

```
/* ITTextCG, Chapter 7, Exercise 1, Polygonal Mesh */
#include <GL/glut.h>
#include <stdio.h>

void MakeMeshData(void);

/* 頂点ブロックのための構造体 */
struct Vertex_block
{
    GLfloat x;
    GLfloat y;
    GLfloat z;

};

/* 稜線ブロックのための構造体 */
struct Edge_block
{
    struct Vertex_block *start_point;
    struct Vertex_block *end_point;
    struct Edge_block *edge_next;

};

/* 多角形ブロックのための構造体 */
struct Polygon_block
{
    GLfloat colorR;
    GLfloat colorG;
    GLfloat colorB;
    struct Edge_block *edge_list;
    struct Polygon_block *polygon_next;
};
```

```
};
```

```
/* ポリゴンメッシュへのポインター */
```

```
static struct Polygon_block *polygonal_mesh;
```

```
static struct Vertex_block vertices[25];
```

```
static GLfloat polygon_color[25][3];
```

```
static int edge_vertex[25][4][2];
```

```
/* ポリゴンメッシュのためのデータを作成する関数 */
```

```
void MakeMeshData(void)
```

```
{
```

```
    GLfloat x1, y1, z1, pitchx1, pitchz1;
```

```
    int      i1, i2, i3, v0, v1, v2, v3;
```

```
    pitchx1 = 30.0;
```

```
    pitchz1 = 30.0;
```

```
    x1 = 0.0;
```

```
    y1 = 0.0;
```

```
    i3 = 0;
```

```
    for(i1 = 1; i1 <= 5; ++i1)
```

```
    {
```

```
        z1 = pitchz1 * 4.0;
```

```
        for(i2 = 1; i2 <= 5; ++i2)
```

```
        {
```

```
            vertices[i3].x = x1;
```

```
            vertices[i3].y = y1;
```

```
            vertices[i3].z = z1;
```

```
            z1 = z1 - pitchz1;
```

```

    i3 = i3 + 1;

} /* for i2 */

x1 = x1 + pitchx1;

} /* for i1 */

v0 = 0;
v1 = 5;
v2 = 6;
v3 = 1;

i3 = 0;
for(i1 = 1; i1 <= 4; ++i1)
{
    for(i2 = 1; i2 <= 4; ++i2)
    {
        polygon_color[i3][0] = 1.0 - 1.0/16.0 * i3;
        polygon_color[i3][1] = 1.0/16.0 * i3;
        polygon_color[i3][2] = 1.0 - 1.0/16.0 * i3;

        edge_vertex[i3][0][0] = v0;
        edge_vertex[i3][0][1] = v1;
        edge_vertex[i3][1][0] = v1;
        edge_vertex[i3][1][1] = v2;
        edge_vertex[i3][2][0] = v2;
        edge_vertex[i3][2][1] = v3;
        edge_vertex[i3][3][0] = v3;
        edge_vertex[i3][3][1] = v0;

        v0 = v0 + 1;

```

```

    v1 = v1 + 1;
    v2 = v2 + 1;
    v3 = v3 + 1;

    i3 = i3 + 1;

} /* for i2 */

v0 = v0 + 1;
v1 = v1 + 1;
v2 = v2 + 1;
v3 = v3 + 1;

} /* for i1 */

vertices[12].y = 40.0;
vertices[7].y = 20.0;

return;

}

void display(void)
{
    struct Vertex_block *ppv;
    struct Edge_block *ppe1, *ppe2;
    struct Polygon_block *ppp1, *ppp2;
    GLfloat red1, green1, blue1, x1, y1, z1;
    int i, j;

```

```

char *malloc();

glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glFrustum(-50.0, 50.0, -50.0, 50.0, 50.0, 1000.0);
gluLookAt(200.0, 80.0, 200.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();

/* ポリゴンメッシュのデータ構造の作成 */
for(i = 15; i >= 0; --i)
{
    for(j = 3; j >= 0; --j)
    {
        ppe1 = (struct Edge_block *)malloc
            (sizeof(struct Edge_block));

        ppe1->start_point =
            &vertices[(int)edge_vertex[i][j][0]];
        ppe1->end_point =
            &vertices[(int)edge_vertex[i][j][1]];

        if(j == 3)
        {
            ppe1->edge_next = NULL;
        }
        else
        {

```

```

    ppe1->edge_next = ppe2;

    } /* if */
    ppe2 = ppe1;

} /* for j */

ppp1 = (struct Polygon_block *)malloc
        (sizeof(struct Polygon_block));
ppp1->colorR = polygon_color[i][0];
ppp1->colorG = polygon_color[i][1];
ppp1->colorB = polygon_color[i][2];
ppp1->edge_list = ppe1;

if(i == 15)
{
    ppp1->polygon_next = NULL;

}
else
{
    ppp1->polygon_next = ppp2;

} /* if */
ppp2 = ppp1;

} /* for i */

/* ポリゴンメッシュへのポインター */
polygonal_mesh = ppp1;
ppp1 = polygonal_mesh;

```

```

/* ポリゴンメッシュのデータ構造の表示 */
while(1)
{
    ppe1 = ppp1->edge_list;
    red1 = ppp1->colorR;
    green1 = ppp1->colorG;
    blue1 = ppp1->colorB;
    glColor3f(red1, green1, blue1);

    glBegin(GL_LINE_LOOP);

    while(1)
    {
        ppv = ppe1->start_point;
        x1 = ppv->x;
        y1 = ppv->y;
        z1 = ppv->z;
        glVertex3f(x1, y1, z1);

        if(ppe1->edge_next ==NULL)
        {
            break;

        }
        else
        {
            ppe1 = ppe1->edge_next;

        } /* if */
    } /* while */
}

```

```

    glEnd();

    if(ppp1->polygon_next == NULL)
    {
        break;

    }
    else
    {
        ppp1 = ppp1->polygon_next;

    } /* if */

} /* while */

glFlush();

}

/* キーボードのエスケープ・キーによりプログラムを終了する関数 */
void keyboard(unsigned char key, int x, int y)
{
    switch(key) {
        case 27: exit(0); break;

    }

}

```



```
int main(int argc, char** argv)
{
    glutInitWindowSize(1000,1000);
    glutInitWindowPosition(0,0);
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA | GLUT_DEPTH);
    glutCreateWindow("PolygonalMesh");

    glEnable(GL_DEPTH_TEST);

    glClearColor(0.0, 0.0, 0.0, 0.0);

    MakeMeshData();

    glutDisplayFunc(display);
    glutKeyboardFunc(keyboard);
    glutMainLoop();
}
```