8.2z缓冲消隐.cpp

```
#define GLUT_DISABLE_ATEXIT_HACK
#include "GLUT.H"
#include <math.h>
#include <string.h>

int nearplane_width = 600; //视景体宽度
int nearplane_height = 600; //视景体高度
int nearplane_distance = 500; //视景体近平面与视点距离
int farplane_distance = nearplane_distance + 300; //视景体远平面与视点距离

float eye_x = 20;
float eye_z = 20;
float theta = 0.1;

struct my_v_homogeneous
{
	float x;
	float y;
	float z;
	float ratio;
};

//box顶点坐标
//每条边都是直线
struct my_v_homogeneous box[8];

//初始化长方形顶点坐标
void init(void)
{
	//前面四个顶点
	box[0].x = 0;
	box[0].y = 0;
	box[0].z = 0;
	box[0].ratio = 1;

	box[1].x = 80;
	box[1].y = 0;
	box[1].z = 0;
	box[1].ratio = 1;

	box[2].x = 80;
	box[2].y = 40;
	box[2].z = 0;
```

```
    box[2].ratio = 1;

    box[3].x = 0;
    box[3].y = 40;
    box[3].z = 0;
    box[3].ratio = 1;

    //后面四个顶点
    box[4].x = 0;
    box[4].y = 0;
    box[4].z = -50;
    box[4].ratio = 1;

    box[5].x = 80;
    box[5].y = 0;
    box[5].z = -50;
    box[5].ratio = 1;

    box[6].x = 80;
    box[6].y = 40;
    box[6].z = -50;
    box[6].ratio = 1;

    box[7].x = 0;
    box[7].y = 40;
    box[7].z = -50;
    box[7].ratio = 1;
}

//绘制坐标系
void draw_coordinate()
{
    glBegin(GL_LINES);
    glColor3f(1.0, 0.0, 0.0); //红色x轴
    glVertex3f(nearplane_width, 0.0, 0.0);
    glVertex3f(-nearplane_width, 0.0, 0.0);

    glColor3f(0.0, 1.0, 0.0); //绿色y轴
    glVertex3f(0.0, nearplane_height, 0.0);
    glVertex3f(0.0, -nearplane_height, 0.0);

    glColor3f(0.0, 0.0, 1.0); //蓝色z轴
    glVertex3f(0.0, 0.0, nearplane_height);
    glVertex3f(0.0, 0.0, -nearplane_height);
```

```
    glEnd();
}

//绘制内容
void display(void)
{
    glClearColor(1.f, 1.f, 1.f, 0.f);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    draw_coordinate(); //绘制坐标系

    glColor3f(0.0, 0.0, 0.0);
    glutSolidSphere(30, 50, 50); //绘制球体

    glColor3f(157.0 / 256, 195.0 / 256, 230.0 / 256);

    //绘制BOX,默认顶点之间通过直线段相连
    glBegin(GL_QUADS);
    //前面
    glVertex3i((floor)(box[0].x + 0.5), (floor)(box[0].y + 0.5), (floor)(box[0].z + 0.5));
    glVertex3i((floor)(box[1].x + 0.5), (floor)(box[1].y + 0.5), (floor)(box[1].z + 0.5));
    glVertex3i((floor)(box[2].x + 0.5), (floor)(box[2].y + 0.5), (floor)(box[2].z + 0.5));
    glVertex3i((floor)(box[3].x + 0.5), (floor)(box[3].y + 0.5), (floor)(box[3].z + 0.5));

    //后面
    glVertex3i((floor)(box[4].x + 0.5), (floor)(box[4].y + 0.5), (floor)(box[4].z + 0.5));
    glVertex3i((floor)(box[5].x + 0.5), (floor)(box[5].y + 0.5), (floor)(box[5].z + 0.5));
    glVertex3i((floor)(box[6].x + 0.5), (floor)(box[6].y + 0.5), (floor)(box[6].z + 0.5));
    glVertex3i((floor)(box[7].x + 0.5), (floor)(box[7].y + 0.5), (floor)(box[7].z + 0.5));

    //左面
    glVertex3i((floor)(box[4].x + 0.5), (floor)(box[4].y + 0.5), (floor)(box[4].z + 0.5));
    glVertex3i((floor)(box[0].x + 0.5), (floor)(box[0].y + 0.5), (floor)(box[0].z + 0.5));
    glVertex3i((floor)(box[3].x + 0.5), (floor)(box[3].y + 0.5), (floor)(box[3].z + 0.5));
    glVertex3i((floor)(box[7].x + 0.5), (floor)(box[7].y + 0.5), (floor)(box[7].z + 0.5));

    //右面
    glVertex3i((floor)(box[1].x + 0.5), (floor)(box[1].y + 0.5), (floor)(box[1].z + 0.5));
    glVertex3i((floor)(box[5].x + 0.5), (floor)(box[5].y + 0.5), (floor)(box[5].z + 0.5));
    glVertex3i((floor)(box[6].x + 0.5), (floor)(box[6].y + 0.5), (floor)(box[6].z + 0.5));
    glVertex3i((floor)(box[2].x + 0.5), (floor)(box[2].y + 0.5), (floor)(box[2].z + 0.5));

    //上面
    glVertex3i((floor)(box[3].x + 0.5), (floor)(box[3].y + 0.5), (floor)(box[3].z + 0.5));
```

```
        glVertex3i((floor)(box[2].x + 0.5), (floor)(box[2].y + 0.5), (floor)(box[2].z + 0.5));
        glVertex3i((floor)(box[6].x + 0.5), (floor)(box[6].y + 0.5), (floor)(box[6].z + 0.5));
        glVertex3i((floor)(box[7].x + 0.5), (floor)(box[7].y + 0.5), (floor)(box[7].z + 0.5));

        //下面
        glVertex3i((floor)(box[0].x + 0.5), (floor)(box[0].y + 0.5), (floor)(box[0].z + 0.5));
        glVertex3i((floor)(box[1].x + 0.5), (floor)(box[1].y + 0.5), (floor)(box[1].z + 0.5));
        glVertex3i((floor)(box[5].x + 0.5), (floor)(box[5].y + 0.5), (floor)(box[5].z + 0.5));
        glVertex3i((floor)(box[4].x + 0.5), (floor)(box[4].y + 0.5), (floor)(box[4].z + 0.5));
        glEnd();

        glutSwapBuffers();
}

//键盘交互事件
void keyboard(unsigned char key, int x, int y)
{
        switch (key)
        {
        case 'z': //打开Zbuffer深度测试
        case 'Z':
        {
                glEnable(GL_DEPTH_TEST); //打开深度缓冲测试
                glDepthFunc(GL_LEQUAL); //判断遮挡关系时，离视点近的物体遮挡离视点远的
物体
                glutPostRedisplay();
                break;
        }
        case 'c': //关闭Zbuffer深度测试
        case 'C':
        {
                glDisable(GL_DEPTH_TEST); //关闭深度缓冲测试
                glutPostRedisplay();
                break;
        }
        case 27:
                exit(0);
                break;
        }
}

//投影方式、modelview方式设置
void reshape(int w, int h)
{
```

```
    glViewport(0, 0, (GLsizei)w, (GLsizei)h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    if (w <= h)
        glOrtho(-0.5 * nearplane_width, 0.5 * nearplane_width, -0.5 * nearplane_height *
(GLfloat)nearplane_height / (GLfloat)nearplane_width, 0.5 * nearplane_height *
(GLfloat)nearplane_height / (GLfloat)nearplane_width,
            -nearplane_distance, farplane_distance); //相对于视点
    else
        glOrtho(-0.5 * nearplane_width, 0.5 * nearplane_width, -0.5 * nearplane_height *
(GLfloat)nearplane_width / (GLfloat)nearplane_height, 0.5 * nearplane_height *
(GLfloat)nearplane_width / (GLfloat)nearplane_height,
            -nearplane_distance, farplane_distance);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();

    gluLookAt(eye_x, 10, eye_z, 0, 0, 0, 0, 1, 0);
}

//鼠标交互事件
void mouse(int button, int state, int x, int y)
{
    switch (button)
    {
    case GLUT_LEFT_BUTTON:
        if (state == GLUT_DOWN)
        {
            eye_x = eye_x * cosf(-theta) + eye_z * sinf(-theta);
            eye_z = eye_z * cosf(-theta) - eye_x * sinf(-theta);
            reshape(nearplane_width, nearplane_height);
            glutPostRedisplay();
        }
        break;
    case GLUT_RIGHT_BUTTON:
        if (state == GLUT_DOWN)
        {
            eye_x = eye_x * cosf(theta) + eye_z * sinf(theta);
            eye_z = eye_z * cosf(theta) - eye_x * sinf(theta);
            reshape(nearplane_width, nearplane_height);
            glutPostRedisplay();
        }
        break;
```

```
        default:
            break;
        }
}

//主调函数
int main(int argc, char** argv)
{
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
        glutInitWindowSize(nearplane_width, nearplane_height);
        glutInitWindowPosition(100, 100);
        glutCreateWindow("z缓冲");

        init();

        glutReshapeFunc(reshape);
        glutDisplayFunc(display);
        glutKeyboardFunc(keyboard);
        glutMouseFunc(mouse);
        glutMainLoop();
        return 0;
}
```