Bài tập thực hành số 3

I. Phép biến đổi điểm nhìn và biến đổi mô hình

Lệnh biến đổi điểm nhìn **gluLookAt**(*eyex*, *eyey*, *eyez*, *centerx*, *centery*, *centerz*, *upx*, *upy*, *upz*)
Chạy chương trình cube.c
1.1 Thay lệnh gluLookAt (0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

- Bởi lệnh gluLookAt (0.0, 0.0, 5.0, 0.0, 0.0, -100.0, 0.0, 1.0, 0.0); kết quả hình ảnh không bị thay đổi, tại sao? Tại vì điểm nhìn của camera vẫn nằm trên vật nên vẫn nhìn thấy

- Bởi lệnh gluLookAt (0.0, 0.0, 5.0, 0.0, 0.0, 100.0, 0.0, 1.0, 0.0); kết quả không nhìn thấy hình ảnh, tại sao? Tại vị trí điểm nhìn của camera nằm dưới vật nên không nhìn th

1.2 Khôi phục lại chương trình gốc cube.c, thay lệnh glutWireCube() bởi lệnh glutWireTeapot(1.5); chạy lại chương trình

Thay lệnh gluLookAt() bởi lần lượt các lệnh sau và giải thích kết quả thu được:

gluLookAt (10.0, 0.0, 10.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0); Vẫn nhìn được tại vị trí 10 0 10
gluLookAt (10.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -1.0, 0.0); Vẫn nhìn được ấm trà sẽ bị lật ngược
gluLookAt (0.0, -5.0, 0.0, 0.0, 0.0, 0.0, 3.0, 0.0, -3.0); Nhìn được vật từ dưới lên
gluLookAt (0.0, 0.0, 5.0, -10.0, 0.0, 0.0, 0.0, 1.0, 0.0); ko nhìn thấy do 0 -10 0 ko nằm (.) hướng nhìn

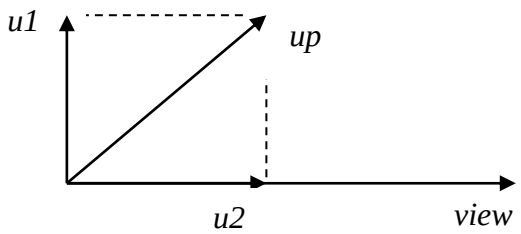
1.3 Khôi phục lại chương trình gốc cube.c, thay lệnh gluLookAt() bởi lần lượt các lệnh sau

gluLookAt (0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, -10.0);
gluLookAt (0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 12.0, 30.0);
gluLookAt (0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 3.7, 11.6);

Ta thấy rằng các lệnh trên chỉ khác nhau ở hướng đỉnh của camera ta sẽ gọi đó là véc tơ *up*(*upx*, *upy*, *upz*). Trong cả ba câu lệnh trên hướng đỉnh của camera không vuông góc với hướng nhìn. Kết quả thu được của các câu lệnh đó là như nhau, tại sao vậy?

Ta quay lại lệnh: **gluLookAt**(*eyex*, *eyey*, *eyez*, *centerx*, *centery*, *centerz*, *upx*, *upy*, *upz*)

Bình thường ta luôn chọn hướng đỉnh vuông góc với hướng nhìn của camera ta gọi đó là véc tơ *view(centerx-eyex*, *centery-eyey*, *centerz-eyez).* Trong trường hợp nếu véc tơ *up* không vuông góc và không cộng tuyến với véc tơ *view* ta luôn có cách phân tích véc tơ *up* theo cách duy nhất như sau: *up= u1+ u2* (trong đó *u1* là véc tơ vuông góc với *view* còn *u2* là một véc tơ cộng tuyến với *view*). Khi đó OpenGL sẽ chọn véc tơ *u1* là hướng đỉnh của camera thay cho véc tơ *up*.



Ví dụ trong lệnh gluLookAt (0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, -10.0); ta có
up(0.0, 1.0, -10.0)=*u1*(0.0, 1.0, 0.0)+ 2**view*(0.0, 0.0, -5.0)

Do đó OpenGL sẽ chọn hướng đỉnh của camera là *u1*(0.0, 1.0, 0.0), đó là lí do vì sao hình ảnh hiển thị không bị thay đổi.

2. Các lệnh
Khôi phục lại chương trình gốc cube.c

2.1 Chèn vào phía trước của lệnh glutWireCube() lần lượt các câu lệnh sau:

glTranslatef(2.0, 0.0, 0.0);

glTranslatef(2.0, 0.0, -3.0);
glRotatef(30, 0.0, 0.0, 1.0);
glRotatef(-60, 1.0, 0.0, 1.0);
glScalef(1.0, 1.0, 1.0);
glScalef(2.0, 1.5, 3.0);

2.2 Khôi phục lại chương trình gốc cube.c và chèn vào phía trước của lệnh glutWireCube() lần lượt các cụm câu lệnh sau:

Cụm lệnh 1:

glTranslatef(2.0, 0.0, 0.0);
glRotatef(45, 0.0, 0.0, 1.0);

Cụm lệnh 2:

glRotatef(45, 0.0, 0.0, 1.0);
glTranslatef(2.0, 0.0, 0.0);

Kết quả thu được khác nhau, tại sao? Thứ tự thực hiện khác nhau lên kết quả khác nhau

II. Chương trình

Một chương trình hoạt cảnh như sau:

```
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>
#include <stdlib.h>
static GLfloat spin = 0.0;
void init(void)
{
glClearColor (0.0, 0.0, 0.0, 0.0);
glShadeModel (GL_FLAT);
}
void display(void)
{
glClear(GL_COLOR_BUFFER_BIT);
glPushMatrix();
glRotatef(spin, 0.0, 0.0, 1.0);
glColor3f(1.0, 1.0, 1.0);
glRectf(-25.0, -25.0, 25.0, 25.0);
glPopMatrix();
glutSwapBuffers();
}
void spinDisplay(void)
{
spin = spin + 0.02;
if (spin > 360.0)
spin = spin - 360.0;
glutPostRedisplay();
}
void reshape(int w, int h)
{
glViewport (0, 0, (GLsizei) w, (GLsizei) h);
```

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(-50.0, 50.0, -50.0, 50.0, -1.0, 1.0);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
}
void mouse(int button, int state, int x, int y)
{
switch (button) {
case GLUT_LEFT_BUTTON:
if (state == GLUT_DOWN)
glutIdleFunc(spinDisplay);
break;
case GLUT_MIDDLE_BUTTON:
if (state == GLUT_DOWN)
glutIdleFunc(NULL);
break;
default:
break;
}}
/* Request double buffer display mode.
 Register mouse input callback functions */
int main(int argc, char** argv){
glutInit(&argc, argv);
glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB);
glutInitWindowSize (250, 250);
glutInitWindowPosition (100, 100);
glutCreateWindow (argv[0]);
init ();
glutDisplayFunc(display);
glutReshapeFunc(reshape);
glutMouseFunc(mouse);
glutMainLoop();
return 0;}
```

Tương tác với các sự kiện đầu vào:

Chuột:

```
void mouse(int button, int state, int x, int y)
{
switch (button) {
case GLUT_LEFT_BUTTON:
{if (state == GLUT_DOWN)
spin = spin + 2;
if (spin > 360.0)
spin = spin - 360.0;
```

```
glutPostRedisplay();
break;}
```

Trong hàm main(), sử dụng lời gọi: `glutMouseFunc(mouse);`

Bàn phím

```
void keyboard (unsigned char key, int x, int y)
{
switch (key) {
case 's': /* s key rotates */
spin = spin + 2;
if (spin > 360.0)
spin = spin - 360.0;
glutPostRedisplay();
break;}
```

Bài 1. Vẽ 1 ấm trà, khi nhấn trái chuột, ấm trà quay quanh trục x,

Bài 2. Vẽ một ngôi sao 5 cánh,

a, Khi nhấn trái chuột ngôi sao quay quanh tâm của nó.

b, Khi nhấn phím 'a' ngôi sao quay quanh một đỉnh cánh của nó.

Bài 3. Vẽ hai khối cầu cạnh nhau

a, Nhấn phím 'a' khối cầu 1 quay quanh trục x,

b, Nhấn trái chuột, khối cầu 1 quay quanh khối cầu 2.