Лабораторная работа №4

Цель работы – освоение приемов моделирования с помощью параметрических кривых и поверхностей и инструментов их реализации с использованием OpenGL.

Задание 1

Необходимо разработать интерактивную OpenGL-программу, позволяющую конструировать плоские параметрические сплайновые кривые одного из следующих типов:

0. Сплайновые кривые на основе полиномов Эрмита;
1. Сплайновые кривые на основе кубических полиномов
2. Сплайны, составленные из кривых Безье

Ваш номер типа определяется по формуле

```
N = YourStudNumber % 3,
```

где `N` – номер типа, а `YourStudNumber` – номер Вашей зачетной книжки.

Программа должна предоставлять возможности

- создания, удаления и изменения положения и (при необходимости) прочих параметров каркасных точек (или каркасных ломаных) для очередного сплайнового сегмента;
- обеспечения стыковки сплайновых сегментов с заданным уровнем гладкости.

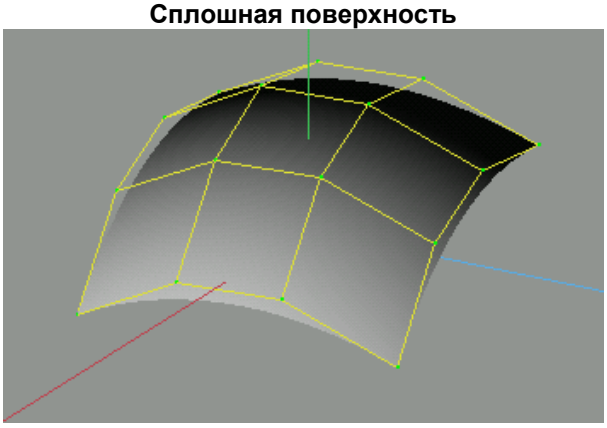
Создаваемую программу следует расценивать как инструмент рисования и моделирования, поэтому управление программой должно осуществляться преимущественно с помощью «мыши». Прибегать к явному вводу каких-либо значений следует лишь в самом крайнем случае. В качестве примеров для подражания следует рассматривать широко известные графические редакторы, реализующие инструменты построения кривых.

Задание 2

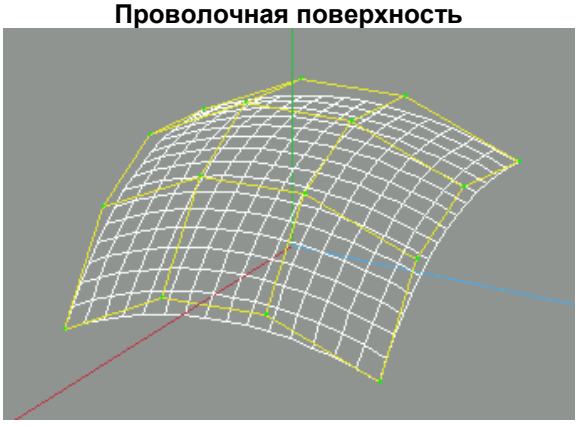
Необходимо создать **один сегмент** поверхности Безье, заданный как минимум 16-ю контрольными точками (см. примеры), используя функции из библиотеки `GL` (`glMap2[fd]`, `glMapGrid2[fd]`, `glEvalMesh2`).

Примеры визуализации поверхности Безье

Сплошная поверхность



Проволочная поверхность



В программе необходимо реализовать возможности

- вращения поверхности (для того, чтобы можно было изучить ее со всех сторон);
- изменения способа отрисовки поверхности (каркасный, сплошной гладкий)

Функции `glMap2[fd]`, `glMapGrid2[fd]`, `glEvalMesh2`

Все ниже приведенные функции вызываются там же, где и примитивы.

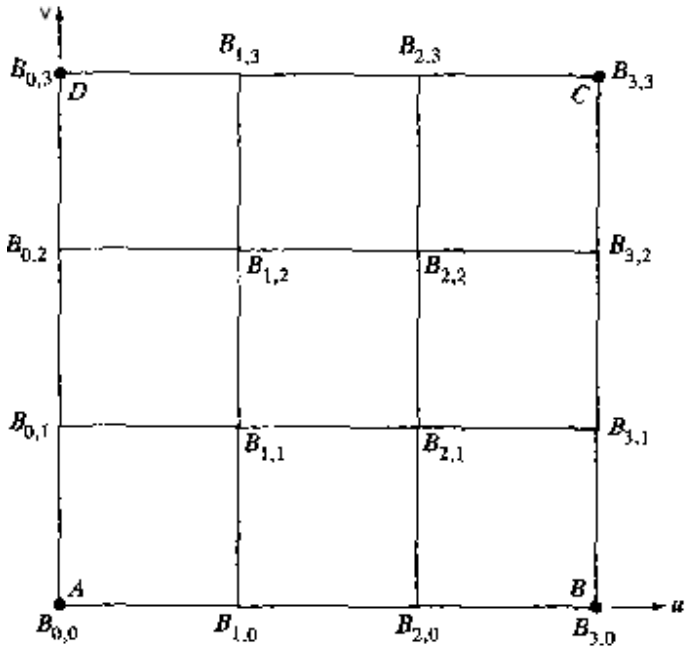


Рис. 1. Контрольные точки $B_{i,j}$. Направления *u*, *v*.

Задание контрольных точек

```
void glMap2d(
      GLenum target,          // тип данных. Если равен GL_MAP2_VERTEX_3, значит
                              // поверхность строится по массиву точек из трех координат
                              // (x, y, z). Можно задать GL_MAP2_VERTEX_4, тогда точка
                              // задается 4-мя координатами (x, y, z, h), где h – это вес.
      GLdouble u1,            // начальное значение параметра в u направлении
      GLdouble u2,            // конечное значение параметра в u направлении
      GLint ustride,          // количество данных типа double между соседними точками
                              // в u направлении
      GLint uorder,           // количество точек в u направлении
      GLdouble v1,            // начальное значение параметра в v направлении
      GLdouble v2,            // конечное значение параметра в v направлении
      GLint vstride,          // количество данных типа double между соседними
                              // точками в v направлении
      GLint vorder,           // количество точек в v направлении
      GLdouble *points        // массив контрольных точек, задающих поверхность
            );
```

Определение аппроксимирующей сетки, по которой строится поверхность

```
void glMapGrid2d(
      GLint un,               // количество точек сетки в u направлении будет равно un+1
      GLdouble u1,            // начальное значение параметра в u направлении
      GLdouble u2,            // конечное значение параметра в u направлении
```

```
      GLint vn,              // количество точек сетки в v направлении будет равно vn+1
      GLdouble v1,           // начальное значение параметра в v направлении
      GLdouble v2            // конечное значение параметра в v направлении
);
```

Определение параметров визуализации поверхности Безье

```
void glEvalMesh2(
      GLenum mode,           // GL_LINE – отображается сетка, аппроксимирующая поверхность
                             // GL_FILL – отображается поверхность
                             // GL_POINT – точки сетки аппроксимирующей поверхности
      GLint i1, GLint i2,    // номера точек аппроксимирующей сетки в u направлении,
                             // с которых начинается визуализация (i1) поверхности и
                             // заканчивается (i2). Причем i1 < i2. i1, i2 принимают
                             // значения от 0 до un (из функции glMapGrid2f)
      GLint j1, GLint j2,    // номера точек аппроксимирующей сетки в v направлении,
                             // с которой начинается визуализация (j1) поверхности и
                             // заканчивается (j2). Причем j1 < j2. j1, j2 принимают
                             // значения от 0 до vn (из функции glMapGrid2f)
);
```