| *voenmeh* | МИНОБРНАУКИ РОССИИ  федеральное государственное бюджетное образовательное учреждение  высшего образования  **«Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова»**  **(БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова»)** |
| --- | --- |
| БГТУ.СМК-Ф-4.2-К5-01 |

| Факультет |  | О |  | Естественнонаучный |
| --- | --- | --- | --- | --- |
|  |  | шифр |  | наименование |
| Кафедра |  | О7 |  | Информационные системы и программная инженерия |
|  |  | шифр |  | наименование |
| Дисциплина |  | Компьютерная геометрия и графика | | |

Отчёт по практической работе № 4

| Трехмерные построения. Буфер глубины. |
| --- |
| Видовые параметры. |
| Параллельная и перспективная проекции. |

| Выполнил студент группы | | | |  | И508Б |
| --- | --- | --- | --- | --- | --- |
| Кабиров К.Р. | | | | | |
| Фамилия И.О. | | | | | |
| **РУКОВОДИТЕЛЬ** | | | | | |
|  | |  |  | | |
| Фамилия И.О. Подпись | | | | | |
| Оценка |  | | | |  |
| «\_\_\_\_\_» |  | | | | 2022г. |

САНКТ-ПЕТЕРБУРГ

2022 г

**Вариант 3**

**Задания к работе**

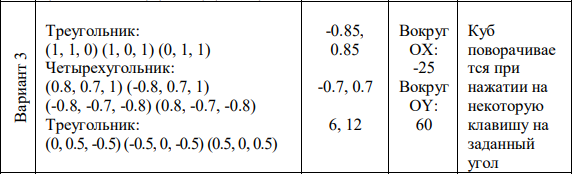


Рисунок 1 - Задание по варианту

1. Построить изображения трех плоских фигур в пространстве (в соответствии с вариантом). Оценить полученный результат: правильно ли изображены фигуры, если нет, объяснить почему

Содержимое функции DrawGLScene:

glClear(GL\_COLOR\_BUFFER\_BIT); // очистка экрана

glEnableClientState(GL\_VERTEX\_ARRAY);

srand(time(NULL));

GLdouble quadVertices[4][3] = {

{0.8, 0.7, 1.0},

{-0.8, 0.7, 1},

{-0.8, -0.7, -0.8},

{0.8, -0.7, -0.8}

};

GLdouble triangle1Vertices[3][3] = {

{1, 1, 0},

{1, 0, 1},

{0, 1, 1}

};

GLdouble triangle2Vertices[3][3] = {

{0, 0.5, -0.5},

{-0.5, 0, -0.5},

{0.5, 0, 0.5}

};

glVertexPointer(3, GL\_DOUBLE, 0, quadVertices);

glColor3f((float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX);

glDrawArrays(GL\_QUADS, 0, 4);

glVertexPointer(3, GL\_DOUBLE, 0, triangle1Vertices);

glColor3f((float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX);

glDrawArrays(GL\_TRIANGLES, 0, 3);

glVertexPointer(3, GL\_DOUBLE, 0, triangle2Vertices);

glColor3f((float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX);

glDrawArrays(GL\_TRIANGLES, 0, 3);

glDisableClientState(GL\_VERTEX\_ARRAY);

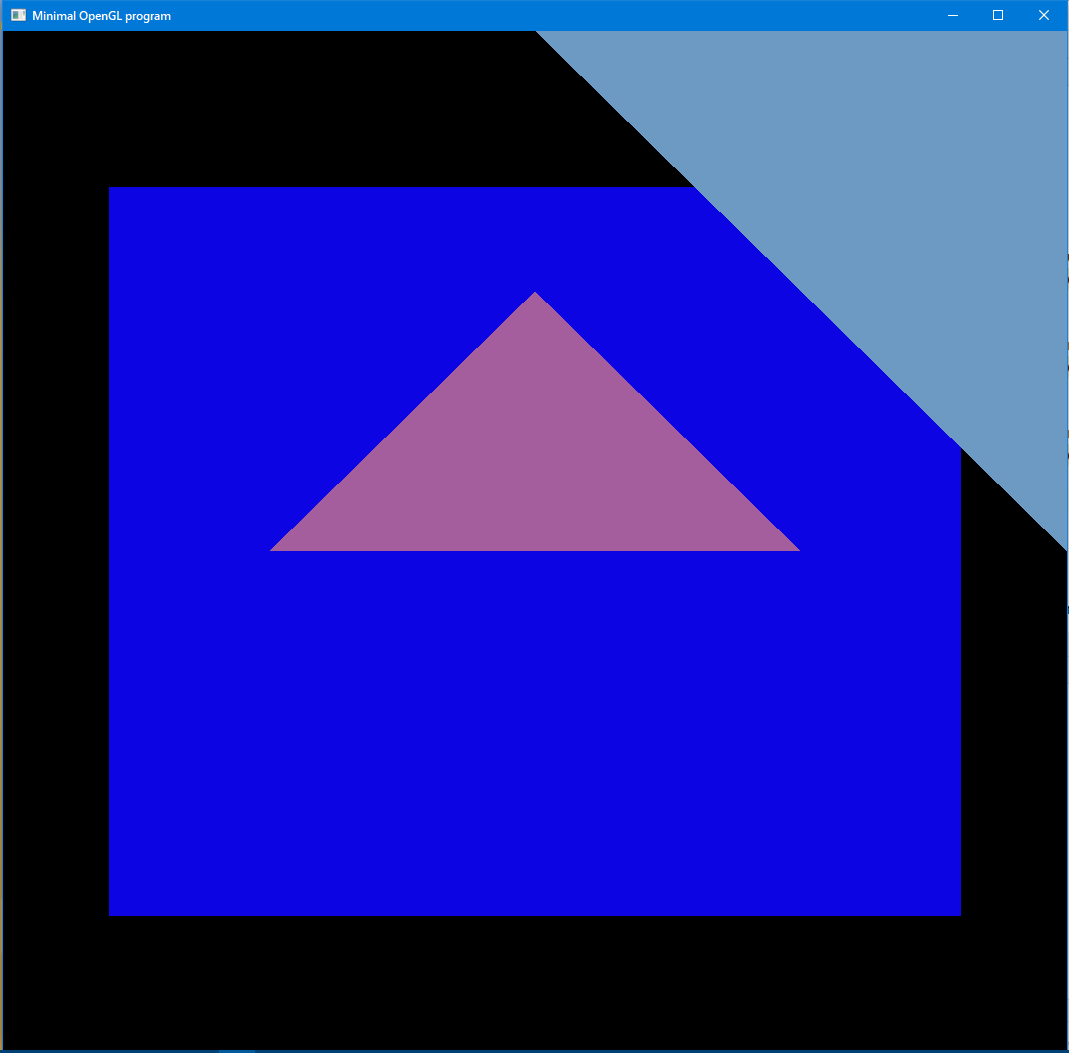


Рисунок 2 - Изображение фигур

Фигуры изображены неправильно. Это происходит из-за отсутствия буфера глубины.

1. Установить режим проверки буфера глубины в программе к заданию 1. Сравнить полученный результат с предыдущим.

Содержимое функции DrawGLScene:

glLoadIdentity; //Сброс всех матриц в 1

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT); // очистка экрана

glEnable(GL\_DEPTH\_TEST);

glEnableClientState(GL\_VERTEX\_ARRAY);

srand(time(NULL));

GLdouble quadVertices[4][3] = {

{0.8, 0.7, 1.0},

{-0.8, 0.7, 1},

{-0.8, -0.7, -0.8},

{0.8, -0.7, -0.8}

};

GLdouble triangle1Vertices[3][3] = {

{1, 1, 0},

{1, 0, 1},

{0, 1, 1}

};

GLdouble triangle2Vertices[3][3] = {

{0, 0.5, -0.5},

{-0.5, 0, -0.5},

{0.5, 0, 0.5}

};

glVertexPointer(3, GL\_DOUBLE, 0, quadVertices);

glColor3f((float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX);

glDrawArrays(GL\_QUADS, 0, 4);

glVertexPointer(3, GL\_DOUBLE, 0, triangle1Vertices);

glColor3f((float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX);

glDrawArrays(GL\_TRIANGLES, 0, 3);

glVertexPointer(3, GL\_DOUBLE, 0, triangle2Vertices);

glColor3f((float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX);

glDrawArrays(GL\_TRIANGLES, 0, 3);

glDisable(GL\_DEPTH\_TEST);

glDisableClientState(GL\_VERTEX\_ARRAY);

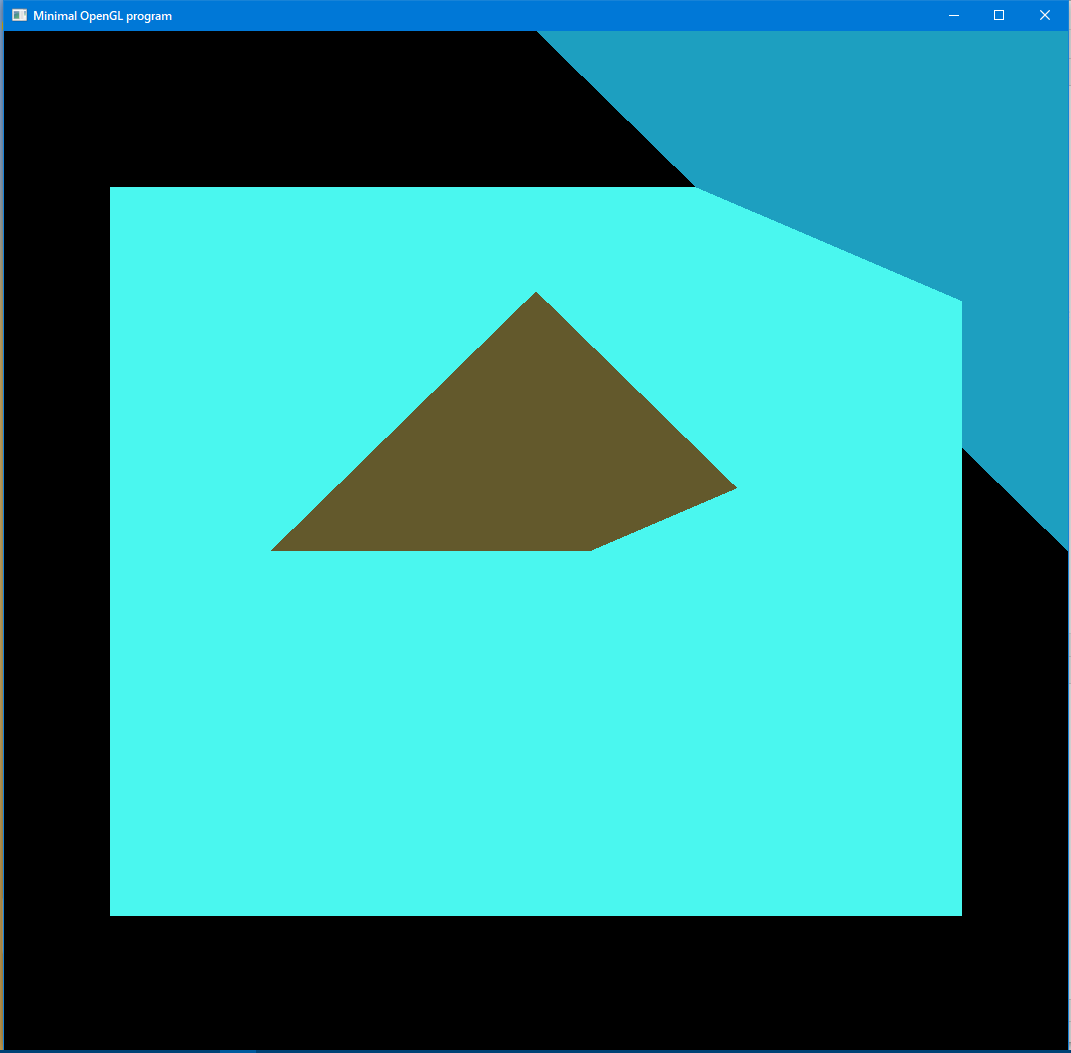


Рисунок 3 - Изображение фигур с буфером глубины

1. Установить видовые параметры в соответствии с вариантом и параллельную проекцию; построить изображение куба с координатами вершин, равными по модулю 1 (куб изображается с помощью шести квадратов). Оценить результат.

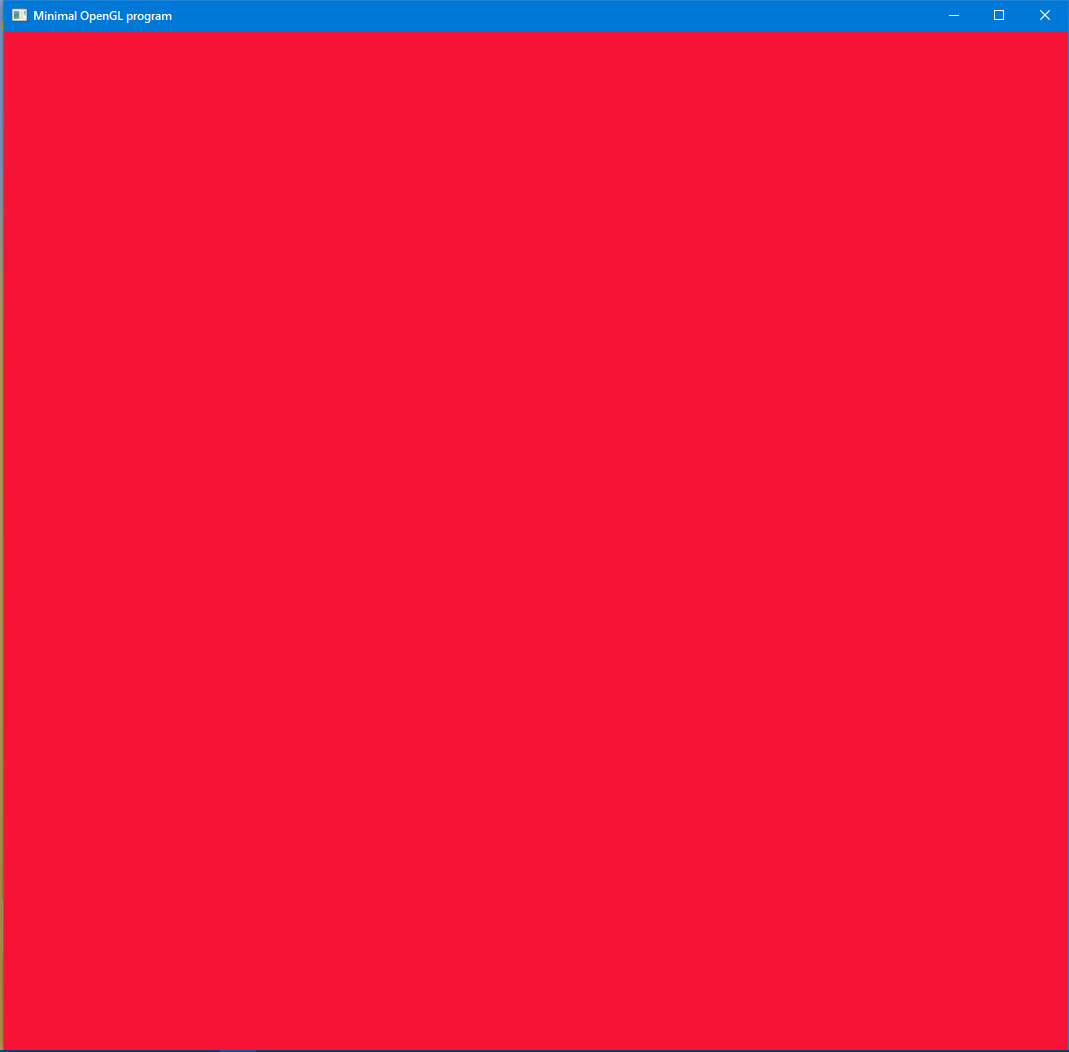


Рисунок 4 – Параллельная проекция куба с видовыми параметрами по варианту

Содержимое функции DrawGLScene:

glClear(**GL\_COLOR\_BUFFER\_BIT** | **GL\_DEPTH\_BUFFER\_BIT**); *// очистка экрана*

glEnableClientState(**GL\_VERTEX\_ARRAY**);

glEnable(**GL\_DEPTH\_TEST**);

srand(time(**NULL**));

GLdouble cubeVertices[8][3] = {

{-1, 1, -1},

{1, 1, -1},

{1, -1, -1},

{-1, -1, -1},

{-1, 1, 1},

{1, 1, 1},

{1, -1, 1},

{-1, -1, 1}

};

glVertexPointer(3, **GL\_DOUBLE**, 0, cubeVertices);

glMatrixMode(**GL\_PROJECTION**);

glLoadIdentity(); *//Сброс всех матриц в 1*

glOrtho(-0.85, 0.85, -0.7, 0.7, 6, 12); *// параллельная проекция с видовыми параметрами*

glTranslatef(0.0, 0.0, -10); *//начальный сдвиг*

glColor3f((float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**, (float)rand() / **RAND\_MAX**);

glBegin(**GL\_QUADS**);

*//1-ый*

glArrayElement(0);

glArrayElement(1);

glArrayElement(2);

glArrayElement(3);

*//2-ой*

glArrayElement(0);

glArrayElement(1);

glArrayElement(5);

glArrayElement(4);

*//3-ий*

glArrayElement(1);

glArrayElement(5);

glArrayElement(6);

glArrayElement(2);

*//4-ый*

glArrayElement(4);

glArrayElement(5);

glArrayElement(6);

glArrayElement(7);

*//5-ый*

glArrayElement(2);

glArrayElement(3);

glArrayElement(7);

glArrayElement(6);

*//6-ой*

glArrayElement(0);

glArrayElement(3);

glArrayElement(7);

glArrayElement(4);

glEnd();

glDisableClientState(**GL\_VERTEX\_ARRAY**);

glDisable(**GL\_DEPTH\_TEST**);

1. Установить перспективную проекцию и построить каркасное изображение куба

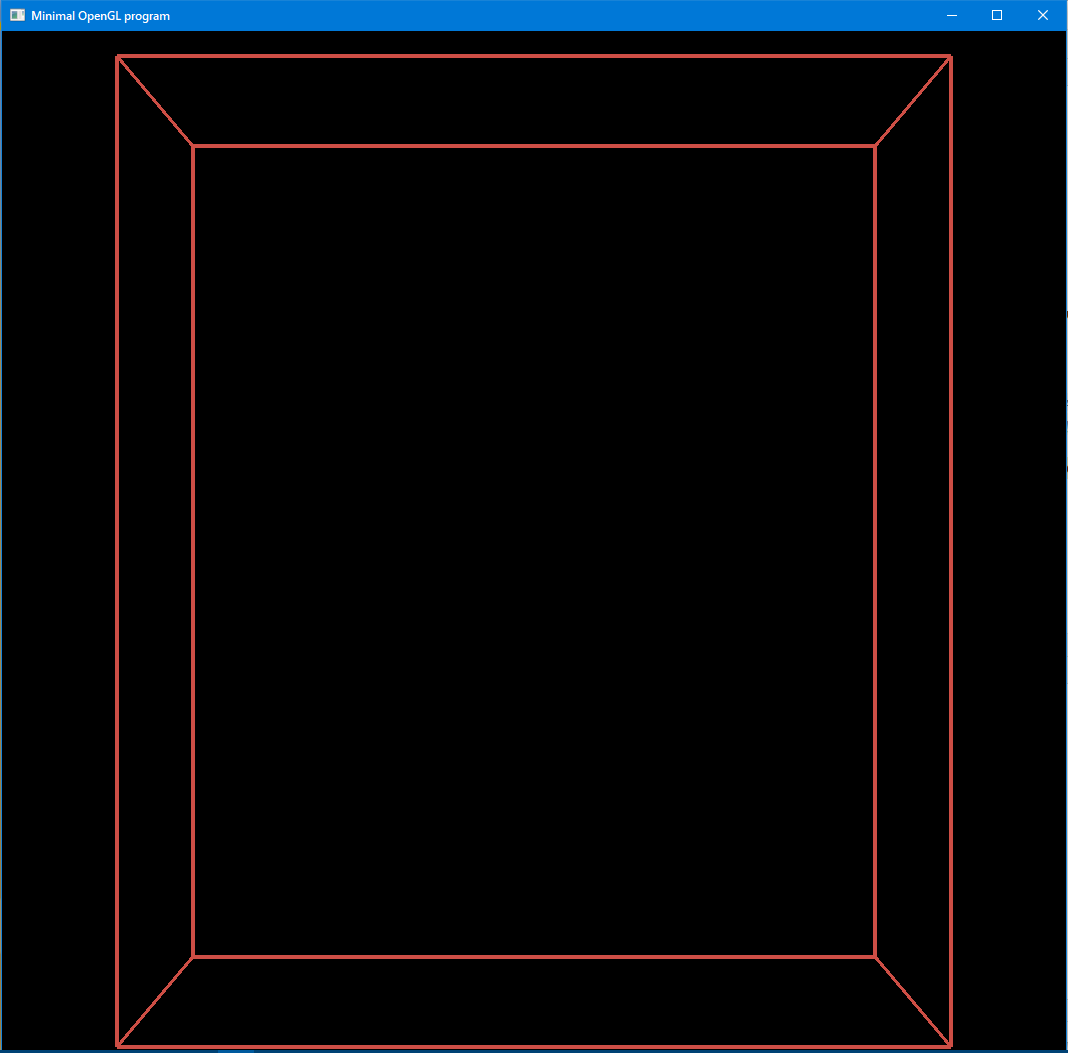


Рисунок 5 – Каркас куба в перспективной проекции с видовыми параметрами по варианту

Содержимое DrawGLScene:

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT); // очистка экрана

glEnableClientState(GL\_VERTEX\_ARRAY);

glEnable(GL\_DEPTH\_TEST);

srand(time(NULL));

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_LINE);

glLineWidth(4);

GLdouble cubeVertices[8][3] = {

{-1, 1, -1},

{1, 1, -1},

{1, -1, -1},

{-1, -1, -1},

{-1, 1, 1},

{1, 1, 1},

{1, -1, 1},

{-1, -1, 1}

};

glVertexPointer(3, GL\_DOUBLE, 0, cubeVertices);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity(); //Сброс всех матриц в 1

glFrustum(-0.85, 0.85, -0.7, 0.7, 6, 12); // параллельная проекция с видовыми параметрами

glTranslatef(0.0, 0.0, -10); //начальный сдвиг

glColor3f((float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX);

glBegin(GL\_QUADS);

//1-ый

glArrayElement(0);

glArrayElement(1);

glArrayElement(2);

glArrayElement(3);

//2-ой

glArrayElement(0);

glArrayElement(1);

glArrayElement(5);

glArrayElement(4);

//3-ий

glArrayElement(1);

glArrayElement(5);

glArrayElement(6);

glArrayElement(2);

//4-ый

glArrayElement(4);

glArrayElement(5);

glArrayElement(6);

glArrayElement(7);

//5-ый

glArrayElement(2);

glArrayElement(3);

glArrayElement(7);

glArrayElement(6);

//6-ой

glArrayElement(0);

glArrayElement(3);

glArrayElement(7);

glArrayElement(4);

glEnd();

glDisableClientState(GL\_VERTEX\_ARRAY);

glDisable(GL\_DEPTH\_TEST);

1. Повернуть оси координат в соответствии с вариантом и построить изображение куба. Оценить результат. Почему части куба отсечены? Изменить видовые параметры таким образом, чтобы куб изображался без отсечений. Изменить код программы таким образом, чтобы каждая грань куба изображалась своим цветом.

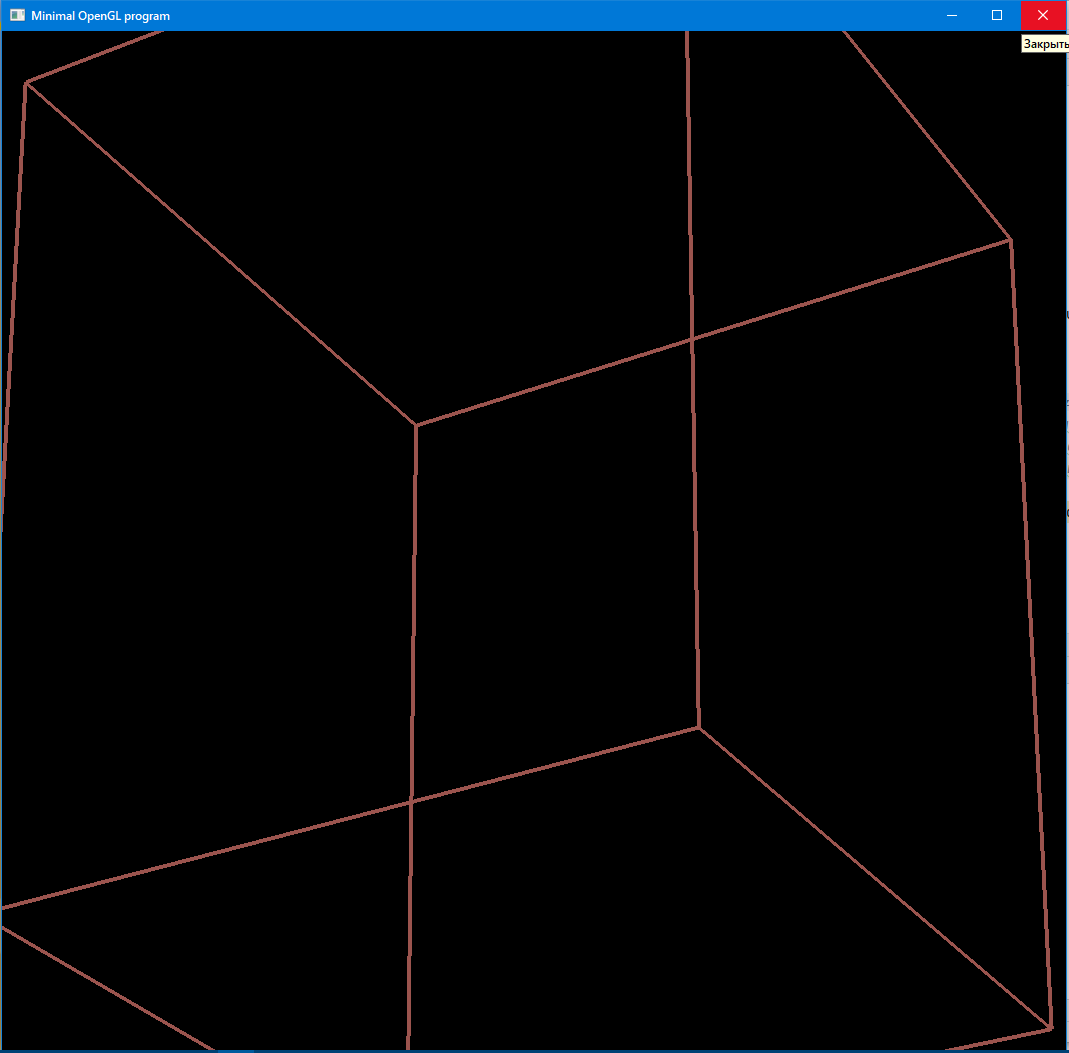


Рисунок 6 – Куб с отсечениями и повернутыми по варианту осями

Части куба отсечены, потому что мы их отсекли видовыми параметрами функции проекции.

Содержимое DrawGLScene:

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT); // очистка экрана

glEnableClientState(GL\_VERTEX\_ARRAY);

glEnable(GL\_DEPTH\_TEST);

srand(time(NULL));

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_LINE);

glLineWidth(4);

GLdouble cubeVertices[8][3] = { {-1, 1, -1}, {1, 1, -1}, {1, -1, -1}, {-1, -1, -1}, {-1, 1, 1}, {1, 1, 1}, {1, -1, 1}, {-1, -1, 1} };

glVertexPointer(3, GL\_DOUBLE, 0, cubeVertices);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity(); //Сброс всех матриц в 1

glFrustum(-0.9, 0.8, -0.8, 0.9, 5, 10); // параллельная проекция с видовыми параметрами

glTranslatef(0.0, 0.0, -7); //начальный сдвиг

//системы координат

glRotatef(45.0, 1.0, 0.0, 0.0); //поворот относительно оси X

glRotatef(45.0, 0.0, 1.0, 0.0); //поворот относительно оси Y

glBegin(GL\_QUADS);

//1-ый

glColor3f((float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX);

glArrayElement(0);

glArrayElement(1);

glArrayElement(2);

glArrayElement(3);

//2-ой

glColor3f((float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX);

glArrayElement(0);

glArrayElement(1);

glArrayElement(5);

glArrayElement(4);

//3-ий

glColor3f((float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX);

glArrayElement(1);

glArrayElement(5);

glArrayElement(6);

glArrayElement(2);

//4-ый

glColor3f((float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX);

glArrayElement(4);

glArrayElement(5);

glArrayElement(6);

glArrayElement(7);

//5-ый

glColor3f((float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX);

glArrayElement(2);

glArrayElement(3);

glArrayElement(7);

glArrayElement(6);

//6-ой

glColor3f((float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX, (float)rand() / RAND\_MAX);

glArrayElement(0);

glArrayElement(3);

glArrayElement(7);

glArrayElement(4);

glEnd();

glDisableClientState(GL\_VERTEX\_ARRAY);

glDisable(GL\_DEPTH\_TEST);