Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский Авиационный Институт»

Национальный Исследовательский Университет

Институт №8 «Информационные технологии и прикладная математика»

Кафедра 806 «Вычислительная математика и программирование»

Курсовой проект

по курсу «Компьютерная графика»

Студент:	Хренникова А. С.
Группа:	М8О-308Б-19
Преподаватель:	Филиппов Г. С.
Подпись:	
Оценка:	
Дата:	

Москва, 2021

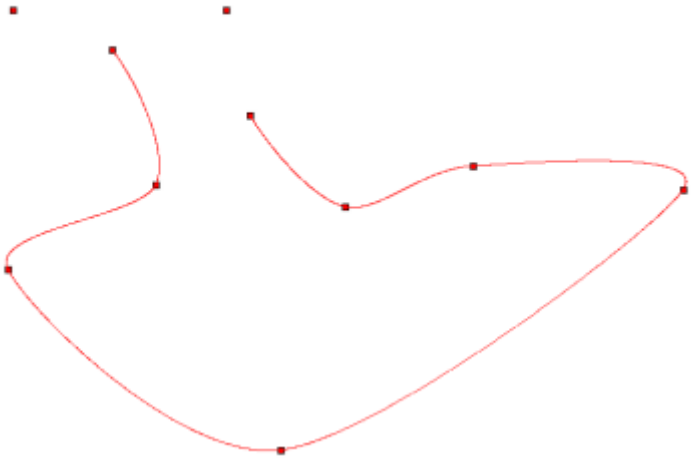
Курсовой проект

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 – эллипс, направляющая – Cardinal Spline 3D.

1 Описание

Cardinal Spline:



Линия представляет собой кривую, а квадраты - представляют контрольные точки P_k. Важно обратить внимание на то, что кривая не достигает первой и последней точек, однако эти точки влияют на форму кривой.

Москва, 2021

Формула для сплайна:

$$P(t) = (-t(1-t)^2 P_0 + (2 - 5t^2 + 3t^3)P_1 + t(1 + 4t - 3t^2)P_2 - t^2(1-t)P_3)/2$$

Программа написана на языке программирования Python с использованием библиотек mathplotlib, Poly3DCollection для отрисовки трехмерного изображения.

В программе с помощью формулы, которая указана выше, вычисляется кривая, вдоль которой строятся эллипсы. Сами эллипсы вычисляются по параметрической формуле. Таким образом строится график.

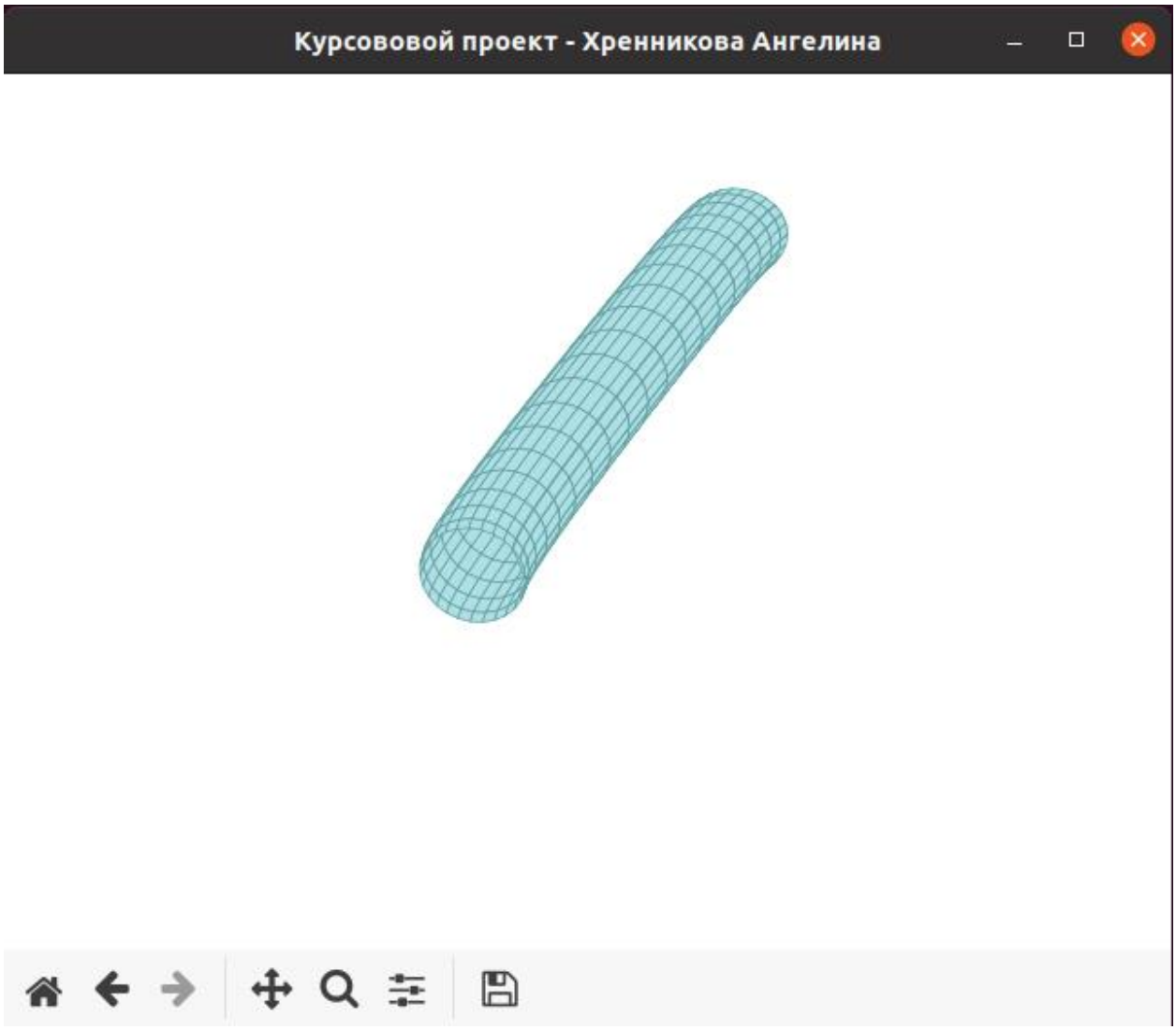
2 Исходный код:

```
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 Poly3DCollection

def zoom_factory(ax, base_scale=2.):
    def zoom_fun(event):
        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
        ydata = event.ydata
        zdata = event.xdata

        if event.button == 'up':
            scale_factor = 1 / base_scale

        elif event.button == 'down':
            scale_factor = base_scale

        else:
            scale_factor = 1
            print(event.button)

        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()

        except:
            print('ERROR')

    fig = ax.get_figure()

    fig.canvas.mpl_connect('scroll_event', zoom_fun)

    return zoom_fun

def interpolate(P1, P2, P3, P4, steps):
    res = []
    for t in range(steps):
        s = t / steps
        h1 = -1 * s * (1 - s) ** 2
        h2 = 2 - 5 * s * s + 3 * s * s * s
        h3 = s * (1 + 4 * s - 3 * s * s * s)
        h4 = -1 * s * s * (1 - s)
        res.append(h1 * P1 + h2 * P2 + h3 * P3 + h4 * P4)
    return res

p0 = np.array([-300, 600, 200])
p1 = np.array([-100, 400, 800])
p2 = np.array([-200, -150, -200])
p3 = np.array([-50, -200, 30])

curve = interpolate(p0, p1, p2, p3, 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)) * 300 + p[0], p[1] * 2,
                       (sin(j * 7 / e)) * 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])]
        ])
fig = plt.figure('Курсововой проект - Хренникова Ангелина')
ax = fig.add_subplot(111, projection='3d')
ax.grid(True)
plt.xlabel('x')
plt.ylabel('y')
plt.axis('off')

ax.set_xlim([-1000, 1000])
```

```
ax.set_ylim([-1000, 1000])
ax.set_zlim([-1000, 1000])

scale = 1.5
f = zoom_factory(ax, base_scale=scale)
ax.add_collection3d(
   Poly3DCollection(
       verts,
       facecolor='powderblue',
       linewidths=0.5,
       edgecolor='cadetblue'))


plt.show()
```

3 Работа программы:



4 Выводы:

Москва, 2021

В ходе выполнения данной лабораторной работы была написана программа на языке Python для построения кинематической поверхности, где образующая – эллипс, а направляющая – cardinal spline. Для решения данной задачи я изучила материал про сплайны, что было довольно интересно и полезно. Также закрепила свой опыт при работе в трехмерном пространстве.

Москва, 2021