マルチメディア工学
プログラム演習手引書
(サーフェイスレンダリング)

教員　佐藤嘉伸　yoshi@image.med.osaka-u.ac.jp
TA　島田隆次　r-simada@ist.osaka-u.ac.jp

1　演習手順

1. Web ページからサンプルプログラムをダウンロードする．
2. SurfaceRendering.c の TODO コメントの部分を埋めてコードを完成させる．
3. パラメータを変化させて生成される CG 画像を比較する．
4. 比較結果を載せたレポートを作成する．

2　プログラム

2.1　サンプルプログラムの入手

マルチメディア工学の Web ページからサンプルプログラム SurfaceRenderingSampleCode.zip をダウンロードし，任意のディレクトリに展開する．

ファイル構成：

```
SurfaceRenderingSampleCode/
 ├ readme.txt
 └ src/                  ソースディレクトリ
   ├ SurfaceRendering.c   超楕円体の生成とレンダリングのメインルーチン
   ├ Point3D.h           三次元空間上の一点を表す
   ├ Triangle.h           3 点のインデックスで三角形を表す
   ├ Vector3D.c          三次元ベクトルを表す
   ├ Vector3D.h                 〃
   ├ Quaternion.c        四元数を表す
   ├ Quaternion.h             〃
   ├ SurfaceModel.c      点と三角形の集合としてサーフェイスモデルを表す
   ├ SurfaceModel.h                    〃
   ├ Image2D.c           二次元ビットマップ画像を表す
   ├ Image2D.h                  〃
   ├ Makefile            gcc 用の Makefile
   └ Makefile.vc         VC++ 用の Makefile
```

SurfaceRendering.c がメインのソースコードである．main 関数では以下の処理をしている．

1. コマンドライン引数を解析

2. createSuperEllipsoid 関数で超楕円体のサーフェイスモデルを作成
3. makeRotater 関数と rotateSurfaceModel 関数でサーフェイスモデルを回転
4. render 関数でサーフェイスモデルを二次元画像に描画
5. 二次元画像をファイルに保存

同梱されているヘッダファイルには SurfaceRendering.c で利用している各関数のプロトタイプ宣言と簡単な説明が書いてあるので，一度目を通しておくこと．

2.2 プログラムの作成

サンプルプログラムは SurfaceRendering.c の一部が未完成となっているので，その部分を埋めてプログラムを完成させる．該当箇所はコメントに**「TODO: (番号) コード作成内容」**と書いてある．以下，各番号に対応したコード作成内容について説明する．

2.2.1 p の計算

超楕円体上の座標 η(eta)，ω(omega)，大きさパラメータ $\alpha_1 \sim \alpha_3$(a1〜a3)，形状パラメータ $\epsilon_1 \sim \epsilon_2$(e1〜e2) を元に三次元空間上の座標を p に求める．

授業スライド「コンピュータグラフィックス（形状表現，陰影表現)」の 23 スライド目に式があるので，それを C 言語で記述すればよい．ただし $cos\eta$ などが負の場合，そのままではべき乗が計算できないので，絶対値のべき乗を求めた上で元の符号をつけるようにする．

2.2.2 頂点数を計算

超楕円体の頂点数を nPoints に求める．なお，etaResolution と omegaResolution はそれぞれ緯度方向と経度方向の頂点数である．31 スライド目の図を参考に．超楕円体の北極と南極ではすべての経度の頂点が重なるが，ここでは考慮する必要はない．

2.2.3 三角形の数を計算

超楕円体表面の三角形の数を nTriangles に求める．三角形 2 個で四角形を作っていることに注意．超楕円体の北極付近と南極付近では四角形を構成する三角形のうち片方の 2 頂点が重なって直線に縮退するが，ここでは考慮する必要はない．（描画の際にそういう三角形を除外すればよい）

2.2.4 η の計算式

超楕円体の緯度である η を計算する．omega が 0〜omegaResolution まで変化するので，それに対応して η が $-\frac{\pi}{2} \sim \frac{\pi}{2}$ まで変化するように式を作る．

2.2.5 ω の計算式

超楕円体の経度である ω を計算する．eta が 0〜etaResolution まで変化するので，それに対応して ω が $-\pi \sim \pi$ まで変化するように式を作る．

2.2.6 左上から順に三角形を登録していく

サーフェイスモデルに超楕円体表面の三角形を登録していく．31 スライド目の図を参考に．

頂点の順番は法線の向きに影響する．全ての三角形は表側から見て反時計回りに点を与える必要があることに注意.

2.2.7 dest の計算

原点と (x,y,z) を通る直線を軸として theta だけ回転させるための四元数を dest に求める．授業スライド「コンピュータグラフィックスの基礎（座標系，剛体変換，3 次元回転の数理，投影）」の 23 スライド目に式があるので，それを C 言語で記述すればよい．dest に値を設定するには Quaternion_set マクロを使う.

2.2.8 q2 = rotater * q * conj(rotater) の計算

q を rotater で回転した結果を q2 に求める．conj(rotater) は事前に c に求めてあるのでそれを利用すればいい．なお conj は共役複素数を表す．四元数同士の乗算は Quaternion_mult 関数を使えばよい.

2.2.9 p から (x, y) へ投影する処理

三次元上の点 p を二次元空間に平行投影する．授業スライド「コンピュータグラフィックス（形状表現，陰影表現）」の 30 スライド目に平行投影の式があるので，それを C 言語で記述すればよい.

2.2.10 直線に縮退した三角形は描画をスキップする

超楕円体の北極付近と南極付近の三角形は 3 つの頂点のうち 2 点が同座標になり，直線に縮退してしまう．直線に縮退した三角形を描画するとゴミが出るので，描画しないようにする必要がある．三角形の各頂点の番号は事前に t に取得してあるのでそれを利用し，SurfaceModel_getPoint 関数でモデルから頂点の座標を取得すればよい.

なお浮動小数点の計算誤差のために頂点が全く同じ座標にはならないことがあるため，頂点間の距離がある小さな値 (10^{-10} など) 以下なら同じ座標と判定するようにしたほうがよい．2 点間の距離を計算するには，2 点間のベクトルを計算するには Vector3D_betweenPoints 関数とベクトルの長さを計算する Vector3D_length 関数を使うとよい.

2.2.11 法線ベクトル n を求める

三角形の法線ベクトルを n に求める．三角形の任意の 2 辺のベクトルの外積は三角形の面に垂直になる（法線ベクトルになる）ことを利用する．31 スライドに法線方向の例が載っているので参考に．外積は Vector3D_outerProduct 関数で求めることができる.

なお外積を計算しただけでは長さが 1 とは限らないので，Vector3D_normalize 関数を用いて正規化するのを忘れないように.

2.2.12 カメラ方向と面の向きを比べて面の表裏を判定

カメラの方向を向いていない（裏を向いている，見えない）三角形は描画しないようにする．カメラの方向は投影面に垂直なので (0,0,1) または (0,0,-1) である．（カメラが Z 軸正の方向にあるとするか負の方向にあるとするかで符号は異なる）32 スライド目を参考に．内積は Vector3D_innerProduct 関数で求めることができる.

2.2.13 RGB 各要素ごとに拡散反射成分を計算

RGB の各要素ごとに拡散反射成分を計算して三角形の色を求める．R は RED(lightColor)，G は GREEN(lightColor)，B は BLUE(lightColor) で取得できるので，各要素ごとに $I_d = K_d * I * cos\theta$ を求めればよい．27 スライド目を参考に．

なお光源が面の裏にある場合 ($cos\theta$ が負の場合) は真っ黒 (RGB(0,0,0)) にする必要があることに注意．

2.2.14 project() を呼び出して三角形の各頂点の二次元画像上の座標を計算

project 関数を 3 回呼び出して三角形の各頂点の二次元座標を求める．

2.2.15 二次元画像の中心が (0,0) になるように平行移動，座標軸を合わせるために y 軸反転

上で求めた二次元座標を画像サイズの半分だけ平行移動して，原点が画像の中央にくるようにする．また三次元空間と二次元画像では Y 軸の向きが逆なので Y 軸を反転させておく．画像サイズは Image2D_getSizeX 関数と Image2D_getSizeY 関数で得られる．

2.3 コンパイルと実行

サンプルプログラムには Makefile が付属している．gcc を使用している場合は「make」で，VisualC++ を使用している場合は「nmake /f Makefile.vc」でコンパイルすればよい．

コンパイルできたら超楕円体の 5 つのパラメータ，回転パラメータ，光源の方向などを変化させて画像を作成し，比較する．コマンドライン引数は以下のとおり．

```
$ SurfaceRendering a1 a2 a3 ε 1 ε 2 η Reso ω Reso
    rotaterX rotaterY rotaterZ rotaterTheta lightDirX lightDirY lightDirZ
    lightColorR lightColorG lightColorB diffRefCoeff imageSizeX imageSizeY filename

  a1 a2 a3 ε 1 ε 2: 超楕円体のパラメータ
  η Reso ω Reso: 緯度/経度方向の頂点数
  rotaterX rotaterY rotaterZ: 原点と(z,y,z)を通る直線を回転軸とする
  rotaterTheta: 回転角
  lightDirX lightDirY lightDirZ: 平行光源の方向
  lightColorR lightColorG lightColorB: 光源の色
  diffRefCoeff: 拡散反射係数
  imageSizeX imageSizeY: 二次元画像のサイズ
  filename: 二次元画像の保存ファイル名
```