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

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Компьютерная графика»

Тема: «Кубические сплайны»

Варианты 9 и 12

Студент гр. 7383 _____ Бергалиев М.

Студент гр. 7383 _____ Власов Р.А.

Преподаватель _____ Герасимова Т.В.

Санкт-Петербург

2020

Задание.

На базе предыдущей лабораторной работы разработать программу реализующую:

Задание 9: Кривая Безье 5-й степени.

Задание 12: Кубический сплайн, состоящий из 2-х сегментов.

Общие сведения.

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.

Ход работы.

Программа разработана на языке программирования Python. Графический интерфейс реализован с помощью библиотеки Qt.

Для успешного отображения изображения был реализован класс OpenGLView. В качестве родительского класса взят QGLWidget. Перепределены методы initializeGL, resizeGL и paintGL, отвечающие за инциализацию, изменение размера виджета и рисование изображения.

Для вычисление кривой Безье была реулизована следующая функция:

```
def calculateBesierCurve(P):
degree = len(P) - 1
T = [np.array([t ** (degree - i) for i in range(0, degree + 1)]) for t in np.arange(0, 1.001, 0.001)]
besie_matrix = np.array([[((-1) ** ((n + k) % 2)) * binomial(degree, n) * binomial(n, k) for k in range(0, degree + 1)] for n in range(degree, -1, -1)])
return [np.dot(np.dot(T[i], besie_matrix), P) for i in range(0, len(T))]
```

Свойства кривой Безье:

- Непрерывность заполнения сегмента между начальной и конечной точками;
- Изменение направления траектории не влияет на форму кривой;
- Изменение координат хотя-бы одной из точек ведет к изменению формы всей кривой;
- Любой частичный отрезок кривой Безье есть тоже кривая Безье.

Для вычисление кубических сплайнов были реализованы следующие методы:

- Метод построения сплайнов

```
def BuildSpline(x, y, n):
splines = [SplineTuple(0, 0, 0, 0, 0) for _ in range(0, n)]
for i in range(0, n):
    splines[i].x = x[i]
    splines[i].a = y[i]
splines[0].c = splines[n - 1].c = 0.0
alpha = [0.0 for _ in range(0, n - 1)]
beta = [0.0 for _ in range(0, n - 1)]
for i in range(1, n - 1):
    hi = x[i] - x[i - 1]
    hi1 = x[i + 1] - x[i]
    A = hi
```

```
        C = 2.0 * (hi + hi1)
        B = hi1
        F = 6.0 * ((y[i + 1] - y[i]) / hi1 - (y[i] - y[i - 1]) /
hi)
        z = (A * alpha[i - 1] + C)
        alpha[i] = -B / z
        beta[i] = (F - A * beta[i - 1]) / z
    for i in range(n - 2, 0, -1):
        splines[i].c = alpha[i] * splines[i + 1].c + beta[i]
    for i in range(n - 1, 0, -1):
        hi = x[i] - x[i - 1]
        splines[i].d = (splines[i].c - splines[i - 1].c) / hi
        splines[i].b = hi * (2.0 * splines[i].c + splines[i -
1].c) / 6.0 + (y[i] - y[i - 1]) / hi
    return splines
```

- Метод вычисления значения в заданной точке по построенному ранее сплайну

```
def Interpolate(splines, x):
    if not splines:
        return None
    n = len(splines)
    s = SplineTuple(0, 0, 0, 0, 0)
    if x <= splines[0].x:
        s = splines[0]
    elif x >= splines[n - 1].x:
        s = splines[n - 1]
    else:
        i = 0
        j = n - 1
        while i + 1 < j:
            k = i + (j - i) // 2
            if x <= splines[k].x:
                j = k
            else:
```

```
                i = k
        s = splines[j]
    dx = x - s.x
    return s.a + (s.b + (s.c / 2.0 + s.d * dx / 6.0) * dx) * dx
```

На каждом отрезке $[x_{i-1}, x_i]$, $i = 1, N$ функция $S(x)$ есть полином третьей степени $S_i(x)$, коэффициенты которого надо определить. Запишем для удобства $S_i(x)$ в виде:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

Вычисление коэффициентов происходит методом прогонки.

Сплайны обладают следующими свойствами:

- На каждом отрезке функция является многочленом степени не выше 3;
- Имеет непрерывные первую и вторую производные;
- В контрольных точках значения совпадают с контрольными.

Тестирование.

Программа протестирована в операционной системе Ubuntu 19.04.

Результаты тестирования представлены на рис. 1-4.

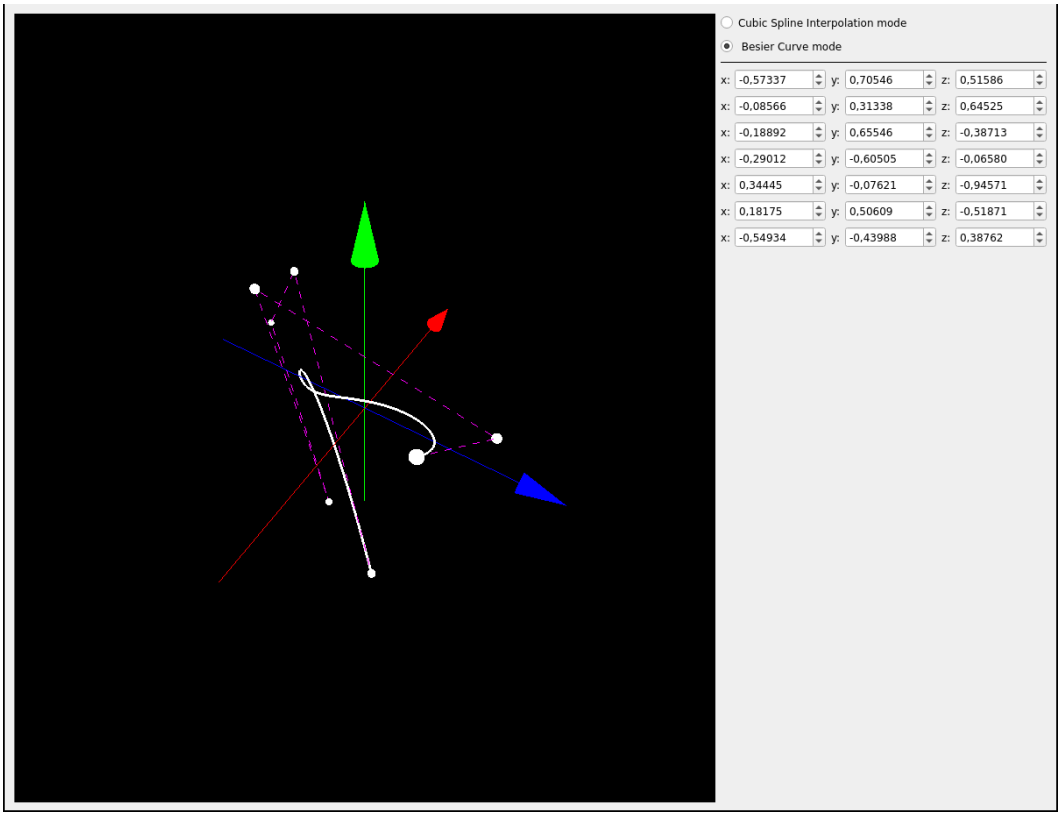

Рисунок 1 – Кривая Безье

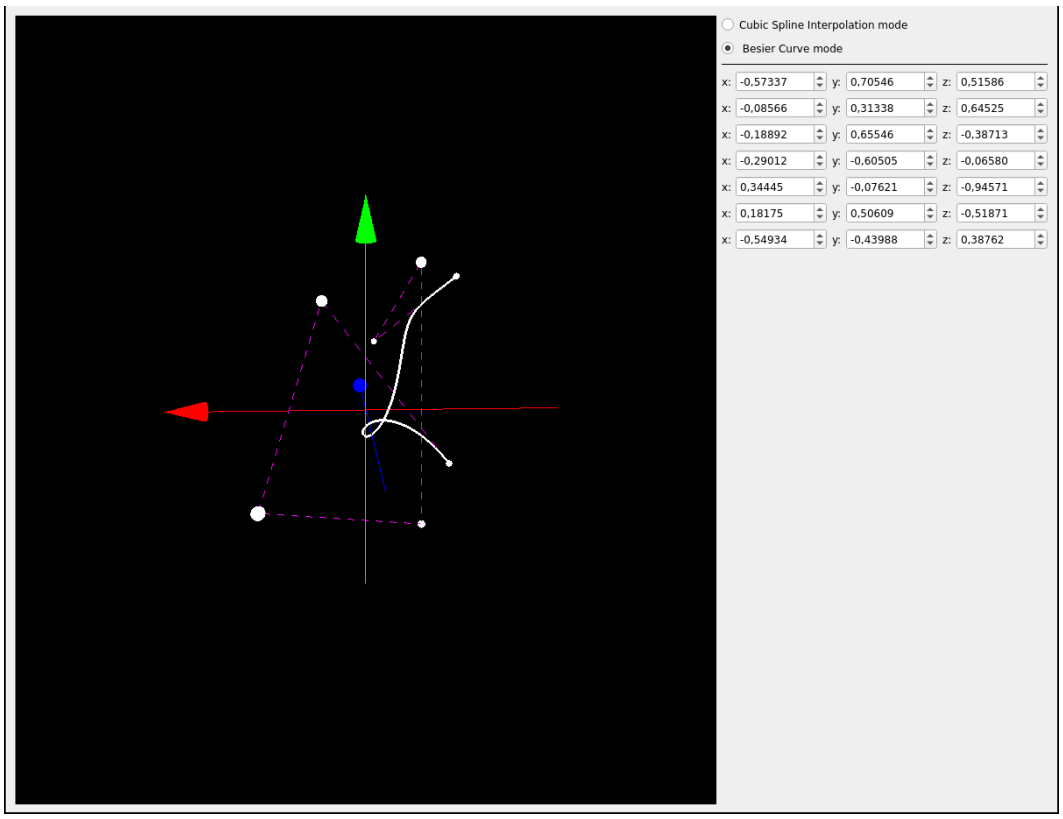

Рисунок 2 – Кривая Безье (другой ракурс)

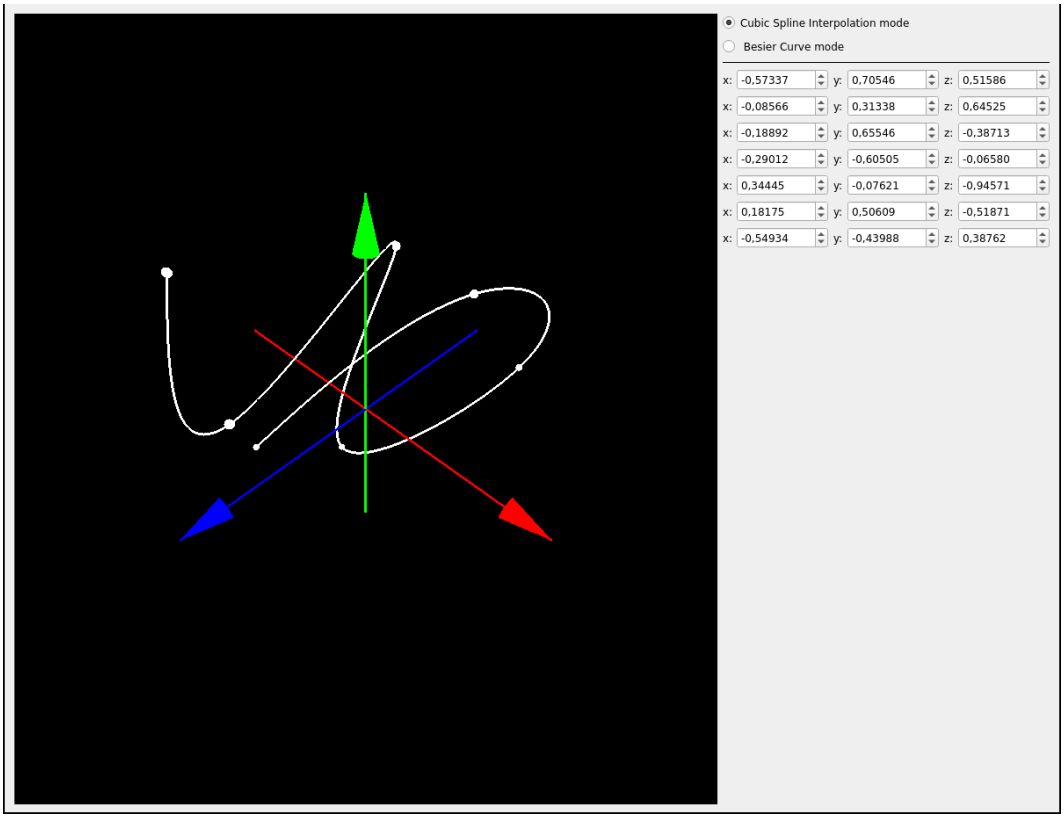

Рисунок 3 – Результат интерполяции кубическими сплайнами

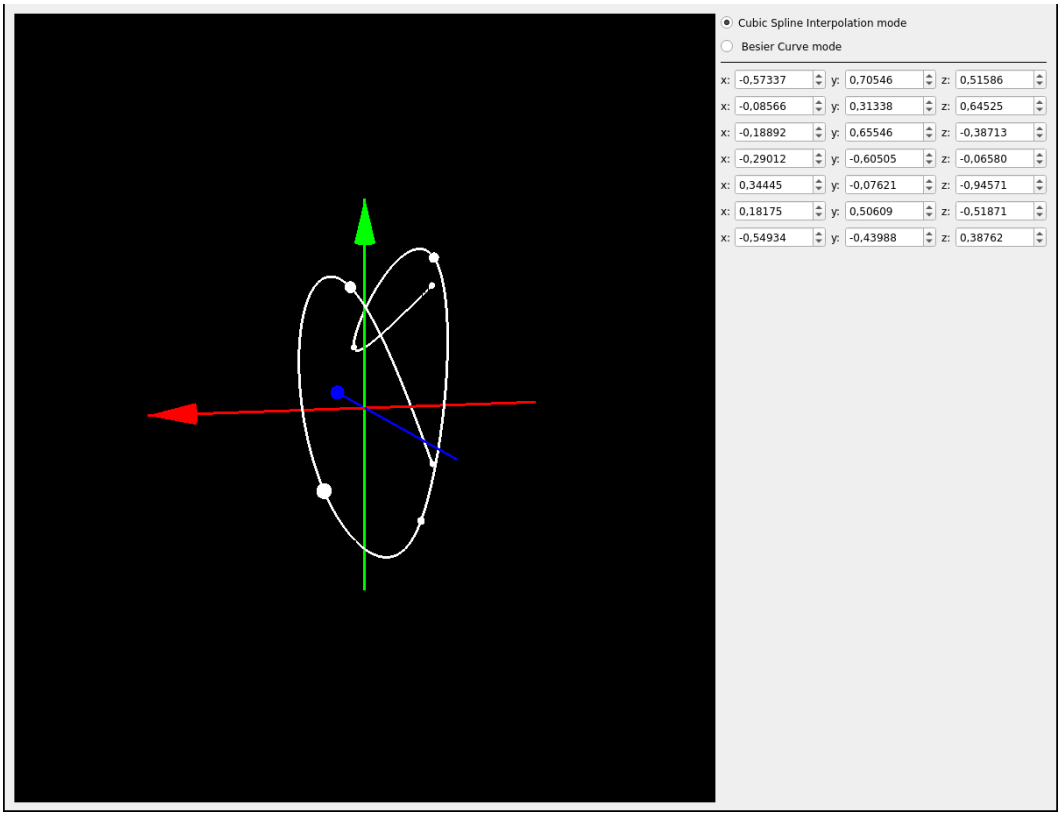

Рисунок 4 – Результат интерполяции кубическими сплайнами (другой ракурс)

Вывод.

В результате выполнения лабораторной работы была разработана программа, реализующая вычислние и отображение кривой Безье и кубических сплайнов. Программа работает корректно. При выполнении работы были приобретены навыки работы с графической библиотекой OpenGL.