「情報工学」ＣＧ 課題５

三浦憲二郎

令和５年１１月２２日（水曜日）

提出締め切り 令和４年１１月２８日（火曜日）２４：００

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

提出：kawai.kota.shizuoka@gmail.com（三浦研川合航太君）

件名：情報工学課題４　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。

正射影変換と透視変換

グラフィック

- 正射影変換と透視変換
- メニューの作成

サンプルプログラム

```
#include <stdlib.h>
#include <math.h>
#include <GL/glut.h>
#include <stdio.h>

/* 視点と注視点のデータ */
static GLdouble eye[3] = { 45.0, 45.0, 25.0 };
        /*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 */

/* 視点の回転角 */
static int spin_eye = 0;
static int globalWidth;
static int globalHeight;

/* 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=1;

/*立方体の描画*/
void
drawCube() {
        float vertex[8][3]={{0.,0.,0.},{10.,0.,0.},{10.,10.,0.},{0.,10.,0.},
        {0.,0.,10.},{10.,0.,10.},{10.,10.,10.},{0.,10.,10.}};
        int i;

        glBegin(GL_LINE_LOOP);
        for(i=0;i<4;++i){
                glVertex3fv(vertex[i]);
        }
        glEnd();

        glBegin(GL_LINE_LOOP);
        for(i=0;i<4;++i){
                glVertex3fv(vertex[i+4]);
        }
        glEnd();

        glBegin(GL_LINES);
        for(i=0;i<4;++i){
                glVertex3fv(vertex[i]);
                glVertex3fv(vertex[i+4]);
        }
        glEnd();
}

/* 座標軸の描画 */
void
drawAxis() {
  /* 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();
}

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

  /* 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）*/

  /* 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*/

  glColor3f(1.,1.,1.);
  glLineWidth(2.);
  drawCube();
        /* drawCube() を呼び出す   */

  /* x, y, z軸の描画 */
  drawAxis();

  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（より手前のフラグメントを描画）*/
  glEnable ( GL_DEPTH_TEST );                        /*デプステストを有効にする*/
}

/*
 *  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 */

        globalWidth=width;
        globalHeight=height;

    glViewport (0, 0, width, height);
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity();

        if(projection==2){  /* 透視変換   */
                gluPerspective(60.0, (GLfloat) width/(GLfloat) height, 0.1, 1000.0);
        }
        else{   /* 正射影変換 */
                glOrtho(-50.,50.,-50.*(GLfloat) height/(GLfloat) width,50.*(GLfloat)
               height/(GLfloat) width,-100.,1000.);
        }

    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 main_menu(int value)                                   /* メインメニュー */
{
    switch(value){
         case 999:
                 exit(1);
                 break;
         }
         glutPostRedisplay();
}

void Rotate(int value)                                      /*  視点回転  */
{
         switch(value){
         case 1:
                 rotEyePlus();
                 break;
         case 2:
                 rotEyeMinus();
                 break;
         }
         glutPostRedisplay();
}

void Projection(int value)                                  /*  投影変換  */
{
         switch(value){
         case 3:
                 projection=1;
                 ourReshape(globalWidth,globalHeight);
                 break;
         case 4:
                 projection=2;
                 ourReshape(globalWidth,globalHeight);
                 break;
         }
         glutPostRedisplay();
}

/*  メイン */
int
main(int argc, char **argv)
{
         int submenu1,submenu2;

         glutInit(&argc, argv);
         glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
         glutInitWindowSize(400, 400);
         glutInitWindowPosition(100, 100);
         glutCreateWindow("exer5");
         ourInit();
         glutReshapeFunc(ourReshape);
         glutDisplayFunc(ourDisplay);

         submenu1=glutCreateMenu(Rotate);    /*Rotate eye のサブメニューの作成 */
         glutAddMenuEntry("positive",1);
         glutAddMenuEntry("negative",2);

         submenu2=glutCreateMenu(Projection);  /* Projection のサブメニューの作成 */
         glutAddMenuEntry("Ortho",3);
         glutAddMenuEntry("Perspective",4);

         glutCreateMenu(main_menu);
```

```
    /* メインメニュー */
    glutAddSubMenu("Rotation mode",submenu1);
    glutAddSubMenu("Projection Mode",submenu2);      /* Projection Mode の作成 */
    glutAddMenuEntry("Exit",999);      /* Exit の作成 */
  glutAttachMenu(GLUT_RIGHT_BUTTON);/* マウス右クリックでポップアップメニュー */

    glutMainLoop();
    return 0;
}
```

課題Ａ

四角錐（ピラミッド）を描画する関数 drawPyramid（）を作成し，それを drawCube()の代わりに呼び出せ。

課題Ｂ

サブメニューを追加して，立方体とピラミッドの描画を変更できるようにせよ。

Hint. １．main 関数に int submenu3; を追加する。

２．main 関数に submenu3=glutCreateMenu(Geometry); を追加する。

３．メニューエントリーを追加する。

４．Rotate()や Projection()と同様に，関数 Geometry()を追加する。

５．glutAddSubMenu("Change geometry",submenu3);

課題Ｃ　（時間に余裕のある人のために）

立方体やピラミッドに対して，回転，平行移動，およびスケーリングを組み合わせて意味のある形状や模様を作成せよ。