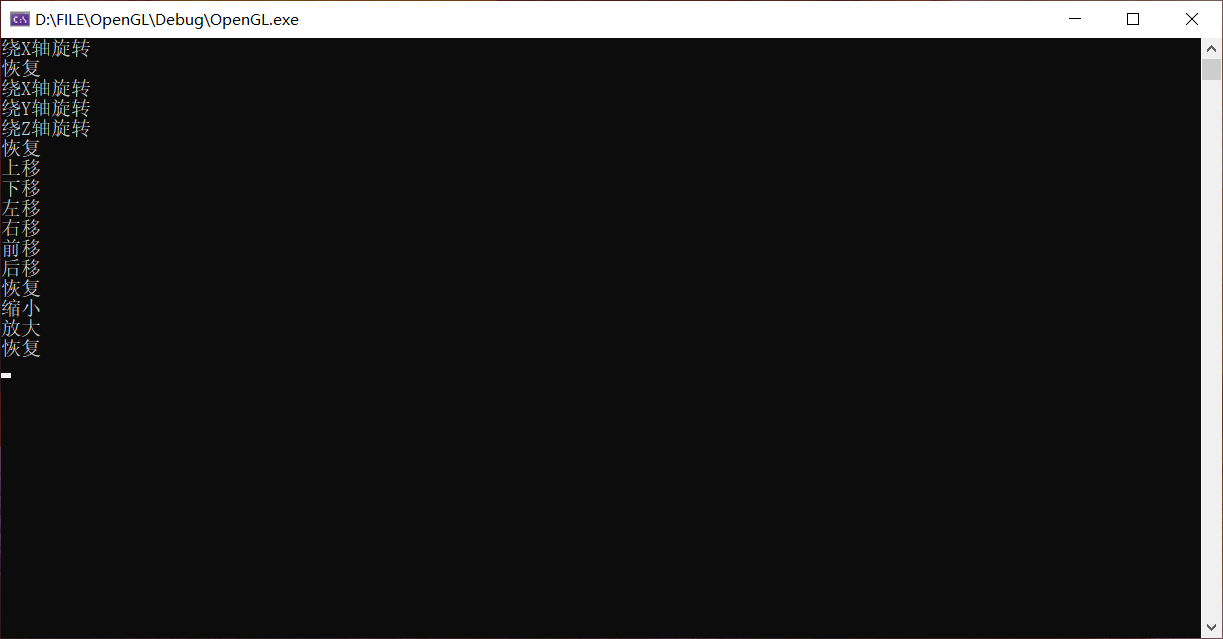
# 计算机图形学第二次作业

作业名：立方体的变换 姓名：方腾 学号：20051511

运行结果：



实验实现内容：

1. 生成三维立方体图形
2. 通过键盘控制，对生成的三维立方体图形进行旋转（X轴，Y轴，Z轴），平移，缩放变换。

实验包含头文件：

1. #include <gl/glut.h>
2. #include<stdio.h>
3. #include <math.h>

程序使用到的变量：

4. 标记键盘点击的按键，用于决定使用变换方式。

int press

5. 正方体每个顶点的坐标。该变量会随着图形变换而改变。

float vertex\_list[8][3] = {

-1.0, -1.0, -1.0,

1.0, -1.0, -1.0,

-1.0, 1.0, -1.0,

1.0, 1.0, -1.0,

-1.0, -1.0, 1.0,

1.0, -1.0, 1.0,

-1.0, 1.0, 1.0,

1.0, 1.0, 1.0

};

6. 正方体的每个顶点坐标。该变量为最初的坐标，不会发生改变。用于恢复变换。

float normal\_list[8][3] = {

-1.0, -1.0, -1.0,

1.0, -1.0, -1.0,

-1.0, 1.0, -1.0,

1.0, 1.0, -1.0,

-1.0, -1.0, 1.0,

1.0, -1.0, 1.0,

-1.0, 1.0, 1.0,

1.0, 1.0, 1.0

};

7. 正方体每个顶点的颜色。用于绘制立方体。

float color[][3] = {

1.0,1.0,0.0,

0.0, 1.0, 0.0,

0.0,1.0,1.0,

1.0,1.0,1.0,

1.0,0.0,1.0,

0.0,0.0,1.0,

0.0,0.0,0.0,

1.0,0.0,0.0

};

8. 正方体每面的个顶点的序号。

GLint index\_list[][4] = {

0, 2, 3, 1,

0, 4, 6, 2,

0, 1, 5, 4,

4, 5, 7, 6,

1, 3, 7, 5,

2, 6, 7, 3,

};

程序使用函数：

旋转变换函数：通过传入的角度，计算其弧度值。之后通过check变量决定其绕X，Y，Z轴中的哪个轴旋转。通过相对应的矩阵计算，改变立方体数组中的八个点的值。存入vertex\_list中。

void Rotate(double angle, int check)

{

double rad = angle \* 3.1415926 / 180;

float temp[8][3] = {};

for (int i = 0; i < 8; i++)

for (int j = 0; j < 3; j++)

{

temp[i][j] = vertex\_list[i][j];

}

// 绕X轴旋转

if (check == 1)

{

for (int i = 0; i < 8; i++)

{

vertex\_list[i][1] = cos(rad) \* temp[i][1] - sin(rad) \* temp[i][2];

vertex\_list[i][2] = sin(rad) \* temp[i][1] + cos(rad) \* temp[i][2];

}

}

// 绕Y轴旋转

if (check == 2)

{

for (int i = 0; i < 8; i++)

{

vertex\_list[i][0] = cos(rad) \* temp[i][0] + sin(rad) \* temp[i][2];

vertex\_list[i][2] = -sin(rad) \* temp[i][0] + cos(rad) \* temp[i][2];

}

}

// 绕Z轴旋转

if (check == 3)

{

for (int i = 0; i < 8; i++)

{

vertex\_list[i][0] = cos(rad) \* temp[i][0] - sin(rad) \* temp[i][1];

vertex\_list[i][1] = sin(rad) \* temp[i][0] + cos(rad) \* temp[i][1];

}

}

}

平移变换函数：通过check变量决定其沿X，Y，Z轴中的哪个轴平移。通过相对应的矩阵计算，对相对应的坐标进行+/- distance变换，改变立方体数组中的八个点的值。存入vertex\_list中。

void Translate(double distance, int check)

{

float temp[8][3] = {};

for (int i = 0; i < 8; i++)

for (int j = 0; j < 3; j++)

{

temp[i][j] = vertex\_list[i][j];

}

// 绕X轴平移

if (check == 1)

{

for (int i = 0; i < 8; i++)

{

vertex\_list[i][0] = temp[i][0] + distance;

}

}

// 绕Y轴平移

if (check == 2)

{

for (int i = 0; i < 8; i++)

{

vertex\_list[i][1] = temp[i][1] + distance;

}

}

// 绕Z轴平移

if (check == 3)

{

for (int i = 0; i < 8; i++)

{

vertex\_list[i][2] = temp[i][2] + distance;

}

}

}

旋转变换函数：通过size变量决定其放大、缩小的比例。通过相对应的矩阵计算，对相对应的坐标进行比例变换，改变立方体数组中的八个点的值。存入vertex\_list中。

void Scale(double size)

{

float temp[8][3] = {};

for (int i = 0; i < 8; i++)

for (int j = 0; j < 3; j++)

{

temp[i][j] = vertex\_list[i][j];

}

// 缩放

for (int i = 0; i < 8; i++)

{

vertex\_list[i][0] = temp[i][0] \* size;

vertex\_list[i][1] = temp[i][1] \* size;

vertex\_list[i][2] = temp[i][2] \* size;

}

}

绘制立方体：

确定其起始点的位置。通过内置的glTranslatef()函数进行相应的平移。

之后对每一个，每个面中的每一个点进行绘制。

1. 给每个顶点选择颜色

2. 绘制异色顶点，以便观察。要保证每次给不同面上色时，都要保证共用的点都是一个色的

3. 绘制过程中先得到顶点的序号，然后选定相应的颜色，再连线。否则最后一次的颜色会覆盖之前的颜色，就无法得到顶点异色的立方体

void Cube()

{

glBegin(GL\_QUADS);

glTranslatef(2, 2, 2); //起始点的位置

for (int i = 0; i < 6; ++i) // 有六个面，循环六次

for (int j = 0; j < 4; ++j) // 每个面有四个顶点，循环四次

{

glColor3f(color[index\_list[i][j]][0], color[index\_list[i][j]][1], color[index\_list[i][j]][2]);

glVertex3fv(vertex\_list[index\_list[i][j]]);//绘制顶点

}

glEnd();

}

显示函数：通过press的值决定调用哪一个变换函数。调用变换函数后，其坐标发生变化，之后对新形成的坐标进行立方体的绘制。

void display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

//绘制立方体

glTranslatef(-2, -2, -12);

if (press == 1)

Rotate(0.05, 1);

if (press == 2)

Rotate(0.05, 2);

if (press == 3)

Rotate(0.05, 3);

if (press == 4)

Translate(-0.001, 1);

if (press == 5)

Translate(0.001, 1);

if (press == 6)

Translate(0.001, 2);

if (press == 7)

Translate(-0.001, 2);

if (press == 8)

Translate(0.001, 3);

if (press == 9)

Translate(-0.001, 3);

if (press == 10)

Scale(1.0001);

if (press == 11)

Scale(0.9999);

if (press == 12)

for (int i = 0; i < 8; i++)

for (int j = 0; j < 3; j++)

{

vertex\_list[i][j] = normal\_list[i][j];

}

Cube();

glEnable(GL\_DEPTH\_TEST);

glutSwapBuffers();//交换缓冲区，否则无法显示正常内容

}

投影函数：通过视图调整进行投影显示，使得在二维平面上显示出三维图形的效果。

参考内容：[(54条消息) 【计算机图形学】c++ OpenGL 三维变换（包括平移、旋转和缩放）\_Arana--的博客-CSDN博客\_c++ 三维变换](https://blog.csdn.net/weixin_44387644/article/details/106163366)

void reshape(int w, int h) {

// 防止除数为0

if (h == 0)

h = 1;

float ratio = 1.0 \* w / h;

// 单位化投影矩阵。

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

// 设置视口大小为整个窗口大小

glViewport(0, 0, w, h);

// 设置正确的投影矩阵

gluPerspective(45, ratio, 1, 1000);

//下面是设置模型视图矩阵

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

//设置观测点

gluLookAt(0.0, 0.0, 5.0, 0.0, 0.0, -12.0, 0.0f, 1.0f, 0.0f);

}

键盘按键函数：根据按键来确定press的值。并在控制台输出相对应的变换。

void myKeyboard(unsigned char key, int x, int y)

{

switch (key)

{

case 'x':

case 'X':

press = 1;

printf("绕X轴旋转\n");

break;

case 'y':

case 'Y':

press = 2;

printf("绕Y轴旋转\n");

break;

case 'z':

case 'Z':

press = 3;

printf("绕Z轴旋转\n");

break;

case 'l':

case 'L':

press = 4;

printf("左移\n");

break;

case 'r':

case 'R':

press = 5;

printf("右移\n");

break;

case 'u':

case 'U':

press = 6;

printf("上移\n");

break;

case 'd':

case 'D':

press = 7;

printf("下移\n");

break;

case 'f':

case 'F':

press = 8;

printf("前移\n");

break;

case 'b':

case 'B':

press = 9;

printf("后移\n");

break;

case 'a':

case 'A':

press = 10;

printf("放大\n");

break;

case 's':

case 'S':

press = 11;

printf("缩小\n");

break;

case 'm':

case 'M':

press = 12;

printf("恢复\n");

break;

default:

break;

}

glutPostRedisplay();

}

主函数：

int main(int argc, char\* argv[])

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DEPTH | GLUT\_DOUBLE | GLUT\_RGBA);

glutInitWindowPosition(100, 100);

glutInitWindowSize(600, 600);

glutCreateWindow("三维图形变换");

glutDisplayFunc(display);

glutKeyboardFunc(myKeyboard);

glutReshapeFunc(reshape); //指定窗口形状变化时的回调函数

glutIdleFunc(display); //指定程序空闲时调用函数

glutMainLoop();

return 0;

}