Graphics with Processing

2007-11 シェーディングと
テクスチャマッピング

http://vilab.org

塩澤秀和

11.1 シェーディング

シェーディング

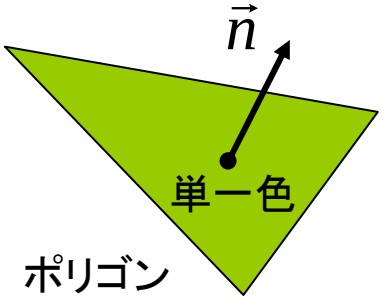
- シェーディングとは
  - Shading＝陰影づけ
  - 光の反射・材質のモデル（前回）
  - ポリゴンの陰影計算モデル
    ＝ シェーディングモデル

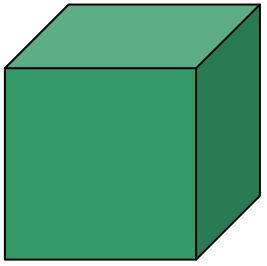
シェーディングモデル

- 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 グローシェーディング

グローシェーディング

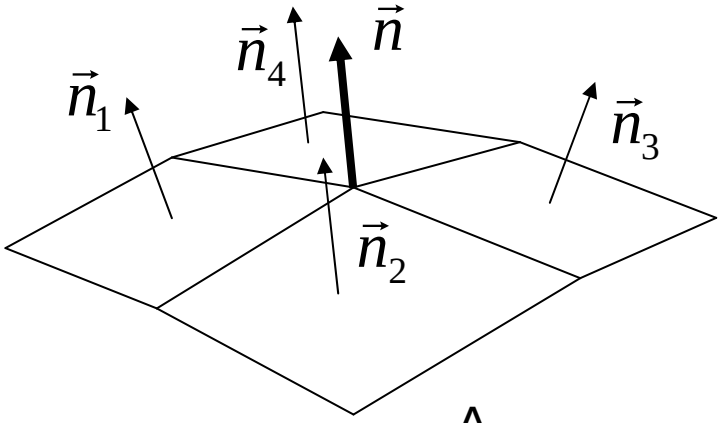
- 頂点間の描画色を補間
  - 周囲の面の法線ベクトルを平均化して，各頂点の向きを計算
  - それを用いて，頂点ごとに光の反射を計算し，描画色を決定
  - 面全体の色は，頂点の間の色を線形補間して，滑らかに描画
  - Processing，OpenGLなどで標準的に使われている
- 描画色の計算式

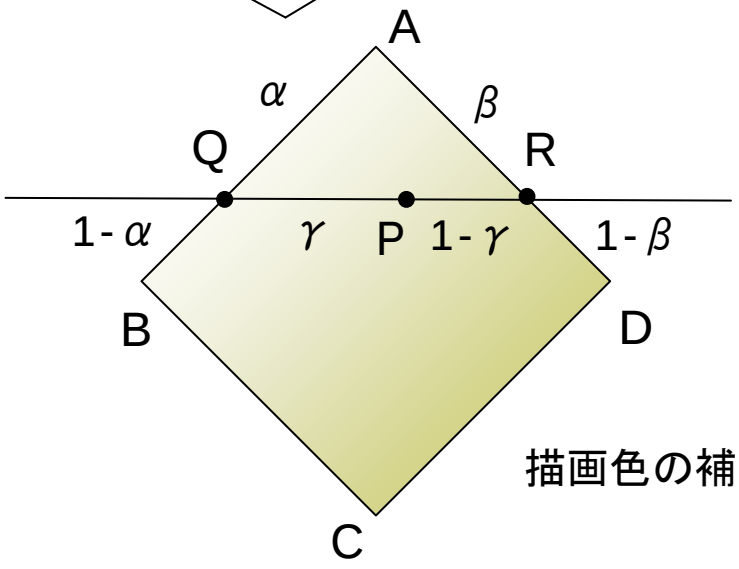
$$C_{\mathrm{Q}} = (1-\alpha)C_{\mathrm{A}} + \alpha C_{\mathrm{B}}$$

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

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

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



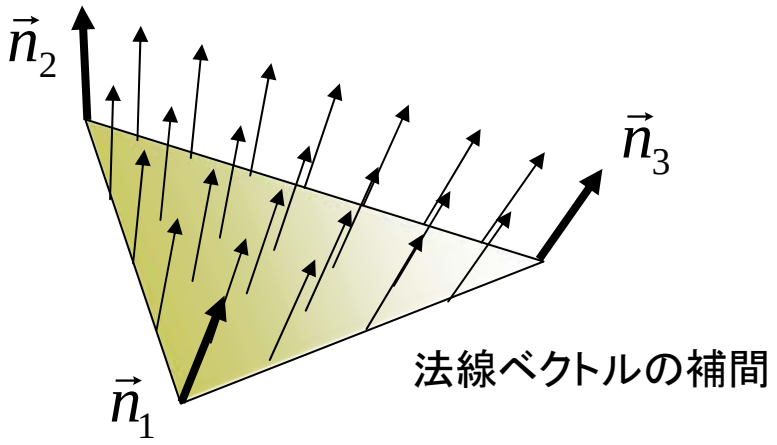


描画色の補間

11.3 フォンシェーディング

フォンシェーディング

- 面全体の法線ベクトルを補間
  - 平面の表面を光の反射についてなめらかな曲面に近似する手法
  - 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



法線ベクトルの補間

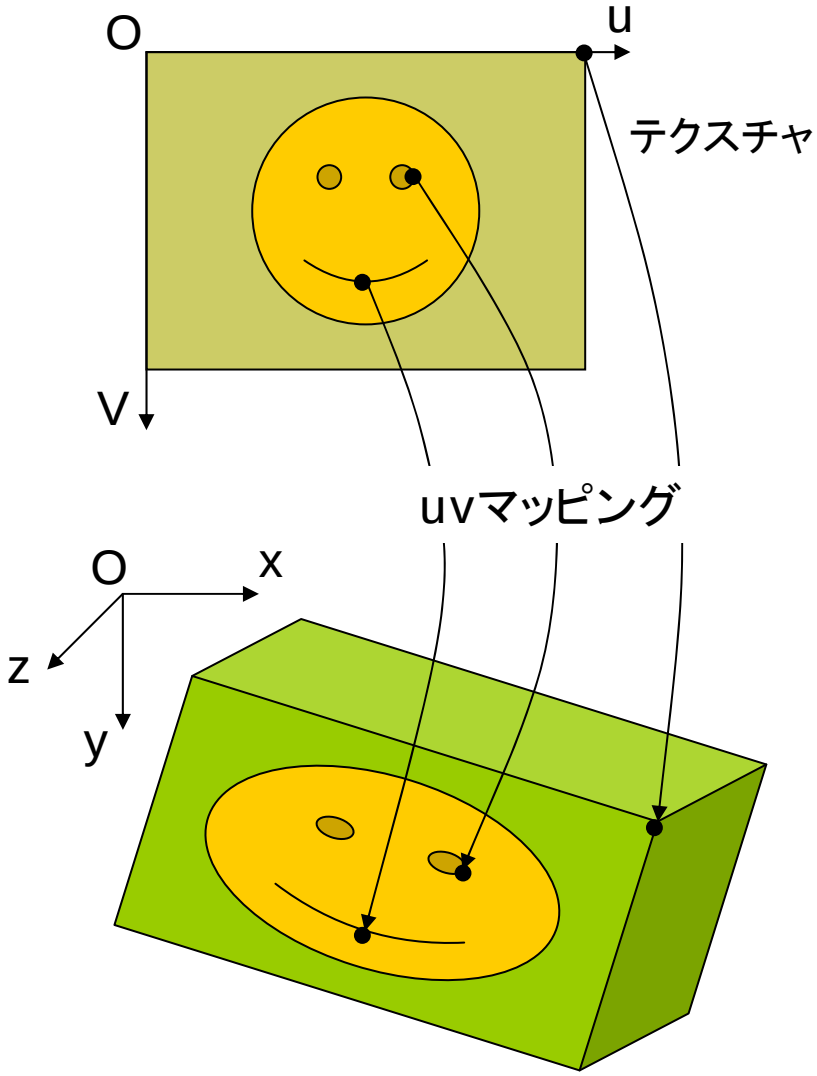
その他参考

- 法線ベクトルの明示設定
  - 通常，システムが算出（10.5）
  - 各頂点の法線ベクトルを自分で設定することも可能である
- 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
  - vertexの前に指定
  - 使用例
    normal(1.0, 0.0, 0.0);
    vertex(2.0, 3.5, 3.4);
- ポリゴンのグラデーション
  - 各頂点に別々の色（fill）をつけるとポリゴン内をなめらかに補間

11.4 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)



11.5 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...
}
```

11.6 サンプルプログラム

```
// 画像はグローバル変数推奨
PImage tex;

void setup() {
  size(300, 300, P3D);
  tex =
    loadImage("kouji50m.jpg");
    // テクスチャファイルは講義ホーム
    // ページからダウンロードし登録
}

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();
}
```

11.7 演習課題

課題

- 立方体(六面体)の各面にテクスチャを貼り付けて，回転表示するプログラムを作成しなさい
  - 各面のテクスチャは同じものでもよい(違うものでもよい)
- 今回のプログラムは**ZIPファイルにまとめてから提出すること**
  - まず，プログラムを保存する
  - 次に，**Tools→Archive Sketch**でZIPファイルにまとめる
  - すると，**workspace**フォルダに「プログラム名.zip」というファイルができるのでこれを提出する
  - アップロード時に**種類**で，**「フォルダ圧縮ZIPファイル」**を選択

参考：文字列の表示

- フォントの作成
  - 事前にフォントファイルを準備
  - Tools → Create Font...
- PFont型
  - フォントを表す変数型
- loadFont("フォントファイル名")
  - フォントの読み込み
  - 例) PFont font = loadFont ("CourierNew36.vlw");
- textFont(フォント，サイズ)
  - 描画フォントの設定
- text(文字列，x，y)
  - 文字列(String型)の描画
  - 例: Basics → Typography

11.8 参考:日本語文字列の画像作成

```
import java.awt.*;
import java.awt.image.*;
import java.awt.font.*;

// 引数は, 文字列, サイズ, 画像の幅, 画像の高さ, 文字の色
PImage makeTextImage(String str, int point, int w, int h, color fg) {
    BufferedImage bi = new BufferedImage(w, h,
                                         BufferedImage.TYPE_4BYTE_ABGR);
    Graphics gc = bi.getGraphics();
    gc.setColor(new Color(0, 0, 0, 0));
    gc.fillRect(0, 0, bi.getWidth(), bi.getHeight());
    gc.setColor(new Color(fg));
    Font fnt = new Font("SansSerif", Font.PLAIN | Font.BOLD, point);
    gc.setFont(fnt);
    gc.drawString(str, 0, bi.getHeight());
    gc.dispose();
    return new PImage(bi);
}
```