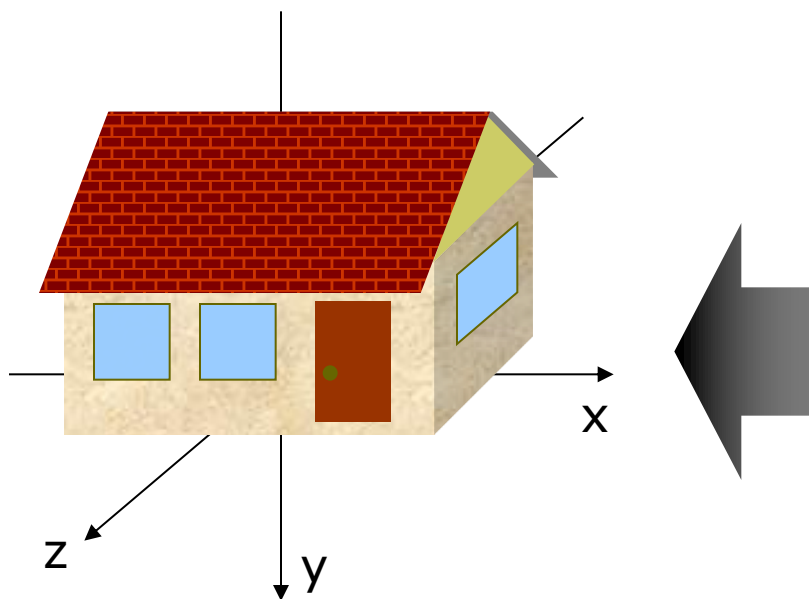
<http://vilab.org>

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(PImage 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2つ
  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();
}

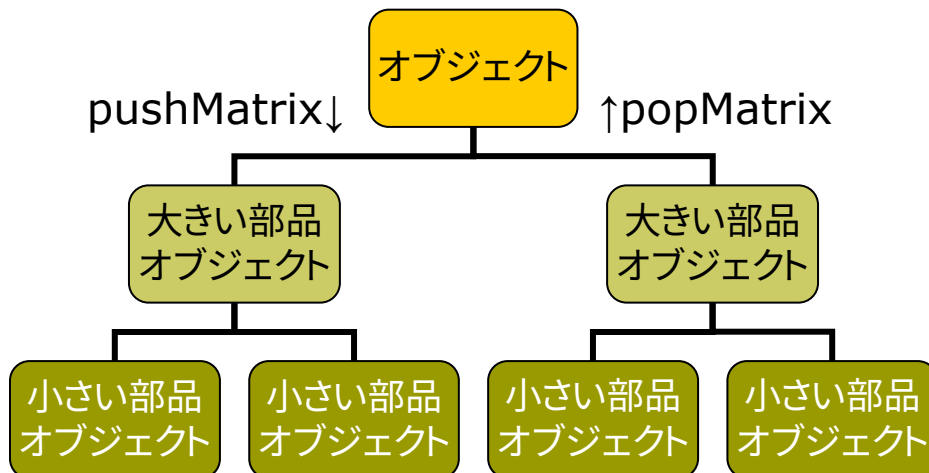
```

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

- 個々の座標系で小さい部品を作り、段階的に上の階層の座標系で組み立てて、大きなオブジェクトを作る
- 可動部は、動きの基準点(関節など)を原点として部品化する
- 行列スタックを階層的に利用する(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();
}
  
```

```

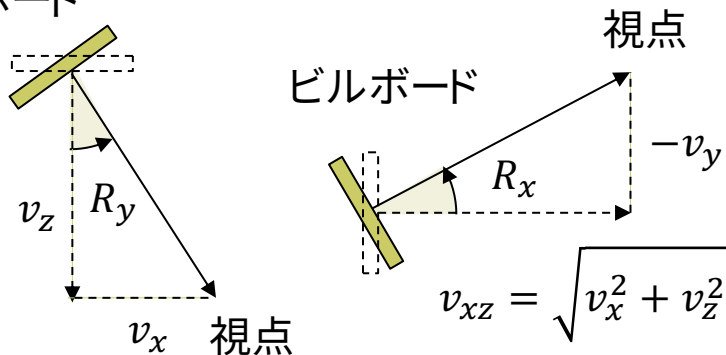
void tree() { // 円錐を重ねた木
  noStroke();
  pushMatrix();
  translate(0, -30, 0);
  scale(0.2, 0.7, 0.2);
  fill(0, 255, 0); cone(); // 円錐1
  popMatrix();
  pushMatrix();
  scale(0.1, 1, 0.1);
  fill(100, 0, 0); cone(); // 円錐2
  popMatrix();
}
  
```

13.4 ビルボード

ビルボード

- Billboard=立看板, 掲示板
 - 3Dのモデルの代わりに, 板に貼った画像でごまかす ⇒ 高速
 - 板はペラペラなので常に視点を向くように調整する
 - 遠方の雲やパーティクル(大量の雪や火花)の表示に有効

ビルボード



上から見た図

横から見た図

```
// ビルボードの使用例(次ページに続く)
// 常に視点を向く「看板」にテクスチャを貼る
PImage tex;
PVector[] snow = new PVector[400];
PVector cam = new PVector(0, -50, 0);
```

```
void setup() {
  size(600, 600, P3D);
  frameRate(30);
  tex = loadImage("particle.png");
  textureMode(NORMAL);

  for (int i = 0; i < snow.length; i++) {
    snow[i] = new PVector(
      random(-400, 400),
      random(-1000, 0),
      random(-400, 400));
  }
}
```

13.5 ビルボード (続き)

回転をコメントアウトしてみよ

```

void draw() {
  background(#000020);
  // 視点を回転する
  float a = radians(frameCount / 2);
  cam.x = 600 * cos(a);
  cam.z = 600 * sin(a);
  camera(cam.x, cam.y, cam.z,
         0, -200, 0, 0, 1, 0);
  perspective(); lights();

  noStroke(); fill(255);
  box(800, 1, 800); // 地面を描く

  // ビルボードの描画
  for (PVector s : snow) {
    pushMatrix();
    translate(s.x, s.y, s.z);
    // 視点へ向かうベクトルを求める
    PVector v = PVector.sub(cam, s);
    // 横にRy回転し, 正面を視点に向ける
    rotateY(atan2(v.x, v.z));
    // 縦にRx回転し, 正面を視点に向ける
    float vxz = dist(0, 0, v.x, v.z);
    rotateX(atan2(-v.y, vxz));

    beginShape(QUADS);
    texture(tex);
    vertex(-10, -20, 0, 0, 0);
    vertex( 10, -20, 0, 1, 0);
    vertex( 10,  0, 0, 1, 1);
    vertex(-10,  0, 0, 0, 1);
    endShape();
    popMatrix();

    s.y += 5;
    if (s.y > 0) s.y = -1000;
  }
}

```

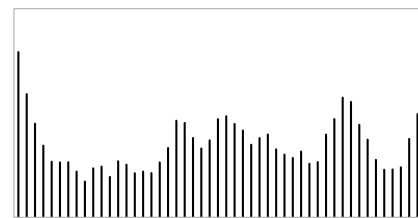
13.6* 複雑な形状の表現

曲面や自然形状

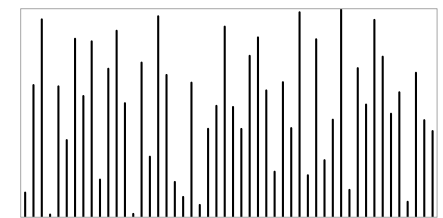
- パラメトリック曲面 (p.87)
 - パラメータ方程式による曲面
 - ベジエ曲面やNURBS曲面など
 - レンダリング時にポリゴンに変換する方式としない方式がある
 - ポリゴン曲面の操作 (p.94)
 - 細分割曲面: ポリゴンを再帰的に分割し、滑らかな面を生成
 - 詳細度制御: 視点から遠い曲面のポリゴン数を削減して簡略化
 - フラクタル (p.109)
 - 自然界によく見られる再帰的な形状(※)のモデリングに適する
- ※ 海岸線や木の枝など, 一部分が全体の縮小のような形状のもの

Perlinノイズ

- noise(x)
 - xの値に対して滑らかに変化するでたらめな値(0~1)を生成
 - さらに大小のノイズの波をフラクタル的に重ねた値を出力
 - 自然物のテクスチャや形状の生成によく利用される(雲, 岩石等)
- noise(x,y), noise(x,y,z)
 - 複数の変数の変化に対しても滑らかに変化する値を生成
- noiseDetail(n)
 - ノイズを重ねる段階数を指定



Perlinノイズ(noise)



擬似乱数(random) 7

13.7 Perlinノイズによる地形生成

```

final int N = 40;
int [][] h = new int[N][N];
float F = 0.1;

void setup() {
  size(600, 600, P3D);
  frameRate(30);

  for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++) {
      h[i][j]
        = (int)(noise(i * F, j * F) * 10);
    }
  }
}

```

Fは出力される値の
変化の速さに影響

```

void draw() {
  background(#6080ff);
  lights();
}

```

```

  translate(width/2, height/2, 0);
  rotateX(-PI/3);
  rotateY(radians(frameCount));
  noStroke();

  translate(-200, 0, -200);
  scale(10);
  for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++) {
      pushMatrix();
      translate(i, -h[i][j] / 2, j);
      if (h[i][j] > 3) fill(#008000);
      else if (h[i][j] > 2) fill(#909000);
      else fill(#2020a0);
      box(1, h[i][j], 1);
      popMatrix();
    }
  }
}

```


13.8 ソフトウェアを使ったモデリング

Art of Illusion

□ 概要

- www.artofillusion.org
- ArtOfIllusion???.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
- 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]→[レンダラー]でレイタレーシングの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

13.9 参考:コードによるモデル生成

```
// 13.3と同様のモデルをデータとして構築する例
```

```
PShape treeModel;

void setup() {
    size(600, 600, P3D);
    treeModel = makeTreeModel();
}

PShape makeConeModel() {
    // 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 makeTreeModel() {
    // 図形を階層的にグループ化する方法
    PShape g = createShape(GROUP);
    PShape c1 = makeConeModel(); // 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 = makeConeModel(); // 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(treeModel); // 構築したモデルの描画
}
```

7.11' 参考：3DCGソフトウェア紹介

- MagicaVoxel ←おすすめ
 - ephtracy.github.io
 - Minecraftのようにボクセル(立方体)でモデリング
- Tinkercad ←おすすめ
 - www.tinkercad.com
 - インストール不要なWebアプリ
- SketchUp Free
 - www.sketchup.com
 - 建物・人工物のモデリングに向く
- ScupltGL
 - stephaneginier.com/sculptgl/
 - 粘土・彫刻のようにモデリング
- 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
- メタセコイア
 - www.metaseq.net
 - 日本製で資料が豊富
- 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
- その他Webアプリ
 - SelfCAD: www.selfcad.com
 - Vectary: www.vectary.com

7.12 参考：3Dモデルデータの取得

- Google Poly
 - poly.google.com
- Free3D
 - free3d.com
- 3DModelFree.com
 - www.3dmodelfree.com
- TurboSquid
 - www.turbosquid.com
- Artist-3D.com
 - artist-3d.com
- XOIO Air
 - xoio-air.de
- Unityアセットストア
 - assetstore.unity.com/search?q=3d&q=models&q=price:0
 - OBJ出力
assetstore.unity.com/packages/tools/utilities/scene-obj-exporter-22250
- SketchUp 3D Warehouse
 - 3dwarehouse.sketchup.com
 - OBJ出力できるプラグイン
sketchup-onigiri.jimdo.com/sketchup-plugins/su2objmtl/