

Задание для группы ЭН-34

Лабораторная работа №6 посвящена разработке программ с использованием библиотеки OpenGL, которые выполняют отображение псевдо-3D объектов. Объекты, которые предстоит рисовать – двумерные, но их нужно представить в виде 3D поверхностей.

Входные данные задаются в файлах.

ЗАДАНИЯ НА МИНИМАЛЬНЫЙ БАЛЛ (7 баллов)

Внимание! Входной файл test_34_00.txt – тестовый, его размер всего 4x4! Он нужен для того, чтобы проверить механизм чтения и отрисовки.

В файлах задаётся матрица из чисел с плавающей запятой, размер матрицы – 99x99.

Задание на лабораторную работу: визуализировать с помощью библиотеки OpenGL поверхности, заданные матрицами во входных файлах.

В процессе выполнения данной лабораторной работы у вас есть возможность получить три плюса за следующие достижения:

1. написать программу, которая считывает данные из файла и выводит в консоль информацию о найденных максимальном и минимальном значениях по оси Z;
2. нарисовать при помощи библиотеки OpenGL поверхность, добавив код для вращения вокруг всех осей (реализацию вращения можно сделать как угодно - можно вращать с клавиатуры, можно мышкой, по вашему усмотрению);
3. дополнить код индивидуальным заданием.

Что нужно сделать обязательно всем вариантам:

- фигура должна рисоваться по центру экрана и вращаться относительно своего центра;
- приближение/удаление – на клавиши «+»/«-»;
- клавиша «Home» - возврат к начальному состоянию по вращению и зуму;
- клавиша Esc – выход из программы.

Обязательное условие!

Весь код, отвечающий за работу с 3D поверхностью, должен располагаться в классе. Например, вид этого класса может быть следующим:

```
class Mesh {  
public:  
    // В этой функции расположен код, ответственный  
    // за считывание сетки из входного файла
```

```
bool LoadFromFile(wchar_t *fileName);  
// В этой функции расположен код, ответственный  
// за отрисовку сетки средствами библиотеки OpenGL  
void Draw();  
};
```

Тогда код, ответственный за рисование, будет примерно следующим (повороты и перемещения могут остаться в RenderScene):

```
void RenderScene() {  
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);  
    glMatrixMode(GL_MODELVIEW);  
    glLoadIdentity();  
    // Начало нашего кода  
  
    mesh.Draw();  
  
    // Конец нашего кода  
    glutSwapBuffers();  
}
```

Варианты на третий плюсик

№	Задания
1, 5, 9, 13, 17, 21, 25	Цвет фона – тёмно-серый. Фигура рисуется сеткой. Цвет узла сетки показывает высоту точки от минимального до максимального значения, меняясь от белого в минимуме до зелёного в максимуме. Последовательное нажатие клавиши F4 приводит к тому, что меняется режим отрисовки поверхности – сеткой или по точкам.
2, 6, 10, 14, 18, 22, 26	Цвет фона – светло-серый. Цвет узла сетки показывает высоту точки от минимального до максимального значения, меняясь от синего в минимуме до чёрного в максимуме. При нажатии на кнопку F6 нужно включать/выключать отрисовку плоскостей, перпендикулярных осям X и Y и проходящих через точку (0,0,0), по одной плоскости за раз.

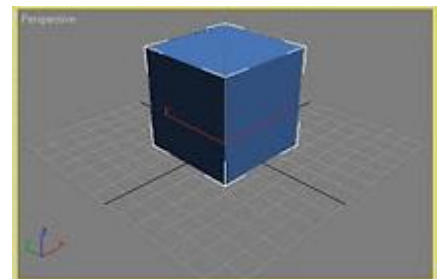
3, 7, 11, 15, 19, 23, 27	Цвет фона – тёмно-зелёный. Цвет узла сетки показывает высоту точки от минимального до максимального значения, меняясь от жёлтого в минимуме до красного в максимуме. При нажатии на клавишу PageDown меняется режим отрисовки фигуры: сплошная заливка и сеткой.
4, 8, 12, 16, 20, 24, 28	Цвет фона – светло-синий. Цвет узла сетки показывает высоту точки от минимального до максимального значения, меняясь от синего в минимуме до чёрного в максимуме. Нажатии на клавишу Del все точки рисуются взятыми по модулю (цвет сохраняется). Повторное нажатие – как обычно.

ЗАДАНИЯ НА ДОПОЛНИТЕЛЬНЫЕ БАЛЛЫ (+3 балла максимум)

Необходимо выполнить все задания на минимум. После этого необходимо выполнить дополнительные задания, приведённые ниже.

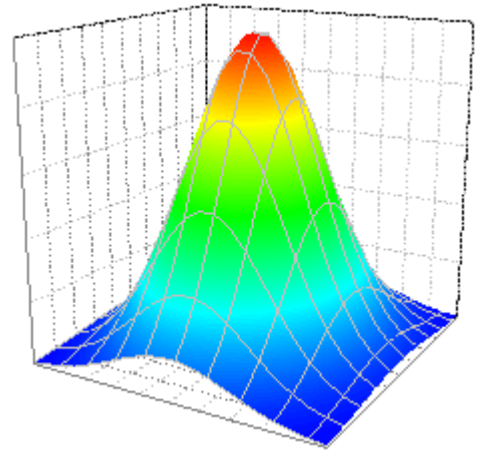
Эти задания выполняются последовательно.

- **Плюс балл.** Нажатием на клавишу F7 включается режим постоянного вращения фигуры вокруг осей. Начинается вращение с оси Z, последовательно нажимая клавишу F7 можно менять оси: X, Y, Z, <выключено>.
- **Плюс балл.** В левом нижнем углу, как это делается в 3D редакторах, всегда должны рисоваться оси небольшого размера, показывающие поворот фигуры (см. рисунок).
- **Плюс балл.** Имя файла и ключ, задающий необходимость рисовать ось из п.2. задаются как аргумент командной строки.
Например (это просто пример, вы можете реализовать по-своему), вы можете реализовать следующий вариант: если в командной строке задаётся ключ (допустим, это ключ «--drawaxis»), то оси рисуются, если не задаётся – не рисуются.



ЗАДАНИЕ ПОВЫШЕННОЙ СЛОЖНОСТИ

Цвета точек высчитываются так, как это обычно происходит в САПР. Цвет точки показывает высоту от минимума до максимума, меняясь в следующем порядке: синий (минимум) – голубой – зелёный – жёлтый – красный (максимум). Цвета должны меняться плавно (должны высчитываться, а не браться порогом).



Дополнительные сведения для выполнения лабораторной работы.

Q: Как поменять размер пикселя при отрисовке по точкам?

A: Для этого нужно воспользоваться функцией **glPointSize**, имеющей следующий формат:

```
void glPointSize(GLfloat size)
```

Q: Как ловить события мыши?

A: Раздельно ловятся факты нажатия и перемещения мыши. Нажатие ловится функцией, связываемой с OpenGL посредством `glutMouseFunc`.

Пример:

```
void mouse(int button, int state, int x, int y) {
    if(button == GLUT_LEFT_BUTTON)
        printf("left button!");
    if(state == GLUT_UP)
        printf("button up!");
}

int main(int argc, char *argv[]) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
    glutInitWindowPosition(100,100);
    glutInitWindowSize(800,800);
    glutCreateWindow("3D");
    glEnable(GL_DEPTH_TEST);
    glutDisplayFunc(renderScene);

    glutMouseFunc(mouse);

    glutMainLoop();
    return 0;
}
```

Перемещение ловится с помощью функции, которая связывается с OpenGL с помощью `glutMotionFunc`. Эта функция вызывается только тогда, когда перемещение происходит с нажатой клавишей, поэтому факт нажатия можно не проверять.

Пример:

```
void motion(int x, int y) {
    printf("Yahoo! I've been moved!\n");
}

int main(int argc, char *argv[]) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
    glutInitWindowPosition(100,100);
    glutInitWindowSize(800,800);
    glutCreateWindow("WaveViewer");
    glEnable(GL_DEPTH_TEST);
    glutDisplayFunc(renderScene);

    glutMotionFunc(motion);

    glutMainLoop();
    return 0;
}
```

Q: Как читать функциональные клавиши?

A: Решение аналогично чтению обычных клавиш, только меняется тип первого аргумента.
Привожу код ниже:

```
void read_special_kb(int key, int, int) {  
    if (key == GLUT_KEY_HOME)  
        ...  
}
```

Связывание обработчика нажатия функциональных клавиш в функции main:

```
glutSpecialFunc(read_special_kb);
```