「情報工学」ＣＧ 課題６

令和３年１１月１７日（水曜日）

提出締め切り 令和３年１１月２３日（火曜日）２４：００提出

提出物：提出物は各課題（課題１、課題２）の最後の状態のプログラムのソースコードとする。

提出：okamoto.shunsuke.17@shizuoka.ac.jp（三浦研岡本隼輔君）
件名：情報工学課題６　学籍番号　名前
提出方法：添付ファイル、本文にも　TA が管理しやすくなるので、学籍番号、名前を必ず記入すること。

曲線の描画と場合分け

グラフィック

- 照明
- 回転と平行移動の順序による違い

C 言語

- 構造体

サンプルプログラム

```
/*
 * rotCube.c
 *
 * Date   : October 31, 2008 (Fri)
 * Author : Kenjiro T. MIURA, Shizuoka University, Japan
 *
 */
#include <stdlib.h>
#include <math.h>
#include <GL/glut.h>
#include <stdio.h>

typedef struct s_vertex {
  float x, y, z; /* 座標 (x, y, z) */
} vertex;

typedef struct s_normal {
  float nx, ny, nz; /* 座標 (x, y, z) */
} normal;

typedef struct s_face {
  int v[4];   /* 頂点インデックス          */
  int n;      /* 法線ベクトルインデックス */
} face;

typedef struct s_fmodel {
  int faceNumber; /* モデルに含まれる面の数 */
  face *faces;
} fmodel;


/* 頂点のデータ */
vertex vertCube[8] =
```

```
{
   0.0f,   0.0f,   0.0f,
  30.0f,   0.0f,   0.0f,
  30.0f,  15.0f,   0.0f,
   0.0f,  15.0f,   0.0f,
   0.0f,   0.0f,  10.0f,
  30.0f,   0.0f,  10.0f,
  30.0f,  15.0f,  10.0f,
   0.0f,  15.0f,  10.0f
};

/* 面の法線ベクトルデータ */
normal normalCube[6] =
{
   0.0f,   0.0f,  -1.0f,
   0.0f,   0.0f,   1.0f,
   0.0f,  -1.0f,   0.0f,
   1.0f,   0.0f,   0.0f,
   0.0f,   1.0f,   0.0f,
  -1.0f,   0.0f,   0.0f
};

/* 面の頂点データ */
face faceCube[6] =
{
  3, 2, 1, 0,  0,
  4, 5, 6, 7,  1,
  0, 1, 5, 4,  2,
  1, 2, 6, 5,  3,
  2, 3, 7, 6,  4,
  0, 4, 7, 3,  5
};

/* モデルのデータ */
fmodel fCube = {
  6,
  faceCube
};

/* 視点と注視点のデータ */
static GLdouble eye[3] = { 100.0, 100.0, 50.0 };
        /*視点の位置*/
static GLdouble center[3] = { 0.0, 0.0, 0.0 };
        /*注視点*/
static GLdouble up[3]; /* ビューアップベクトル */

/* 照明の位置 */
static GLfloat light_position0[] = { 0.0, 0.0, 1.0, 0.0 }; /*  平行光線  */
static GLfloat light_position1[] = { 50.0, 0.0, 50.0, 1.0 };/*  点光源  */
static GLfloat light_position2[] = { -50.0, 0.0, 50.0, 1.0 };/*  点光源  */

/* 視点の回転角 */
static int spin_eye = 0;

/* 視点の正方向への回転 */
void
rotEyePlus(void)
{
    spin_eye = ( spin_eye + 15 ) % 360;           /*１５°加える*/
}

/* 視点の負方向への回転 */
void
rotEyeMinus(void)
{
    spin_eye = ( spin_eye - 15 ) % 360;           /*１５°差し引く*/
}
static int projection=20;

/* 照明の設定 */
void
initLights(void)
{
```

```
    GLfloat light0_diffuse[] = { 0.9, 0.9, 0.9, 1.0 }; /* 拡散成分 */
    GLfloat light1_diffuse[] = { 0.5, 0.5, 0.5, 1.0 }; /* 拡散成分 */

    GLfloat light_specular[] = { 0.3, 0.3, 0.3, 1.0 }; /* 鏡面成分 */

    GLfloat lmodel_ambient[] = { 0.2, 0.2, 0.2, 1.0 }; /* 環境光成分 */

    glLightModelfv(GL_LIGHT_MODEL_AMBIENT, lmodel_ambient );       /*環境光*/

    glLightfv ( GL_LIGHT0, GL_DIFFUSE, light0_diffuse );
    /*light0の拡散光の設定*/
    //glLightfv ( GL_LIGHT0, GL_SPECULAR, light_specular );

    glLightfv ( GL_LIGHT1, GL_DIFFUSE, light1_diffuse );
    /*light1の拡散光の設定*/
    glLightfv ( GL_LIGHT1, GL_SPECULAR, light_specular );
    /*light1の鏡面反射光の設定*/
    glLightfv ( GL_LIGHT2, GL_DIFFUSE, light1_diffuse );
    /*light2の拡散光の設定*/
    glLightfv ( GL_LIGHT2, GL_SPECULAR, light_specular );
    /*light2の鏡面反射光の設定*/

    glEnable ( GL_LIGHTING );
                /*照明効果を有効にする*/
    glEnable ( GL_LIGHT0 );
                /*light0による照明を有効にする*/
    glEnable ( GL_LIGHT1 );
                /*light1による照明を有効にする*/
    glEnable ( GL_LIGHT2 );
                /*light2による照明を有効にする*/
}

/*オブジェクトの描画*/
void
drawFModel(fmodel *fmo) {
  int i, j;         /* カウンタ     */
  float x, y, z;    /* 頂点の座標値 */
  float nx, ny, nz; /* 法線ベクトルの座標値 */
  int faceNum;

  faceNum = fmo->faceNumber;
                /*構造体ポインタ(fmo)の指す構造体のメンバー(faceNumber)へのアクセス
*/
  glBegin(GL_QUADS);
                /*四角形を描く*/
    for (i=0; i<faceNum; ++i) {
                nx = normalCube[fmo->faces[i].n].nx;

        ny = normalCube[fmo->faces[i].n].ny;
        nz = normalCube[fmo->faces[i].n].nz;
        glNormal3f(nx, ny, nz);
                /*１つの面を構成する４頂点の法線の座標*/
                for (j=0; j<4; ++j) {

                    x = vertCube[fmo->faces[i].v[j]].x;
                    y = vertCube[fmo->faces[i].v[j]].y;
                    z = vertCube[fmo->faces[i].v[j]].z;
                    glVertex3f(x, y, z);
                /*四角形を構成する４点（頂点）の座標*/
                }
    }
        glEnd();
}

/* 座標軸の描画 */
void
drawAxis() {
  glDisable( GL_LIGHTING );                         /* 照明効果を無効にする。 */

  /* x軸(レッド) */
  glColor3f(1.0f, 0.0f, 0.0f);
  glBegin(GL_LINES);
    glVertex3f(-100.0f, 0.0f, 0.0f);
```

```
    glVertex3f(100.0f, 0.0f, 0.0f);
  glEnd();

  /* y軸(グリーン) */
  glColor3f(0.0f, 1.0f, 0.0f);
  glBegin(GL_LINES);
    glVertex3f(0.0f, -100.0f, 0.0f);
    glVertex3f(0.0f, 100.0f, 0.0f);
  glEnd();

  /* z軸(ブルー) */
  glColor3f(0.0f, 0.0f, 1.0f);
  glBegin(GL_LINES);
    glVertex3f(0.0f, 0.0f, -100.0f);
    glVertex3f(0.0f, 0.0f, 100.0f);
  glEnd();

  glEnable(GL_LIGHTING);                    /* 照明効果を有効にする。 */
}

void
ourDisplay(void)
{
  GLfloat material_color[4] = { 1.0, 0.0, 0.0, 1.0 }; /* 拡散反射成分 */
  GLfloat material_specular[4] = { 0.2, 0.2, 0.2, 1.0 }; /* 鏡面反射成分 */

  /* バッファのクリア */
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

  /* 鏡面反射成分のセット */
  glMaterialfv(GL_FRONT, GL_SPECULAR, material_specular);
  /*材質の特性（前面、鏡面反射光、成分(material_specular)）の設定*/

  /* モデルビュー行列 */
  glMatrixMode ( GL_MODELVIEW );
  glLoadIdentity ();
  /* ビュー行列をセットする。*/
  gluLookAt(eye[0], eye[1], eye[2], center[0], center[1], center[2],
            up[0], up[1], up[2]);
                /*視点の設定（視点、注視点、方向）*/

  glLightfv ( GL_LIGHT0, GL_POSITION, light_position0 );
  /*light0をlight_position0の位置に設置*/
  glLightfv ( GL_LIGHT1, GL_POSITION, light_position1 );
  /*light0をlight_position1の位置に設置*/
  glLightfv ( GL_LIGHT1, GL_POSITION, light_position2 );
  /*light0をlight_position2の位置に設置*/

  /* 視点の回転移動        */
  glRotated ( (GLdouble) spin_eye, 0.0, 0.0, 1.0 );
        /*z軸正の方向にspin_eye度だけ回転*/

  /* x, y, z軸の描画 */
  drawAxis();

  /* レッド */
  glMaterialfv(GL_FRONT, GL_DIFFUSE, material_color);
  /*材質の特性（前面、拡散光成分、成分（material_color0））の設定*/

  drawFModel( &fCube );                     /*fCubeについてdrawModelを実行する*/

  glFlush();
}

void
ourInit (void)
{
  glClearColor(0.0, 0.0, 0.0, 1.0);                 /*背景色の指定*/
  glDepthFunc ( GL_LESS );/*デプステストのための比較関数GL_LESS */
  glEnable ( GL_DEPTH_TEST );                       /*デプステストを有効にする*/

  initLights();                                     /*照明の設定*/
}
```

```
/*
 *  ウインドウが最初にオープンした時やウインドウが移動やリサイズされた時
 *  呼ばれる。
 */
void
ourReshape(int width, int height)
{
    int i; /* カウンター */
    GLdouble vector[3]; /* ビューアップベクトル計算用ベクトル */
    GLdouble norm; /* ベクトルのノルム */

    glViewport (0, 0, width, height);
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity();
        gluPerspective(60.0, (GLfloat) width/(GLfloat) height, 10.0, 1000.0);
        glutPostRedisplay();
        glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();

    /* ビューアップベクトルを計算する。*/
    for ( i=0; i<3; i++ ){
        vector[i] = center[i] - eye[i];
    }
    up[0] = - vector[0] * vector[2];
    up[1] = - vector[1] * vector[2];
    up[2] = vector[0] * vector[0] + vector[1] * vector[1];
    norm = up[0] * up[0] + up[1] * up[1] + up[2] * up[2];
    norm = sqrt ( norm );

    for ( i=0; i<3; ++i ) up[i] /= norm;
}



void ourKeyboard(unsigned char key,int x,int y)
{
        switch(key)
        {
        case 'p':
        case 'P':
                rotEyePlus();
                break;

        case 'm':
        case 'M':
                rotEyeMinus();
        }

        glutPostRedisplay();
        /* ourKeyboard()が呼ばれた後でourDisplay()を呼ぶことを指示  */
}

/*  メイン */
int main(int argc, char **argv)
{
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(400, 400);
        glutInitWindowPosition(100, 100);
        glutCreateWindow("rotCube");
        ourInit();
        glutReshapeFunc(ourReshape);
        glutDisplayFunc(ourDisplay);
        glutKeyboardFunc(ourKeyboard);

        glutMainLoop();
        return 0;
}
```

課題 1

(a) 直方体の拡散光成分の色を赤から青(0,0,1)に変更する.

(b) 直方体の鏡面光成分の色を(0.2, 0.2,0.2,1.0)を(0.6,0.6,0.6,1.0)に変更して，直方体を回転させ見え方の違いを確認する。

課題 2

平行移動させてから回転移動させた場合と回転移動させてから平行移動させると位置が異なることを示す例を描画する。

Hint. 平行移動させてから回転移動させた直方体を赤で描画し, 回転移動させてから平行移動させた直方体を青で描画する。

課題 3 （時間に余裕のある人のために）

キー入力により，正射影変換と透視変換を変更できるようにせよ.