МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА» (БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

РГЗ

по дисциплине: Компьютерная графика

тема: «Кубик Рубика на OpenGL»

Выполнил: ст. группы ПВ-223

Дмитриев Андрей Александрович

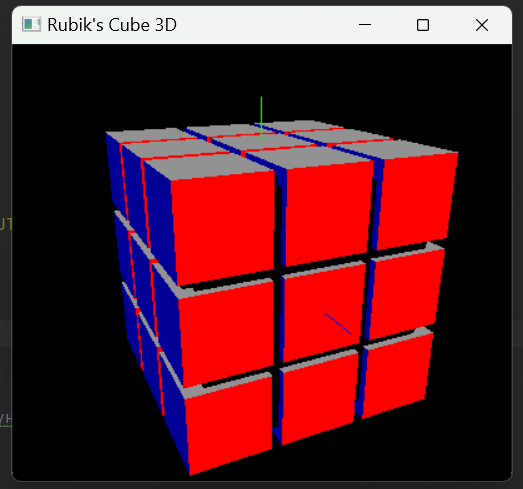
Проверил:

Осипов Олег Васильевич

Белгород 2024 г.

**Цель работы:** Написать программу, которая будет отрисовывать кубик Рубика, с функцией вращения граней.

Предпросмотр:

****

Github: <https://github.com/AnDreV133/RubiksCube>

Ход работы:

**Отрисовка кубика Рубика**

Кубик Рубика состоит из однообразных маленьких кубиков с окрашенными гранями:

// Отрисовка кубика  
void draw\_cube(int x, int y, int z) {  
 vector<cube\_rotate> lrot = cube\_rotations[x][y][z];  
  
 glPushMatrix();  
  
 // Установка позиции кубика  
 glTranslatef(  
 (x - 1) \* (cube\_size + gap),  
 (y - 1) \* (cube\_size + gap),  
 (z - 1) \* (cube\_size + gap)  
 );  
  
 // Поворот кубика по его истории поворотов  
 for(int i = lrot.size() - 1; i >= 0; --i)  
 glRotatef(lrot[i].angle, lrot[i].x, lrot[i].y, lrot[i].z);  
  
 glColor3f(1.0f, 0.0f, 0.0f);  
 glBegin(GL\_QUADS); // front  
 glNormal3f(0.0, 0.0, 1.0);   
 glVertex3f(cube\_size / 2, cube\_size / 2, cube\_size / 2);  
 glVertex3f(-cube\_size / 2, cube\_size / 2, cube\_size / 2);  
 glVertex3f(-cube\_size / 2, -cube\_size / 2, cube\_size / 2);  
 glVertex3f(cube\_size / 2, -cube\_size / 2, cube\_size / 2);  
 glEnd();  
  
 glColor3f(1.0f, 0.5f, 0.0f);  
 glBegin(GL\_QUADS); // back  
 glNormal3f(0.0, 0.0, -1.0);   
 glVertex3f(cube\_size / 2, cube\_size / 2, -cube\_size / 2);  
 glVertex3f(cube\_size / 2, -cube\_size / 2, -cube\_size / 2);  
 glVertex3f(-cube\_size / 2, -cube\_size / 2, -cube\_size / 2);  
 glVertex3f(-cube\_size / 2, cube\_size / 2, -cube\_size / 2);  
 glEnd();  
  
 glColor3f(0.0f, 0.0f, 1.0f);  
 glBegin(GL\_QUADS); // left  
 glNormal3f(-1.0, 0.0, 0.0);   
 glVertex3f(-cube\_size / 2, cube\_size / 2, cube\_size / 2);  
 glVertex3f(-cube\_size / 2, cube\_size / 2, -cube\_size / 2);  
 glVertex3f(-cube\_size / 2, -cube\_size / 2, -cube\_size / 2);  
 glVertex3f(-cube\_size / 2, -cube\_size / 2, cube\_size / 2);  
 glEnd();  
  
 glColor3f(0.0f, 1.0f, 0.0f);  
 glBegin(GL\_QUADS); // right  
 glNormal3f(1.0, 0.0, 0.0);   
 glVertex3f(cube\_size / 2, cube\_size / 2, cube\_size / 2);  
 glVertex3f(cube\_size / 2, -cube\_size / 2, cube\_size / 2);  
 glVertex3f(cube\_size / 2, -cube\_size / 2, -cube\_size / 2);  
 glVertex3f(cube\_size / 2, cube\_size / 2, -cube\_size / 2);  
 glEnd();  
  
 glColor3f(1.0f, 1.0f, 1.0f);  
 glBegin(GL\_QUADS); // top  
 glNormal3f(0.0, 1.0, 0.0);   
 glVertex3f(-cube\_size / 2, cube\_size / 2, -cube\_size / 2);  
 glVertex3f(-cube\_size / 2, cube\_size / 2, cube\_size / 2);  
 glVertex3f(cube\_size / 2, cube\_size / 2, cube\_size / 2);  
 glVertex3f(cube\_size / 2, cube\_size / 2, -cube\_size / 2);  
 glEnd();  
  
 glColor3f(1.0f, 1.0f, 0.0f);  
 glBegin(GL\_QUADS); // bottom  
 glNormal3f(0.0, -1.0, 0.0);   
 glVertex3f(-cube\_size / 2, -cube\_size / 2, -cube\_size / 2);  
 glVertex3f(cube\_size / 2, -cube\_size / 2, -cube\_size / 2);  
 glVertex3f(cube\_size / 2, -cube\_size / 2, cube\_size / 2);  
 glVertex3f(-cube\_size / 2, -cube\_size / 2, cube\_size / 2);  
 glEnd();  
  
 glPopMatrix();  
}

Функция библиотеки – glTranslatef – устанавливает кубик на своё место; glRotatef – последовательно применяет повороты кубика (см. Алгоритм поворота граней).

Маленькие кубики расставляются с помощью вложенных циклов:

// Отрисовка кубика Рубика  
void draw\_rubiks\_cube() {  
 // Отрисовка кубиков  
 for (int i = 0; i < 3; ++i)  
 for (int j = 0; j < 3; ++j)  
 for (int k = 0; k < 3; ++k)  
 draw\_cube(i, j, k);  
  
 // Отрисовка цилиндров внутри кубика Рубика  
 int cylinder\_half\_size = cube\_size \* 1.4 + gap;  
 int cylinder\_size = cylinder\_half\_size \* 2;  
 int cylinder\_radius = cylinder\_half\_size;  
  
 // Отрисовка цилиндров  
 glPushMatrix();  
  
 // Черный цвет для цилиндров  
 glColor3f(0.0f, 0.0f, 0.0f);  
  
 // Цилиндр вдоль оси X  
 glPushMatrix();  
 glTranslatef(-cylinder\_half\_size, 0, 0);  
 glRotatef(90.0f, 0.0f, 1.0f, 0.0f); // Поворот вокруг оси Y, чтобы цилиндр был вдоль оси X  
 glutSolidCylinder(cylinder\_radius, cylinder\_size, 16, 1); // Высота 3.0f, чтобы проходить через весь куб  
 glPopMatrix();  
  
 // Цилиндр вдоль оси Y  
 glPushMatrix();  
 glTranslatef(0, -cylinder\_half\_size, 0);  
 glRotatef(-90.0f, 1.0f, 0.0f, 0.0f); // Поворот вокруг оси X, чтобы цилиндр был вдоль оси Y  
 glutSolidCylinder(cylinder\_radius, cylinder\_size, 16, 1); // Высота 3.0f, чтобы проходить через весь куб  
 glPopMatrix();  
  
 // Цилиндр вдоль оси Z  
 glPushMatrix();  
 glTranslatef(0, 0, -cylinder\_half\_size);  
 glutSolidCylinder(cylinder\_radius, cylinder\_size, 16, 1); // Цилиндр по умолчанию вдоль оси Z  
 glPopMatrix();  
  
 glPopMatrix();  
}

В этой функции также строятся цилиндры, как основание конструкции цельного кубика.

**Алгоритм поворота граней**

Алгоритм основан на 3-мерной матрице массивов, элементы которой составляют параметры для поворота кубиков:

void rotate\_edge(rot\_edge\_params params) {  
 GLfloat rotate\_angle = params.is\_positive\_rot ? 90 : -90;  
  
 // Поворот грани по параметрам  
 reset\_selected\_face();  
 if (params.is\_vertical && params.is\_close\_edge) {  
 x\_k = x\_0;  
 } else if (!params.is\_vertical && params.is\_close\_edge) {  
 y\_k = y\_0;  
 } else if (params.is\_vertical && !params.is\_close\_edge) {  
 x\_0 = x\_k;  
 } else if (!params.is\_vertical && !params.is\_close\_edge) {  
 y\_0 = y\_k;  
 }  
  
 // Формирование параметров поворота грани  
 vector<cube\_rotate> face[3][3];  
 int indexI;  
 int indexJ;  
 cube\_rotate rotation;  
 for (int i = 0; i < 3; ++i)  
 for (int j = 0; j < 3; ++j) {  
 indexI = i;  
 indexJ = j;  
  
 if (x\_0 == x\_k) {  
 rotation = {-rotate\_angle, 1.0, 0.0, 0.0};  
 face[indexJ][indexI] = cube\_rotations[x\_k][i][j];  
 }  
  
 if (y\_0 == y\_k) {  
 rotation = {rotate\_angle, 0.0, 1.0, 0.0};  
 face[indexJ][indexI] = cube\_rotations[i][y\_k][j];  
 }  
  
 if (z\_0 == z\_k) {  
 rotation = {-rotate\_angle, 0.0, 0.0, 1.0};  
 face[indexJ][indexI] = cube\_rotations[i][j][z\_k];  
 }  
  
 face[indexJ][indexI].push\_back(rotation);  
 }  
  
 // Копируем получившийся face в cube\_rotations  
 for (int i = 0; i < 3; ++i)  
 for (int j = 0; j < 3; ++j) {  
 indexI = i;  
 indexJ = j;  
  
 if (x\_0 == x\_k)  
 cube\_rotations[x\_k][i][j] = face[indexI][indexJ];  
  
 if (y\_0 == y\_k)  
 cube\_rotations[i][y\_k][j] = face[indexI][indexJ]; // покрутить i j  
  
 if (z\_0 == z\_k)  
 cube\_rotations[i][j][z\_k] = face[indexI][indexJ];  
 }  
}

На вход приходят параметры поворота грани – они поворачивают, относительно лицевой стороны кубика, верх, низ, лево и право. Параметры поворота грани изменяют пары координат. Например, если x\_0 = 0 и x\_k = 0, то поворачиваем левый край по оси х, далее по аналогии.

В face сохраняем соответствующие грани массивы поворотов. И создаём новый поворот. Далее соответственно записываем получившийся поворот.

**Элементы управления**

Реализован поворот мышью и масштабирование колёсиком, вращение граней с помощью 8 клавиш.

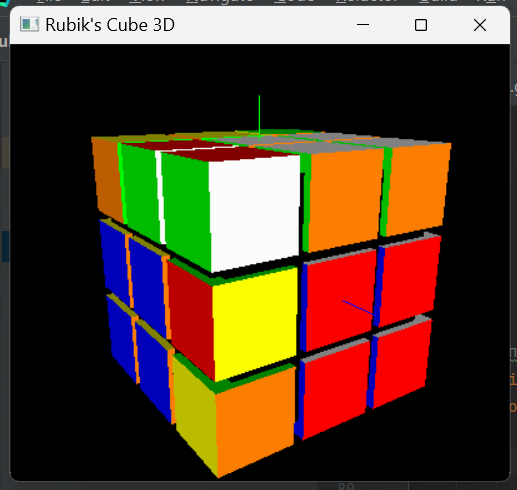
Поворот мышью реализован 2-я функциями:

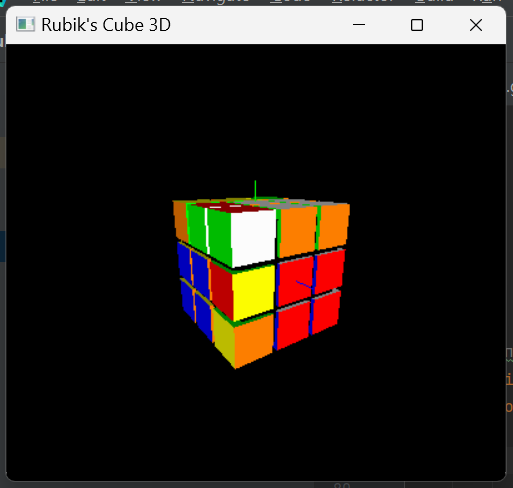
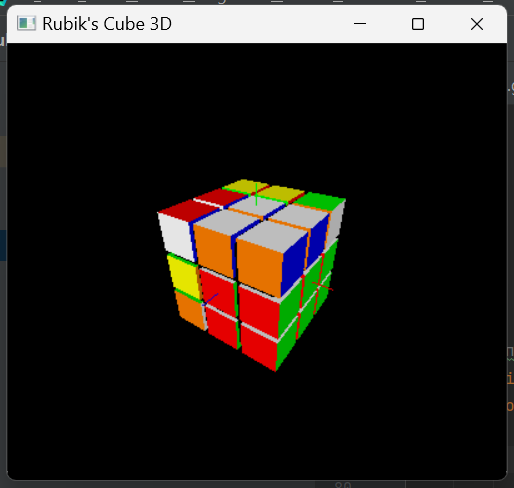
void mouse\_func(int button, int state, int x, int y) {  
 if (button == GLUT\_LEFT\_BUTTON) {  
 if (state == GLUT\_DOWN) {  
 mouseDragging = true;  
 lastMouseX = x;  
 lastMouseY = y;  
 } else if (state == GLUT\_UP) {  
 mouseDragging = false;  
 }  
 }  
}  
  
void mouse\_motion\_func(int x, int y) {  
 if (mouseDragging) {  
 // Вычисляем разницу в координатах мыши  
 int deltaX = x - lastMouseX;  
 int deltaY = y - lastMouseY;  
  
 // Обновляем углы поворота  
 rot\_y += deltaX \* 0.5f; // Чувствительность поворота  
 rot\_x += deltaY \* 0.5f;  
  
 // Сохраняем текущие координаты мыши  
 lastMouseX = x;  
 lastMouseY = y;  
  
 // Перерисовываем сцену  
 load\_visualization\_parameters();  
 glutPostRedisplay();  
 }  
}

Масштабирование колёсиком мыши:

void mouse\_wheel\_func(int wheel, int direction, int x, int y) {  
 if (direction > 0) {  
 if (angle >= 10) angle -= 5; // Zoom-in  
 } else {  
 if (angle <= 130) angle += 5; // Zoom-out  
 }  
 load\_visualization\_parameters();  
 glutPostRedisplay();  
}

**Результаты работы**



 ****

**Вывод:** в ходе работы написана программа для рисования кубика Рубика, с функцией вращения граней.