**Задание для группы ЭН-35**

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_35_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 – 101x101.

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 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,<br>5,<br>9,<br>13,<br>17,<br>21,<br>25 | Цвет фона – светло-серый.<br><br>Цвет узла сетки показывает высоту точки от минимального до максимального значения, меняясь от синего в минимуме до белого в максимуме.<br><br>При нажатии на клавишу F5 меняется режим отрисовки фигуры: сплошная заливка и сетка. |
| 2,<br>6,<br>10,<br>14,<br>18,<br>22<br>26 | Цвет фона – тёмно-серый.<br><br>Цвет узла сетки показывает высоту точки от минимального до максимального значения, меняясь от зелёного в минимуме до белого в максимуме.<br><br>При нажатии на клавишу F2 из центра координат рисуются оси OX, OY, OZ. Повторное нажатие – оси не рисуются. |
| 3,<br>7,<br>11,<br>15,<br>19, | Цвет фона – тёмно-зелёный.<br><br>Цвет узла сетки показывает высоту точки от минимального до максимального значения, меняясь от жёлтого в минимуме до красного в максимуме. |

| | |
|---|---|
| 23,<br>27 | При нажатии на клавишу Page Up меняется режим отрисовки фигуры: сплошная заливка и по точкам. |
| 4,<br>8,<br>12,<br>16,<br>20,<br>24,<br>28 | Цвет фона – светло-синий.<br><br>Цвет узла сетки показывает высоту точки от минимального до максимального значения, меняясь от синего в минимуме до чёрного в максимуме.<br><br>Нажатии на клавишу Ins все точки рисуются в плоскости Z=0. Повторное нажатие – как обычно. |

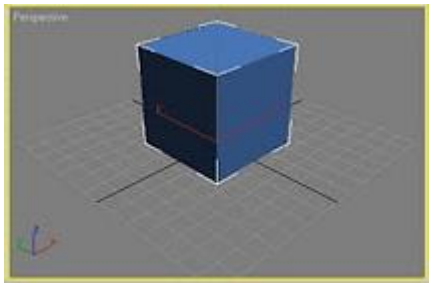
## ЗАДАНИЯ НА ДОПОЛНИТЕЛЬНЫЕ БАЛЛЫ (+3 балла максимум)

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

***Эти задания выполняются последовательно.***

- **Плюс балл.** Нажатием на клавишу F1 включаются и выключаются оси, находящиеся в центре фигуры (цвета – R для X, G для Y, B для Z). Оси должны иметь диапазон, совпадающий с размером фигуры (из центра идти во все стороны по размеру фигуры).

- **Плюс балл.** В левом нижнем углу, как это делается в 3D редакторах, всегда должны рисоваться оси небольшого размера, показывающие поворот фигуры (см. рисунок).

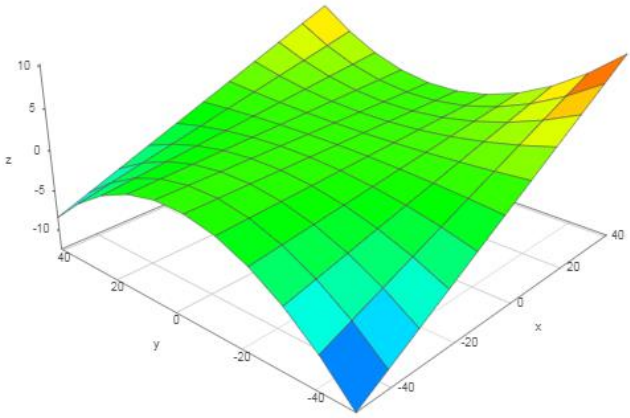


- **Плюс балл.** Имя файла и ключ, задающий необходимость рисовать ось из п.2. задаются как аргумент командной строки.
  Например (это просто пример, вы можете реализовать по-своему), вы можете реализовать следующий вариант: если в командной строке задаётся ключ (допустим, это ключ «--drawaxis»), то оси рисуются, если не задаётся – не рисуются.

  Оси рисуются:
  proga.exe test_data1.txt --drawaxis
  Оси не рисуются:
  proga.exe test_data1.txt

# 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);
```