**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР**

отчет

**по практической работе №2**

**по дисциплине «Компьютерная графика»**

|  |  |  |
| --- | --- | --- |
| Студенты гр. 5371 |  | Локкина О.С. |
|  |  | Бергер Э.Э. |
|  |  | Уруков С.Д. |
| Преподаватель |  | Матвеева И.В. |

**Цель работы:**

Научиться строить составную кривую Безье 3-го порядка по заданным точкам.

**Задание:**

Используя генератор случайных чисел сформировать 10 точек и на их основе нарисовать задающую ломанную. После этого построить составную кривую Безье третьей степени.

**Ход работы:**

Сперва генерируются 10 точек (можно задать своё количество точек, но не меньше 4), у которых координата Х и Y это случайное число из отрезка [-9,9]. Эти точки выделяются на рисунке синим цветом. По ним сроится ломанная в той последовательности, в которой происходила их генерация.

После этого строится кривая Безье третьего порядка. Формула для построения следующая:

Где p – значение компоненты координаты (х или y) опорных точек; t – величина, изменяющаяся от 0 до 1.

В данной работе кривая, составленная из четырёх точек является ломанной из 20 маленьких отрезков. При этом на дисплее наблюдается сглаженная кривая.

Достигается сглаженность составной кривой Безье при помощи вспомогательных точек. Вместо 4-й точки при построении берется середина отрезка, образованного 3-й и 4-й точками. Вторая кривая начинает строится из вспомогательной точки, подхватывая заданные по условию, и заканчивает опять же вспомогательной. Таким образом получается составная кривая.

Для наглядности рассмотрим на примере кривой по 10 точкам:

Пусть заданы точки:

[[1.4, -7.6], [-6.7, 2.4], [8.0, -4.4], [2.9, -8.1], [-3.5, -0.7], [1.7, -0.5], [5.0, -7.2],   
[-4.6, 1.4], [3.9, 1.8], [2.1, 0.6]]

При построении кривой мы опираемся на три точки из условия, а четвертую берем как среднюю между т.3 и т.4.

[[1.4, -7.6], [-6.7, 2.4], [8.0, -4.4], **[5.45, -6.25]**]

Для второй кривой мы используем эту же вспомогательную, дополняем двумя точками из условия и в качестве последней берем новую вспомогательную.

[**[5.45, -6.25],** [2.9, -8.1], [-3.5, -0.7], **[-0.9, -0.6]**]

Аналогичным образом строим третью кривую.

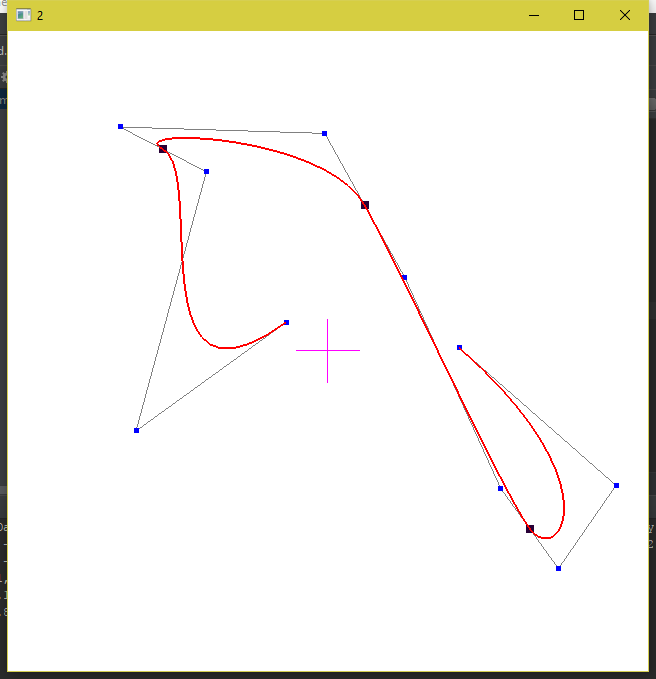
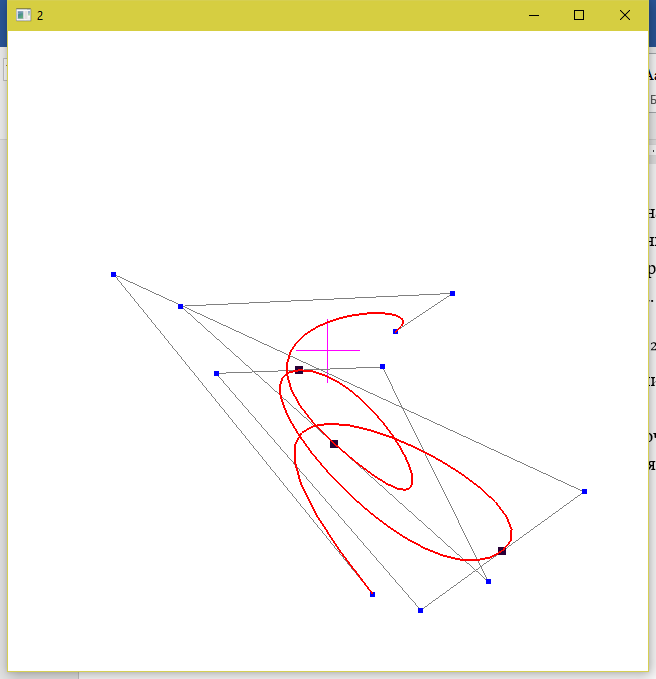
[**[-0.9, -0.6]**, [1.7, -0.5], [5.0, -7.2], **[0.2, -2.9]**]

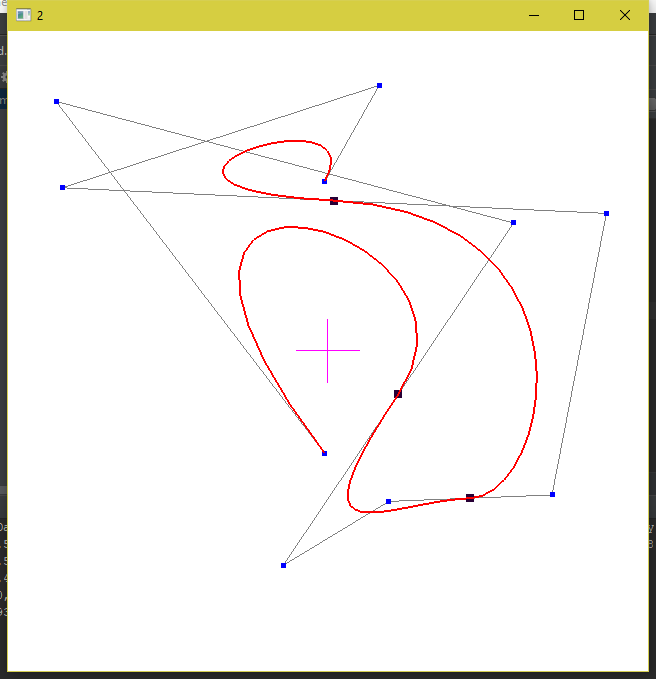
И в завершении используем оставшиеся точки.

[**[0.2, -2.9]**, [-4.6, 1.4], [3.9, 1.8], [2.1, 0.6]]

Вспомогательные точки вычисляются по формуле нахождения середины отрезка:

Примеры полученных кривых изображены ниже. На каждом изображении видно перекрестие: это начало координат, отрезки, исходящие из этой точки, являются единичными.





**Код программы**

**from** random **import** choice

**from** OpenGL.GL **import** \*

**from** OpenGL.GLUT **import** \*

points = list()

curve = list()

window = 0

countOfPoints = 10

**def** createPoints(count, start, stop, step=0.1):

listOfNumbers = []

result = start

**while** result < stop:

listOfNumbers.append(round(result, 1))

result += step

**for** i **in** range(count):

points.append([choice(listOfNumbers), choice(listOfNumbers)])

**def** get\_middle\_dot(coords):

**return** [0.5 \* (coords[0][0] + coords[1][0]), 0.5 \* (coords[0][1] + coords[1][1])]

**def** beizer4(my\_list):

**global** curve

a = 20

k = [i / a **for** i **in** range(0, a + 1)]

**for** t **in** k:

x = (1 - t) \*\* 3 \* my\_list[0][0] + 3 \* (1 - t) \*\* 2 \* t \* my\_list[1][0] + 3 \* (1 - t) \* t \*\* 2 \* my\_list[2][

0] + t \*\* 3 \* my\_list[3][0]

y = (1 - t) \*\* 3 \* my\_list[0][1] + 3 \* (1 - t) \*\* 2 \* t \* my\_list[1][1] + 3 \* (1 - t) \* t \*\* 2 \* my\_list[

2][1] + t \*\* 3 \* my\_list[3][1]

curve.append([x, y])

**def** drawPoints():

**global** points

glPointSize(5)

glBegin(GL\_POINTS)

glColor3f(0, 0, 1)

**for** point **in** points:

glVertex2dv(point)

glEnd()

glPointSize(8)

glBegin(GL\_POINTS)

glColor3f(0.155, 0, 0.211)

**for** i **in** range(2, countOfPoints - 3, 2):

glVertex2dv(get\_middle\_dot(points[i:i + 2]))

glEnd()

**def** draw\_polyline(coords):

**for** x **in** range(len(coords)):

**if** x + 1 != len(coords):

glVertex2dv(coords[x])

glVertex2dv(coords[x + 1])

**def** draw():

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)

glLoadIdentity()

glClearColor(1.0, 1.0, 1.0, 1.0)

glScalef(0.1, 0.1, 1)

glLineWidth(1)

glBegin(GL\_LINES)

glColor3f(1, 0, 1)

glVertex2d(1, 0)

glVertex2d(-1, 0)

glColor3f(1, 0, 1)

glVertex2d(0, -1)

glVertex2d(0, 1)

glColor3f(0.5, 0.5, 0.5)

draw\_polyline(points)

glEnd()

glLineWidth(2)

glBegin(GL\_LINES)

glColor3f(1, 0, 0)

draw\_polyline(curve)

glEnd()

drawPoints()

glutSwapBuffers()

**def** main():

**global** curve, window, countOfPoints

countOfPoints = int(input("Count of points (min 4):\n>>> "))

createPoints(countOfPoints, -9, 9)

*# Дублирует последнюю точку, если не хватает 1 точки для завершения кривой*

**if** countOfPoints % 2 == 1:

countOfPoints += 1

points.append(points[-1])

super\_arr = points[:3] + [get\_middle\_dot(points[2:4])]

beizer4(super\_arr)

**if** countOfPoints > 4:

**for** i **in** range(4, countOfPoints - 3, 2):

super\_arr = [get\_middle\_dot(points[i - 2:i])] + points[i - 1:i + 1] + [get\_middle\_dot(points[i:i + 2])]

beizer4(super\_arr)

super\_arr = [get\_middle\_dot(points[countOfPoints - 4:countOfPoints - 2])] + points[countOfPoints - 3:countOfPoints]

beizer4(super\_arr)

glutInit()

glutInitDisplayMode(GLUT\_RGBA | GLUT\_DOUBLE | GLUT\_ALPHA | GLUT\_DEPTH)

glutInitWindowSize(640, 640)

glutInitWindowPosition(200, 10)

window = glutCreateWindow(b"2")

glutDisplayFunc(draw)

glutIdleFunc(draw)

glutMainLoop()

main()

**Выводы**

В ходе работы был рассмотрен метод построения составной кривой Безье по нескольким точкам. Разобран способ отображения кривой по нескольким отрезкам. Накоплен практический опыт в работе с 2D графикой в OpenGL.