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

# ОТЧЕТ

**по лабораторным работам №3**
**по дисциплине «Компьютерная графика»**
**Тема: Кубические сплайны**

Студентка гр. 1304 _____ Чернякова В.А.

Студентка гр. 1304 _____ Ярусова Т.В.

Преподаватель _____ Герасимова Т.В.

Санкт-Петербург
2024

**Цель работы.**

Реализовать интерактивное приложение, отображающее заданные полиномиальные кривые.

**Задание.**

Вариант 7. NURB-сплайн (n = 5, k = 4) с равномерным узловым вектором и изменяемыми весами точек.

**Теоретические положения.**

Неоднородный рациональный *B*-сплайн, *NURBS* (*Non-uniform rational B-spline*) – математическая форма, применяемая в компьютерной графике для генерации и представления кривых и поверхностей.

Разработка *NURBS* началась в 1950-х годах инженерами, которым требовалось математически точное представление поверхностей произвольной формы (таких как корпуса кораблей, самолётов, космических аппаратов и автомобилей) с возможностью точного копирования и воспроизведения всякий раз, когда это нужно. До появления представлений такого рода проектировщик создавал единичную физическую (материальную) модель, которая и служила эталоном.

Разберем плюсы и минусы *NURBS*-сплайнов.

Плюсы:

1. **Гибкость**: *NURBS*-сплайны обеспечивают большую гибкость при создании различных форм и поверхностей. Они могут быть использованы для моделирования разнообразных объектов, от автомобилей до аэрокосмических конструкций.
2. **Плавность**: Кривые и поверхности, созданные с использованием *NURBS*-сплайнов, обычно очень плавные и естественные, что делает их идеальными для моделирования органических объектов и форм.

3. **Математическая точность**: *NURBS*-сплайны определяются математически, что позволяет точно контролировать их форму и свойства. Это обеспечивает высокую степень предсказуемости и точности при моделировании.

4. **Масштабируемость**: *NURBS*-сплайны хорошо масштабируются без потери качества, что делает их удобными для использования в различных масштабах, от мелкой детализации до крупномасштабных моделей.

5. **Интерактивность**: Многие программы для моделирования предоставляют интерактивные инструменты для работы с *NURBS*-сплайнами, что облегчает создание и редактирование кривых и поверхностей.

Минусы:

1. **Сложность**: Использование *NURBS*-сплайнов может потребовать некоторого времени и опыта для освоения. Некоторые аспекты их работы, такие как настройка весов узлов, могут быть сложными для понимания.

2. **Ограничения формы**: Хотя *NURBS*-сплайны предоставляют широкий диапазон форм и поверхностей, иногда может быть сложно достичь определенных форм, особенно если они требуют большой детализации или необычных геометрических характеристик.

3. **Вычислительная сложность**: Вычисление *NURBS*-сплайнов может быть вычислительно затратным, особенно при работе с крупномасштабными моделями или сложными кривыми и поверхностями.

4. **Неэффективность для некоторых типов геометрии**: В некоторых случаях другие методы моделирования могут быть более эффективными для создания определенных типов геометрии.

5. **Трудности при анимации**: В некоторых случаях анимация объектов, созданных с использованием *NURBS*-сплайнов, может быть сложной из-за специфических характеристик кривых и поверхностей.

*NURBS*-сплайны широко применяются в различных областях компьютерной графики и дизайна. Некоторые из наиболее распространенных областей применения включают:

- *3D* моделирование и анимация: Программы для трехмерного моделирования и анимации, такие как *Autodesk Maya, Blender, 3ds Max, Cinema 4D,* используют *NURBS*-сплайны для создания плавных и реалистичных кривых и поверхностей, таких как каракули, лица, автомобили и другие объекты.
- *CAD*: В инженерном дизайне и архитектурном проектировании *NURBS*-сплайны применяются для создания точных и гладких поверхностей, таких как кривые кузовов автомобилей, корпуса самолетов, архитектурных форм и многое другое.
- Производство видеоигр: В разработке видеоигр *NURBS*-сплайны могут использоваться для создания анимированных персонажей, предметов окружения и анимационных эффектов.
- Дизайн судов и аэрокосмической техники: В проектировании судов, самолетов и космических аппаратов *NURBS*-сплайны могут использоваться для моделирования корпусов, крыльев, фюзеляжей и других частей судов.
- Медицинское моделирование и визуализация: В медицинском моделировании *NURBS*-сплайны могут использоваться для создания анатомически точных моделей органов и тканей для образовательных целей, хирургического планирования и визуализации.
- Проектирование и производство промышленных деталей и компонентов: В промышленном дизайне и производстве *NURBS*-сплайны используются для моделирования и создания прецизионных деталей, таких как литьевые формы, прототипы и инструменты.

Это лишь несколько примеров областей, где *NURBS*-сплайны активно применяются в компьютерной графике и дизайне.

Теперь рассмотрим построение *NURBS*-сплайна. Для этого, в первую очередь, необходимо задать следующие параметры:

- узловой вектор $T = [T_0 \dots T_m]$ – данные узлы определяют, где и как контрольные точки будут влиять на сплайн.
- массив контрольных точек $P = [P_0 \dots P_n]$ – контрольные точки задают форму сплайна, ограничивая его.

- массив весов контрольных точек $W = [W_0 \dots W_n]$ – веса позволяют контролировать вклад каждой контрольной точки в итоговый сплайн.

Для расчёта степени сплайна используется формула*:* $k = m - n - 1$

Следующий шаг в построении *NURBS*-сплайна – определение базисных функций $N_{i,j}$ , где *j* – степень базисной функции, а *i* – её номер, $j \in [0 \dots k]$ , $\forall j :$ $i \in [0 \dots n - j + k]$

Базисные функции *0* степени являются константными кусочно-заданными функциями следующего вида:

$$N_{i,0}(t) = \begin{cases} 1 \text{ , } t_i \le t \le t_{i+1} \\ 0 \text{ , иначе} \end{cases}$$

Базисные функции высших порядков можно построить рекурсивно:

$$N_{i,j}(t) = f_{i,j}(t) \cdot N_{i,j-1}(t) + g_{i+1,j}(t) \cdot N_{i+1,j-1}(t),$$

где $f_{i,j}(t) = \frac{t - T_i}{T_{i+j} - T_i}$ , а $g_{i,j}(t) = 1 - f_{i,j}(t) = \frac{T_{i+j} - t}{T_{i+j} - T_i}$

После построения базисных функций степени *k* (степень итогового сплайна), по следующей формуле можно получить функцию полученной кривой:

$$\mathrm{B}(t) = \frac{\sum_{i=1}^{n} N_{i,k}(t) * W_i * P_i}{\sum_{i=1}^{n} N_{i,k}(t) * W_i}$$

И перебирая параметр $t \in [T_0, T_m]$ можно получить все точки сплайна.

**Выполнение работы.**

В предложенном варианте лабораторной работы дано следующее:

*n = 5* (количество контрольных точек)

*k = 4* (степень полинома)

Количество значений узлового вектор *T = 10 = n + k + 1*.

Так как узловой вектор равномерный (по условию задания), то он будет выглядеть следующим образом:

*T = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]*

Остальные данные, такие как координаты контрольных точек и веса, будут задаваться в программном коде, так как для них нет ограничений.

Рассмотрим реализацию построения *NURBS*-сплайна на языке программирования *Python* с применением библиотек *PyQt5* и *OpenGL*.

Подключение графической библиотеки осуществляется с помощью виджета *QOpenGLWidget*.

Создан виджет *GLSence* наследуемый от данного класса, в котором с помощью переопределённых методов *initializeGL, resizeGL* и *paintGL* осуществляется соответственно подготовка кадра, переопределение размеров и отрисовка изображения.

В программе выполняется отрисовка 3-х основных компонент. Контрольные точки, они отрисовываются с помощью примитивы *GL_POINTS*. Линия через контрольные точки, она отрисовываются с помощью примитивы *GL_LINE_STRIP* через основные координаты контрольных точек, чтобы при построении *NURBS*-сплайна отследить влияние положения и веса контрольных точек на итоговый вид сплайна. *NURBS*-сплайн обрисовывается также с помощью примитивы *GL_LINE_STRIP* но уже по следующему принципу: создается одномерный массив значений равномерно распределенных на узловом векторе, точки, которые определяют сплайн, выбираются из такого массива значений путем поиска совпадений с посчитанным значением переменной *F*, в которой хранятся результаты расчеты *NURBS*-сплайна.

```
# Построение контрольных точек
glBegin(GL_POINTS)
for i, point in enumerate(self.P):
    if i == self.selected_point:
        glColor3dv((0, 1, 0))
    else:
        glColor3dv((1, 0, 0))
    glVertex2dv(point)
glEnd()

# Построение основных линий через контрольные точки
glBegin(GL_LINE_STRIP)
glColor3dv((0, 0, 1))
for points in self.P:
    glVertex2dv(points)
glEnd()

# Построение NURBS-слпайна
glBegin(GL_LINE_STRIP)
glColor3dv((0, 0, 0))
x = np.linspace(1, 8, 1000)
points_ = [self.F(x_) for x_ in x]
for p in points_:
```

```
    glVertex2dv(p)
glEnd()
```

Для расчета *NURBS*-сплайна по формулам, представленным в теоретических положениях, написана функция *buildNurbs*. Код функции генерирует сплайн по заданным узлам *T*, управляющим точкам *P* и весам *W*. Переменные *m, n* и *k* определяют количество узлов, управляющих точек и степень сплайна соответственно. Переменная *N* — это словарь, который будет содержать базисные функции сплайна. Функция *buildNk1* создает базисные функции первого порядка (т.е. базисные функции для порядка *0*). Функция *buildNkm* создает базисные функции для *k*-го порядка. Она использует рекурсивный подход, чтобы определить базисные функции более высоких порядков на основе уже определенных функций более низких порядков. Функция *F* вычисляет значение *NURBS*-сплайна в заданной точке *t*. Она использует базисные функции *N*, заданные в словаре, а также управляющие точки *P* и веса *W* для вычисления значения сплайна. Возвращаются функция *F* и словарь *N,* содержащий все базисные функции сплайна.

```
def buildNurbs(T: List[float], P: List[np.array], W: List[float]):
    # Узлы - 1
    m = 9
    # Управляющие точки - 1
    n = 4
    # Степень сплайна
    k = 4

    # Базисные функции N(i, j)
    N = dict()

    # Генерация базисных функций порядка 0
    def buildNk1(t_i, t_i1):
        return lambda t: 1 if (t_i <= t <= t_i1) else 0

    for i in range(m):
        N[(i, 0)] = buildNk1(T[i], T[i + 1])

    # Генерация базисных функций k-го порядка
    def buildNkm(i, j):
        nonlocal N, T

        @cache
        def Nin(t):
            f = (t - T[i]) / (T[i + j] - T[i]) if (T[i + j] - T[i]) != 0 else
0
            g = (T[i + 1 + j] - t) / (T[i + 1 + j] - T[i + 1]) if (T[i + 1 +
j] - T[i + 1]) != 0 else 0
            return f * N[(i, j - 1)](t) + (g) * N[(i + 1, j - 1)](t)

        return Nin
```

```
    for j in range(1, k + 1):
        for i in range(n - j + k + 1):
            N[(i, j)] = buildNkm(i, j)

    def F(t):
        f = [N[(i, k)](t) * W[i] for i in range(n + 1)]
        b1 = sum([f[i] * P[i] for i in range(n + 1)])
        b2 = sum([f[i] for i in range(n + 1)])
        return b1 / b2 if b2 != 0 else b1

    return F, N
```

Для возможности изменения положения контрольных точек были написаны методы, которые реагируют на действия мыши.

*mousePressEvent(self, event):* Этот метод вызывается, когда пользователь нажимает кнопку мыши. В данном случае он отслеживает нажатие левой кнопки мыши. Когда это происходит, он определяет координаты точки на экране, в которой произошло нажатие, и ищет ближайшую к этой точке управляющую точку. Затем он обновляет выбранную управляющую точку, чтобы пользователь мог взаимодействовать с ней.

*mouseMoveEvent(self, event):* Этот метод вызывается при перемещении мыши. Он отслеживает перемещение мыши при нажатой левой кнопке и, если выбрана управляющая точка (*self.selected_point* не равен *None*), обновляет ее координаты в соответствии с текущим положением указателя мыши.

*mouseReleaseEvent(self, event):* Этот метод вызывается, когда пользователь отпускает кнопку мыши после ее нажатия. В данном случае он отслеживает отпускание левой кнопки мыши и сбрасывает выбранную управляющую точку (*self.selected_point*), что прекращает действие перемещения или редактирования точки.

```
def mousePressEvent(self, event):
    if event.button() == Qt.LeftButton:
        x = (event.x() / self.width) * 2 - 1
        y = ((self.height - event.y()) / self.height) * 2 - 1
        min_distance = float('inf')
        self.selected_point = None
        for i, point in enumerate(self.P):
            distance = np.linalg.norm(point - np.array([x, y]))
            if distance < min_distance:
                min_distance = distance
                self.selected_point = i
        self.update()

def mouseMoveEvent(self, event):
    if event.buttons() == Qt.LeftButton and self.selected_point is not None:
```

```
        pos = event.pos()
        x_point = pos.x() / self.width * 2 - 1
        y_point = 1 - pos.y() / self.height * 2
        self.P[self.selected_point] = np.array([x_point, y_point])
        self.update()

def mouseReleaseEvent(self, event):
    if event.button() == Qt.LeftButton:
        self.selected_point = None
        self.update()
```

Управление контрольными точками в приложении осуществляется с помощью виджета *ControlPanel*, который содержит в себе ползунки для регулирования весов контрольных точек.

При изменении местоположения точки и веса вызывается метод *update()* класса, отвечающего за отображение сцены с использованием *OpenGL*. После вызова метода *update(), Qt* запланирует перерисовку элемента управления и графической сцены. Когда это происходит, *Qt* отправляет событие перерисовки в очередь событий, и это событие будет обработано при следующем цикле событий.

Элементы управления и графический виджет объединены компонентом *MainWindow* (наследуемом от *QMainWindow*), в нём происходит связывание событий интерфейса управления с обновлениями изображения.

**Тестирование.**

Возможные изображения сплайна и интерфейса программы представлены на рисунках 1-3.

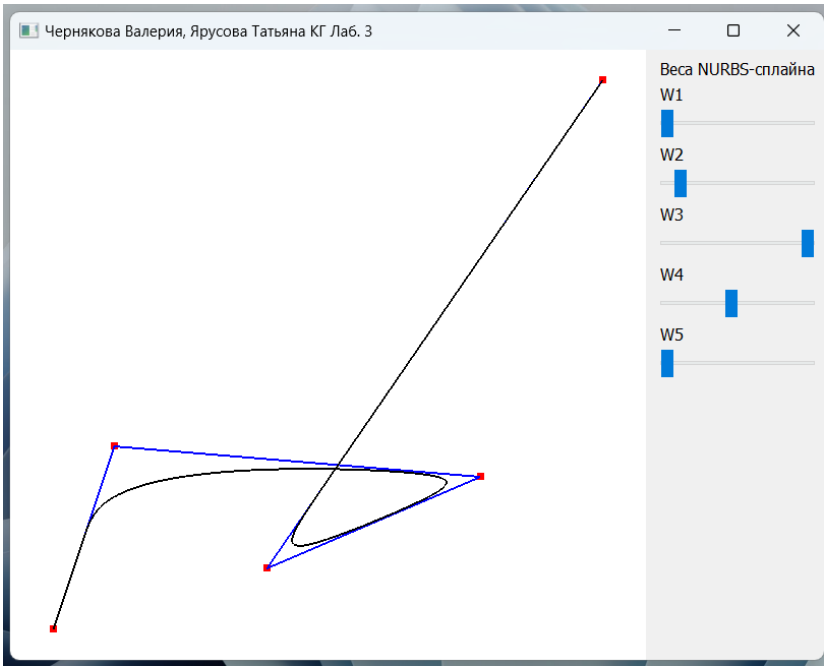


Рисунок 1. Сплайн с заданными программно параметрами

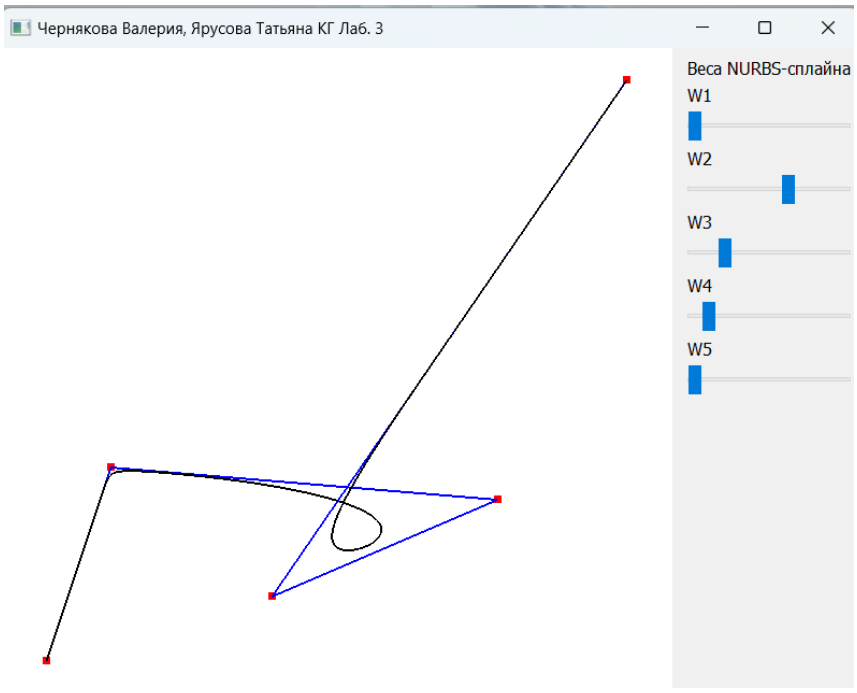


Рисунок 2. Сплайн при изменении веса контрольных точек

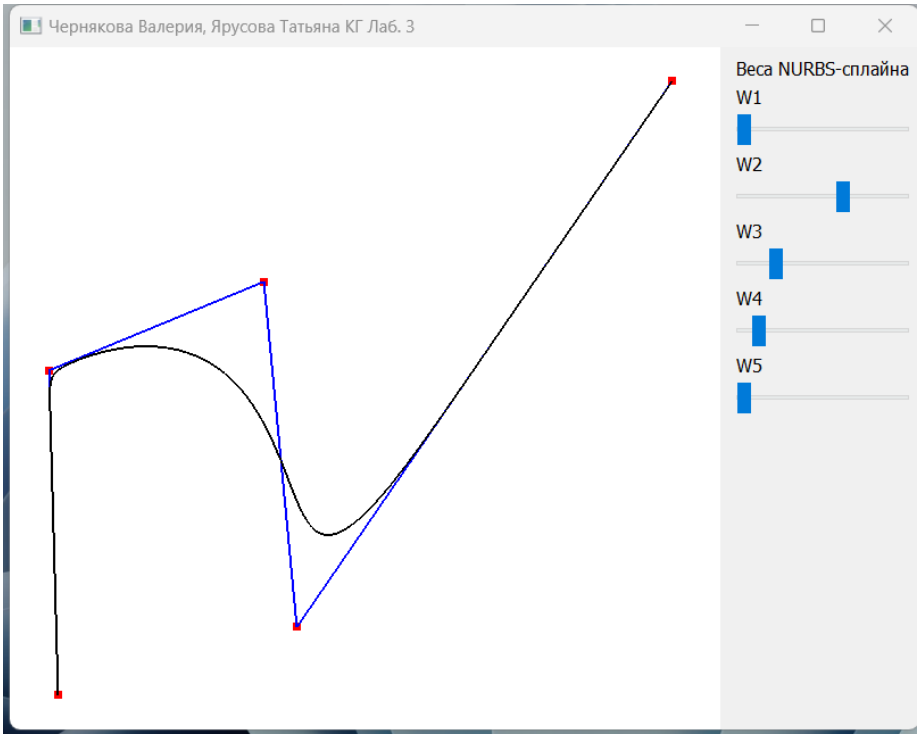


Рисунок 3. Сплайн при перемещении контрольных точек

**Выводы.**

В ходе лабораторной работы были изучены принципы построения *NURBS* сплайнов. Была реализована программа позволяющая настраивать параметры сплайна, такие как веса контрольных точек и их расположение. Полученный опыт в построении таких сплайнов является значимым, так как *NURBS* является важным инструментом в компьютерной графике благодаря своей гибкости, точности и возможности создания плавных и реалистичных форм.