МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ
по лабораторной работе №4
по дисциплине «Компьютерная графика»
Тема: «Кубические сплайны»
Вариант 38

Студентка гр. 7381	________________________	Алясова А.Н.
Студентка гр. 7381	________________________	Кушкоева А.О.
Преподаватель	________________________	Герасимова Т.В.

Санкт-Петербург

2020

Цель работы.

Реализовать интерактивное приложение, отображающее заданные полиномиальные кривые.

Задание.

NURB-кривая. n = 5, k = 4. Узловой вектор равномерный. Веса точек различны и модифицируются

В отчете д.б. представлена реализуемая в программе формула, описан алгоритм построения и показаны основные характеристики кривой.

Общие сведения.

Интерполяция В-сплайнами

Чуть более сложный тип интерполяции – так называемая полиномиальная сплайн-интерполяция, или интерполяция В-сплайнами. В отличие от обычной сплайн-интерполяции, сшивка элементарных В-сплайнов производится не в точках (t_i, x_i), а в других точках, координаты которых обычно предлагается определить пользователю. Таким образом, отсутствует требование равномерного следования узлов при интерполяции В-сплайнами.

Сплайны могут быть полиномами первой, второй или третьей степени (линейные, квадратичные или кубические). Применяется интерполяция Всплайнами точно так же, как и обычная сплайн-интерполяция, различие состоит только в определении вспомогательной функции коэффициентов сплайна.

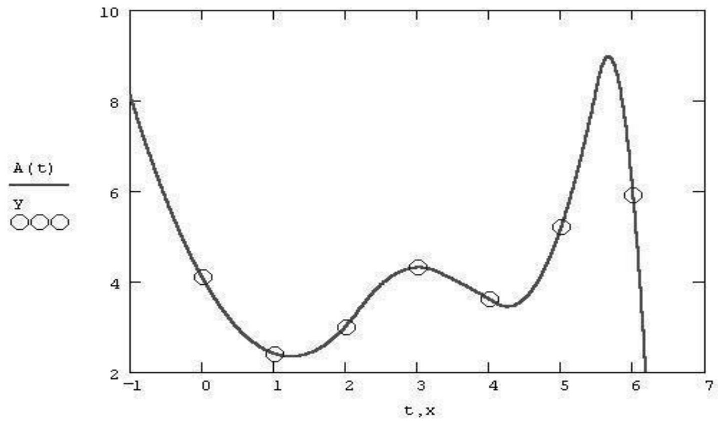


Рис.4 Интерполяция В-сплайнами

Наиболее приемлем способ, при котором кривая описывается многочленом 3-й степени:

x(t) =	A11 t3	+	A12 t2	+	A_{13} t	+	A_{14};
y(t) =	A21t3	+	A22t2	+	A_{23}t	+	A_{24};
z(t) =	A31 t3	+	A32t2	+	A_{33}t	+	A34,

0<t<1 (переход от точки i к i+1 точке)

Кубические уравнения выбраны потому, что для

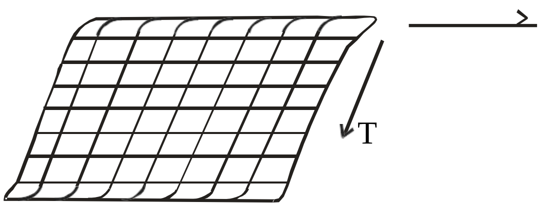
сегментов произвольной кривой:

- -не существует представление более низкого порядка, которая обеспечивает сопряжение на границах связи
- -при более высоком порядке, появляются осцилляции и волнистость. Из ряда способов описания бикубических кривых (метод Эрмита, метод Безье и т.п.) наиболее применяем метод В-сплайнов, для которого характерно несовпадение кривой с аппроксимируемыми точками что, однако гарантирует равенство 1-й и 2-й производных при стыковке сегментов. В-сплайн описывается следующей формулой:

  x(t)=TMsGsx – обобщенная форма описания кривой для всех методов где: $T=[t^3,t^2,t,1]$ – параметр, определяющий переход от точки Pi к Pi +1 M – матрица обобщения для В – сплайна.

Для трехмерных поверхностей определяется два параметра S и T, изменение которых дают координату любой точки на поверхности.

S



Фиксация одной переменной позволяет перейти к построению кривой на поверхности. Общая форма записи (для направления x):

$x(S,t)=SCxT^T$

где: Cx – коэффициенты кубического многочлена (для определения коэффициентов y,z соответственно Cy,Cz)

Для B-сплайна:

$X(S,t)=SMsPxMs^TT^T$

$Y(S,t)=SMsPyMs^TT^T$

$Z(S,t)=SMsPzMs^TT^T$

P – управляющие точки (16 точек) (4 по S и 4 по T).

Кривые и поверхности NURBS

Рассмотрим NURBS-кривые, поскольку это дает базовое понимание Всплайнов, а затем обобщим их на поверхности.

Неоднородный рациональный B-сплайн, NURBS (Non-uniform rational B-spline) - математическая форма, применяемая в компьютерной графике для генерации и представления кривых и поверхностей. В общем случае B-сплайн состоит из нескольких сплайновых сегментов, каждый из которых определен как набор управляющих точек. Поэтому коэффициенты многочлена будут зависеть только от управляющих точек на рассматриваемом сегменте кривой.

Этот эффект называется локальным управлением, поскольку перемещение управляющей точки будет влиять не на все сегменты кривой. На рисунке 5 показано, как управляющие точки влияют на форму кривой.

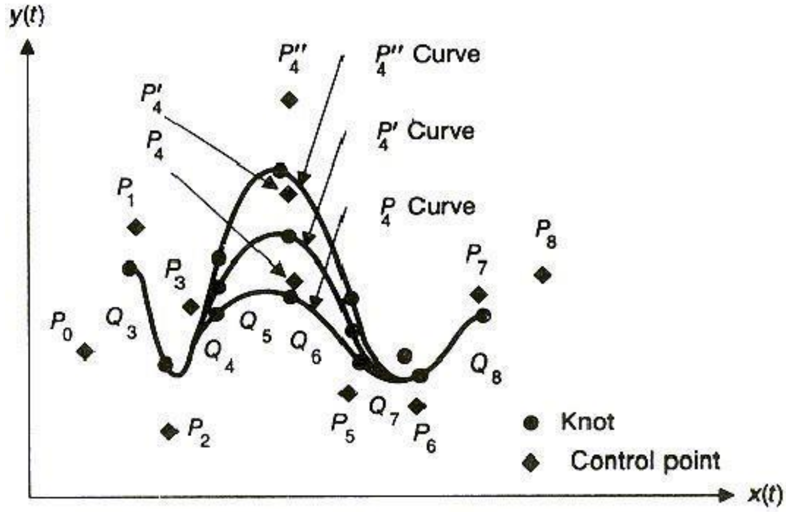


Рис. 5 В-сплайн с управляющей точкой Р4 в нескольких положениях

В-сплайн интерполирует набор из p+1 управляющей точки $\{P_o, P_1, ..., P_p\}, p \geq n$, и состоит из p-(n-1) сегментов кривой $\{Q_n, Q_{n+1}, ..., Q_p\}$. Кроме того, мы можем определить общий параметр t, нежели отдельный для каждого сегмента в интервале от 0 до 1. Таким образом, для каждого сегмента кривой Q_i t будет принадлежать интервалу $[t_i, t_{i+1}], n \leq i \leq p$. Более того, на каждый сегмент Q_i будет влиять ровно n управляющих точек от P_{i-n} до P_i.

Для каждого i >= n существует узел между Q_i и Q_{i+1} для значения t_i параметра t. Для В-сплайна существует p-n-2 узлов. Отсюда исходит понятие однородности: если узлы равномерно распределены на интервале от 0 до 1, т.е. $\forall i \in [n, p], t_{i+1} - t_i = t_{i+2} - t_{i+1}$, то говорят, что В-сплайн равномерный. В противном случае – неравномерный. Стоит также обратить внимание на факт, что эти определения касаются узлов, возрастающих по значению, т.е. $\forall i \in [n, p], t_i \leq t_{i+1}$.

Теперь предположим, что координаты (x,
y, z) точки кривой представлены в виде рациональной дроби. В этом случае говорят, что В-сплайн рациональный, иначе – нерациональный:

Подводя итог, можно указать на существование 4 типов В-сплайнов:

- равномерные нерациональные;

$$x = \frac{X(t)}{W(t)}, y = \frac{Y(t)}{W(t)}, z = \frac{Z(t)}{W(t)}$$

- неравномерные нерациональные;
- равномерные рациональные;
- неравномерные рациональные.

Последний тип и представляет собой NURBS как наиболее общий случай B-сплайнов.

Ход работы.

Формулы для получения б-сплайнов:

Базисная функция

$$B_{i,0}(x) := \begin{cases} 1 & \text{if} \quad t_i \le x < t_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$B_{i,k}(x) := \frac{x - t_i}{t_{i+k} - t_i} B_{i,k-1}(x) + \frac{t_{i+k+1} - x}{t_{i+k+1} - t_{i+1}} B_{i+1,k-1}(x).$$ [7]

Реализация показана в листинге 1.

```
float MyWidget::B(float x, int n, int d){
    if(d == 0)
    {
        if(knots[n] <= x && x < knots[n+1])
        {
            return 1.0f;
        }
        return 0.0f;
    }
    float a = B(x,n,d-1);
    float b = B(x,n+1,d-1);
    float c = 0.0f, e = 0.0f;

    if(a != 0.0f)
    {
        c = (x - knots[n]) / (knots[n+d] - knots[n]);
    }
    if(b != 0)
    {
        e = (knots[n+d+1] - x) / (knots[n+d+1] - knots[n+1]);
    }
    return (a*c + b*e);
}
```

Листинг 1 - Реализация базисной функции

t_i - i-й узел в узловом векторе. Узловой вектор имеет длину равную количеству контрольных точек + степень сплайна + 1. Пример открытого равномерного узлового вектора для степени 4

$$[0, 0, 0, 0, 1, 2, 3, 4, 4, 4, 4] \qquad (k = 4)$$

Реализация расчета узлового вектора представлена в листинге 2.

```
}

void MyWidget::CrKnotVector(){
    QVector <float> knots;

    for(int i = 0; i < d; i++)
    {
        knots.append(0.0f);
    }

    for(int i=0; i < points.length()-d+1; i++)
    {
        knots.append((float)i);
    }

    for(int i=0; i< d;i++)
    {
        knots.append((float)(points.length()-d));
    }

    this->knots=knots;
}
```

Листинг 2 - Расчет узлового вектора

Сам б-сплайн рассчитывается параметрически, что позволяет функции пересекаться, иметь несколько значений при одном x. Формула вычисления параметра

$$X(t) = \sum_i x_i B_{i,n}(t),$$
$$Y(t) = \sum_i y_i B_{i,n}(t),$$

Реализация показана в листинге 3.

```
void MyWidget::NURBspline(){
    if(d >= points.length())
    {
        return;
    }
    glColor3d(0,1,1);
    glBegin(GL_LINE_STRIP);

    CrKnotVector();

    float xmin=knots[0];
    float xmax=knots.last();

    float delta = xmax - xmin;
    float step = delta/300;

    for(float t = xmin; t < xmax; t += step){
        float x = 0.0f, y = 0.0f;

        for(int i = 0; i < points.length(); i++){
            x+=B(t,i,d) * points[i].x() * g_Weight[i];
            y+=B(t,i,d) * points[i].y() * g_Weight[i];
        }
        glVertex2f(x,y);
    }
    glVertex2f(points.last().x(),points.last().y());
    glEnd();
}
```

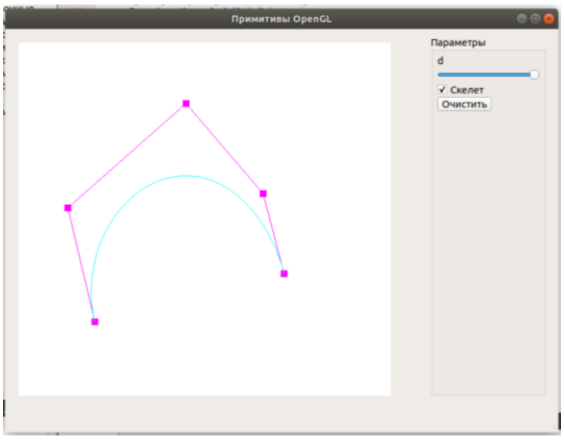
Листинг 3 - Реализация отрисовки сплайна

Степень сплайна и количество контрольных точек выбирается пользователем. Нажатие правой кнопки мыши добавляет контрольную точку, также их можно перемещать удержанием левой кнопки мыши.

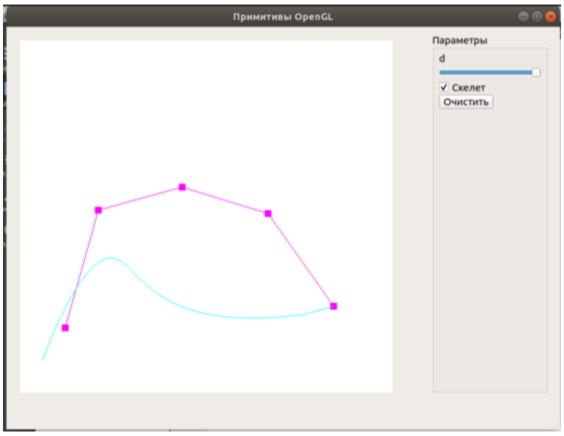
Свойства кривой:

Кривая обладает непрерывностью в точках стыковки сегментов, кроме того непрерывны первые две производные. Кривая обладает гладкостью. У каждой точки прямой разный вес. Приведем примеры работы программы с различными значения весов.

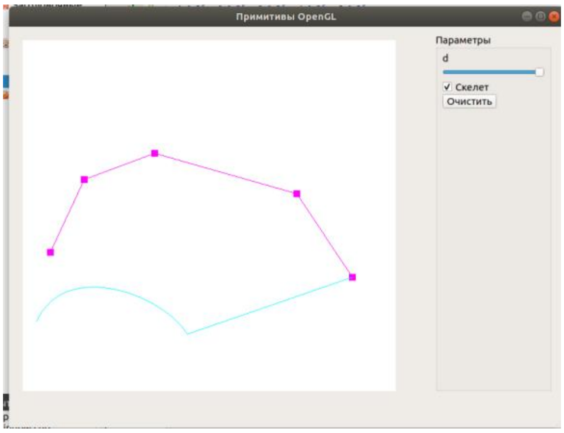
Веса 1 – 1 – 1 – 1 – 1



Веса 0.5 - 1 – 0.8 – 0.3 – 0.9



Веса 0.5 – 0.5 – 0.5 – 0.5 – 0.5



Выводы.

В итоге лабораторной работы разработа приложение отрисовки NURBсплайнов, поддерживающее интерактивное взаимодействие с пользователем, улучшены навыки владения с OpenGL.

ПРИЛОЖЕНИЕ А

Исходный код

```
#include "mywidget.h"

//PIPµCЃP°
float p1=0.5f, p2=0.5f, p3=0.5f, p4=0.5f, p5=0.5f;
float g_Weight[] = {p1, p2, p3, p4, p5};
float q_Size;

MyWidget::MyWidget(QWidget *parent) // PєPsPSCЃC,CЂCѓPєC,PsCЂ
    : QGLWidget(parent)
{
    selected = NULL;
    start = QPointF(0.0,0.0);
    d = 4;
    bones = true;
}


void MyWidget::initializeGL()
{
    qglClearColor(Qt::white); // P·P°PїPsP»PSCЏPµPj CЌPєCЂP°PS
P±PµP»C‹Pj C†PIPµC,PsPj
    glShadeModel(GL_SMOOTH);
}


void MyWidget::resizeGL(int nWidth, int nHeight)
{
    glViewport(0, 0, nWidth, nHeight);
    glMatrixMode(GL_PROJECTION);
    glOrtho(0, nWidth, 0, nHeight, -10.0, 1.0);
}


void MyWidget::paintGL() // CЂPёCЃPsPIP°PSPёPµ
{
    glClear ( GL_COLOR_BUFFER_BIT );
    glColor3f(1.0,0.0,1.0);
    glPointSize(10.0f);

    if(bones){
        glBegin(GL_LINE_STRIP);
        for(auto& p : points){
            glVertex2f(p.x(),p.y());
        }
        glEnd();
        glBegin(GL_POINTS);

        for(auto& p : points){
            glVertex2f(p.x(),p.y());
        }
        glEnd();
    }
    NURBspline();
}
```

```
void MyWidget::mousePressEvent(QMouseEvent *event){

    if(event->button()==Qt::RightButton)
    {
        start=QPointF(event->x(),(-1)*event->y()+520);
        qDebug()<<event->button();
        points.append(start);
        updateGL();
    }
    if(event->button()==Qt::LeftButton)
    {
        qDebug()<<event->pos();
        for(auto& p : points)
        {
            if(QLineF(QPointF(event->x(),(-1)*event-
>y()+520),p).length()<=20.0f)
            {
                selected=&p;
                qDebug()<<selected;
            }
        }
    }
}


void MyWidget::mouseReleaseEvent(QMouseEvent *event){
    selected=NULL;
}

void MyWidget::mouseMoveEvent(QMouseEvent *event){
    if(selected){
        start.setX(event->x()-start.x());
        start.setY(event->y()-start.y());
        selected->setX(event->x());
        selected->setY((-1)*event->y()+520);
        start=event->pos();
        updateGL();
    }
}

void MyWidget::setd(int d){
    this->d = d;
}

void MyWidget::CrKnotVector(){
    QVector <float> knots;

    for(int i = 0; i < d; i++)
    {
        knots.append(0.0f);
    }

    for(int i=0; i < points.length()-d+1; i++)
    {
        knots.append((float)i);
    }
```

```
    for(int i=0; i< d;i++)
    {
        knots.append((float)(points.length()-d));
    }

    this->knots=knots;
}

float MyWidget::B(float x, int n, int d){
    if(d == 0)
    {
        if(knots[n] <= x && x < knots[n+1])
        {
            return 1.0f;
        }
        return 0.0f;
    }
    float a = B(x,n,d-1);
    float b = B(x,n+1,d-1);
    float c = 0.0f, e = 0.0f;

    if(a != 0.0f)
    {
        c = (x - knots[n]) / (knots[n+d] - knots[n]);
    }
    if(b != 0)
    {
        e = (knots[n+d+1] - x) / (knots[n+d+1] - knots[n+1]);
    }
    return (a*c + b*e);
}


void MyWidget::NURBspline(){
    if(d >= points.length())
    {
        return;
    }
    glColor3d(0,1,1);
    glBegin(GL_LINE_STRIP);

    CrKnotVector();

    float xmin=knots[0];
    float xmax=knots.last();

    float delta = xmax - xmin;
    float step = delta/300;

    for(float t = xmin; t < xmax; t += step){
        float x = 0.0f, y = 0.0f;

        for(int i = 0; i < points.length(); i++){
            x+=B(t,i,d) * points[i].x() * g_Weight[i];
            y+=B(t,i,d) * points[i].y() * g_Weight[i];
        }
        glVertex2f(x,y);
    }
```

```
    glVertex2f(points.last().x(),points.last().y());
    glEnd();
}

void MyWidget::setb(bool f){
    bones = f;
}

void MyWidget::clear(){
    this->points.clear();
}
```