

ГУАП

КАФЕДРА № 43

ОТЧЕТ  
ЗАЩИЩЕН С ОЦЕНКОЙ  
ПРЕПОДАВАТЕЛЬ

\_\_\_\_\_  
должность, уч. степень, звание

\_\_\_\_\_  
подпись, дата

Щёкин С. В.  
\_\_\_\_\_  
инициалы, фамилия

## ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ

Вывод трехмерных объектов с динамическим расчетом  
проекционных теней.

по курсу: Компьютерная графика

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №

4136

\_\_\_\_\_  
подпись, дата

Бобрович Н. С.  
\_\_\_\_\_  
инициалы, фамилия

Санкт-Петербург 2022

**Задание:**

Вывести трехмерную сцену с движущимся объектом, который отбрасывает тень на другой объект (напр. плоскость). Тень должна перемещаться вместе с движением объекта исходя из взаимного положения источника света, объекта, который отбрасывает тень и объектов, на которые тень проецируется.

**Листинг программы:**

```

1  #include <iostream>
2  #include <glut.h>
3  #ifndef CALLBACK
4  #define CALLBACK
5  #endif
6  double rotate_y = 0;
7  double rotate_x = 0;
8  float xx = 1.0;
9  float yy = 1.0;
10 GLint Torus;
11 void display();
12 class Lab
13 {
14 public:
15     float x, y, z, w; // Вершинные x, y, и z координаты.
16     Lab(float X, float Y, float Z, float W); // перегруженный конструктор
17     float DotProduct4(Lab v1);
18 };
19 Lab::Lab(float X, float Y, float Z, float W)
20 {
21     // Инициализирует переменные значениями X, Y, Z и W.
22     x = X;
23     y = Y;
24     z = Z;
25     w = W;
26 }
27 float Lab::DotProduct4(Lab v1)
28 {
29     // Получаем dot product из v1 и этого объекта и возвращаем результат
30     return x * v1.x + y * v1.y + z * v1.z + w * v1.w;
31 }
32
33 class Matrix
34 {
35 public:
36     float matrix[16]; // 4x4 матрица в одномерном массиве.
37     Matrix(); // Конструктор.
38     bool CreateShadowMatrix(Lab planeNormal, Lab lightPos);
39     void Clear();
40 };
41 Matrix::Matrix()
42 {
43     // Инициализируем все переменные
44     Clear();
45 }
46 void Matrix::Clear()
47 {
48     matrix[0] = 1.0f; matrix[1] = 0.0f; matrix[2] = 0.0f; matrix[3] = 0.0f;

```

```

49     matrix[4] = 0.0f; matrix[5] = 1.0f; matrix[6] = 0.0f; matrix[7] = 0
50     matrix[8] = 0.0f; matrix[9] = 0.0f; matrix[10] = 1.0f; matrix[11] =
51     matrix[12] = 0.0f; matrix[13] = 0.0f; matrix[14] = 0.0f; matrix[15]
52 }
53 bool Matrix::CreateShadowMatrix(Lab planeNormal, Lab lightPos)
54 {
55     Clear();
56     // Чтобы создать матрицу теней, сначала нужно получить скалярное пр
57     // поверхности и позиции света.
58     float dotProduct = planeNormal.DotProduct4(lightPos);
59     // Создаем матрицу теней путем добавления наших значений...
60     matrix[0] = dotProduct - lightPos.x * planeNormal.x;
61     matrix[4] = 0.0f - lightPos.x * planeNormal.y;
62     matrix[8] = 0.0f - lightPos.x * planeNormal.z;
63     matrix[12] = 0.0f - lightPos.x * planeNormal.w;
64     matrix[1] = 0.0f - lightPos.y * planeNormal.x;
65     matrix[5] = dotProduct - lightPos.y * planeNormal.y;
66     matrix[9] = 0.0f - lightPos.y * planeNormal.z;
67     matrix[13] = 0.0f - lightPos.y * planeNormal.w;
68     matrix[2] = 0.0f - lightPos.z * planeNormal.x;
69     matrix[6] = 0.0f - lightPos.z * planeNormal.y;
70     matrix[10] = dotProduct - lightPos.z * planeNormal.z;
71     matrix[14] = 0.0f - lightPos.z * planeNormal.w;
72     matrix[3] = 0.0f - lightPos.w * planeNormal.x;
73     matrix[7] = 0.0f - lightPos.w * planeNormal.y;
74     matrix[11] = 0.0f - lightPos.w * planeNormal.z;
75     matrix[15] = dotProduct - lightPos.w * planeNormal.w;
76     return true;
77 }
78 void init(void)
79 {
80     // Очищаем буфер цвета и глубины
81     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
82     //Включаем нужные механизмы
83     glEnable(GL_LIGHTING);
84     glEnable(GL_LIGHT0);
85     glEnable(GL_DEPTH_TEST);
86 }
87 //Отображение
88 void display(void)
89 {
90     Matrix ShadowMatrix;
91     GLfloat light_position[] = { xx, yy, 3.0 , 1.0 };
92     GLfloat white_light[] = { 1.0,1.0,1.0 ,1.0 };
93     glLightfv(GL_LIGHT0, GL_POSITION, light_position);
94     glLightfv(GL_LIGHT0, GL_DIFFUSE, white_light);
95     glLightfv(GL_LIGHT0, GL_SPECULAR, white_light);
96     Lab lightPos(xx, yy, 3.0f, 1.0f);

```



```

97     Lab planeNormal(0.0f, 0.0f, 1.0f, 0.0f);
98     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
99     glBegin(GL_POLYGON);
100    GLfloat mat_specular01[] = { 1.0, 1.0, 1.0, 1.0 };
101    GLfloat mat_ambient[] = { 0.9, 0.9, 0.9, 1.0 };
102    GLfloat mat_diffuse[] = { 0.8, 0.8, 0.8, 1.0 };
103    GLfloat mat_emission1[] = { 0.0, 0.0, 0.0, 1.0 };
104    GLfloat mat_shininess[] = { 100.0 };
105    glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, mat_specular01);
106    glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, mat_ambient);
107    glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, mat_diffuse);
108    glVertex3f(-2.0, -2.0, -1.0);
109    glVertex3f(2.0, -2.0, -1.0);
110    glVertex3f(2.0, 2.0, -1.0);
111    glVertex3f(-2.0, 2.0, -1.0);
112    glEnd();
113    // Создадим матрицу тени на основе нормали поверхности и позиции и
114    ShadowMatrix.CreateShadowMatrix(planeNormal, lightPos);
115    glEnable(GL_BLEND);
116    glDisable(GL_DEPTH_TEST);
117    glDisable(GL_LIGHTING);
118    glPushMatrix();
119    glMultMatrixf(ShadowMatrix.matrix);
120    glColor3f(0.0f, 0.0f, 0.0f);
121    //glLoadIdentity();
122    glutSolidTeapot(1);/////
123    glPopMatrix();
124    glEnable(GL_LIGHTING);
125    glEnable(GL_DEPTH_TEST);
126    glDisable(GL_BLEND);
127    glPushMatrix();
128    glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, mat_specular01);
129    glMaterialfv(GL_FRONT_AND_BACK, GL_SHININESS, mat_shininess);
130    glutSolidTeapot(1);
131    glPopMatrix();
132    glFlush();
133    glutSwapBuffers();
134 }
135 //Изменение размеров окна
136 void reshape(int w, int h)
137 {
138     glViewport(0, 0, (GLsizei)w, (GLsizei)h);
139     glMatrixMode(GL_PROJECTION);
140     glLoadIdentity();
141     if (w <= h)
142         glOrtho(-1.5, 1.5, -1.5 * (GLfloat)h / (GLfloat)w, 1.5 * (GLfloat)h / (GLfloat)w, -10.0, 10.0);
143     else
144         glOrtho(-1.5, 1.5, -10.0, 10.0, -1.5 * (GLfloat)w / (GLfloat)h, 1.5 * (GLfloat)w / (GLfloat)h);

```

```

145         else
146             glOrtho(-1.5 * (GLfloat)w / (GLfloat)h, 1.5 * (GLfloat)w / (GL-
147                 -
148                 10.0, 10.0);
149         glMatrixMode(GL_MODELVIEW);
150         glLoadIdentity();
151     }
152     void specialKeys(int key, int x, int y)
153     {
154         if (key == GLUT_KEY_RIGHT)
155             xx += 0.01;
156         else if (key == GLUT_KEY_LEFT)
157             xx -= 0.01;
158         else if (key == GLUT_KEY_UP)
159             yy += 0.01;
160         else if (key == GLUT_KEY_DOWN)
161             yy -= 0.01;
162         glutPostRedisplay();
163     }
164     int main(int argc, char** argv)
165     {
166         glutInit(&argc, argv);
167         glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
168         glutInitWindowSize(700, 500);
169         glutInitWindowPosition(100, 100);
170         glutCreateWindow("KG7");
171         init();
172         glutDisplayFunc(display);
173         glutSpecialFunc(specialKeys);
174         glutReshapeFunc(reshape);
175         glutMainLoop();
176         return 0;
177     }

```

Результат работы:









**Выводы:**

В результате выполнения работы были получены навыки работы с выводом трехмерных объектов с динамическим расчетом проекционных теней.