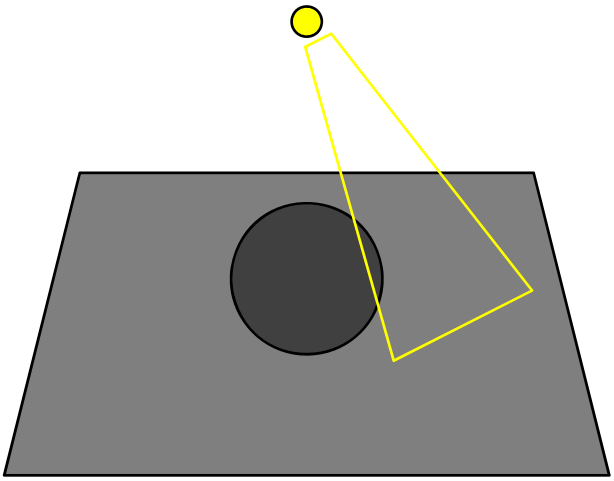
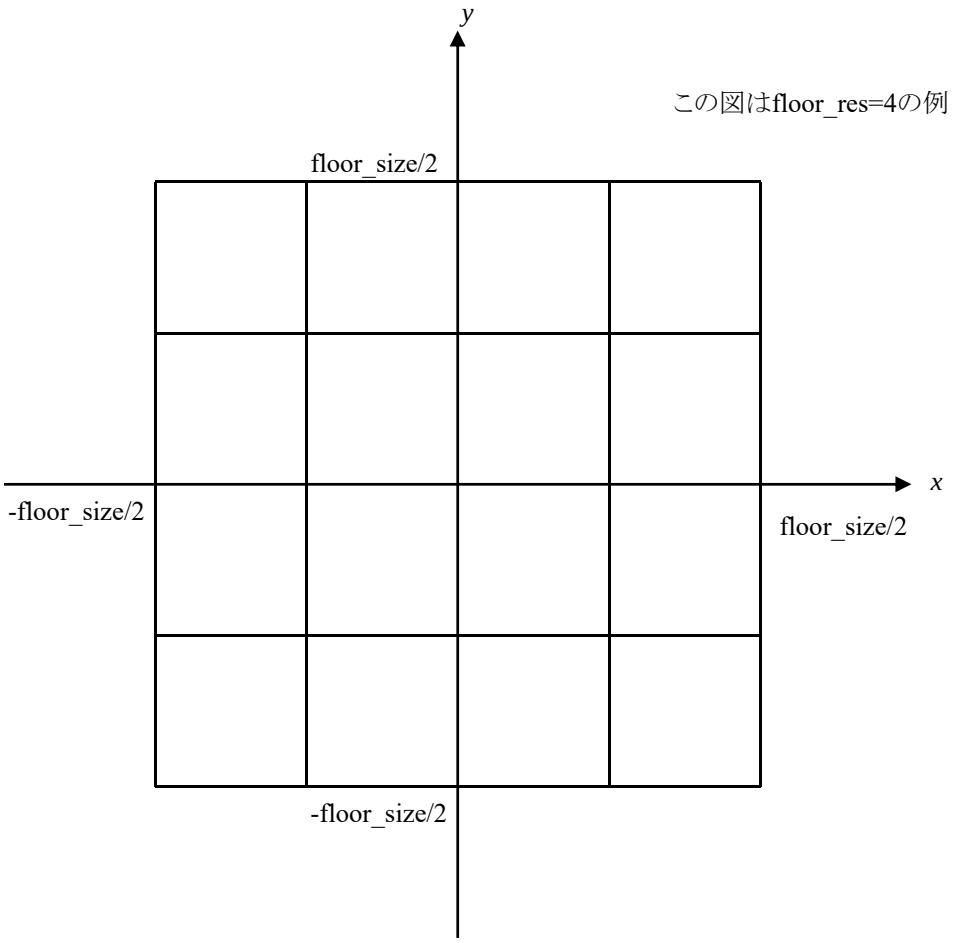
スポットライト

- スポットライトで平面と球を照らすプログラム





サンプルソースコード

```
#include <stdlib.h>
#include <stdio.h>
#include <GL/glut.h>

int winw, winh;
const int delay = 50;
float light0_dir[3] = { 0.0f, 0.0f, -1.0f };
float light0_dir_vel[3] = { 0.03f, 0.05f, 0.0f };
const double floor_size = 10.0;

void myGround()
{
    const int floor_res = 100;
    int x, y;

    glNormal3d(0.0, 0.0, 1.0);
    glBegin(GL_QUADS);
    for (y = -floor_res / 2; y < floor_res / 2; y++) {
        for (x = -floor_res / 2; x < floor_res / 2; x++) {
            glVertex3d((double)x / (double)floor_res * floor_size, (double)y / (double)floor_res * floor_size,
0.0);
            glVertex3d((double)(x + 1) / (double)floor_res * floor_size, (double)y / (double)floor_res *
floor_size, 0.0);
            glVertex3d((double)(x + 1) / (double)floor_res * floor_size, (double)(y + 1) / (double)floor_res *
floor_size, 0.0);
            glVertex3d((double)x / (double)floor_res * floor_size, (double)(y + 1) / (double)floor_res *
floor_size, 0.0);
        }
    }
    glEnd();
}

void myDisplay()
{
    const float scene_amb[4] = { 0.5f, 0.5f, 0.5f, 1.0f };
    const float light0_diff[4] = { 10.0f, 10.0f, 10.0f, 1.0f };
    const float light0_pos[4] = { 0.0f, 0.0f, 3.0f, 1.0f };
    const float light0_exp = 100.0f;
    const float light0_cut = 75.0f;
    const float light0_atten_const = 0.0f;
    const float light0_atten_lin = 0.0f;
    const float light0_atten_quad = 1.0f;

    glClearColor(0.0, 0.0, 0.0, 1.0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glShadeModel(GL_SMOOTH);
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(60.0, (double)winw / (double)winh, 0.1, 100.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(0.0, -3.0, 3.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0);

    glLightModelfv(GL_LIGHT_MODEL_AMBIENT, scene_amb);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, light0_diff);
    glLightfv(GL_LIGHT0, GL_POSITION, light0_pos);
    glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, light0_dir);
    glLightf(GL_LIGHT0, GL_SPOT_EXPONENT, light0_exp);
    glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, light0_cut);
    glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, light0_atten_const);
    glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, light0_atten_lin);
    glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, light0_atten_quad);

    myGround();
    glutSolidSphere(1.0, 50, 50);

    glutSwapBuffers();
}
```

サンプルソースコード

```
void myTimer(int value)
{
    int i;
    if (value == 1) {
        for (i = 0; i < 3; i++) {
            light0_dir[i] += light0_dir_vel[i];
        }
        if (light0_dir[0] < -floor_size * 0.1) {
            light0_dir[0] = -floor_size * 0.1;
            light0_dir_vel[0] = -light0_dir_vel[0];
        }
        if (light0_dir[0] > floor_size * 0.1) {
            light0_dir[0] = floor_size * 0.1;
            light0_dir_vel[0] = -light0_dir_vel[0];
        }
        if (light0_dir[1] < -floor_size * 0.1) {
            light0_dir[1] = -floor_size * 0.1;
            light0_dir_vel[1] = -light0_dir_vel[1];
        }
        if (light0_dir[1] > floor_size * 0.1) {
            light0_dir[1] = floor_size * 0.1;
            light0_dir_vel[1] = -light0_dir_vel[1];
        }
        glutPostRedisplay();
        glutTimerFunc(delay, myTimer, 1);
    }
}

void myKeyboard(unsigned char key, int x, int y)
{
    if (key == 0x1B) exit(0);
}

void myReshape(int width, int height)
{
    winw = width;
    winh = height;
    glViewport(0, 0, winw, winh);
}

int main(int argc, char* argv[])
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
    glutInitWindowSize(640, 480);
    glutInitWindowPosition(0, 0);
    glutCreateWindow(argv[0]);
    glutKeyboardFunc(myKeyboard);
    glutTimerFunc(delay, myTimer, 1);
    glutReshapeFunc(myReshape);
    glutDisplayFunc(myDisplay);
    glutMainLoop();
    return 0;
}
```