Graphics with Processing

2009-12 レンダリング技術

http://vilab.org

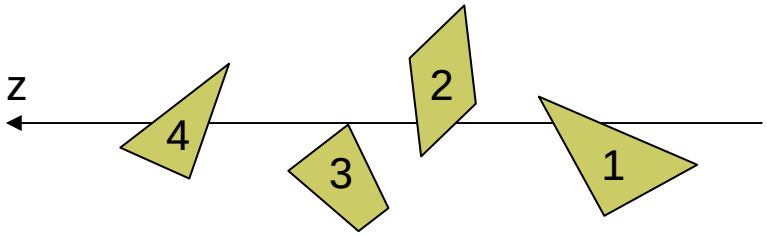
塩澤秀和

12.1 隠面消去(1)

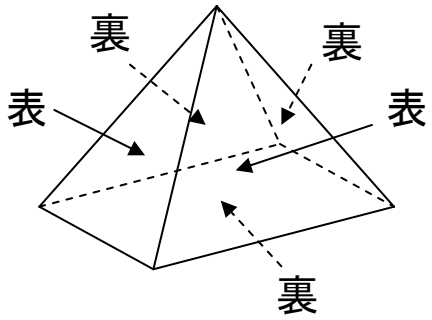
隠面消去(隠線・隠面処理)

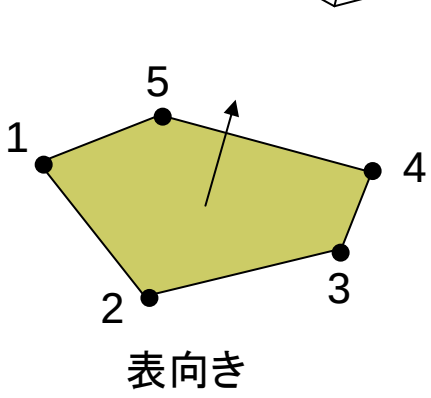
- 隠面消去とは
  - 他の物体などに**隠れて見えない**物体(の全部または一部)を描画しない処理
  - 弱点を補い合ういくつかの手法を組み合わせることもある

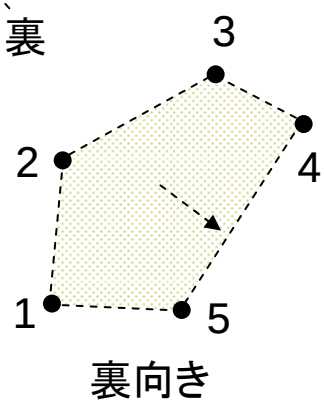
- 奥行きソート法(p.102)
  - ポリゴンをz座標(視点座標)で並び替え，遠くから順に描画
  - 細長いポリゴンで問題が生じる



- バックフェースカリング(p.100)
  - ポリゴンに表裏を設定し，裏側を向いているポリゴンを描画しない
  - 表裏はポリゴン作成時の頂点の順序(右回り・左回り)で指定
  - 1つの凸多面体上の隠面消去

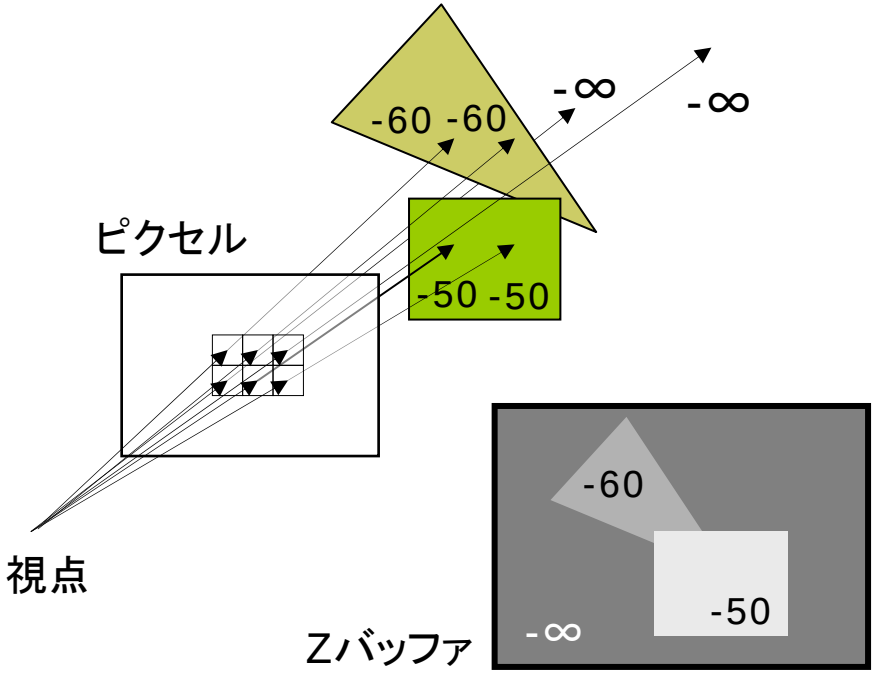




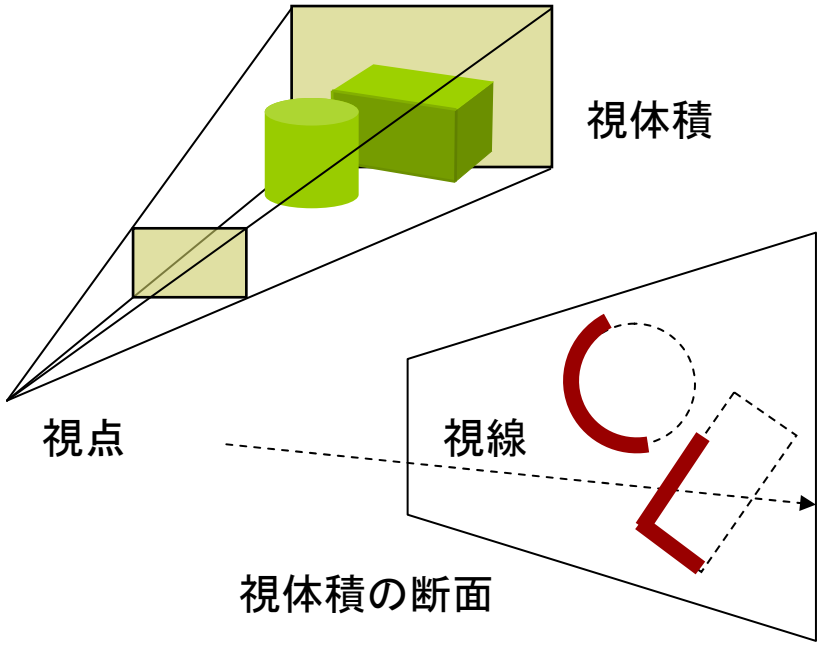


12.2 隠面消去(2)

- Zバッファ法(p.107)
  - 画面上の全ピクセルに z座標を持たせ，1点1点描画するときに遠近関係をチェックする
  - 単純＆高速 ⇒ ハードウェア化
  - 半透明の重なりの処理に難点



- スキャンライン法(p.105)
  - ピクセル横1行(スキャンライン)ごとにポリゴンの断面の重なりを数学的に計算し，描画する
  - 計算は複雑だが，使用メモリが少ない

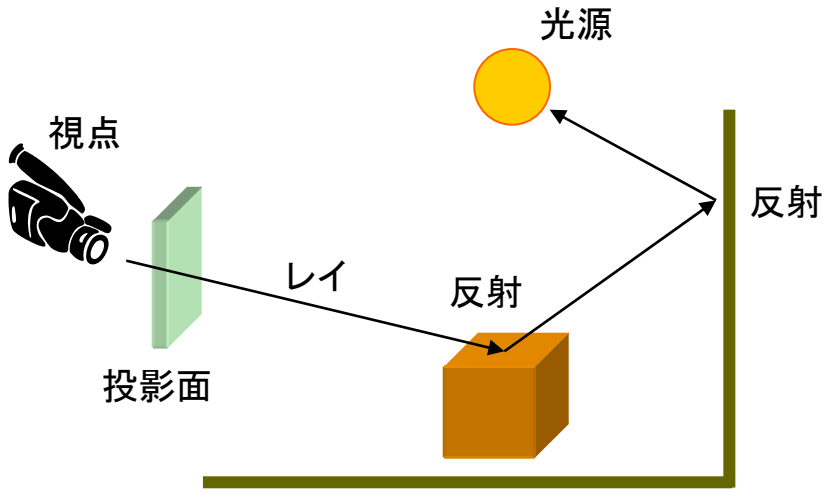


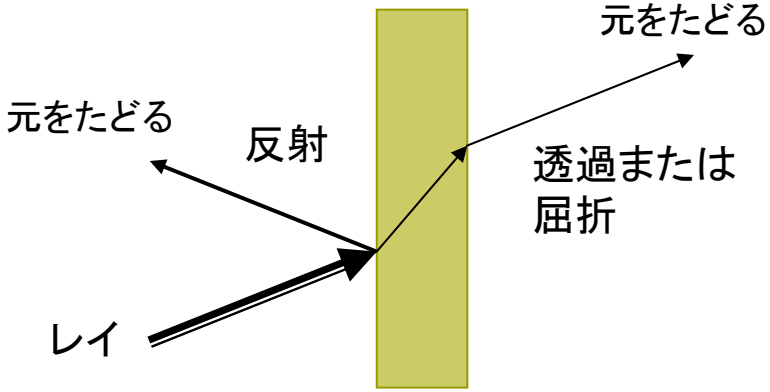
12.3 レイトレーシング (p.110)

レイトレーシング法

- 概要
  - Ray Tracing＝光線追跡
  - 各ピクセルに届く光の軌跡を，視点から光源にさかのぼるように追跡するレンダリング技術
- 高品質
  - 3DCGの初期からあるが，より正しく光学現象を再現するように研究され続けている
  - 原理的に隠面消去される
  - 透明，影，レンズも自然に表現
- 用途
  - リアルだが時間がかかるので，まだゲームなどには向かない
  - 映像作品（映画）製作で一般的

- 光線追跡の概念図





12.4 影付け(p.136)

影の種類

- 本影と半影
  - 点光源・平行光線 ⇒ 本影のみ

影付けのアルゴリズム

- レイトレーシング法
  - 原理的に影がつくが遅い
- スキャンライン単位の方法
  - スキャンライン法において，影の区間も計算によって求める
- 投影マッピングによる方法
  - （物体空間における2段階法）
  - **視点を光源**にして，物体のシルエットを描画すると影が分かる
  - 例えば，それを影の画像として，各物体の上にテクスチャ投影
- シャドウボリューム法
  - 各物体が光源からの光をさえぎってできる影の空間を計算し，物体データに「シャドウポリゴン」として付け加える
  - レンダリング時に，各ポリゴンと影の交差を判定しながら描画
  - 「ステンシルバッファ」を用いると，高速に実現できる
- シャドウマップ法
  - （Zバッファを用いた2段階法）
  - 視点を光源にしてZバッファだけを描画すると，光源から光のあたる部分までの距離が分かる
  - ピクセルをレンダリングするときに，このシャドウマップも参照し，光のあたっている点だけを描画する

12.5 大域照明モデル(1)

大域照明モデル

- 間接光の計算
  - 環境光モデルでは空間で均一としている間接光を，よりリアルに計算する手法
  - 特に室内の陰影がより自然

ラジオシティ法(p.141)

- 概要
  - Radiosity＝放射発散度
  - すべてのポリゴンをパッチ(より小さなポリゴン)に分割する
  - 2つのパッチの位置関係から，相互反射による“光の授受”(フォームファクタ)を計算
  - 全パッチ間での光の放射発散の平衡状態を計算する

- 特徴
  - 照明工学の技術の応用
  - 輪郭の柔らかい自然な影(ソフトシャドウ)などが表現できる
  - 光源が変化しても，フォームファクタは再計算の必要がない

- ラジオシティ方程式(p.158)

$$B_i = E_i + \rho_i \sum_{j=1}^{n} F_{ij} B_j$$

n	シーン全体のパッチ数
Bi	パッチiのラジオシティ
ρ i	パッチiの反射率
Fij	パッチの間のフォームファクタ

  - 本質的には「連立一次方程式」
    ⇒ ガウス・ザイデル法など

12.6 大域照明モデル(2)

フォトンマッピング(p.142)

- 概要
  - Photon＝光子(光の粒子)
  - 光源から出る大量の光子を考え，その軌跡をシミュレーションする(レイトレーシングとは逆方向)
  - すると，シーン全体の間接光の分布が概算できる
  - 最後に，求めた間接光を環境光の代わりにして，視点からのレイトレーシングを行う
- 特徴
  - レイトレーシング法と相性がよい
  - 集光現象(凸レンズや凹面鏡で光が集まる効果)が表現できる
  - 着想は簡単だが，アルゴリズムは複雑で，膨大な時間がかかる

フリーソフトによるレンダリングの例

- POV-Ray
  http://www.povray.org
  → Hall of Fame
- Blender+Yafray
  http://www.blender.org
  → Feature & Gallery
  http://www.yafaray.org
  → Gallery
- Sunflow
  http://sunflow.sourceforge.net
  → Gallerly
- Art of Illusion
  http://www.artofillusion.org
  → Art Gallery

12.7 その他のレンダリング技術

ぼかし(ボケ)系

- CG画像の違和感
  - すべてがはっきりくっきりしすぎ
  - 現実感を出すために,「はっきり見えなくする」ことも必要
- アンチエイリアシング
  - ドットのギザギザが目立たないように, 輪郭を中間色でぼかす
- フォグ(霧)
  - 水蒸気やチリなどによる空気の「濁り」を再現する
  - 遠くにあるものがかすんでいき, 色が落ちていく効果を与える
- 被写界深度(DOF)
  - レンズの効果を再現し, ピントが合っていないところをぼかす
- モーションブラー
  - 速く動くものに見える残像(ボケ)をわざと表示する
  - 軌跡の画像を重ね合わせる

ノンフォトリアリスティック(非写実的)レンダリング

- 概要
  - 現実のマネではないレンダリング
  - 例) 油絵風, 手書きタッチの再現, 製図風, 2次元アニメ, 芸術作品
- 背景
  - 既に, 写実的(フォトリアリスティック)なCG技術はかなり完成
  - 芸術などへのCG利用の広がり
  - アニメーション製作への利用

12.8 演習

Processingでレイトレーシング

- P5Sunflowのインストール
  - p5sunflow-fix.zip を講義ページからダウンロードし，展開する
  - p5sunflow という名前のフォルダを選び，Processingのlibrariesフォルダの下にコピーする
    - さらにその下にlibraryができる
  - http://hipstersinc.com/p5sunflow/
- 使用例

```
import hipstersinc.*;
void setup() {
  size(300, 200,
       "hipstersinc.P5Sunflow");
  noLoop();
}
```

まとめテスト（12/21）

- 日時
  - 12月21日（月） 3～4限
- 形式
  - 持ち込み不可
  - **配布したA4用紙のみ持込可（手書きで記入し，必ず回収）**
- 範囲
  - 配布資料（BlackBoard）
  - 教科書の該当部分
- 内容
  - CGにおけるコンピュータ処理の手順と概要について
  - 主に，用語の“意味”や考え方
  - プログラミングそのものは出ない

12.9 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（12.10参照）

12.10 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();
}
```

12.11 参考:影付けの例

```
// これは難しいのであくまでも参考です
// (Processingではここらへんが限界)
import processing.opengl.*;

PGraphics3D pg;
PImage tex;

float fw =256;
float fh = 256;

float lightX = -100;
float lightY = -200;
float lightZ = -100;

void setup() {
  size(400, 400, OPENGL);
  tex = loadImage("davinci.jpg");
  pg = (PGraphics3D)
    createGraphics(256, 256, P3D);
}
```

```
void draw() {
  pg.beginDraw();
  pg.background(255, 255, 255, 0);
  pg.copy(tex,
     0, 0, tex.width, tex.height,
     0, 0, pg.width, pg.height);

  pg.camera(lightX, lightY, lightZ,
                lightX, 0, lightZ, 0, 0, 1);
  pg.frustum((-pg.width/2  - lightX) * 0.1,
               ( pg.width/2  - lightX) * 0.1,
               (-pg.height/2 - lightZ) * 0.1,
               ( pg.height/2 - lightZ) * 0.1,
               -lightY/10, -lightY);
  pg.noLights();
  pg.scale(pg.width/fw, 1.0 , pg.height/fh);
  pg.stroke(0, 0, 0, 50);
  pg.fill(0, 0, 0, 128);
  drawScene(pg);
  pg.endDraw();
```

12.12 参考:影付けの例(続き)

```
background(0);

camera(0, -300, -300,
       0, 0, 0,  0, 1, 0);
rotateY(radians(frameCount));

pointLight(128, 128, 128,
           lightX, lightY, lightZ);
ambientLight(128, 128, 128);

stroke(0);
fill(255, 0, 0);
drawScene(this.g);

pushMatrix();
translate(lightX, lightY, lightZ);
noStroke();
fill(255, 255, 0);
sphere(5);
popMatrix();
```

```
  textureMode(NORMAL);
  fill(255);
  beginShape(QUADS);
  texture(pg);
  vertex(-fw/2, 0, -fh/2,   0,   0);
  vertex( fw/2, 0, -fh/2,   1,   0);
  vertex( fw/2, 0,  fh/2,   1,   1);
  vertex(-fw/2, 0,  fh/2,   0,   1);
  endShape();
}

void drawScene(PGraphics3D g) {
  g.pushMatrix();
  g.translate(0, -50, 0);
  g.rotateX(radians(frameCount));
  g.rotateZ(radians(frameCount));
  g.box(30, 20, 40);
  g.popMatrix();
}
```