Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №7 по курсу «Компьютерная графика»

Студент: Ф.М. Шавандрин
Преподаватель: Г.С. Филиппов
Группа: М8О-308Б-19
Дата: 20.12.2021
Оценка:
Подпись:

Москва, 2021

Лабораторная работа №7

Построение плоских полиномиальных кривых

Задача: Написать программу, строящую полиномиальную кривую по заданным точкам. Обеспечить возможность изменения позиции точек и, при необходимости, значений касательных векторов и натяжения.
Вариант 7: Кривая Безье 5-й степени.

Описание

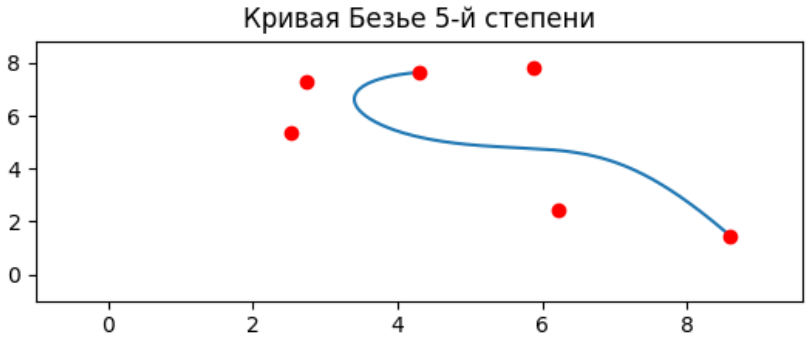
Кривая Безье записывается следующим параметрическим выражением: $\sum_{k=0}^{n} P_k * b_{k,n}(t), 0 \le t \le 1$, где $b_{k,n}(t) = \frac{n!}{k! * (n-k)!} * t^k * (1-t)^{n-k}$. Для построения кривой Безье 5-го порядка используется 6 опорных точек, координаты которых задаются случайным образом. Также использовал виджет Slider из matplotlib.widgets, который позволяет с помощью ползунка изменять координаты x и y для каждой опорной точки. После изменения координат точек программа строит кривую Безье заново.

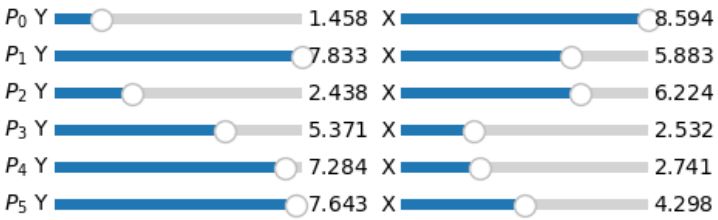
Исходный код

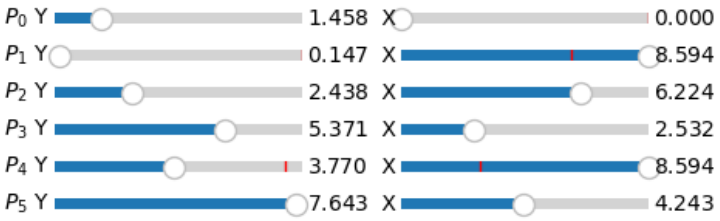
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.special import comb
from matplotlib.widgets import Slider

def bernstein_poly(i, n, t):
    # The Bernstein polynomial of n, i as a function of t=
    return comb(n, i) * ( t**(n-i) ) * (1 - t)**i


def bezier_curve(points, nTimes=1000):
    nPoints = len(points)
    xPoints = np.array([p[0] for p in points])
    yPoints = np.array([p[1] for p in points])
    t = np.linspace(0.0, 1.0, nTimes)
      polynomial_array = np.array([ bernstein_poly(i, nPoints-1, t) for i in
range(0, nPoints)])
    xvals = np.dot(xPoints, polynomial_array)
    yvals = np.dot(yPoints, polynomial_array)
    return xvals, yvals


# plotting figure
nPoints = 6
fig = plt.figure()
ax = fig.add_subplot(211)
#points = [np.array([float(i) for i in input("Введите координаты x и y для " +
str(ind + 1) + " опорной точки: ").split()]) for ind in range(nPoints)]
points = np.random.rand(nPoints,2)*10
xpoints = [p[0] for p in points]
ypoints = [p[1] for p in points]

xvals, yvals = bezier_curve(points, nTimes=1000)
M, = plt.plot(xvals, yvals)
P, = plt.plot(xpoints, ypoints, "ro")

plt.title("Кривая Безье 5-й степени")
```

```
x_max_lim = np.max(xpoints) + 1
y_max_lim = np.max(ypoints) + 1
plt.xlim(-1, x_max_lim)
plt.ylim(-1, y_max_lim)

sliders_x = []
for i in range(nPoints):
    slider_ax = plt.axes([0.5, 0.3 - 0.05 * i, 0.25, 0.03])
    slider = Slider(slider_ax, r' X'.format(i), 0, x_max_lim - 1, xpoints[i])
    sliders_x.append(slider)

def update_x(val):
    """
    Updates the plot after changing a slider
    """
    for i in range(nPoints):
        xpoints[i] = sliders_x[i].val
        points[i][0] = xpoints[i]
    newxval,_ = bezier_curve(points, nTimes=1000)
    P.set_xdata(xpoints)
    M.set_xdata(newxval)

sliders_y = []
for i in range(nPoints):
    slider_ax = plt.axes([0.15, 0.3 - 0.05 * i, 0.25, 0.03])
       slider  =  Slider(slider_ax,  r'$P_{0}$  Y'.format(i),  0,  y_max_lim  -  1,
ypoints[i])
    sliders_y.append(slider)


def update_y(val):
    """
    Updates the plot after changing a slider
    """
    for i in range(nPoints):
        ypoints[i] = sliders_y[i].val
        points[i][1] = ypoints[i]
    _, newyval = bezier_curve(points, nTimes=1000)
    P.set_ydata(ypoints)
    M.set_ydata(newyval)


for i in range(nPoints):
    sliders_x[i].on_changed(update_x)
    sliders_y[i].on_changed(update_y)

plt.show()
```

Результат работы







Выводы

Выполнив седьмую лабораторную работу по курсу «Компьютерная графика», я познакомился с кривой Безье n-ого порядка, построил данную кривую по шести опорным точкам, закрепил навыки работы с библиотекой matplotlib для Python, изучил виджет Slider.