Московский Авиационный Институт(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Курсовой проект по курсу “Компьютерная Графика”

Студент: Т.А.Габдуллин

Преподаватель: Г. С. Филиппов

Группа: М8О-306Б

Оценка:

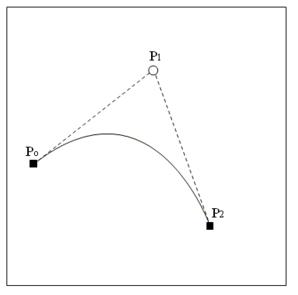
Подпись:

# 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

Кривая Безье 2 порядка



Исходныйкод

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

**import matplotlib.pyplot**

**import numpy**

**from mpl\_toolkits import mplot3d**

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

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

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

**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**

**ydata = event.ydata**

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

**scale\_factor = 1 / base\_scale**

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

**scale\_factor = base\_scale**

**else:**

**scale\_factor = 1**

**print(event.button)**

**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**

**])**

**matplotlib.pyplot.draw()**

**fig = ax.get\_figure()**

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

**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 = numpy.array([-300, 600, 200])**

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

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

**p3 = numpy.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 = numpy.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 = matplotlib.pyplot.figure()**

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

**matplotlib.pyplot.xlabel('x')**

**matplotlib.pyplot.ylabel('y')**

**matplotlib.pyplot.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)**

**ax.add\_collection3d(**

**Poly3DCollection(**

**verts,**

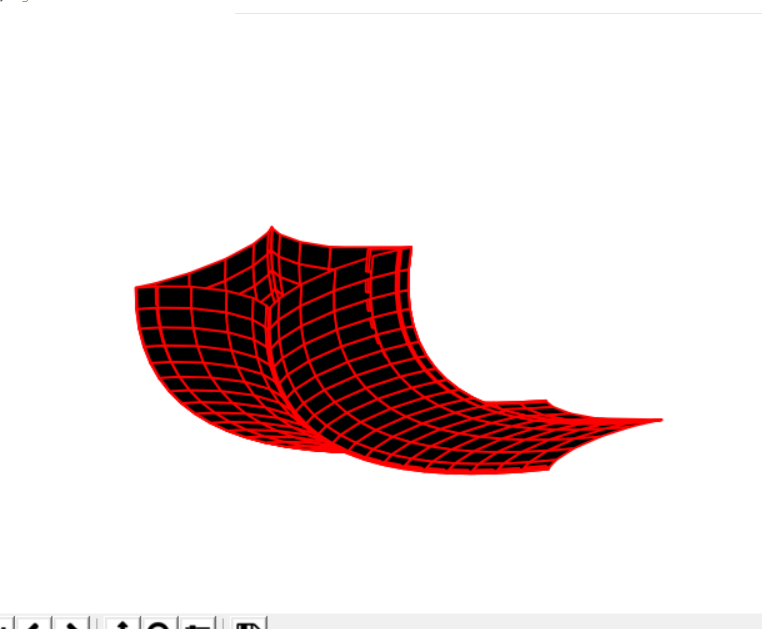
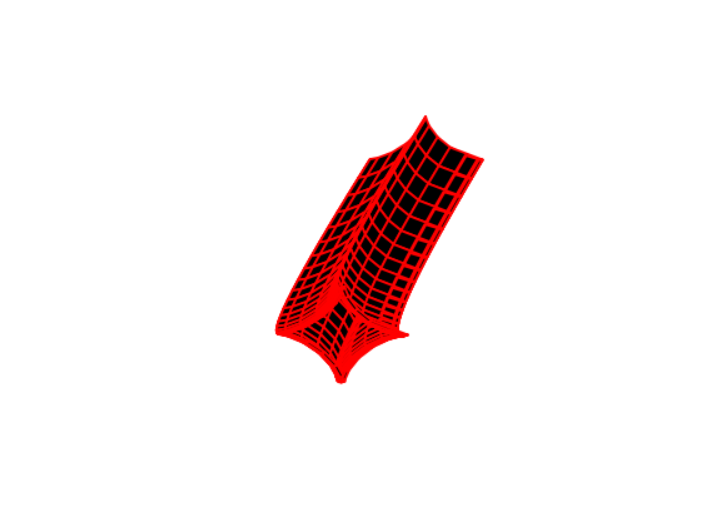
**facecolor=(0, 0, 0),**

**linewidths=1.3,**

**edgecolor=(1, 0, 0)))**

**matplotlib.pyplot.show()**

# Скриншоты



# Выводы

Выполнив курсовую работу, я закрепил основные навыки работы с языком программирования Python, который хорошо справляется с визуализацией объемных фигур и объектов.