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

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Компьютерная графика»
Тема: Кубические сплайны

Студенты гр. 8383 _____ Кормщикова А. О.
Бессуднов Г. И.

Преподаватель _____ Герасимова Т. В.

Санкт-Петербург
2021

Цель работы

Реализовать интерактивное приложение, отображающее заданные полиномиальные кривые. При этом для кривых, состоящих из нескольких сегментов, должно быть обеспечено свойство непрерывной кривизны. Программа должна позволять пользователю: интерактивно менять положение контрольных точек, касательных, натяжений.

Вариант 16

16. NURB-сплайн (n = 5, k = 3) с равномерным узловым вектором и изменяемыми весами точек.

Основные теоретические положения

Сплайны - это гладкие (имеющие несколько непрерывных производных) кусочно-полиномиальные функции, которые могут быть использованы для представления функций, заданных большим количеством значений и для которых неприменима аппроксимация одним полиномом. Так как сплайны гладки, экономичны и легки в работе, они используются при построении произвольных функций для:

- моделирования кривых;
- аппроксимации данных с помощью кривых;
- выполнения функциональных аппроксимаций;
- решения функциональных уравнений.

Важным их свойством является простота вычислений. На практике часто используют сплайны вида полиномов третьей степени. С их помощью довольно удобно проводить кривые, которые интуитивно соответствуют человеческому субъективному понятию гладкости.

Кривые и поверхности NURBS

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.

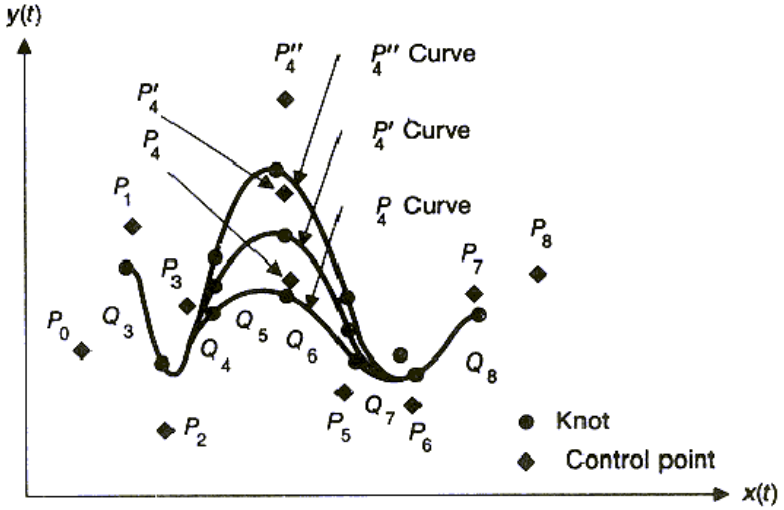


Рис. 5 B-сплайн с управляющей точкой P4 в нескольких положениях

B-сплайн интерполирует набор из p+1 управляющей точки $\{P_o, P_1, \ldots, P_p\}, p \geq n$, и состоит из p-(n-1) сегментов кривой $\{Q_n, Q_{n+1}, \ldots, Q_p\}$. 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 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 ti параметра t. Для B-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 B-сплайн равномерный. В противном случае – неравномерный. Стоит также обратить внимание на факт,

что эти определения касаются узлов, возрастающих по значению, т.е. $\forall i \in [n, p], t_i \le t_{i+1}$.

Теперь предположим, что координаты (x, y, z) точки кривой представлены в виде рациональной дроби.

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

В этом случае говорят, что B-сплайн рациональный, иначе – нерациональный:

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

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

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

Выполнение работы

Работа была выполнена в среде Visual Studio.

В папки библиотек и заголовков Visual Studio, а также в системную папку были добавлены соответствующие файлы. В проект были подключены: opengl32.lib, glut32.lib, glu32.lib.

Были созданы глобальные переменные с начальными условиями:

KnotVector – массив узлов с равномерным распределением, не изменяется в ходе работы программы.

WeigthVector – массив весов точек.

Ctrlpoints – массив контрольных точек.

P – степень NURBS

N – количество контрольных сегментов.

А также вспомогательные переменные для изменения координат и веса точки.

Была написана базисная функция basisFunc, функция определена рекурсивно формулами Кокса-де Бура:

$$B_{i,n}(t):\quad B_{i,0}(t)=\begin{cases}1, & t_i \le t < t_{i+1}\\ 0, & \text{иначе}\end{cases}\qquad \forall k>0,\ B_{i,k}(t)=\frac{t-t_i}{t_{i+k}-t_i}B_{i,k-1}(t)+\frac{t_{i+k+1}-t}{t_{i+k+1}-t_{i+1}}B_{i+1,k-1}(t)$$

Была написана функция NURBS сплайна NURBSSpline, которая возвращает координаты точки в момент времени t. Сплайн рассчитывается по следующей формуле :

$$P(t)=\frac{\sum_{i=0}^{p} B_{i,n}(t)P_i w_i}{\sum_{i=0}^{p} B_{i,n}(t)w_i}$$

Bi,n – базисная функция, wi – вес, ассоциированный с управляющей точкой Pi , $1 \le k \le n$, n=p-1 – степень полиномов, p – порядок B-сплайна и число сегментов поддержки, p+1 – число узлов поддержки, принято, что 0/0=0.

Была написана функция Display, которая управляет всем выводом на экран: зеленым выводятся контрольные точки с размером 5,а также линии между контрольными точками.

Желтым цветом выводится сплайн: параметр quality = 500 отвечает за то, сколько шагов будет просчитываться сплайн. В каждый момент времени (i/quality, где i изменяется от 0 до quality) вызывается функция NURBSSpline. По координатам, которые она возвратила, строится точка.

Была написана функция Reshape, которая вызывается при изменении размеров окна.

Была написана функция SearchPiont, которая находит существующую точку в небольшой области клика мышки и делает ее «активной».

Была написана функция movePoint, которая меняет координаты активной точки, в конце функции вызывается функция перерисовки.

Была написана функция changeWeight, которая изменяет вес активной точки.

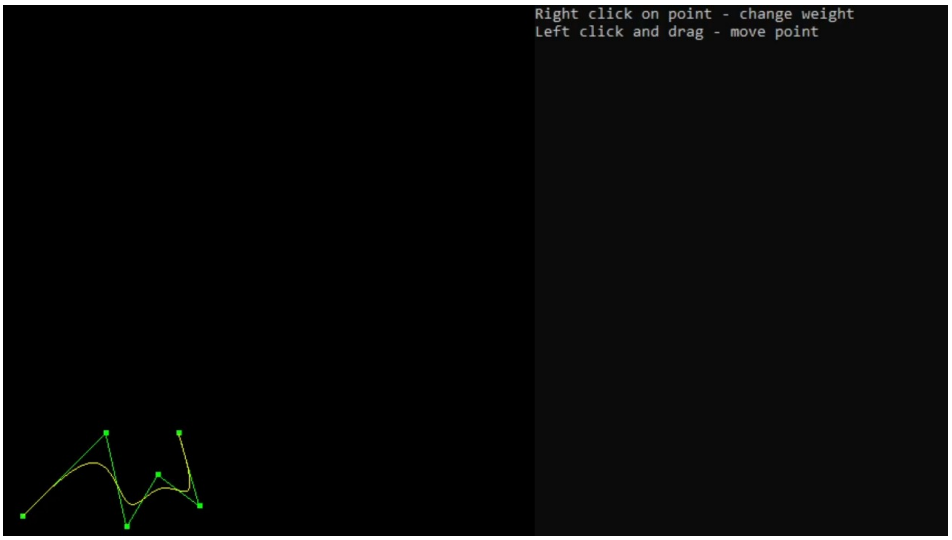
Была написана функция MouseButtonHandler, которая нажатия мыши: с помощью левой кнопки мыши можно выбрать и перетащить точку, щелчок правой кнопкой мыши по точке позволяет изменить ее вес. Вес запрашивается через консоль: значения от 0 до 10.

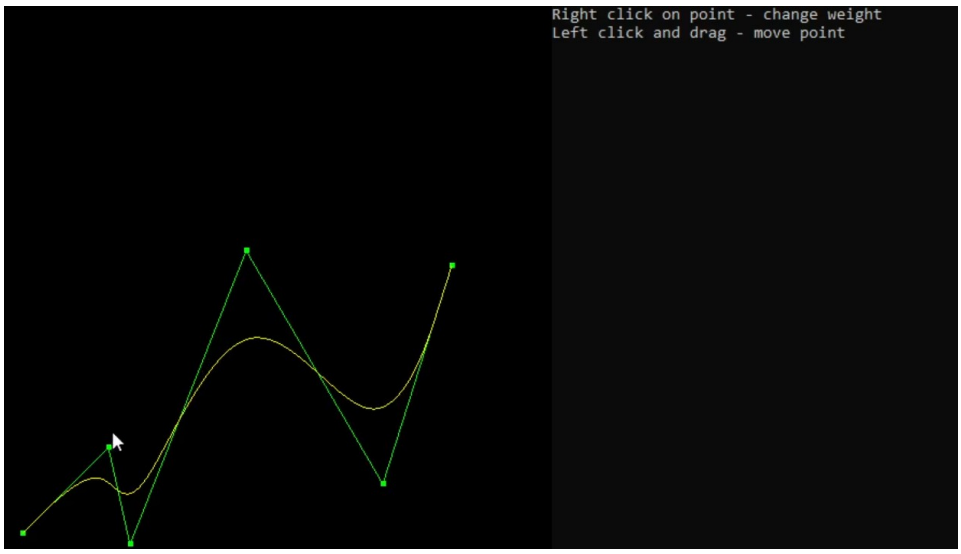
Была написана функция MouseMotionHandler, которая отслеживает активное перемещение мыши, функция посылает новые координаты точки при перетаскивании в функцию изменения координат.

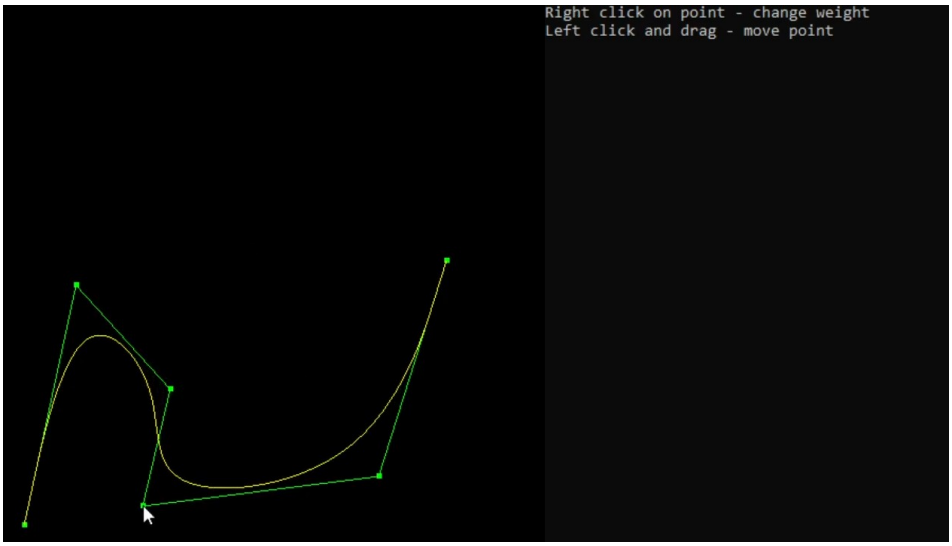
Была написана функция Keyboard, которая обрабатывает сообщения от клавиатуры. С помощью кнопки Esc можно закрыть приложение.

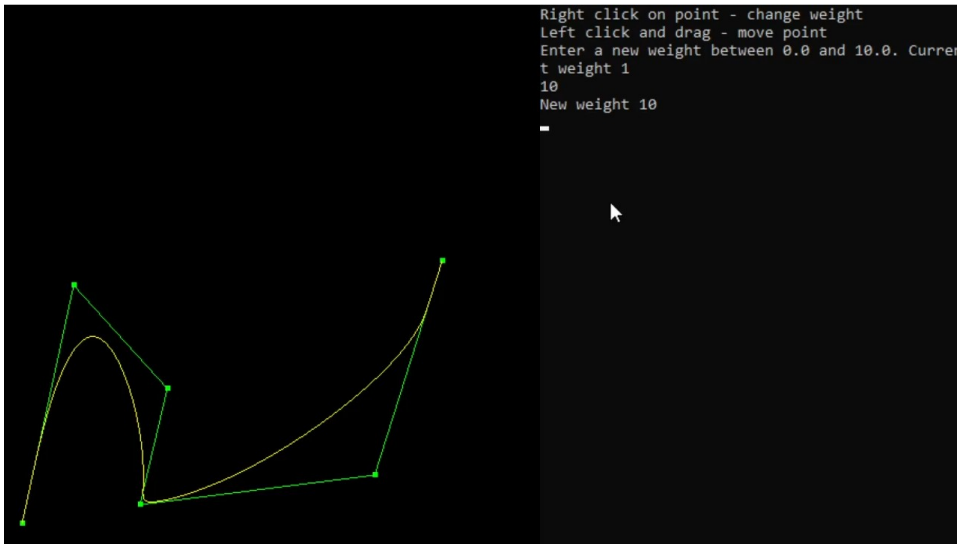
Код представлен в приложении А.

Тестирование.









Выводы

Во время лабораторной была разработана программа в среде Visual Studio, которая реализует NURB-сплайн (n = 5, k = 3) с равномерным узловым

вектором и изменяемыми весами точек. Программа также позволяет пользователю интерактивно менять положение контрольных точек

ПРИЛОЖЕНИЕ А

КОД ПРОГРАММЫ

```
#INCLUDE <STDLIB.H>
#INCLUDE <GLUT.H>
#INCLUDE <CMATH>
#INCLUDE <CTIME>
#INCLUDE <IOSTREAM>

//РАЗМЕРЫ ОКНА
GLINT WIDTH = 512;
GLINT HEIGHT = 512;

//УЗЛЫ С РАВНОМЕРНЫМ РАСПРЕДЕЛЕНИЕМ.
FLOAT KNOTVECTOR[10] = { 0.0F, 1.0F / 9.0F, 2.0F / 9.0F, 3.0F / 9.0F, 4.0F /
9.0F, 5.0F / 9.0F, 6.0F / 9.0F, 7.0F / 9.0F, 8.0F / 9.0F, 1.0F };
//ВЕСА ДЛЯ ТОЧЕК
FLOAT WEIGTHVECTOR[6] = { 1.0F, 1.0F, 1.0F, 1.0F, 1.0F, 1.0F };

//СТАРТОВЫЕ ПАРАМЕТРЫ
INT P = 3; //СТЕПЕНЬ NURBS
INT N = 5; //КОЛИЧЕСТВО КОНТРОЛЬНЫХ СЕГМЕНТОВ

//МАССИВ КОНТРОЛЬНЫХ ТОЧЕК
GLFLOAT CTRLPOINTS[6][2] = { {20.0F, 20.0F}, {100.0F, 100.0F},
               {120.0F, 10.0F}, {150.0F, 60.0F},
               {190.0F, 30.0F}, {170.0F, 100.0F} };

//ВСПОМОГАТЕЛЬНЫЕ ПЕРЕМЕННЫЕ ДЛЯ ИЗМЕНЕННИЙ КООРДИНАТ/ВЕСА ТОЧКИ
GLFLOAT* ACTIVEPOINT = NULL;
INT INDEXACTIVEPOINT = 0;
BOOL MOVEMODE = FALSE;

//БАЗИСНАЯ ФУНКЦИЯ
FLOAT BASISFUNC(INT K, INT P, FLOAT T) {
    //УСЛОВИЕ ВЫХОДА ИЗ ФУНКЦИИ
    IF (P == 0) {
        //УСЛИ ПОПАЛИ В ПРОМЕЖУТОК МЕЖДУ К И К+1 УЗЛОМ, ТО ВОЗВРАЩАЕМ 1,
ЕСЛИ НЕТ, ТО 0.
        IF (KNOTVECTOR[K] <= T && T < KNOTVECTOR[K + 1]) {
            RETURN 1.0F;
        }
        ELSE {
            RETURN 0.0F;
        }
    }


    FLOAT F = (T - KNOTVECTOR[K]) / (KNOTVECTOR[K + P] - KNOTVECTOR[K]);
    FLOAT G = (KNOTVECTOR[K + P + 1] - T) / (KNOTVECTOR[K + P + 1] -
KNOTVECTOR[K + 1]);

    //РЕККУРЕНТНЫЙ ВЫЗОВ ФУНКЦИИ
```

```
    RETURN F * BASISFUNC(K, P - 1, T) + G * BASISFUNC(K + 1, P - 1, T);
}

//ФУНКЦИЯ ДЛЯ ПОСТРОЕНИЯ NURBS СПЛАЙНА
//ВОЗВРАЩАЕТ КООРДИНАТЫ ТОЧКИ В МОМЕНТ ВРЕМЕНИ T
FLOAT* NURBSSPLINE(FLOAT T) {
    FLOAT X = 0.0F;
    FLOAT Y = 0.0F;
    FLOAT BASISWEIGTHSUM = 0.0F;
    FOR (INT I = 0; I <= N; I++) {
        FLOAT COEFF = BASISFUNC(I, P, T) * WEIGTHVECTOR[I];
        BASISWEIGTHSUM += COEFF;
        X += CTRLPOINTS[I][0] * COEFF;
        Y += CTRLPOINTS[I][1] * COEFF;
    }
    FLOAT ANSWER[2] = { X / BASISWEIGTHSUM, Y / BASISWEIGTHSUM };
    RETURN ANSWER;
}


VOID DISPLAY(VOID) {

    GLCLEARCOLOR(0.0F, 0.0F, 0.0F, 1.0F);
    GLCLEAR(GL_COLOR_BUFFER_BIT);
    //КОНТРОЛЬНЫЕ ТОЧКИ
    GLCOLOR3F(0.0, 1.0, 0.0);
    GLPOINTSIZE(5.0);
    GLBEGIN(GL_POINTS);
    FOR (INT I = 0; I <= N; I++)
        GLVERTEX2FV(CTRLPOINTS[I]);
    GLEND();

    //ЛИНИИ МЕЖДУ КОНТРОЛЬНЫМИ ТОЧКАМИ
    GLLINEWIDTH(1.0F);
    GLBEGIN(GL_LINE_STRIP);
    FOR (INT I = 0; I <= N; I++)
        GLVERTEX2FV(CTRLPOINTS[I]);
    GLEND();

    //NURBS СПЛАЙН

    GLBEGIN(GL_LINE_STRIP);
    INT QUALITY = 500;
    FOR (INT I = 0; I <= QUALITY; I++) {
        //СЧИТАЕМ СПЛАЙН И ДОБАВЛЯЕМ К ТОЧКАМ
        GLCOLOR3F(1.0F, 1.0F, 0.0F);
        GLVERTEX2FV(NURBSSPLINE((FLOAT)I / (FLOAT)QUALITY));
    }
    GLEND();
    GLFLUSH();
}

//ФУНКЦИЯ ИЗМЕНЕНИЯ РАЗМЕРОВ ОКНА
VOID RESHAPE(GLINT W, GLINT H) {
    WIDTH = W;
```

```
    HEIGHT = H;

    GLVIEWPORT(0, 0, W, H);

    GLMATRIXMODE(GL_PROJECTION);
    GLLOADIDENTITY();
    GLORTHO(0, W, 0, H, -1.0F, 1.0F);

    GLMATRIXMODE(GL_MODELVIEW);
    GLLOADIDENTITY();
}

//ФУНКЦИЯ ПОИСКА ТОЧКИ В ОБЛАСТИ КЛИКА МЫШИ
VOID SEARCHPIONT(INT X, INT Y) {
    GLFLOAT* NEARESTPOINT = NULL;
    INT DISTANSE = 10;
    INT INDEXNP = 0;
    FOR (INT I = 0; I < N + 1; I++) {
        INT TMPDISTANCE = ABS(CTRLPOINTS[I][0] - X) + ABS(CTRLPOINTS[I][1] -
Y);
      //  STD::COUT << I << "\T" << TMPDISTANCE << "\T" << CTRLPOINTS[I][0]
<< "\T" << CTRLPOINTS[I][1] << STD::ENDL;
        IF (TMPDISTANCE < DISTANSE) {
            NEARESTPOINT = CTRLPOINTS[I];
            INDEXNP = I;
        }
    }
    IF (NEARESTPOINT != NULL) {
       // STD::COUT << NEARESTPOINT[0] <<"\T"<<NEARESTPOINT[1]<<"\
T"<<INDEXNP<< STD::ENDL;
        ACTIVEPOINT = NEARESTPOINT;
        INDEXACTIVEPOINT = INDEXNP;
    }
}


//ФУНКЦИЯ ПЕРЕМЕЩЕНИЯ АКТИВНОЙ ТОЧКИ
VOID MOVEPOINT(INT X, INT Y) {
    IF (ACTIVEPOINT == NULL) {
        RETURN;
    }
    ACTIVEPOINT[0] = X;
    ACTIVEPOINT[1] = Y;
    GLUTPOSTREDISPLAY();
}
//ФУНКЦИЯ ИЗМЕНЕНИЯ АКТИВНОЙ ТОЧКИ
VOID CHANGEWEIGHT() {
    IF (ACTIVEPOINT == NULL) {
        RETURN;
    }
    FLOAT TMPW;
    STD::COUT << "ENTER A NEW WEIGHT BETWEEN 0.0 AND 10.0. CURRENT WEIGHT "
<< WEIGTHVECTOR[INDEXACTIVEPOINT] << STD::ENDL;
    STD::CIN >> TMPW;
    IF (TMPW > 10.0) {
```

```
        TMPW = 10.0;
    }
    ELSE IF (TMPW < 0.0) {
        TMPW = 0.0;
    }
    STD::COUT << "NEW WEIGHT " << TMPW << STD::ENDL;
    WEIGTHVECTOR[INDEXACTIVEPOINT] = TMPW;
    GLUTPOSTREDISPLAY();
}
//ФУНКЦИЯ ОБРАБОТКИ НАЖАТИЙ МЫШИ
VOID MOUSEBUTTONHANDLER(INT BUTTON, INT STATE, INT X, INT Y) {
    IF (BUTTON == GLUT_LEFT_BUTTON) {
       // STD::COUT << "LEFT\T" << X << "\T" << HEIGHT - Y << STD::ENDL;
        IF (STATE == GLUT_DOWN) {
            SEARCHPIONT(X, HEIGHT - Y);
            IF (ACTIVEPOINT != NULL) {
                MOVEMODE = TRUE;
            }
        }
        IF (STATE == GLUT_UP) {
            ACTIVEPOINT = NULL;
            MOVEMODE = FALSE;
        }
    }

    IF (BUTTON == GLUT_RIGHT_BUTTON) {
     //   STD::COUT << "RIGHT\T" << X << "\T" << HEIGHT - Y << STD::ENDL;
        IF (STATE == GLUT_DOWN) {
            SEARCHPIONT(X, HEIGHT - Y);
            CHANGEWEIGHT();
        }
    }
}

//ФУНКЦИЯ ОТСЛЕЖИВАНИЕ АКТИВНОГО ПЕРЕМЕЩЕНИЯ МЫШИ
VOID MOUSEMOTIONHANDLER(INT X, INT Y) {
    IF (MOVEMODE == TRUE) {
        IF (X < 0) {
            X = 0;
        }
        ELSE IF (X > WIDTH) {
            X = WIDTH;
        }
        IF (Y < 0) {
            Y = 0;
        }
        ELSE IF (Y > HEIGHT) {
            Y = HEIGHT;
        }
        MOVEPOINT(X, HEIGHT - Y);
    }
}

VOID KEYBOARD(UNSIGNED CHAR KEY, INT X, INT Y) {
    IF (KEY == '\033') {
```

```
        EXIT(0);
    }
}

INT MAIN(INT ARGC, CHAR* ARGV[]) {
    STD::COUT << "RIGHT CLICK ON POINT - CHANGE WEIGHT\NLEFT CLICK AND DRAG
- MOVE POINT" << STD::ENDL;
    GLUTINIT(&ARGC, ARGV);
    GLUTINITDISPLAYMODE(GLUT_RGB);
    GLUTINITWINDOWSIZE(WIDTH, HEIGHT);
    GLUTCREATEWINDOW("LAB03");

    GLMAP1F(GL_MAP1_VERTEX_3, 0.0, 1.0, 3, 4, &CTRLPOINTS[0][0]);
    GLENABLE(GL_MAP1_VERTEX_3);
    GLUTDISPLAYFUNC(DISPLAY);
    GLUTRESHAPEFUNC(RESHAPE);
    GLUTKEYBOARDFUNC(KEYBOARD);
    GLUTMOUSEFUNC(MOUSEBUTTONHANDLER);
    GLUTMOTIONFUNC(MOUSEMOTIONHANDLER);
    GLUTMAINLOOP();
}
```