Graphics with Processing

2020-12 モデリング

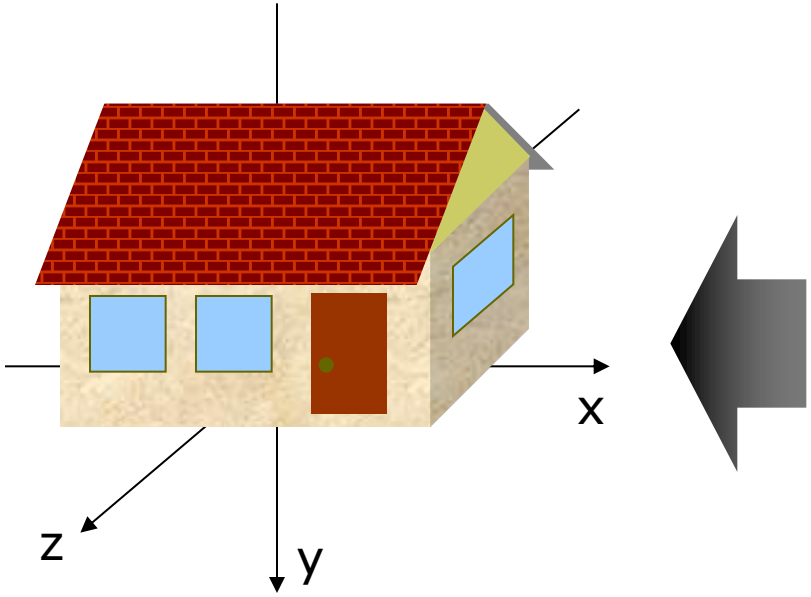
http://vilab.org

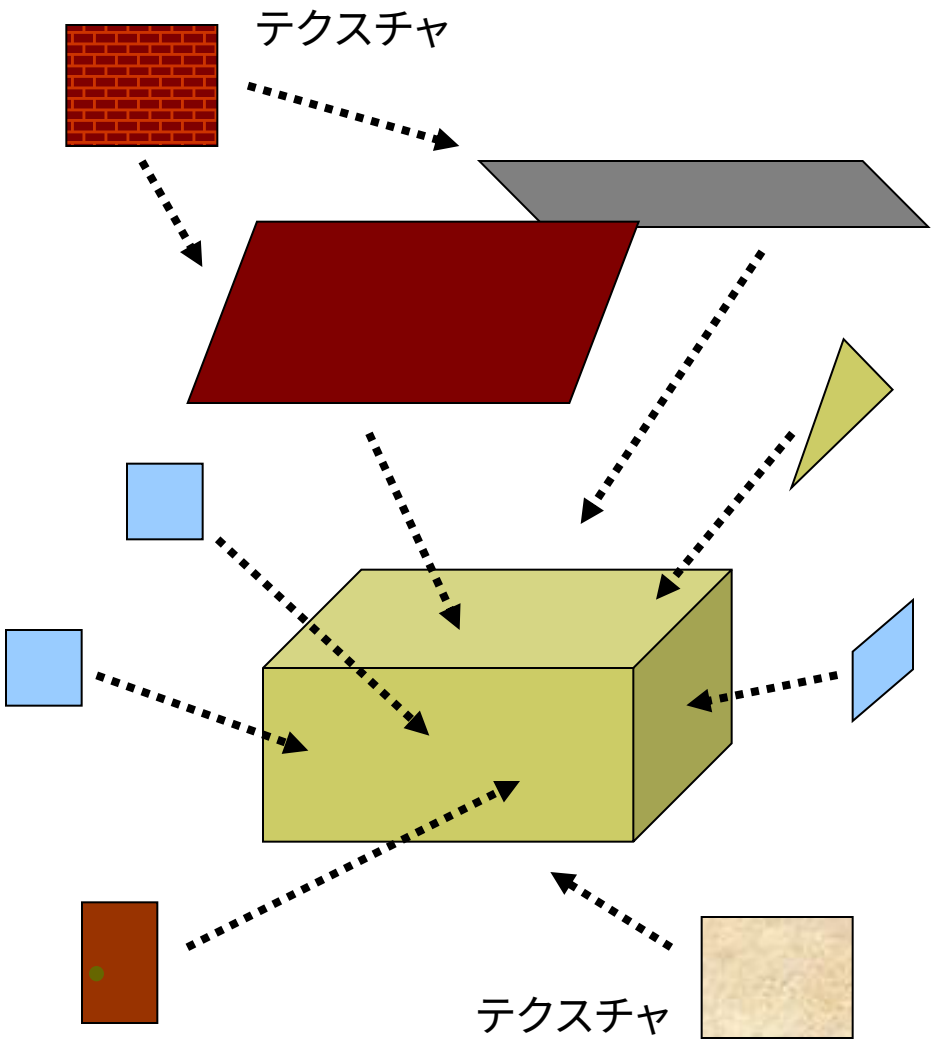
塩澤秀和

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





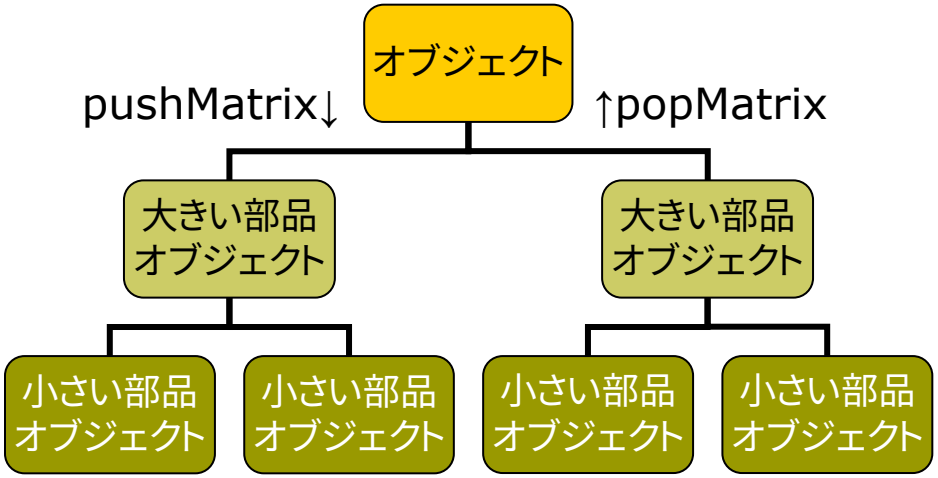
12.2 3Dモデルの描画例

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

12.3* 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();
}
```

12.4 階層的モデルデータ

```
// 12.3のモデルをデータとして構築する例
// 階層的なデータ構造になるのが分かる

PShape coneModel() {
  // まず空の図形を作成する
  PShape s = createShape();

  // ポリゴンを追加してモデルを構成する
  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 treeModel() {
  // 図形を階層的にグループ化する方法
  PShape g = createShape(GROUP);
  g.setStroke(false);  // noStroke

  // 円錐1
  PShape c1 = coneModel();
  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
  // 円錐2
  PShape c2 = coneModel();
  c2.scale(0.1, 1, 0.1);
  c2.setFill(color(0, 100, 0));
  g.addChild(c2);  // 親図形に追加
  return g;
}
```

12.5* 3Dモデルのデータ構造

境界表現(p.65)

- 頂点リスト+面リスト
  - 単純な構造だが,処理が遅い
  - OBJファイル等で利用

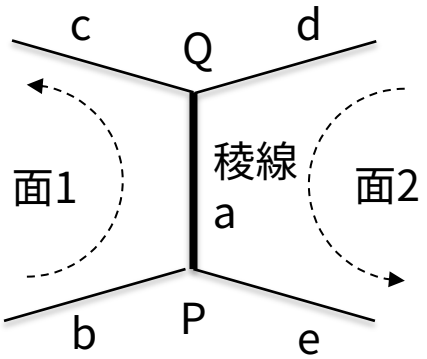
頂点	座標		
P	x_1	y_1	z_1
Q	x_2	y_2	z_2
…			

面	頂点			
1	P	Q	R	
2	Q	R	S	T
…				

- ウィングドエッジ構造
  - 稜線(edge;辺)の周りの接続関係を保持するデータ構造
  - レコードが固定長で処理が速い

頂点	座標			稜線
P	x_1	y_1	z_1	a
Q	x_2	y_2	z_2	c
…				

面	稜線
1	a
2	e
…	



稜線	頂点		面		稜線			
	始	終	左	右	左前	左後	右前	右後
a	P	Q	1	2	b	c	d	e
b	R	P	1	3	f	a	e	g
…								

12.6* 複雑な形状の表現

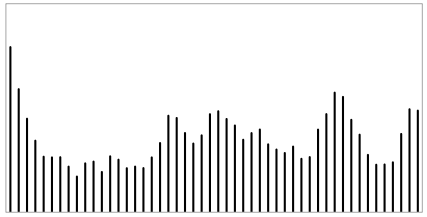
曲面や自然形状

- パラメトリック曲面(p.87)
  - パラメトリック曲線(2.3～2.6)の3次元への拡張
  - ベジエ曲面,Bスプライン曲面,NURBS曲面などがある
- 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て面を滑らかに見せる
  - 詳細度制御: 視点から遠い面のポリゴンを結合して簡略化する
- 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

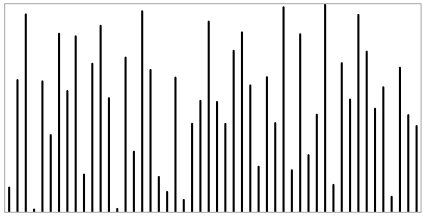
※ 海岸線や木の枝など,一部分が全体の縮小のような形状のもの

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
- noiseDatail(n)
  - ノイズを重ねる段階数を指定



Perlinノイズ(noise)



擬似乱数(random)

12.7 Perlinノイズによる地形生成

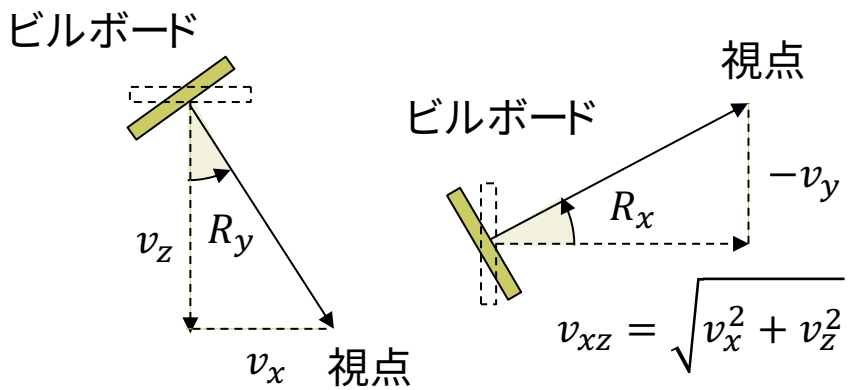
```
final int N = 40;
int [][] h = new int[N][N];
float F = 0.1;

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);
    }
  }
}

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();
    }
  }
}
```

Fは出力される値の
変化の速さに影響

12.8 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



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));
  }
}
```

12.9 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;
  }
}
```

12.10 ソフトウェアを使ったモデリング

Art of Illusion

- 概要
  - www.artofillusion.org
  - 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)
  - Downloadから取得
- インストール
  - [Edit]→[Preferences…]→[Language]で,[日本語]化
  - Macではアプリをフォルダから出して戻す必要があるかも
- 日本語の参考サイト
  - yunzu.qee.jp/artofillusion/documentation.htm
  - ei-www.hyogo-dai.ac.jp/~masahiko/moin.cgi/AOI

モデリング操作

- 基本図形
  - アイコンのリストで図形を選択し,シーンをドラッグして配置する
  - 移動・回転・変形等のアイコンを選択してからマウスで操作できる
- メッシュ(曲面)
  - 市松模様のアイコンを選択して,メッシュの平面を配置する
  - 図形を右クリックして[メッシュに変換…]することもできる
  - 選択してダブルクリックで変形
- 曲線の変形
  - 曲線アイコンを選択して描画する
  - [ツール]メニューで,[回転体…],[管…]などを選択して変形する

12.11 Art of Illusionの操作(続き)

色とテクスチャの設定

- 色の設定
  - オブジェクトを右クリックして[テクスチャ,材質を指定…]を選ぶと,設定ウィンドウが開く
  - [テクスチャ]タブを選択し,タイプ:[単一]→[新規テクスチャ…]→[Uniform](※1)
  - [拡散反射色]や[鏡面反射色]を適切に設定する
- テクスチャの設定
  - [シーン]→[マッピング用画像]で画像ファイルを登録しておく
  - 色の設定と同様に進み,(※1)で[Image Mapped]を選択する
  - 拡散反射色の横の□をクリックし,テクスチャ画像を選択する

モデリング以外

- レンダリングとアニメーション
  - [シーン]→[レンダー]でレイトレーシングのCGも生成できる
  - 小規模なソフトだが,アニメーションも作成できるのが特徴

Processingで利用

- OBJ形式で出力
  - [ファイル]→[データ書き出し]→[Wavefront(.obj)]
  - [テクスチャを .mtl ファイルに書き出し]を選択する
  - 発光色(Ke)が環境反射色(Ka)に変換されてしまうことに注意
  - 座標系の原点に注意すること

12.12 演習課題

期末レポートに向けて

- チームを結成し，以下の内容を報告せよ
  - メンバーの氏名（3人以内）
  - チーム名（個人名はダメ）
  - 選択したいテーマ
  - 本日中にシートに記入すること

自由課題

- 3DCGソフトウェア，または今回説明したプログラミング技術を利用して，期末レポートの作品の構成要素を作成せよ
  - 提出は3Dモデルファイルまたは Processingのプログラム
  - 今回の課題の提出は自由とする

ベクトルクラス（12.8で利用）

コンストラクタ	v = new PVector(x, y, z)
複製	u = v.copy()
ベクトルの和	v.add(u) w = PVector.add(v, u)
ベクトルの差	v.sub(u) w = PVector.sub(v, u)
スカラー倍	v.mult(s) w = PVector.mult(v, s)
内積	s = v.dot(u)
外積	w = v.cosss(u)
大きさ（ノルム）	s = v.mag()
ノルムの2乗	s = v.magSq()
正規化	v.normalize()

12.13 MagicaVoxelの利用(追加)

MagicaVoxel

- ボクセルモデリング
  - http://ephtracy.github.io
  - Minecraftのようにボクセル(立方体)でモデリング
- OBJ形式で出力
  - ウィンドウの右下の[Export]→[obj]を選び,保存する
  - そうすると,モデル本体(.obj),マテリアル(.mtl),色(.png)の3つのファイルが出力される
- 使用例
  - サンプルchr_knightをロードし,OBJに変換する
  - 右のプログラムを入力し,dataフォルダにobj,mtl,pngの3つのファイルを入れて実行する

```
PShape model;

void setup() {
  size(800, 800, P3D);
  model = loadShape("chr_knight.obj");
}

void draw() {
  background(0, 0, 128);  lights();
  translate(width/2, height/2, -500);

  scale(200); // 適当に拡大

  // ProcessingとMagicaVoxelは座標系が
  // 異なるので、x軸方向とy軸方向に反転
  scale(-1, -1, 1);

  // MagicaVoxelのモデルのローカル座標系は
  // 原点が左下手前にあるので中心付近に移動
  translate(-1, -2, 1);
  shape(model);
}
```