Graphics with Processing

2008-11 シェーディングとマッピング

http://vilab.org

塩澤秀和

11.1 シェーディング

シェーディング

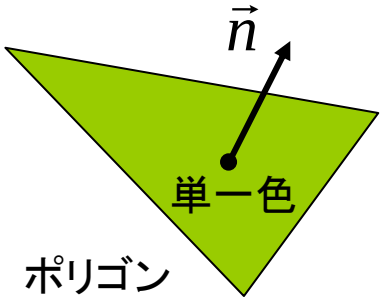
- シェーディングとは
  - Shading＝陰影づけ
  - ポリゴンの陰影計算モデル
    ＝ シェーディングモデル
  - 光の反射・材質のモデル（前回）

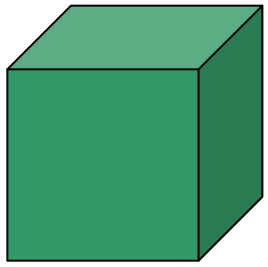
シェーディングモデル

- フラットシェーディング
  - ポリゴンを単一色で描画
- スムースシェーディング
  - ポリゴンの色を滑らかに描画
    ⇒ グローシェーディング
    ⇒ フォンシェーディング

フラットシェーディング

- 各ポリゴンを単一色で描画
  - もっとも単純で高速な方法
  - ポリゴンの代表点（例：重心）の法線ベクトルを面の向きとする
  - 面の向きから光の反射を計算し，面全体の描画色を決定する
  - 各面は単一色で塗りつぶす

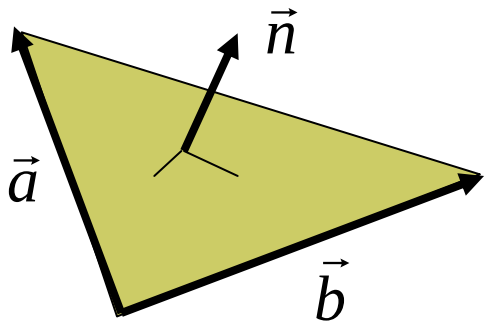




フラットシェーディング

11.2 法線ベクトル

ポリゴンの法線ベクトル



平面の方程式から

$$ax + by + cz + d = 0$$

$$\vec{N} = (a, b, c)$$

ポリゴンの辺（ベクトル）から

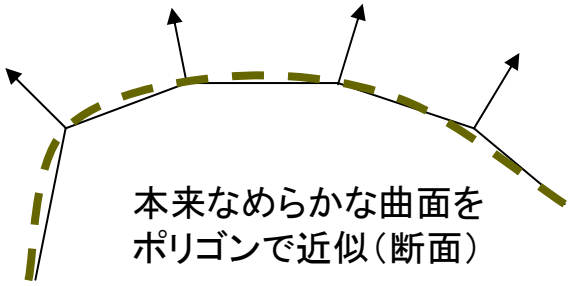
$$\vec{N} = \vec{a} \times \vec{b}$$ （ベクトルの外積）

単位法線ベクトル（大きさ1）

$$\vec{n} = \vec{N} / |\vec{N}|$$

- 法線ベクトルの設定
  - 通常は，左の式で自動算出
  - 自分で設定することも可能
  - 使用例：ポリゴンによる曲面近似

元の曲面の法線ベクトルを設定



- normal(nx, ny, nz)
  - 頂点に法線ベクトルを明示的に設定したいときに使う関数
  - 対応するvertexの直前で使用
  - 例）normal(1.0, 0.0, 0.0);
    vertex(2.0, 3.5, 3.4);

11.3 グローシェーディング

グローシェーディング

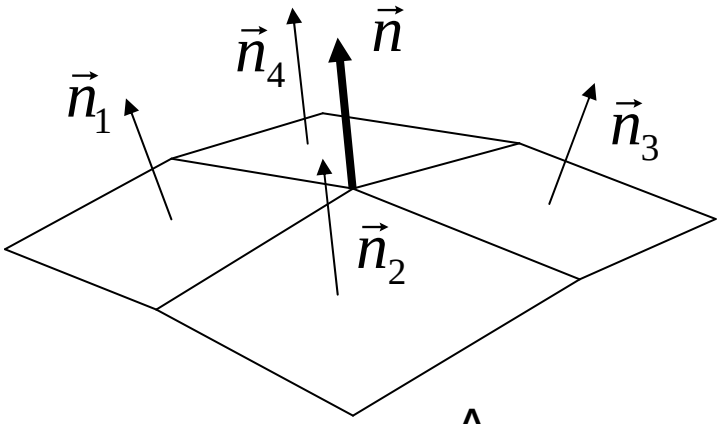
- 頂点間の描画色を補間
  - 周囲の面の法線ベクトルを平均化して，各頂点の向きを計算
  - それを用いて，頂点ごとに光の反射を計算し，描画色を決定
  - 面全体の色は，頂点の間の色を線形補間し，滑らかに描画する
  - Processing，OpenGLなどで標準的に使われている
- 描画色の計算式

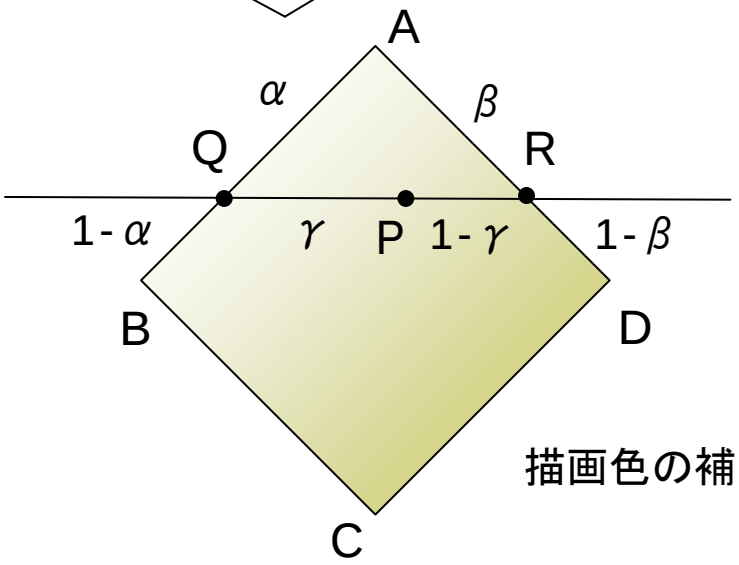
$$C_{\mathrm{Q}} = (1-\alpha)C_{\mathrm{A}} + \alpha C_{\mathrm{B}}$$

$$C_{\mathrm{R}} = (1-\beta)C_{\mathrm{A}} + \beta C_{\mathrm{D}}$$

$$C_{\mathrm{P}} = (1-\gamma)C_{\mathrm{Q}} + \gamma C_{\mathrm{R}}$$

隣接面の法線ベクトルを
平均化した頂点の法線ベクトル



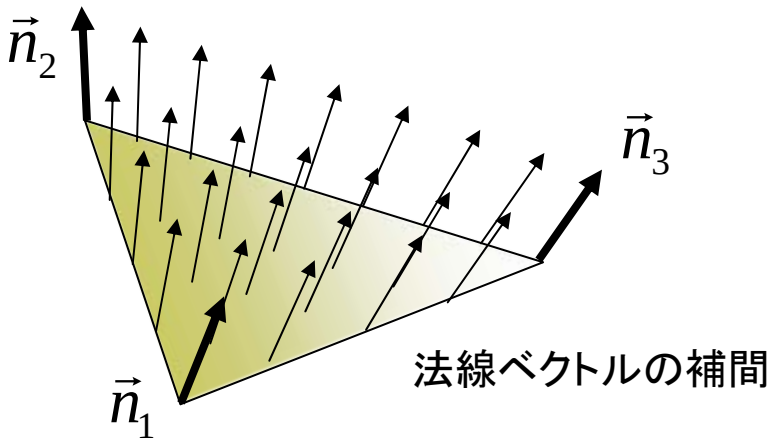


描画色の補間

11.4 フォンシェーディングとバンプマッピング

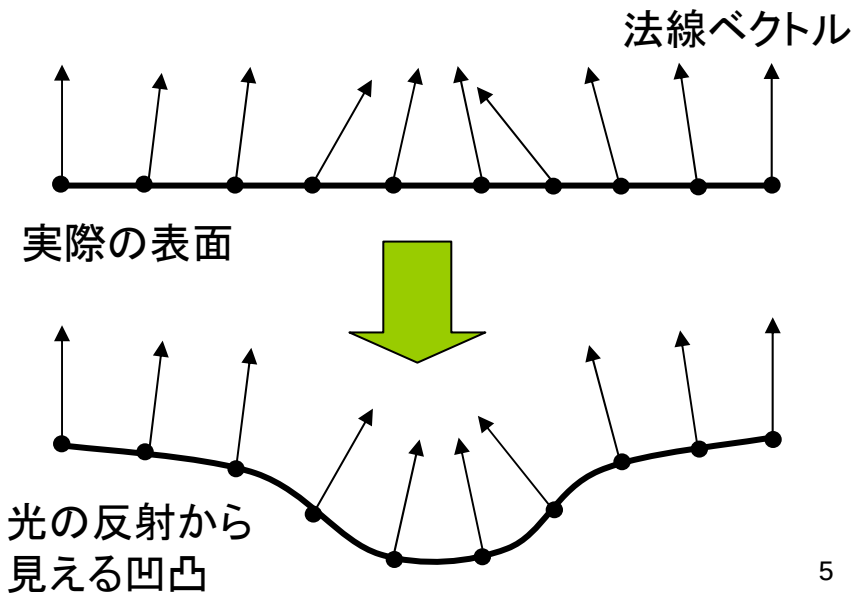
フォンシェーディング

- 法線ベクトル自体を補間
  - 色を補間するのでなく，面全体の法線ベクトルを線形補間
  - 描画時に各ピクセルの法線ベクトルを計算し，光の反射からピクセルごとの描画色を決定する
  - グローシェーディングより，光沢（つや）のある反射がリアル



バンプマッピング

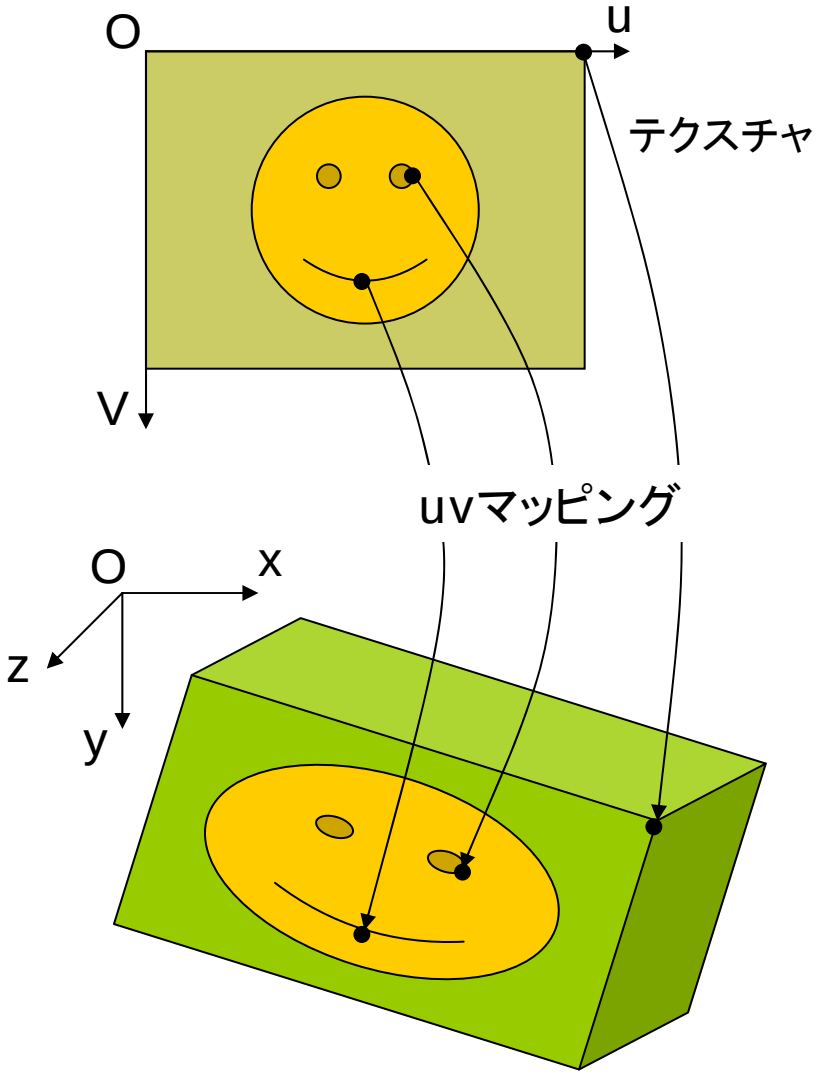
- 法線ベクトルで凹凸を表現
  - 平らな面で法線ベクトルだけを変化させることで，まるで凹凸があるかのように見せる
  - 表面の細かい凹凸を簡単かつ少ない計算量で表現できる



11.5 テクスチャマッピング

テクスチャマッピング

- テクスチャマッピングの役割
  - テクスチャ＝模様画像
  - 立体にテクスチャ(画像)を，シールのように貼りつける
  - 質感を表すのに効果てきめん
  - 例）球に世界地図を貼りつける，人体モデルに肌を貼りつける
- uv座標(テクスチャ座標)
  - テクスチャ画像の2次元座標
  - (x,y)のかわりに(u,v)を用いる
- uvマッピング
  - 2次元のテクスチャ画像を3次元空間の面に貼りつける対応づけ
  - 画像(u, v) → 空間(x, y, z)



11.6 テクスチャマッピング関数

テクスチャマッピング

- texture(画像)
  - 画像: PImage型(5.3参照)
  - テクスチャの設定
  - beginShape(), endShape()の中で指定する
- vertex(x, y, z, u, v)
  - 通常のvertex(x, y, z)の処理に加え，その点をテクスチャ座標(u, v)に対応づける
  - vertex(x, y, u, v): 2次元用
- textureMode(座標モード)
  - uv座標の指定モード
  - IMAGE: 実際の画像の座標
  - NORMALIZED: 0.0～1.0

- 使い方

```
PImage tex;  // テクスチャ画像

void setup() {
  // 省略...
  tex = loadImage("画像ファイル");
}

void draw() {
  // 省略...
  beginShape(図形モード);
  texture(tex);
  textureMode(座標モード);
  vertex(x1, y1, z1, u1, v1);
  vertex(x2, y2, z2, u2, v2);
  // 省略...
}
```

11.7 サンプルプログラム

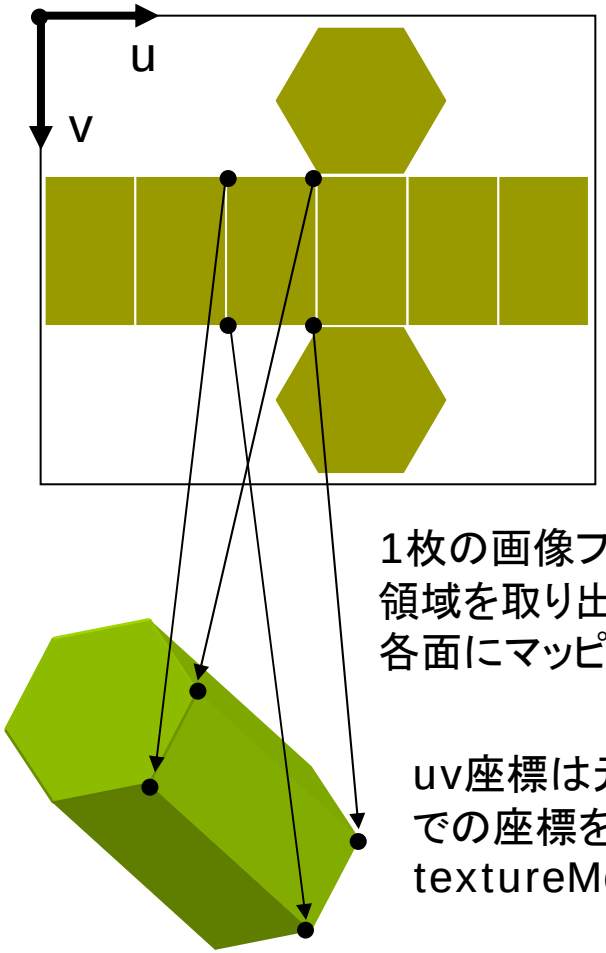
```
import processing.opengl.*;
PImage tex;
// 画像はグローバル変数推奨

void setup() {
  size(300, 300, OPENGL);
  tex =
    loadImage("kouji50m.jpg");
    // 画像は講義ホームページにある
}

void draw() {
  background(0);
  translate(width/2, height/2);
  scale(0.5);
  rotateY(-radians(frameCount));
  beginShape(QUADS);
  noStroke();
  texture(tex);
  textureMode(NORMALIZED);
  vertex(-40,-100,  0, 0, 0);
  vertex( 40,-100,  0, 1, 0);
  vertex( 40, 100, 30, 1, 1);
  vertex(-40, 100, 30, 0, 1);

  fill(#ffffff); stroke(#555555);
  vertex(-40,-100,   0);
  vertex( 40,-100,   0);
  vertex( 40, 100, -30);
  vertex(-40, 100, -30);
  endShape();
}
```

11.8 演習課題

課題

- 立体(直方体など)に，テクスチャマッピングで画像を貼りつけて，現実の物のCGを作りなさい
  - 電車やバス(直方体)
  - 本，CD，菓子などの箱(直方体)
  - ビル(直方体)や塔(四角すい)
- 提出方法
  - プログラムを保存したら，**Tools → Archive Sketch** で，**画像をまとめたZIPファイルを作る**
  - Processingフォルダにできた「プログラム名(スケッチ名).zip」というファイルを提出する
  - アップロード時に，**種類**で**「フォルダ圧縮ZIPファイル」**を選ぶ

参考：展開図画像などの利用



1枚の画像ファイルから領域を取り出して立体の各面にマッピングできる

uv座標は元の画像上での座標を使うとよい
textureMode(IMAGE)