Задание для группы ЭН-34

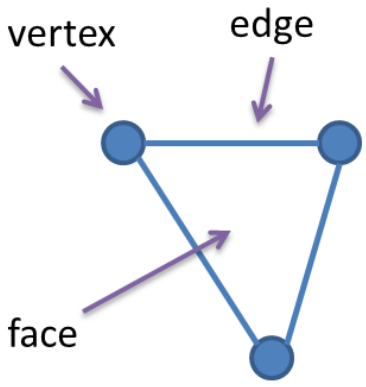
На этой лабораторной работе вы выступаете в роли разработчиков САПР для этапа технологического моделирования (TCAD). В качестве примеров для вас подготовлены существенно упрощённые файлы, описывающие сетку трёхмерного объекта. Основой для подготовленных для вас примеров являются файлы настоящего САПР технологического моделирования Synopsys TCAD в формате DF-ISE.

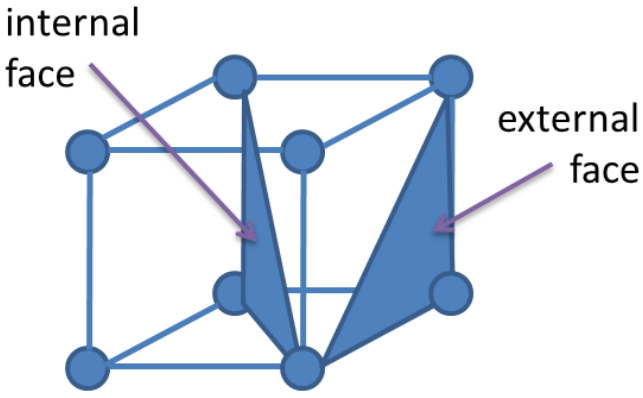
В формате DF-ISE данные сетки хранятся в файле с расширением .grd, который устроен следующим образом (с учётом сделанного для вас упрощения).

Все данные разделены на секции, названия секций соответствуют тому, что в этих секциях описывается. Всего в файле присутствует три вида секций.

1. **Vertices** – это массив координат вершин точек-узлов сетки.
2. **Edges** – массив рёбер сетки. Для каждого ребра хранятся индексы вершин тх массива **Vertices**, которые в него входят.
3. **Faces** – массив поверхностей. Все поверхности – треугольники, в каждой строчке есть три числа, число показывает номер ребра из массива **Edges**, образующее треугольник. Если число отрицательное, это значит, что ребро переориентируется, то есть вместо того, чтобы быть нарисованным от 1 к 2 точке, ребро рисуется от 2 к 1 точке и его индекс берётся положительным и на единицу меньше.
   Например, в файле cube.grd в разделе Faces есть такая поверхность: «-14 2 7». Это значит, что первым берётся ребро 13, а его начальная и конечная точки меняются местами, остальные рёбра (2 и 7) рисуются так, как и должны.
4. Locations – массив, показывающий положение соответсвующей поверхности из массива Faces: 'i' означает, что поверхность располагается внутри фигуры, а 'e' – что снаружи.

Пример взаимного отношения элементов и их расположение показано на рисунке ниже.





ЗАДАНИЯ НА МИНИМАЛЬНЫЙ БАЛЛ (7 баллов)

Что нужно сделать обязательно всем вариантам:

1. фигура должна рисоваться по центру экрана и вращаться относительно центра;
2. приближение/удаление – на клавиши «+»/«-»;
3. клавиша Esc – выход из программы.

Обязательное условие!

Весь код для работы с сеткой устройства, точки которого читаются из файла, должен быть реализован в виде класса.

Класс должен иметь интерфейсные методы:

- метод для чтения данных из файла, возвращает bool, чтобы было понятно, удалось прочитать файл или нет;
- метод для рисования фигуры;
- метод для реализации индивидуального задания.

Пример того, как должен выглядеть ваш код для работы с сеткой устройства, приведён ниже.

```
#pragma once

class Grid {
  std::vector<Vertex> vertices;
  std::vector<Edge>   edges;
  std::vector<Face>   faces;
public:
  bool Load(std::string fileName);
  void Draw();
  ...
};
```

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 всем осям;
2. нарисовать при помощи библиотеки OpenGL фигуру;
3. дополнить код небольшим индивидуальным заданием.

Варианты на третий плюс

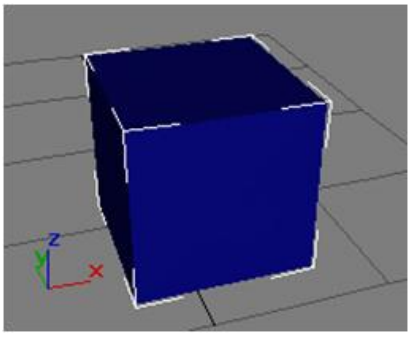
№	Задания
1 5 9 13 17 21	Фон окна – светло-серый. Фигура – красная. Сетка фигуры – чёрная. При нажатии на клавишу «M» должен меняться режим отрисовки фигуры: вместо заливки отрисовка проводится по массиву вершин, точки рисуются точки размером 10.0. Сетка, если она включена, по-прежнему должна рисоваться. Рисование сетки включается/выключается клавишей «пробел».
2 6 10 14 18 22	Фон окна – тёмно-серый. Фигура – зелёная. Сетка фигуры – синяя. При нажатии на клавишу «M» должен меняться перечень рисуемых поверхностей. Рисоваться должны все / только внутренние поверхности. Клавишей «Ввод» можно включать/выключать отрисовку самой фигуры (рисуется фигура и сетка или только сетка).
3 7 11 15 19 23	Фон окна – чёрный. Фигура – синяя. Сетка фигуры – белая. При нажатии на клавишу «M» должен меняться режим отрисовки фигуры: вместо заливки отрисовка проводится по массиву Edges, линии рисуются линии размером 4.0. Сетка по-прежнему должна рисоваться. Рисование сетки включается/выключается клавишей «пробел».
4 8 12 16 20 24	Фон окна – белый. Фигура – чёрная. Сетка фигуры – жёлтая. При нажатии на клавишу «M» должен меняться перечень рисуемых поверхностей. Рисоваться должны все / только внешние поверхности. Клавишей «Ввод» можно включать/выключать отрисовку самой фигуры (рисуется фигура и сетка или только сетка).

ЗАДАНИЯ НА ДОПОЛНИТЕЛЬНЫЕ БАЛЛЫ (+3 балла максимум)

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

Эти задания выполняются последовательно.

- **Плюс балл.** В точках минимальных и максимальных координат рисуются ограничивающие направляющие фигуры (см. рисунок). Цвет для направляющей: синий для минимальных значений, красный – для максимальных.



- **Плюс балл.** Нажатием на клавишу F10 включаются и выключаются оси X, Y, Z, которые рисуются разными цветами в левом нижнем углу окна (как показано на рисунке) и вращаются вместе с фигурой.

- **Плюс балл.** Клавишей F11 включается-выключается рисование осей из центра координат фигуры в размер фигуры (цвета осей – как в предыдущем пункте). Если оси включены, фигура рисуется точками.

ЗАДАНИЕ ПОВЫШЕННОЙ СЛОЖНОСТИ

Задание повышенной сложности.

Фигура должна вращаться с помощью мыши. Если зажата клавиша Ctrl и происходит движение в вертикальном направлении – фигура должна масштабироваться.

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);
```