Московский Авиационный Институт (Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Курсовой проект по курсу “Компьютерная Графика”

Студент: А. Т. Бахарев

Преподаватель: Г. С. Филиппов

Группа: М8О-306Б

Дата: 25.12.2018

Оценка:

Подпись:

# Курсовая работа

**Задача:** Составить и отладить программу, обеспечивающую каркасную визуализацию порции поверхности заданного типа. Исходные данные готовятся самостоятельно и вводятся из файла или в панели ввода данных. Должна быть обеспечена возможность тестирования программы на различных наборах исходных данных. Программа должна обеспечивать выполнение аффинных преобразований для заданной порции поверхности, а также возможность управлять количеством изображаемых параметрических линий. Для визуализации параметрических линий поверхности разрешается использовать только функции отрисовки отрезков в экранных координатах

**Вариант:** Кинематическая поверхность. Образующая – астроида, направляющая – кривая Безье 3D 2-й степени

Пояснение

**Кривы́е Безье́** или **Кривы́е Бернште́йна — Безье́** — типы кривых, предложенные в 60-х годах XX века независимо друг от друга Пьером Безье из автомобилестроительной компании «Рено» и Полем де Кастельжо из компании «*Ситроен*», где применялись для проектирования кузовов автомобилей.

Несмотря на то, что открытие де Кастельжо было сделано несколько ранее Безье (1959), его исследования не публиковались и скрывались компанией как производственная тайна до конца 1960-х.

Кривая Безье является частным случаем многочленов Бернштейна, описанных Сергеем Натановичем Бернштейном в 1912 году.

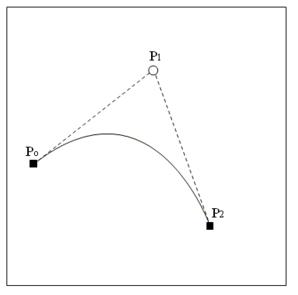
Впервые кривые были представлены широкой публике в 196 году французским инженером Пьером Безье, который, разработав их независимо от де Кастельжо, использовал их для компьютерного проектирования автомобильных кузовов. Кривые были названы именем Безье, а именем де Кастельжо назван разработанный им рекурсивный способ определения кривых (алгоритм де Кастельжо).

Впоследствии это открытие стало одним из важнейших инструментов систем автоматизированного проектирования и программ компьютерной графики.

Формула и изображение

Кривая Безье относится к частному классу алгебраических кривых, а именно: к кривым 3-го и 2-го порядков соответственно.

Квадратичная кривая Безье (n = 2) задаётся тремя опорными точками: **P**0, **P**1 и **P**2:



Исходный код

**from** math **import** cos, pi, sin

**import** matplotlib.pyplot **as** plt

**import** numpy **as** np

**from** mpl\_toolkits **import** mplot3d

**from** mpl\_toolkits.mplot3d.art3d **import** Line3DCollection, Poly3DCollection

**def** zoom\_factory(ax, base\_scale=2.):

**def** zoom\_fun(event):

# get the current x and y limits

cur\_xlim = ax.get\_xlim()

cur\_ylim = ax.get\_ylim()

cur\_xrange = (cur\_xlim[1] - cur\_xlim[0]) \* .5

cur\_yrange = (cur\_ylim[1] - cur\_ylim[0]) \* .5

xdata = event.xdata # get event x location

ydata = event.ydata # get event y location

**if** event.button == 'up':

# deal with zoom in

scale\_factor = 1 / base\_scale

**elif** event.button == 'down':

# deal with zoom out

scale\_factor = base\_scale

**else**:

# deal with something that should never happen

scale\_factor = 1

**print**(event.button)

# set new limits

ax.set\_xlim([

xdata - cur\_xrange \* scale\_factor,

xdata + cur\_xrange \* scale\_factor

])

ax.set\_ylim([

ydata - cur\_yrange \* scale\_factor,

ydata + cur\_yrange \* scale\_factor

])

plt.draw() # force re-draw

fig = ax.get\_figure() # get the figure of interest

# attach the call back

fig.canvas.mpl\_connect('scroll\_event', zoom\_fun)

#return the function

**return** zoom\_fun

**def** interpolate(P1, P2, T1, T2, steps):

res = []

**for** t **in** range(steps):

s = t / steps

h1 = (1 - s) \*\* 2

h2 = 2 \* s \* (1 - s)

h3 = s\*\*2

#h4 = s\*\*3 - s\*\*2

res.append(h1 \* P1 + h2 \* P2 + h3 \* T1)

**return** res

p0 = np.array([-300, 600, 200])

p1 = np.array([-100, 400, 800])

p2 = np.array([-500, 100, -500])

p3 = np.array([100, -150, 300])

t1 = 0.3 \* (p2 - p0)

t2 = 0.3 \* (p3 - p1)

curve = interpolate(p1, p2, t1, t2, 20)

x, y, z = zip(\*curve)

e = 30

ell = []

**for** p **in** curve:

points = []

**for** j **in** range(0, e + 1):

points.append(((cos(j \* 7 / e) \*\* 3) \* 300 + p[0], p[1] \* 2,

(sin(j \* 7 / e) \*\* 3) \* 300 + p[2]))

points = np.array(points)

ell.append(points)

verts = []

**for** i **in** range(len(ell) - 1):

**for** j **in** range(len(ell[i])):

verts.append([

ell[i][j], ell[(i + 1) % len(ell)][j],

ell[(i + 1) % len(ell)][(j + 1) % len(ell[i])],

ell[i][(j + 1) % len(ell[i])]

])

plt.style.use('dark\_background')

fig = plt.figure()

ax = fig.add\_subplot(111, projection='3d')

ax.grid(True)

plt.xlabel('x')

plt.ylabel('y')

plt.axis('off')

ax.set\_xlim([-800, 800])

ax.set\_ylim([-100, 800])

ax.set\_zlim([-800, 800])

scale = 1.5

f = zoom\_factory(ax, base\_scale=scale)

# plot sides

ax.add\_collection3d(

Poly3DCollection(

verts,

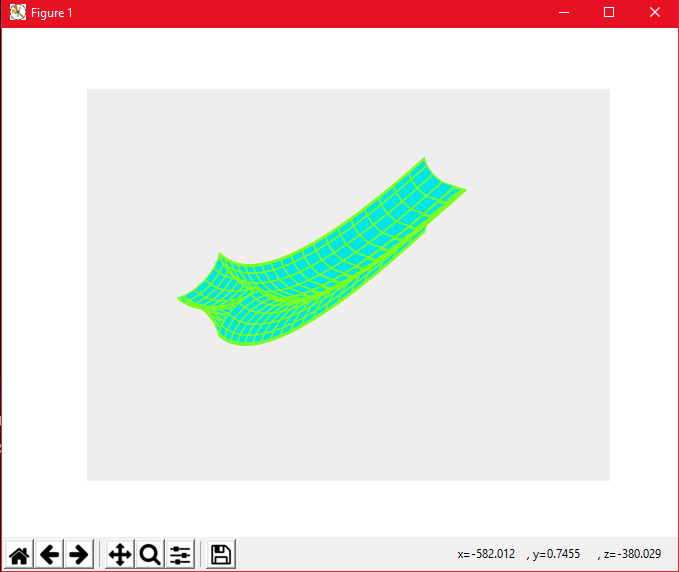
facecolor=(0.0, 0.9, 0.9),

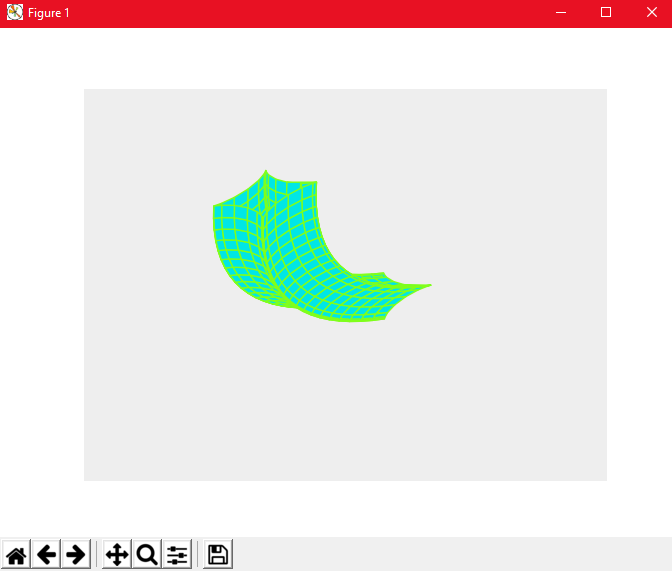
linewidths=1,

edgecolor=(0.5, 1, 0.1)))

plt.show()

# Скриншоты







# Выводы

Выполнив курсовую работу по курсу “Компьютерная графика”, я узнал о кривой Безье и о ее применении в машиностроении. В 60-х годах прошлого века Пьер Безье вывел свою знаменитую кривую для проектирования кузовов автомобилей. Данная работа помогла визуализировать эту параметрическую кривую в трехмерном пространстве, что дало более подробные сведения о ней. Так же, я закрепил основные навыки работы с языком программирования Python, который вполне не плох для работы с визуализацией объемных фигур и объектов.