Computer graphics practical 6

code:

```
#include <iostream>
#include <stdlib.h>

#ifdef __APPLE__
#include <OpenGL/OpenGL.h>
#include <GLUT/glut.h>
#else
#include <GL/glut.h>
#endif

using namespace std;

float ballX = -0.8f;
float ballY = -0.3f;
float ballZ = -1.2f;
float colR = 3.0;
float colG = 1.5;
float colB = 1.0;
float bgColR = 0.0;
float bgColG = 0.0;
float bgColB = 0.0;
static int flag = 1;

void drawBall() {
   glColor3f(colR, colG, colB);
   glTranslatef(ballX, ballY, ballZ);
   glutSolidSphere(0.05, 30, 30);
}

void drawAv() {
   glBegin(GL_POLYGON);
   glColor3f(1.0, 1.0, 1.0);
   glVertex3f(-0.9, -0.7, -1.0);
   glVertex3f(-0.5, -0.1, -1.0);
   glVertex3f(-0.2, -1.0, -1.0);
   glVertex3f(0.5, 0.0, -1.0);
   glVertex3f(0.6, -0.2, -1.0);
   glVertex3f(0.9, -0.7, -1.0);
   glEnd();
}

void drawClouds() {

}

void keyPress(int key, int x, int y) {
   switch (key) {
```

```
        case GLUT_KEY_RIGHT:
            ballX -= 0.05f;
            break;
        case GLUT_KEY_LEFT:
            ballX += 0.05f;
            break;
        case GLUT_KEY_UP:
            ballY += 0.05f;
            break;
        case GLUT_KEY_DOWN:
            ballY -= 0.05f;
            break;
    }
    glutPostRedisplay();
}


void initRendering() {
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_COLOR_MATERIAL);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glEnable(GL_LIGHT1);
    glEnable(GL_NORMALIZE);
    glShadeModel(GL_SMOOTH);
}

void handleResize(int w, int h) {
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0, (double)w / (double)h, 1.0, 200.0);
}

void drawScene() {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glClearColor(bgColR, bgColG, bgColB, 0.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    GLfloat ambientColor[] = {0.2f, 0.2f, 0.2f, 1.0f};
    glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambientColor);
    GLfloat lightColor0[] = {0.5f, 0.5f, 0.5f, 1.0f};
    GLfloat lightPos0[] = {4.0f, 0.0f, 8.0f, 1.0f};
    glLightfv(GL_LIGHT0, GL_DIFFUSE, lightColor0);
    glLightfv(GL_LIGHT0, GL_POSITION, lightPos0);
    GLfloat lightColor1[] = {0.5f, 0.2f, 0.2f, 1.0f};
    GLfloat lightPos1[] = {-1.0f, 0.5f, 0.5f, 0.0f};
    glLightfv(GL_LIGHT1, GL_DIFFUSE, lightColor1);
    glLightfv(GL_LIGHT1, GL_POSITION, lightPos1);

    // Push and pop the matrix around each drawing operation
    glPushMatrix();
```

```
    drawBall();
    glPopMatrix();

    glPushMatrix();
    drawAv();
    glPopMatrix();

    glPushMatrix();
    drawClouds();
    glPopMatrix();

    glutSwapBuffers();
}

void update(int value) {
    if (ballX > 0.9f) {
        ballX = -0.8f;
        ballY = -0.3f;
        flag = 1;
        colR = 2.0;
        colG = 1.5;
        colB = 1.0;
        bgColB = 0.0;
    }
    if (flag) {
        ballX += 0.001f;
        ballY += 0.0007f;
        colR -= 0.001;
        colB += 0.005;
        bgColB += 0.001;
        if (ballX > 0.01) {
            flag = 0;
        }
    }
    if (!flag) {
        ballX += 0.001f;
        ballY -= 0.0007f;
        colR += 0.001;
        colB -= 0.01;
        bgColB -= 0.001;
        if (ballX < -0.3) {
            flag = 1;
        }
    }
    glutPostRedisplay();
}

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_DEPTH);
    glutInitWindowSize(400, 400);
    glutCreateWindow("SUN");
```

```
    initRendering();
    glutDisplayFunc(drawScene);
    glutFullScreen();
    glutSpecialFunc(keyPress);
    glutReshapeFunc(handleResize);
    glutTimerFunc(25, update, 0);
    glutMainLoop();
    return 0;
}
```

output: