

Московский Авиационный Институт (Национальный Исследовательский
Университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования
Курсовой проект по курсу “Компьютерная Графика”

Студентка: Чурсина Н.А.

Преподаватель: Филиппов Г.С.

Группа: М8О-308Б-18

Дата: 24.12.2020

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

Кривые Безье — 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 1961 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).

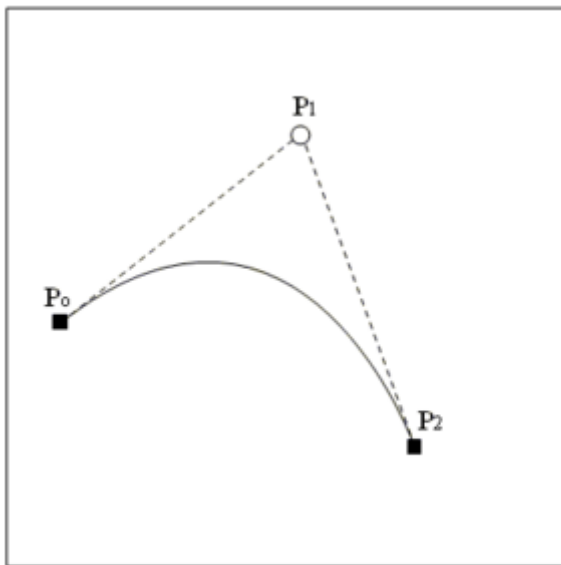
Впоследствии это открытие стало одним из важнейших инструментов систем автоматизированного проектирования и программ компьютерной графики.

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 :

$$B(t) = (1 - t)^2 P_0 + 2t(1 - t)P_1 + t^2 P_2, t \in [0, 1]$$



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 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_zlim = ax.get_zlim()
        cur_xrange = (cur_xlim[1] - cur_xlim[0]) * .5
        cur_yrange = (cur_ylim[1] - cur_ylim[0]) * .5
        cur_zrange = (cur_zlim[1] - cur_zlim[0]) * .5
        xdata = event.xdata # get event x location
        ydata = event.ydata # get event y location
        zdata = event.xdata # get event z 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
    try:
        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
        ])
        ax.set_zlim([
            zdata - cur_zrange * scale_factor,
            zdata + cur_zrange * scale_factor
        ])

        plt.draw() # force re-draw
    except:
        print('Курсор вне поля')

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
        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('bmh')
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d', facecolor = 'w', )
ax.grid(True)
plt.xlabel('x')
plt.ylabel('y')
plt.axis('off')

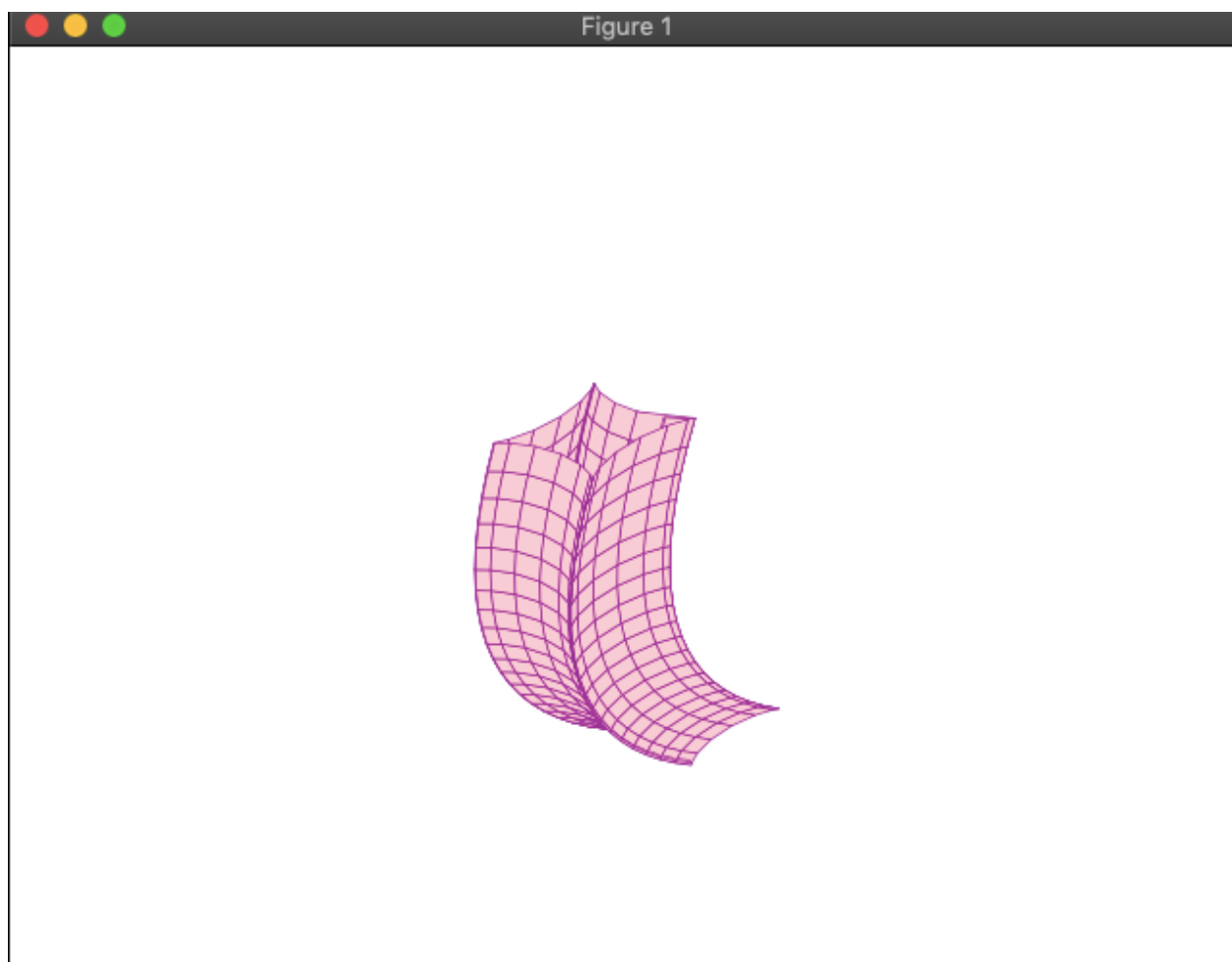
ax.set_xlim([-1000, 1000])
ax.set_ylim([-100, 1000])
ax.set_zlim([-1000, 1000])

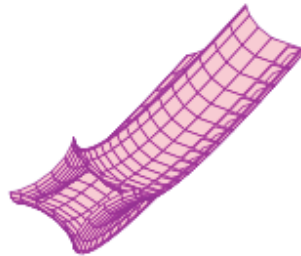
scale = 1.5
f = zoom_factory(ax, base_scale=scale)
# plot sides
ax.add_collection3d(
    Poly3DCollection(
        verts,
        facecolor='pink',
        linewidths=0.5,
        edgecolor='purple'))

plt.show()

```

Демонстрация работы





Выводы

Выполнив курсовую работу, я изучила много теории о кривой Безье, о применении кривой в машиностроении. В 60-х годах прошлого века Пьер Безье вывел данную кривую для проектирования кузовов автомобилей.

Мой прежний опыт работы с Python помог визуализировать эту параметрическую кривую в трехмерном пространстве, но мне было интересно изучать данную тему, искать подробные сведения о ней. Также, я закрепила основные навыки работы с языком программирования Python и некоторыми библиотеками, использование которых подходит для работы с визуализацией фигур и объектов в 3D пространстве.