Министерство науки и высшего образования
Московский Авиационный Институт
(Национальный Исследовательский Университет)

Институт №8
«Компьютерные науки и прикладная математика»

Отчёт о выполнении лабораторных работ по курсу

«Компьютерная графика»

студент: Шерматов Егор Дмитриевич

5 Вариант

преподаватель: Филиппов Глеб Сергеевич

Москва, 2023

Лабораторная работа N1 (4 часа).

Тема: Построение изображений 2D- кривых.

1.1 Задание
Написать и отладить программу, строящую изображение заданной замечательной кривой.

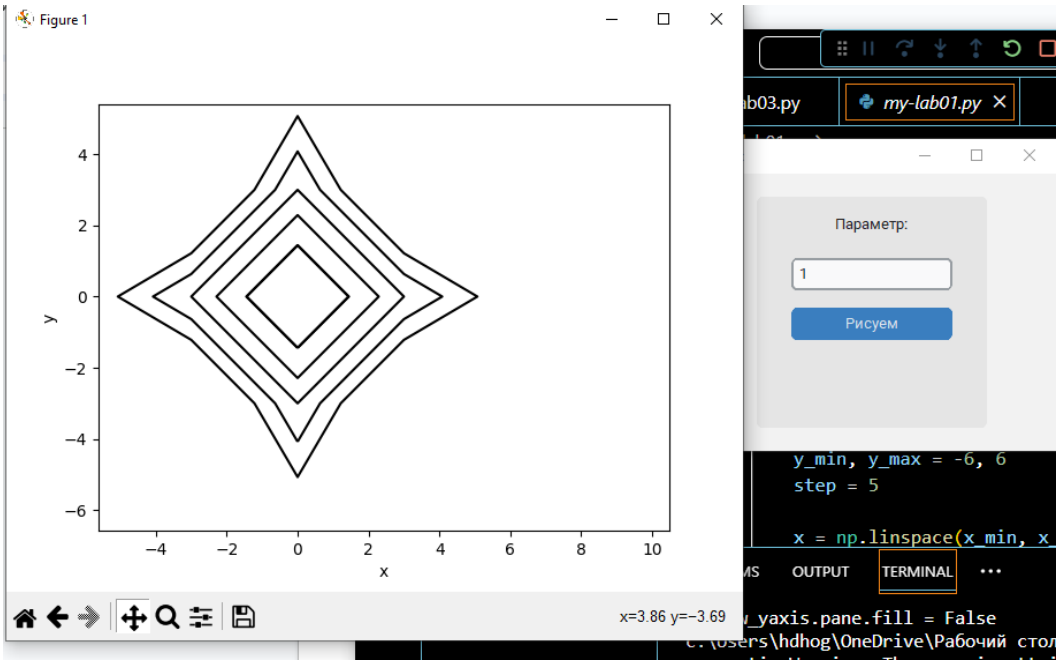
$$x^{\frac{2}{3}} + y^{\frac{2}{3}} = a^{\frac{2}{3}}$$

ρ,φ- полярные координаты, x,y – декартовы координаты t – независимый параметр.
a,b, k,A,B, - константы, значения которых выбираются пользователем (вводятся в окне программы). a,b>0

Обеспечить автоматическое масштабирование и центрирование кривой при изменении размеров окна.

1.2 Выполнение
В данной лабораторной работе я использовал модули *matplotlib.pyplot* для построения кривой, *numpy* для работы с данными(в данном случае для обработки координат) и *customtkinter* для отображения всего на интерфейсе.

Результат работы программы: для параметра a от 0 до 5



Вывод: удалось написать и отладить программу, строящую изображение заданной замечательной кривой и обеспечить автоматическое масштабирование и центрирование кривой при изменении размеров окна.

Лабораторная работа N2 (4 часа).

Тема: Каркасная визуализация выпуклого многогранника. Удаление невидимых линий.

2.1 Задание

Разработать формат представления многогранника и процедуру его каркасной отрисовки в ортографической и изометрической проекциях. Обеспечить удаление невидимых линий и возможность пространственных поворотов и масштабирования многогранника. Обеспечить автоматическое центрирование и изменение размеров изображения при изменении размеров окна. Фигура 5) Обелиск(усеченный клин).

2.2 Выполнение

В данной лабораторной работе я использовал модули *matplotlib.pyplot* для построения графических примитивов, *numpy* для работы с данными(в данном случае для обработки координат), модуль *Axes3D* из библиотеки *mpl_toolkits.mplot3d*

Описание удаления невидимых граней:

1) Сначала находим точку, расположенную внутри многогранника. Это можно сделать, например, таким образом:

1а) Находим центроиды всех граней, покоординатно суммируя каждую из точек, задающих эту грань, затем так же, по координатам делим получившиеся суммы на количество точек. Получаем точку - центр грани.

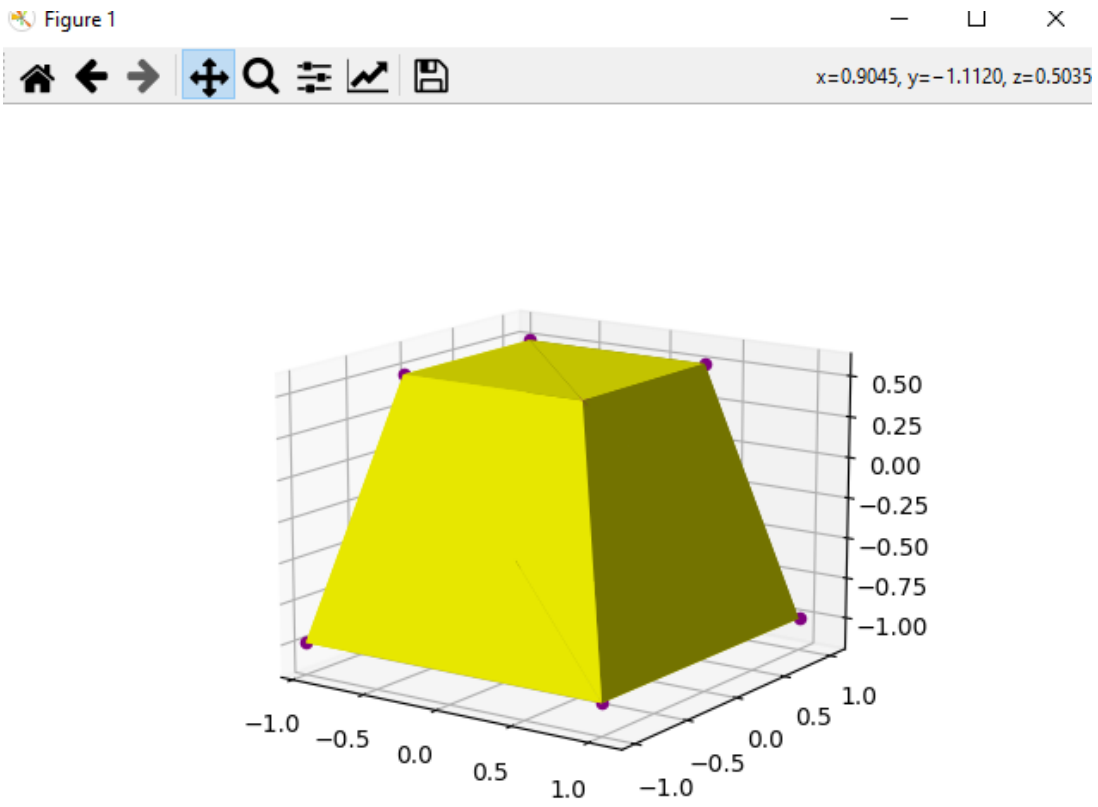
1б) Аналогично суммируем центры и делим их на количество граней

2) Для каждой грани ищем вектор нормали, это можно сделать через векторное произведение векторов, образованных тремя соседними точками грани.

3) Для каждой грани считаем скалярное произведение вектора, направленного из центра многогранника к центру грани, и вектора нормали этой грани. Если произведение <0, то домножаем компоненты вектора нормали на -1, чтобы он смотрел "наружу" многогранника.

4) При отрисовке для каждой грани проверяем скалярное произведение его и вектора, направленного от центра данной грани до точки наблюдения(для перспективной проекции это (0,0,-K), где K - расстояние от начала координат до точки наблюдения). Если скалярное произведение >0, то данная грань проецируется на экран.

На выходе работы программы получаем такое изображение:



Вывод: Удалось разработать формат представления многогранника и процедуру его каркасной отрисовки в ортографической и изометрической проекциях. Обеспечить удаление невидимых линий и возможность пространственных поворотов и масштабирования многогранника. Обеспечить автоматическое центрирование и изменение размеров изображения при изменении размеров окна.

Лабораторная работа N3 (4 часа).

Тема: Основы построения фотореалистичных изображений.

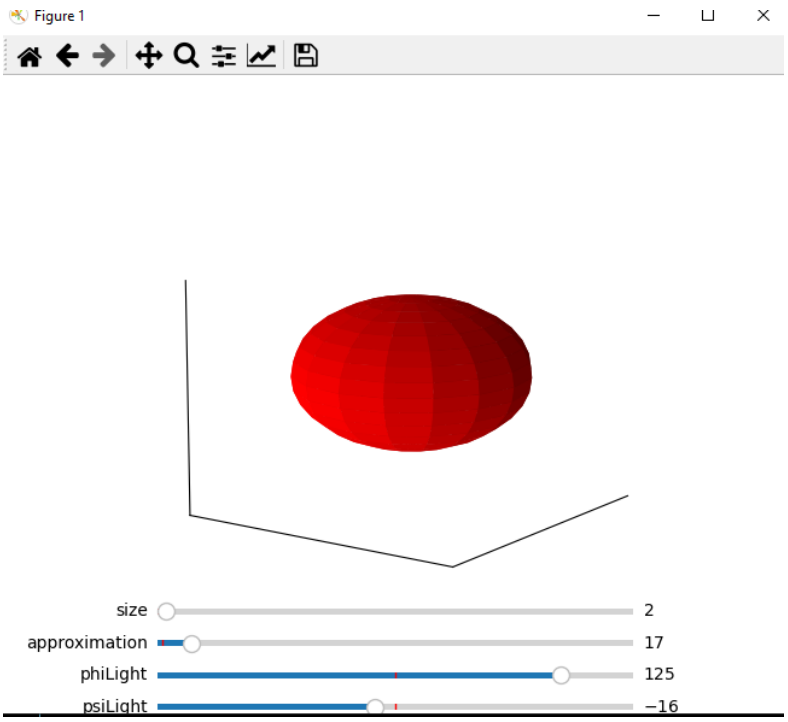
3.1 Задание

Используя результаты Л.Р.No2, аппроксимировать заданное тело выпуклым многогранником. Точность аппроксимации задается пользователем. Обеспечить возможность вращения и масштабирования многогранника и удаление невидимых линий и поверхностей. Реализовать простую модель закраски для случая одного источника света. Параметры освещения и отражающие свойства материала задаются пользователем в диалоговом режиме. Фигура 5) Эллипсоид.

3.2 Выполнение

В данной лабораторной работе я использовал модули *matplotlib.pyplot* для построения графических примитивов, *numpy* для работы с данными(в данном случае для обработки координат), math для реализации расчетов в полярной система координат, *matplotlib.widgets* для ввода и изменения пользователем параметров аппроксимации, углов для источника света и размера, *matplotlib.colors* для использования экземпляра класса *LightSourse*, который моделирует точечный источник света.

На выходе работы программы получаем такое изображение:



Вывод: Используя результаты Л.Р.No2, удалось аппроксимировать эллипсоид выпуклым многогранником (точность аппроксимации задается пользователем), обеспечить возможность вращения и масштабирования многогранника и удаление невидимых линий и поверхностей, реализовать простую модель закраски для случая одного источника света(параметры освещения и отражающие свойства материала задаются пользователем в диалоговом режиме)

Лабораторная работа N7 (4 часа).

Тема: Построение плоских полиномиальных кривых.

7.1 Задание

Написать программу, строящую полиномиальную кривую по заданным точкам. Обеспечить возможность изменения позиции точек и, при необходимости, значений касательных векторов и натяжения.

7.2 Выполнение

В данной лабораторной работе я использовал модули *matplotlib.pyplot* для построения кривой, *numpy* для работы с данными(в данном случае для обработки координат) и *customtkinter* для отображения всего на интерфейсе.

Квадратичная кривая Безье (n=2) задается опорными точками P0,P1и P2:

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

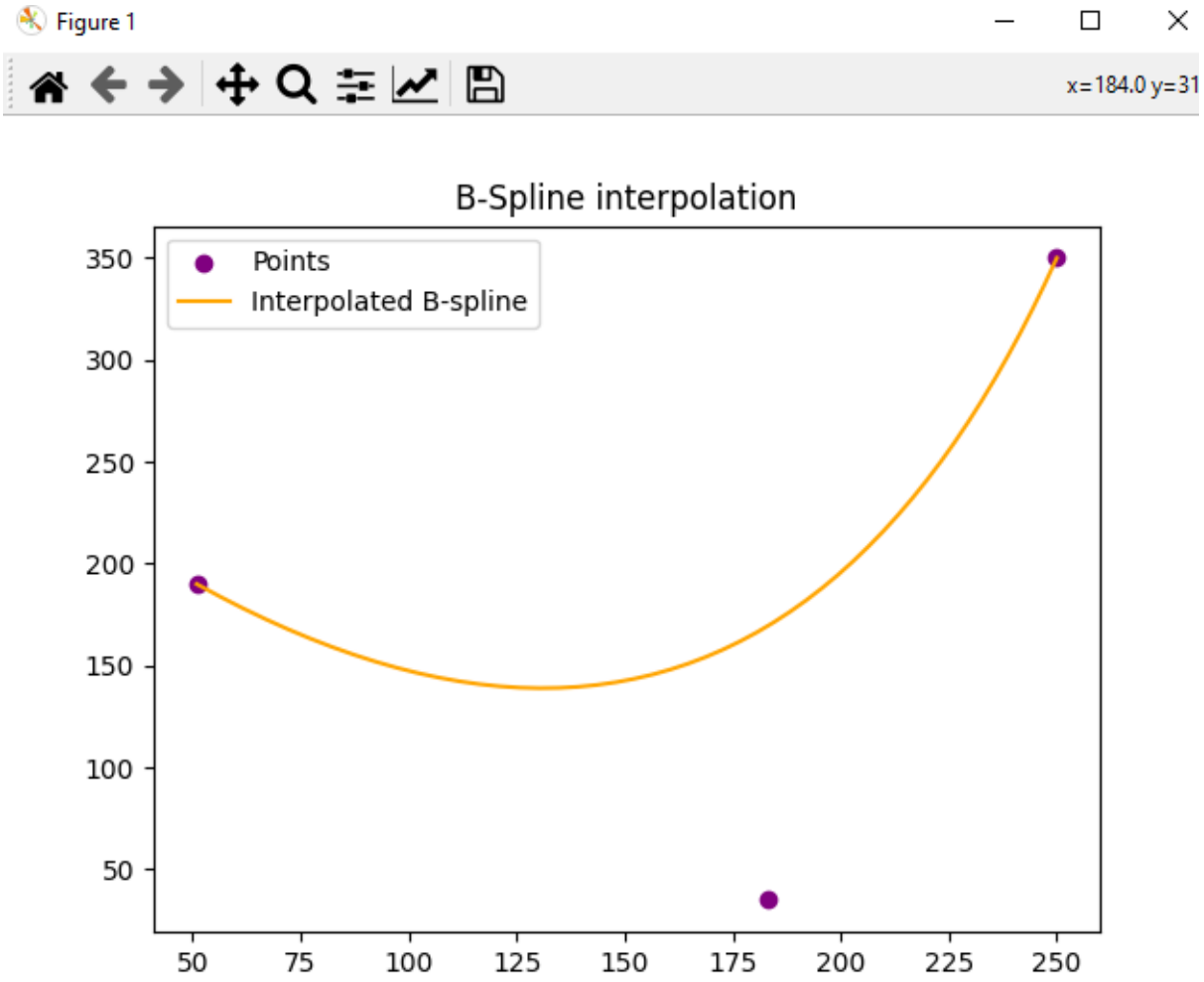
Квадратичные кривые Безье в составе сплайнов используя для описания формулы:

$$t = \frac{P_0 - P_1 \pm \sqrt{(P_0 - 2P_1 + P_2)B + P^2{}_1 - P_0 \cdot P_2}}{P_0 - 2P_1 + P_2},\ P_0 - 2P_1 + P_2 \neq 0$$

$$t = \frac{B - P_0}{2(P_1 - P_0)},\ P_0 - 2P_1 + P_2 = 0,\ P_0 \neq P_1$$

$$t = \sqrt{\frac{B - P_0}{P_2 - P_1}},\ P_0 = P_1 \neq P_2$$

На выходе работы программы получаем такое изображение:



Вывод: удалось написать программу, строящую полиномиальную кривую по заданным точкам и обеспечить возможность изменения позиции точек и, при необходимости, значений касательных векторов и натяжения.

Код для всех лабораторных работ предоставлен в публичном репозитории: https://github.com/HaPPyDutCHoGGG/Comp_Graphics_MAI