コンピュータグラフィックスＳ 演習資料

第４回　シェーディング、マッピング

九州工業大学 情報工学部 システム創成情報工学科
講義担当：尾下真樹

1. 演習準備

今回の演習も、前回までの演習で作成したプログラムに続けて変更を行う。
まずは、シェーディングの演習のため、描画処理で、回転する一つの四角すいを描画するように変更する。

```
void   display( void )
{
        // 画面をクリア（ピクセルデータとＺバッファの両方をクリア）
        glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );

        // 変換行列を設定（ワールド座標系→カメラ座標系）
        glMatrixMode( GL_MODELVIEW );
        glLoadIdentity();
        glTranslatef( 0.0,   0.0, - camera_distance );
        glRotatef( - camera_pitch, 1.0, 0.0, 0.0 );
        glTranslatef( 0.0, -1.0, 0.0 );

        // 光源位置を設定（モデルビュー行列の変更にあわせて再設定）
        float   light0_position[] = { 10.0, 10.0, 10.0, 1.0 };
        glLightfv( GL_LIGHT0, GL_POSITION, light0_position );

        // 地面を描画
        glBegin( GL_POLYGON );
                glNormal3f( 0.0, 1.0, 0.0 );
                glColor3f( 0.5, 0.8, 0.5 );

                glVertex3f( 5.0, 0.0, 5.0 );
                glVertex3f( 5.0, 0.0,-5.0 );
                glVertex3f(-5.0, 0.0,-5.0 );
                glVertex3f(-5.0, 0.0, 5.0 );
        glEnd();
/*
        他の描画処理は、全て、削除するかコメントアウト
*/
        // 中心に赤い四角すいを描画（地面にめり込まないように (0,1,0) に移動）
        glTranslatef( 0.0, 1.0, 0.0 );
        glRotatef( theta_cycle, 0.0, 1.0, 0.0 );
        glColor3f( 1.0, 0.0, 0.0 );
        renderPyramid3();

        // バックバッファに描画した画面をフロントバッファに表示
        glutSwapBuffers();
}
```

2. シェーディング（光源設定の変更）

本演習では、光源の設定の変更による、シェーディングの効果の変化を確認する。
現在のプログラムでは、ワールド座標系の（10.0, 10.0, 10.0）の位置に、点光源を置く設定になっている。

まず、点光源の効果をより大きくするために、光源の位置を（5.0, 3.0, 5.0）に移動してみる。

```
void   display( void )
{
        ・・・・・・

        // 光源位置を設定（モデルビュー行列の変更にあわせて再設定）
        float   light0_position[] = { 5.0, 3.0, 5.0, 1.0 };
        glLightfv( GL_LIGHT0, GL_POSITION, light0_position );

        // 地面を描画
        glBegin( GL_POLYGON );
                glNormal3f( 0.0, 1.0, 0.0 );
                glColor3f( 0.5, 0.8, 0.5 );

                glVertex3f( 5.0, 0.0, 5.0 );
                glVertex3f( 5.0, 0.0,-5.0 );
                glVertex3f(-5.0, 0.0,-5.0 );
                glVertex3f(-5.0, 0.0, 5.0 );
        glEnd();

        ・・・・・・
}
```

以上の修正を行い、プログラムをコンパイル・実行し、点光源の効果を確認すること。
地面が、原点を中心に x 方向・z 方向の幅がともに 10 の四角形として描画されており、光源の位置が(5.0, 3.0, 5.0)にあることから、ちょうど、地面の右手前の角に、光が最も良く当たり、明るくなっていることが分かる。

現在は、地面の四角形の 4 つの頂点に対してのみ、照明効果の計算（色の計算）が行われ、四角面の内部の各ピクセルは、4 つの頂点の色を補間することで計算されている。そのため、四角面の内部では、照明効果があまり正確ではない。
そこで、照明効果をより正確に表現するために、地面を x 方向・z 方向の幅がともに 1 の四角形を、10×10＝100 個並べたものとして描画するように修正してみる。

```
void   display( void )
{
        ・・・・・・

        // 光源位置を設定（モデルビュー行列の変更にあわせて再設定）
        float   light0_position[] = { 5.0, 3.0, 5.0, 1.0 };
        glLightfv( GL_LIGHT0, GL_POSITION, light0_position );

        // 地面を描画
        int   i, j;
        glBegin( GL_QUADS );
                glNormal3f( 0.0, 1.0, 0.0 );
                glColor3f( 0.5, 0.8, 0.5 );
                for ( i=0; i<10; i++ )
                {
                        for ( j=0; j<10; j++ )
                        {
                                glVertex3f( i - 4.0, 0.0, j - 4.0 );
                                glVertex3f( i - 4.0, 0.0, j - 5.0 );
                                glVertex3f( i - 5.0, 0.0, j - 5.0 );
                                glVertex3f( i - 5.0, 0.0, j - 4.0 );
                        }
                }
        glEnd();

        ・・・・・・
}
```

このプログラムでは、並べて描画する四角形の 4 つの頂点が適切な座標になるように、i, j にもとづいて頂点座標を計算している。

以上の修正を行い、プログラムをコンパイル・実行し、さきほどのプログラムと比較して、どのように描画結果が変わるかを確認すること。
結果的に、描画される地面の大きさは同じだが、各四角形の頂点ごとに光源計算が行われるため、照明効果がより正確に計算され、結果的に、点光源の効果がよく分かるようになっている。

次に、点光源ではなく、ワールド座標系の原点から見て(5.0, 3.0, 5.0)の方向からの平行光源を設定してみる。
光源位置の w 座標を 0 とすることで、点光源ではなく、平行光源となる。

```
void   display( void )
{
        ・・・・・・

        // 光源位置を設定（モデルビュー行列の変更にあわせて再設定）
        float   light0_position[] = { 5.0, 3.0, 5.0, 0.0 };
        glLightfv( GL_LIGHT0, GL_POSITION, light0_position );

        ・・・・・・
}
```

以上の修正を行い、プログラムをコンパイル・実行し、さきほどのプログラムと比較して、どのように描画結果が変わるかを確認すること。
平行光源に変更することで、地面を構成する頂点に対して、同じ方向から光が来ることになり、結果的に、地面全体が単一の色で描画されることになる。（地面の全ての頂点で法線が等しいため。）

最後に、演習課題として、最初の状態のように地面の右手前が照らされるのではなく、地面の左手前が照らされるように、光源の位置を修正してみよ。

```
void   display( void )
{
        ・・・・・・

        // 光源位置を設定（モデルビュー行列の変更にあわせて再設定）
        float   light0_position[] = {              };
        glLightfv( GL_LIGHT0, GL_POSITION, light0_position );

        ・・・・・・
}
```

3. テクスチャマッピング

本演習では、テクスチャマッピングの例として、地面にテクスチャマッピングを適用してみる。
ファイルから読み込むテクスチャ画像のデータを格納するためのグローバル変数を追加する。

```
// アニメーションのための変数
float     theta_cycle = 0.0;

// 視点操作のための変数
float     camera_pitch = -30.0;          // X軸を中心とする回転角度
```

```
// マウスのドラッグのための変数
int       drag_mouse_r = 0;              // 右ボタンがドラッグ中かどうかのフラグ
                                         // (1:ドラッグ中, 0:非ドラッグ中)
int       last_mouse_x;                  // 最後に記録されたマウスカーソルのX座標
int       last_mouse_y;                  // 最後に記録されたマウスカーソルのY座標

// テクスチャ画像データ
int       tex_width;
int       tex_height;
unsigned char *   tex_image = NULL;
```

ファイルから BMP 画像を読み込むために、サンプルプログラムとして用意されている、BMP 画像読み込みを行う loadBitmap 関数が記述されたソースファイル（bitmap.h, bitmap.cpp）を使用する。
そのため、まず、プログラムの先頭に、ヘッダファイルの読み込みを追加する。

```
// GLUT ヘッダファイルのインクルード
#include <GL/glut.h>

// Bitmap 読み込み関数のためのヘッダファイルのインクルード
#include "bitmap.h"
```

次に、最初に呼ばれる環境初期化関数に、テクスチャ画像を読み込みんで、OpenGL が使うテクスチャとして登録する処理を追加する。
ここでは、最初に loadBitmap 関数を呼び出して、kyushu.bmp というファイル名の画像を読み込んでいる。
その後、読み込んだ画像データを、テクスチャとして設定している。

```
//
//   環境初期化関数
//
void   initEnvironment( void )
{
          ・・・・・・

          // テクスチャ画像の読み込み
          loadBitmap( "kyushu.bmp", &tex_image, &tex_width, &tex_height );
          if ( tex_image == NULL )
                    return;

          // テクスチャオブジェクトの設定
          glTexImage2D( GL_TEXTURE_2D, 0, GL_RGB, tex_width, tex_height, 0,
                              GL_RGB, GL_UNSIGNED_BYTE, tex_image );

          // テクスチャマッピングの方法を設定
          glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT );
          glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT );
          glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR );
          glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR );
}
```

最後に、地面のポリゴンを描画するときに、テクスチャマッピングを行うための処理を追加する。テクスチャ画像を使用するように設定し、各頂点にテクスチャ座標を指定する。描画が終わったら、テクスチャマッピングをオフにする。

```
//
//   ウィンドウ再描画時に呼ばれるコールバック関数
//
void   display( void )
{
          ・・・・・・
```

```
        // 地面を描画（テクスチャマッピング）
        glEnable( GL_TEXTURE_2D );
        glTexEnvf( GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL );

        int   i, j;
        glBegin( GL_QUADS );
                glNormal3f( 0.0, 1.0, 0.0 );
                glColor3f( 1.0, 1.0, 1.0 );
                for ( i=0; i<10; i++ )
                {
                        for ( j=0; j<10; j++ )
                        {
                                glTexCoord2f( (i+1) * 0.1, (j+1) * 0.1 );
                                glVertex3f( i - 4.0, 0.0, j - 4.0 );
                                glTexCoord2f( (i+1) * 0.1, j * 0.1 );
                                glVertex3f( i - 4.0, 0.0, j - 5.0 );
                                glTexCoord2f( i * 0.1, j * 0.1 );
                                glVertex3f( i - 5.0, 0.0, j - 5.0 );
                                glTexCoord2f( i * 0.1, (j+1) * 0.1 );
                                glVertex3f( i - 5.0, 0.0, j - 4.0 );
                        }
                }
        glEnd();

        glDisable( GL_TEXTURE_2D );

        ・・・・・・
}
```

以上の処理の追加により、テクスチャマッピングが行われた状態で地面が描画されるので、コンパイル・実行して確認する。
このプログラムでは、テクスチャ座標（u,v）の（0.0, 0.0）～（1.0, 1.0） の範囲が地面に貼り付けられるように、i, j にもとづいてテクスチャ座標を計算している。例えば、i=0, j=0 のときには、4 頂点のテクスチャ座標は（0.1, 0.1）（0.1, 0.0）（0.0, 0.0）（0.0, 0.1）となり、i=9, j=9 のときには、4 頂点のテクスチャ座標は（1.0, 1.0）（1.0, 0.9）（0.9, 0.9）（0.9, 1.0）となる。

なお、コンパイルの際には、bitmap.h と bitmap.cpp の 2 つのソースファイルを Visual Studio プロジェクトに追加して、一緒にコンパイル・リンクを行う必要がある。（Visual Studio プロジェクトへのソースファイルの追加方法は、コンパイル方法の資料を参照する。）
また、実行時には、プロジェクトの実行ディレクトリ（プロジェクトの実行ディレクトリの設定を変更していなければ、プロジェクトファイルやソースファイルとディレクトリ）に、画像ファイル kyushu.bmp を置いておく必要がある。

以上の修正で、テクスチャマッピングが正しく行われることを確認したら、次は、テクスチャマッピングの適用方法を、以下のように、変更してみる。

```
//
//   ウィンドウ再描画時に呼ばれるコールバック関数
//
void   display( void )
{
        ・・・・・・

        // 地面を描画（テクスチャマッピング）
        glEnable( GL_TEXTURE_2D );
        glTexEnvf( GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE );

        ・・・・・・
}
```

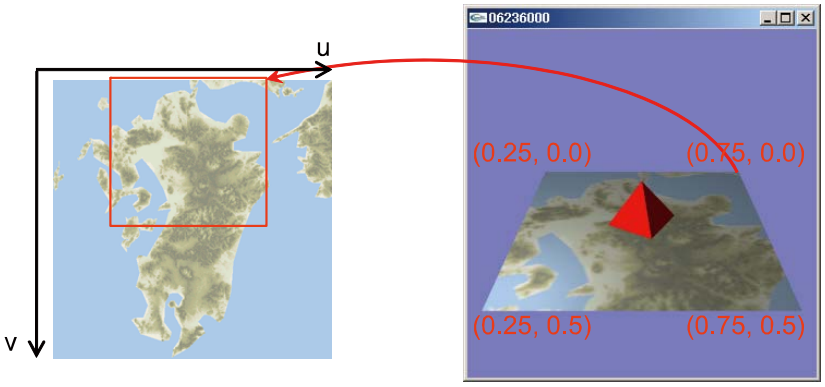
適用方法を GL_MODULATE に変更することで、光源処理により色が変化するポリゴンのもとの色の影響を受けるようになる。コンパイル・実行して、確認してみること。

次に、地面を描画する際の、各頂点のテクスチャ座標（u,v）を変更することで、テクスチャの張り方を変更してみる。

```
//
//   ウィンドウ再描画時に呼ばれるコールバック関数
//
void   display( void )
{
          ・・・・・・

          // 地面を描画（テクスチャマッピング）
          glEnable( GL_TEXTURE_2D );
          glTexEnvf( GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL );

          int   i, j;
          glBegin( GL_QUADS );
                    glNormal3f( 0.0, 1.0, 0.0 );
                    glColor3f( 1.0, 1.0, 1.0 );
                    for ( i=0; i<10; i++ )
                    {
                              for ( j=0; j<10; j++ )
                              {
                                        glTexCoord2f( (i+1) * 0.2, (j+1) * 0.2 );
                                        glVertex3f( i - 4.0, 0.0, j - 4.0 );
                                        glTexCoord2f( (i+1) * 0.2, j * 0.2 );
                                        glVertex3f( i - 4.0, 0.0, j - 5.0 );
                                        glTexCoord2f( i * 0.2, j * 0.2 );
                                        glVertex3f( i - 5.0, 0.0, j - 5.0 );
                                        glTexCoord2f( i * 0.2, (j+1) * 0.2 );
                                        glVertex3f( i - 5.0, 0.0, j - 4.0 );
                              }
                    }
          glEnd();

          glDisable( GL_TEXTURE_2D );

          ・・・・・・
}
```

以上のように、各頂点のテクスチャ座標（u,v）が 2 倍になるように修正し、コンパイル・実行した結果を確認せよ。
このプログラムでは、テクスチャ座標（u,v）の（0.0, 0.0）～（2.0, 2.0） の範囲が地面に貼り付けられる。テクスチャの初期化時に、glTexParameteri() 関数を使ってテクスチャが繰り返されるように設定しているので（GL_REPEATE）、テクスチャ座標値が 1.0 を超えた箇所には、テクスチャが繰り返し貼り付けられることになる。

最後に、演習課題として、下記の図の通り、テクスチャ画像の 1/4 の領域のみを地面に貼り付けるように、テクスチャ座標を適切に修正してみよ。




```
void   display( void )
{
        . . . . . .

        int   i, j;
        glBegin( GL_QUADS );
                glNormal3f( 0.0, 1.0, 0.0 );
                glColor3f( 1.0, 1.0, 1.0 );
                for ( i=0; i<10; i++ )
                {
                        for ( j=0; j<10; j++ )
                        {
                                glTexCoord2f(                    );
                                glVertex3f( i - 4.0, 0.0, j - 4.0 );
                                glTexCoord2f(                    );
                                glVertex3f( i - 4.0, 0.0, j - 5.0 );
                                glTexCoord2f(                    );
                                glVertex3f( i - 5.0, 0.0, j - 5.0 );
                                glTexCoord2f(                    );
                                glVertex3f( i - 5.0, 0.0, j - 4.0 );
                        }
                }
        glEnd();

        . . . . . .
}
```

もし、うまくテクスチャ座標を設定することが難しければ、まずは左上 1/4 のみを地面全体に貼り付けるように各頂点のテクスチャ座標を修正して、その後、各頂点のテクスチャ座標を右方向に 0.25 ずらすように修正すると良い。