

**Московский авиационный институт  
(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»  
Кафедра: 806 «Вычислительная математика и программирование»

**Лабораторная работа № 2  
по курсу «Компьютерная графика»**

Студент:	Валов В.В
Группа:	М8О-308Б-18
Вариант:	
Преподаватель:	Филиппов Г.С.
Оценка:	
Дата:	

Москва, 2020

# **Каркасная визуализация выпуклого многогранника.**

## **Удаление невидимых линий**

### **Постановка задачи**

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.

**Вариант задания:** Правильный октаэдр

### **Решение задачи**

Язык программирования - Python

Библиотеки: tkinter (графическая библиотека), numpy (содержит линейную алгебру).

Для удаления невидимых граней вычисляется нормаль, которая скалярно умножается на позицию наблюдателя. Если скалярное произведение отрицательное - грань невидима.

### **Общие сведения о программе**

vertexes = np.array - массив вершин

sides = np.array - массив индексов, по которым будет строиться треугольник, например строка [0, 1, 4] построит треугольник с вершинами № 0,1,4

rotate\_y - возвращает матрицу поворота по оси y

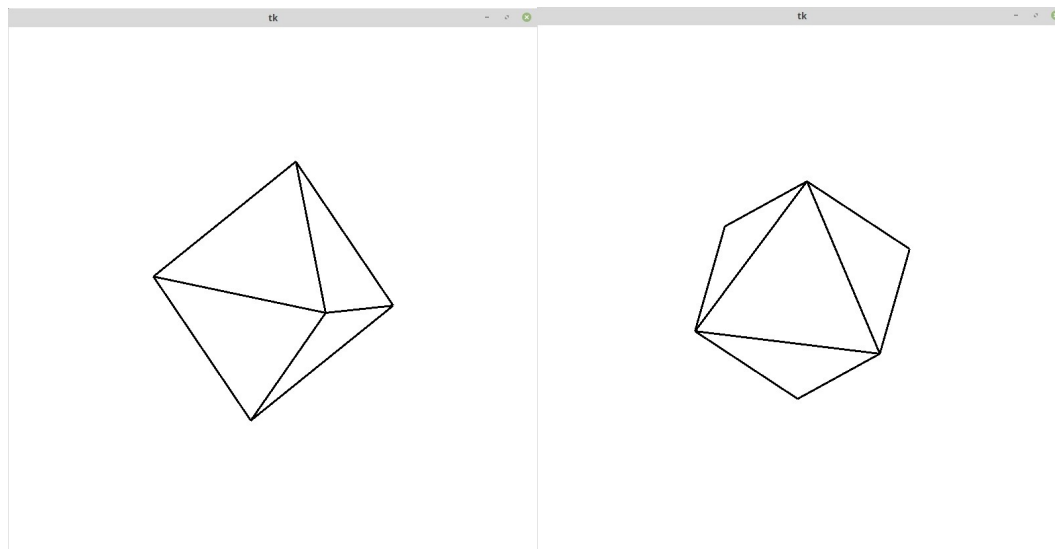
norm\_to\_side - функция нормали к стороне

project - проекция на экранные координаты

### **Руководство по использованию программы**

~:\$ python3 "lab2.py"

Вращение фигуры с помощью клавиш A, S, D, W.



## Код программы

```
import numpy as np
from tkinter import Tk, Canvas
user_pos = np.array([1, 0, 0], dtype=np.float64)
unit_seq = 200
vertexes = np.array([
    [+0, +1, +0], # 0
    [+1, +0, +0], # 1
    [+0, -1, +0], # 2
    [-1, +0, +0], # 3
    [+0, +0, +1], # 4
    [+0, +0, -1] # 5
], dtype=np.float64)
number_of_sides = 8
sides = np.array([
    [0, 1, 4], # 0
    [1, 2, 4], # 1
    [2, 3, 4], # 2
    [3, 0, 4], # 3
    [5, 1, 0], # 4
    [5, 2, 1], # 5
```

```

[5, 3, 2], # 6
[5, 0, 3] # 7
], dtype=np.uint8)
def rotate_y(angle):
    c = np.cos(angle)
    s = np.sin(angle)
    return np.array([
        [c, -s, 0],
        [s, c, 0],
        [0, 0, 1]
    ])
def rotate_z(angle):
    c = np.cos(angle)
    s = np.sin(angle)
    return np.array([
        [c, 0, s],
        [0, 1, 0],
        [-s, 0, c]
    ])
def norm_to_side(side_index):
    p0, p1, p2 = sides[side_index]
    v1 = vertexes[p2] - vertexes[p1]
    v2 = vertexes[p0] - vertexes[p1]
    return np.cross(v2, v1)
def project(vertex):
    return 400 + unit_seq * vertex[2], 400 - unit_seq * vertex[1]
def draw(c):
    for side_index in range(number_of_sides):

        x0, y0 = project(vertexes[sides[side_index, 0]])
        x1, y1 = project(vertexes[sides[side_index, 1]])

```

```

x2, y2 = project(vertexes[sides[side_index, 2]])

if np.dot(norm_to_side(side_index), user_pos) > 0:

    c.create_line(x0, y0, x1, y1, width=3)
    c.create_line(x1, y1, x2, y2, width=3)
    c.create_line(x2, y2, x0, y0, width=3)

# else:
#     c.create_line(x0, y0, x1, y1, width=3, dash=(10, 10))
#     c.create_line(x1, y1, x2, y2, width=3, dash=(10, 10))
#     c.create_line(x2, y2, x0, y0, width=3, dash=(10, 10))
def main():
    def right_arrow(event):
        global vertexes

        vertexes = np.dot(rotate_z(0.05), vertexes.T).T
        canvas.delete('all')
        draw(canvas)

    def left_arrow(event):
        global vertexes

        vertexes = np.dot(rotate_z(-0.05), vertexes.T).T
        canvas.delete('all')
        draw(canvas)

    def up_arrow(event):
        global vertexes

        vertexes = np.dot(rotate_y(0.05), vertexes.T).T

```

```

    canvas.delete('all')
    draw(canvas)

def down_arrow(event):
    global vertexes

    vertexes = np.dot(rotate_y(-0.05), vertexes.T).T
    canvas.delete('all')
    draw(canvas)

root = Tk()

root.bind('d', right_arrow)
root.bind('a', left_arrow)
root.bind('w', up_arrow)
root.bind('s', down_arrow)
canvas = Canvas(root, width=800, height=800, bg='white')
canvas.pack()
draw(canvas)
root.geometry("800x800")
root.mainloop()

if __name__ == '__main__':
    main()

```

## Вывод

Я научился строить выпуклый многогранник, удалять невидимые линии и ознакомился с библиотекой tkinter.