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

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

# Лабораторная работа № 2

## Тема: Каркасная визуализация выпуклого многогранника. Удаление невидимых линий.

Студент: Махмудов Орхан
Группа: О8-305
Преподаватель: Чернышов Л.Н.
Дата:
Оценка:

Москва, 2020

1. 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 №8** - 8-гранная прямая правильная пирамида

2. Описание программы

**Язык программирования:** Python

**Используемые библиотеки**: numpy, math, tkinter, cv2, PIL, functools

**Используемая среда программирования:** Visual Studio Code

Запускаем программу, рисуется 8-ми гранная правильная пирамида, которую можно крутить в трёх плоскостях x,y,z.

3. Набор тестов

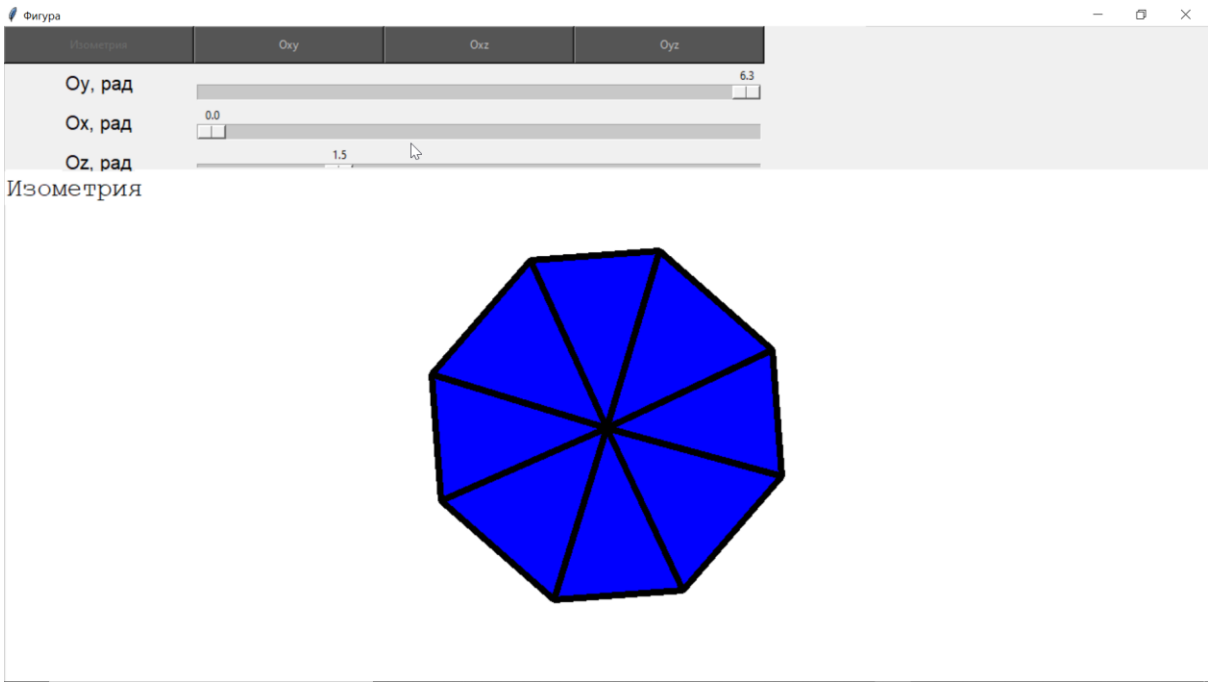
**Тест №1 (вид сверху)**

**Тест №2 (вид сбоку)**

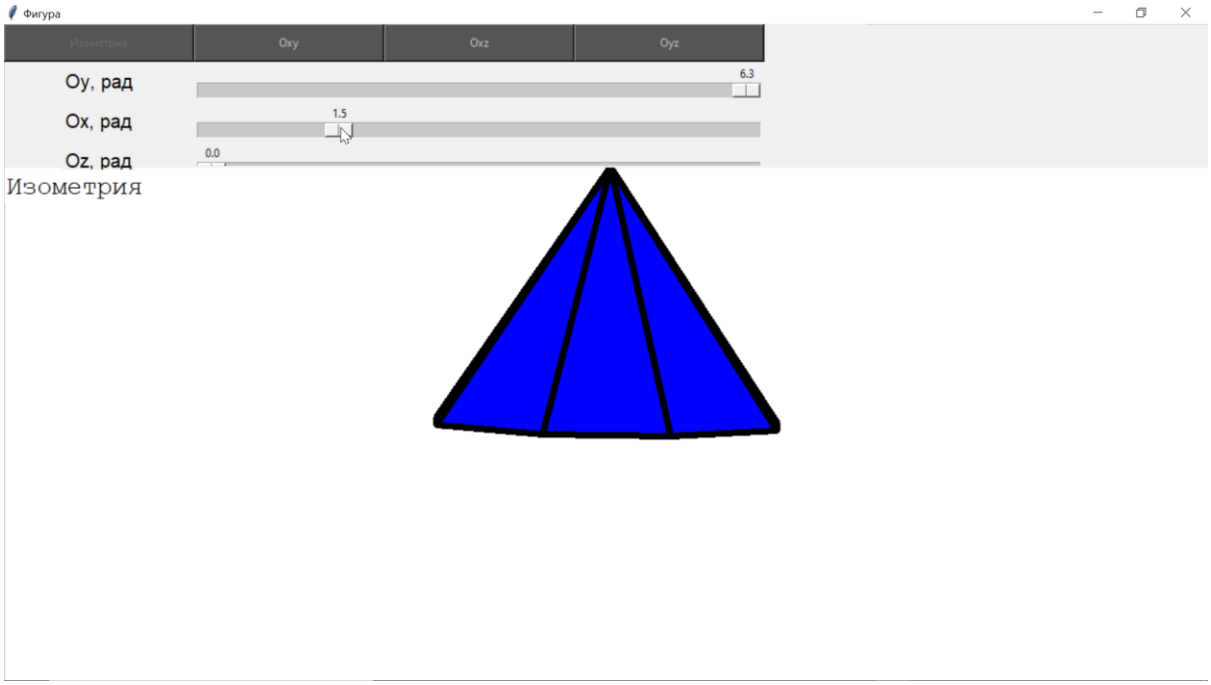
**Тест №3 (вид снизу)**

4. Результаты выполнения тестов

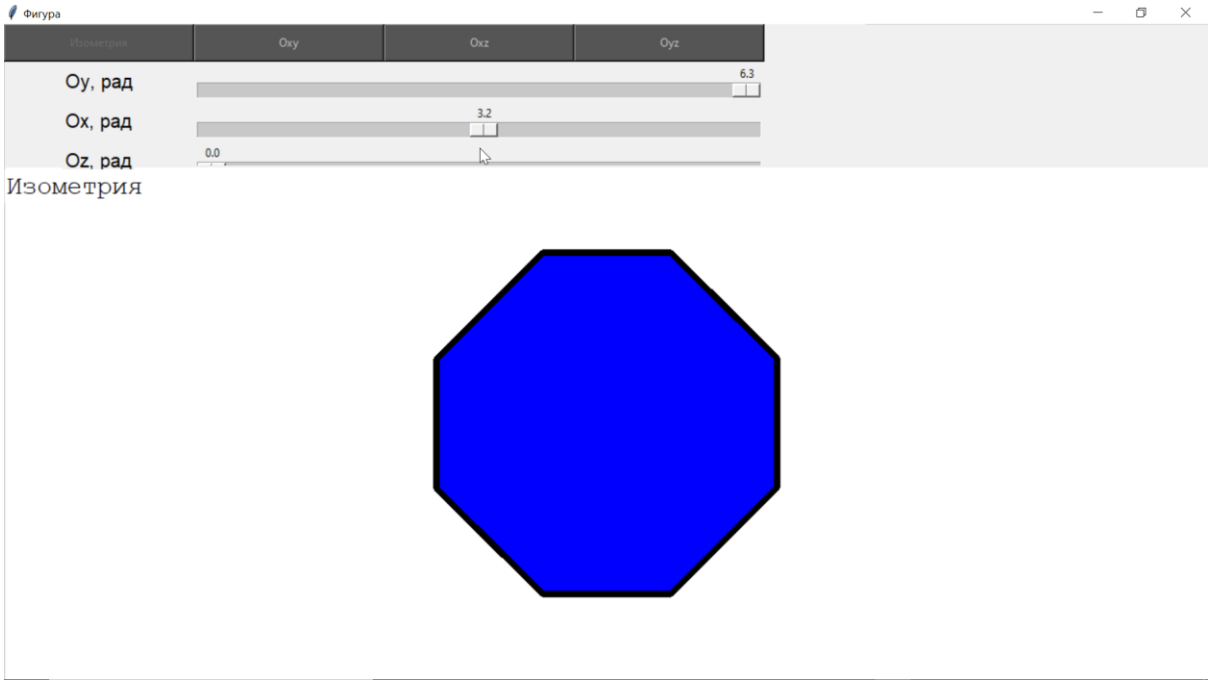
## Тест №1 (вид сверху)




## Тест №2 (вид сбоку)




## Тест №3 (вид снизу)




## 5. Листинг программы

```
#Махмудов                Орхан                группа                М8О-305Б-18
import cv2
import numpy as np
import math
from tkinter import *
from PIL import ImageTk, Image
import os
import functools

def sort_gr(el1, el2):
    ell1 = el1.copy()
    ell2 = el2.copy()
    for i in el1:
        if i in el2:
            ell1.remove(i)
            ell2.remove(i)

    e1 = []
    e2 = []
    for i in ell1:
        e1.append(ppp[i][2])
    for i in ell2:
        e2.append(ppp[i][2])
    e1 = sorted(e1)
    e2 = sorted(e2)

    if (len(ell1) == 1 or len(ell2) == 1):
        if ((e1[len(e1) // 2] + e1[len(e1) // 2 - 1]) / 2 < e2[0]):
            return -1
        elif (e1[len(e1) // 2] == e2[0]):
            return 0
        else:
```

```
            return 1

    if (len(ell1) == 2 or len(ell2) == 2):
        if (e1[0] < e2[0]):
            return -1
        elif(e1[0] > e2[0]):
            return 1
        else:
            if (e1[1] < e2[1]):
                return -1
            elif (e1[1] == e2[1]):
                return 0
            else:
                return 1
    if (e1[0] < e2[0]):
        return -1
    elif(e1[0] > e2[0]):
        return 1
    else:
        if (e1[1] < e2[1]):
            return -1
        elif(e1[1] > e2[1]):
            return 1
        else:
            if (e1[2] < e2[2]):
                return -1
            elif(e1[2] > e2[2]):
                return 1
            else:
                return 0


def white_img(img):
    img[:] = (255, 255, 255)

def points(w,h):
    ss = min(w, h)
    k1 = int(200 * ss / 600)
    k2 = int(153 * ss / 600)
    pnt1 = (( - k1), int( - k2 / 2), 0)
    pnt2 = (( - k1), int( + k2 / 2), 0)
    pnt3 = (( + k1), int( - k2 / 2), 0)
    pnt4 = (( + k1), int( + k2 / 2), 0)
    pnt5 = (int( - k2 / 2), int( + k1), 0)
    pnt6 = (int( + k2 / 2), int( + k1), 0)
    pnt7 = (int( - k2 / 2), int( - k1), 0)
    pnt8 = (int( + k2 / 2), int( - k1), 0)
    pnt9 = (0, 0, ss / 2)
    return (pnt1, pnt2, pnt3, pnt4, pnt5, pnt6, pnt7, pnt8, pnt9)

def turnX(points, angle):
    pnt1  =  (int(points[0][0]),  int(points[0][1]  *  math.cos(angle)  -
points[0][2]  *  math.sin(angle)),  int(points[0][1]  *  math.sin(angle)  +
points[0][2] * math.cos(angle)))
    pnt2  =  (int(points[1][0]),  int(points[1][1]  *  math.cos(angle)  -
points[1][2]  *  math.sin(angle)),  int(points[1][1]  *  math.sin(angle)  +
points[1][2] * math.cos(angle)))
    pnt3  =  (int(points[2][0]),  int(points[2][1]  *  math.cos(angle)  -
points[2][2]  *  math.sin(angle)),  int(points[2][1]  *  math.sin(angle)  +
points[2][2] * math.cos(angle)))
    pnt4  =  (int(points[3][0]),  int(points[3][1]  *  math.cos(angle)  -
```

```
points[3][2] * math.sin(angle)), int(points[3][1] * math.sin(angle) +
points[3][2] * math.cos(angle)))
    pnt5 = (int(points[4][0]), int(points[4][1] * math.cos(angle) -
points[4][2] * math.sin(angle)), int(points[4][1] * math.sin(angle) +
points[4][2] * math.cos(angle)))
    pnt6 = (int(points[5][0]), int(points[5][1] * math.cos(angle) -
points[5][2] * math.sin(angle)), int(points[5][1] * math.sin(angle) +
points[5][2] * math.cos(angle)))
    pnt7 = (int(points[6][0]), int(points[6][1] * math.cos(angle) -
points[6][2] * math.sin(angle)), int(points[6][1] * math.sin(angle) +
points[6][2] * math.cos(angle)))
    pnt8 = (int(points[7][0]), int(points[7][1] * math.cos(angle) -
points[7][2] * math.sin(angle)), int(points[7][1] * math.sin(angle) +
points[7][2] * math.cos(angle)))
    pnt9 = (int(points[8][0]), int(points[8][1] * math.cos(angle) -
points[8][2] * math.sin(angle)), int(points[8][1] * math.sin(angle) +
points[8][2] * math.cos(angle)))
    return (pnt1, pnt2, pnt3, pnt4, pnt5, pnt6, pnt7, pnt8, pnt9)

def turnY(points, angle):
    pnt1 = (int(points[0][0] * math.cos(angle) + points[0][2] *
math.sin(angle)), int(points[0][1]), int(-points[0][0] * math.sin(angle) +
points[0][2] * math.cos(angle)))
    pnt2 = (int(points[1][0] * math.cos(angle) + points[1][2] *
math.sin(angle)), int(points[1][1]), int(-points[1][0] * math.sin(angle) +
points[1][2] * math.cos(angle)))
    pnt3 = (int(points[2][0] * math.cos(angle) + points[2][2] *
math.sin(angle)), int(points[2][1]), int(-points[2][0] * math.sin(angle) +
points[2][2] * math.cos(angle)))
    pnt4 = (int(points[3][0] * math.cos(angle) + points[3][2] *
math.sin(angle)), int(points[3][1]), int(-points[3][0] * math.sin(angle) +
points[3][2] * math.cos(angle)))
    pnt5 = (int(points[4][0] * math.cos(angle) + points[4][2] *
math.sin(angle)), int(points[4][1]), int(-points[4][0] * math.sin(angle) +
points[4][2] * math.cos(angle)))
    pnt6 = (int(points[5][0] * math.cos(angle) + points[5][2] *
math.sin(angle)), int(points[5][1]), int(-points[5][0] * math.sin(angle) +
points[5][2] * math.cos(angle)))
    pnt7 = (int(points[6][0] * math.cos(angle) + points[6][2] *
math.sin(angle)), int(points[6][1]), int(-points[6][0] * math.sin(angle) +
points[6][2] * math.cos(angle)))
    pnt8 = (int(points[7][0] * math.cos(angle) + points[7][2] *
math.sin(angle)), int(points[7][1]), int(-points[7][0] * math.sin(angle) +
points[7][2] * math.cos(angle)))
    pnt9 = (int(points[8][0] * math.cos(angle) + points[8][2] *
math.sin(angle)), int(points[8][1]), int(-points[8][0] * math.sin(angle) +
points[8][2] * math.cos(angle)))
    return (pnt1, pnt2, pnt3, pnt4, pnt5, pnt6, pnt7, pnt8, pnt9)

def turnZ(points, angle):
    pnt1 = (int(points[0][0] * math.cos(angle) - points[0][1] *
math.sin(angle)), int(points[0][0] * math.sin(angle) + points[0][1] *
math.cos(angle)), int(points[0][2]))
    pnt2 = (int(points[1][0] * math.cos(angle) - points[1][1] *
math.sin(angle)), int(points[1][0] * math.sin(angle) + points[1][1] *
math.cos(angle)), int(points[1][2]))
    pnt3 = (int(points[2][0] * math.cos(angle) - points[2][1] *
math.sin(angle)), int(points[2][0] * math.sin(angle) + points[2][1] *
math.cos(angle)), int(points[2][2]))
    pnt4 = (int(points[3][0] * math.cos(angle) - points[3][1] *
math.sin(angle)), int(points[3][0] * math.sin(angle) + points[3][1] *
```

```
math.cos(angle)), int(points[3][2]))
    pnt5   =   (int(points[4][0]   *   math.cos(angle)   -   points[4][1]   *
math.sin(angle)),  int(points[4][0]  *  math.sin(angle)  +  points[4][1]  *
math.cos(angle)), int(points[4][2]))
    pnt6   =   (int(points[5][0]   *   math.cos(angle)   -   points[5][1]   *
math.sin(angle)),  int(points[5][0]  *  math.sin(angle)  +  points[5][1]  *
math.cos(angle)), int(points[5][2]))
    pnt7   =   (int(points[6][0]   *   math.cos(angle)   -   points[6][1]   *
math.sin(angle)),  int(points[6][0]  *  math.sin(angle)  +  points[6][1]  *
math.cos(angle)), int(points[6][2]))
    pnt8   =   (int(points[7][0]   *   math.cos(angle)   -   points[7][1]   *
math.sin(angle)),  int(points[7][0]  *  math.sin(angle)  +  points[7][1]  *
math.cos(angle)), int(points[7][2]))
    pnt9   =   (int(points[8][0]   *   math.cos(angle)   -   points[8][1]   *
math.sin(angle)),  int(points[8][0]  *  math.sin(angle)  +  points[8][1]  *
math.cos(angle)), int(points[8][2]))
    return (pnt1, pnt2, pnt3, pnt4, pnt5, pnt6, pnt7, pnt8, pnt9)

def fig(image):
    w,h,c = image.shape
    global ppp
    pnt1 = points(w,h)
    i = sc1.get()
    j = sc2.get()
    k = sc3.get()

    pnt = turnY(pnt1, i)
    pnt = turnX(pnt, j)
    pnt = turnZ(pnt, k)

    white_img(image)

    minz = math.inf
    n = -1
    for x in range(8):
        if(pnt[x][2] < minz and x != 8):
            minz = pnt[5][2]
            n = x

    p = [0,6,7,2,3,5,4,1]

    if(n != 8):
        ind = p.index(n)

    ppp = pnt
    gran = sorted([[0, 6, 8], [6, 7, 8], [7, 2, 8], [2, 3, 8], [3, 5, 8],
[5,    4,    8],    [4,    1,    8],    [1,    0,    8],    [0,1,2,3,4,5,6,7]],
key=functools.cmp_to_key(sort_gr))
    if (vid == 0):

        for zx in gran:
            r = []
            if(len(zx) == 3):
                for i in zx:
                    r.append(
                    (h // 2 + pnt[i][0], w // 2 + pnt[i][1]))
                a3 = np.array( [r],
                     dtype=np.int32 )
            else:
                a3 = np.array( [[
                    [h // 2 + pnt[0][0],  w // 2 + pnt[0][1]],
```

```
                    [h // 2 + pnt[1][0],w // 2 + pnt[1][1]],
                    [h // 2 + pnt[4][0],w // 2 + pnt[4][1]],
                    [h // 2 + pnt[5][0],w // 2 + pnt[5][1]],
                    [h // 2 + pnt[3][0],w // 2 + pnt[3][1]],
                    [h // 2 + pnt[2][0],w // 2 + pnt[2][1]],
                    [h // 2 + pnt[7][0],w // 2 + pnt[7][1]],
                    [h // 2 + pnt[6][0],w // 2 + pnt[6][1]],
                    ]],dtype=np.int32 )
            cv2.fillPoly(image, a3, (255, 0, 0))
            if(len(zx) == 3):
                cv2.line(image,  (h  //  2  +  pnt[zx[0]][0],  w  //  2  +
pnt[zx[0]][1]), (h // 2 + pnt[zx[1]][0], w // 2 + pnt[zx[1]][1]),(0,0,0), 5)
                cv2.line(image,  (h  //  2  +  pnt[zx[0]][0],  w  //  2  +
pnt[zx[0]][1]), (h // 2 + pnt[8][0], w // 2 + pnt[8][1]),(0,0,0), 5)
                cv2.line(image, (h // 2 + pnt[8][0], w // 2 + pnt[8][1]), (h
// 2 + pnt[zx[1]][0], w // 2 + pnt[zx[1]][1]),(0,0,0), 5)
            else:
                cv2.line(image, (h // 2 + pnt[0][0], w // 2 + pnt[0][1]), (h
// 2 + pnt[1][0], w // 2 + pnt[1][1]),(0,0,0), 5)
                cv2.line(image, (h // 2 + pnt[2][0], w // 2 + pnt[2][1]), (h
// 2 + pnt[3][0], w // 2 + pnt[3][1]),(0,0,0), 5)
                cv2.line(image, (h // 2 + pnt[4][0], w // 2 + pnt[4][1]), (h
// 2 + pnt[5][0], w // 2 + pnt[5][1]),(0,0,0), 5)
                cv2.line(image, (h // 2 + pnt[6][0], w // 2 + pnt[6][1]), (h
// 2 + pnt[7][0], w // 2 + pnt[7][1]),(0,0,0), 5)



                cv2.line(image, (h // 2 + pnt[0][0], w // 2 + pnt[0][1]), (h
// 2 + pnt[6][0], w // 2 + pnt[6][1]),(0,0,0), 5)
                cv2.line(image, (h // 2 + pnt[2][0], w // 2 + pnt[2][1]), (h
// 2 + pnt[7][0], w // 2 + pnt[7][1]),(0,0,0), 5)
                cv2.line(image, (h // 2 + pnt[1][0], w // 2 + pnt[1][1]), (h
// 2 + pnt[4][0], w // 2 + pnt[4][1]),(0,0,0), 5)
                cv2.line(image, (h // 2 + pnt[3][0], w // 2 + pnt[3][1]), (h
// 2 + pnt[5][0], w // 2 + pnt[5][1]),(0,0,0), 5)
    elif(vid == 1):
        for zx in gran:
            r = []
            for i in zx:
                r.append(
                (h // 2 + pnt[i][1], w // 2 + pnt[i][2]))
            a3 = np.array( [r],
                 dtype=np.int32 )
            cv2.fillPoly(image, a3, (128, 128, 128))
            a3 = np.array( [[
                [h // 2 + pnt[0][1],  w // 2 + pnt[0][2]],
                [h // 2 + pnt[1][1],w // 2 + pnt[1][2]],
                [h // 2 + pnt[4][1],w // 2 + pnt[4][2]],
                [h // 2 + pnt[5][1],w // 2 + pnt[5][2]],
                [h // 2 + pnt[3][1],w // 2 + pnt[3][2]],
                [h // 2 + pnt[2][1],w // 2 + pnt[2][2]],
                [h // 2 + pnt[7][1],w // 2 + pnt[7][2]],
                [h // 2 + pnt[6][1],w // 2 + pnt[6][2]],
                ]],dtype=np.int32 )
            cv2.fillPoly(image, a3, (128, 128, 128))


        pa = np.array([
            (h // 2 + pnt[0][1], w // 2 + pnt[0][2]), (h // 2 + pnt[6][1],
w // 2 + pnt[6][2]),
```

```
            (h // 2 + pnt[7][1], w // 2 + pnt[7][2]), (h // 2 + pnt[2][1],
w // 2 + pnt[2][2]),
            (h // 2 + pnt[3][1], w // 2 + pnt[3][2]), (h // 2 + pnt[5][1],
w // 2 + pnt[5][2]),
            (h // 2 + pnt[4][1], w // 2 + pnt[4][2]), (h // 2 + pnt[1][1],
w // 2 + pnt[1][2]),
            (h // 2 + pnt[0][1], w // 2 + pnt[0][2])
            ])
        pa2 = np.array([
            (h // 2 + pnt[8][1], w // 2 + pnt[8][2]), (h // 2 + pnt[0][1],
w // 2 + pnt[0][2]),
            (h // 2 + pnt[8][1], w // 2 + pnt[8][2]), (h // 2 + pnt[1][1],
w // 2 + pnt[1][2]),
            (h // 2 + pnt[8][1], w // 2 + pnt[8][2]), (h // 2 + pnt[2][1],
w // 2 + pnt[2][2]),
            (h // 2 + pnt[8][1], w // 2 + pnt[8][2]), (h // 2 + pnt[3][1],
w // 2 + pnt[3][2]),
            (h // 2 + pnt[8][1], w // 2 + pnt[8][2]), (h // 2 + pnt[4][1],
w // 2 + pnt[4][2]),
            (h // 2 + pnt[8][1], w // 2 + pnt[8][2]), (h // 2 + pnt[5][1],
w // 2 + pnt[5][2]),
            (h // 2 + pnt[8][1], w // 2 + pnt[8][2]), (h // 2 + pnt[6][1],
w // 2 + pnt[6][2]),
            (h // 2 + pnt[8][1], w // 2 + pnt[8][2]), (h // 2 + pnt[7][1],
w // 2 + pnt[7][2])
            ])
        cv2.polylines(image, [pa, pa2], False, (128,128,128), 5)
    elif(vid == 2):
        for zx in gran:
            r = []
            for i in zx:
                r.append(
                (h // 2 + pnt[i][0], w // 2 + pnt[i][2]))
            a3 = np.array( [r],
                 dtype=np.int32 )
            cv2.fillPoly(image, a3, ((128, 128, 128)))
            a3 = np.array( [[
                [h // 2 + pnt[0][0],  w // 2 + pnt[0][2]],
                [h // 2 + pnt[1][0],w // 2 + pnt[1][2]],
                [h // 2 + pnt[4][0],w // 2 + pnt[4][2]],
                [h // 2 + pnt[5][0],w // 2 + pnt[5][2]],
                [h // 2 + pnt[3][0],w // 2 + pnt[3][2]],
                [h // 2 + pnt[2][0],w // 2 + pnt[2][2]],
                [h // 2 + pnt[7][0],w // 2 + pnt[7][2]],
                [h // 2 + pnt[6][0],w // 2 + pnt[6][2]],
                ]],dtype=np.int32 )
            cv2.fillPoly(image, a3, ((128, 128, 128)))
        pa3 = np.array([
            (h // 2 + pnt[0][0], w // 2 + pnt[0][2]), (h // 2 + pnt[6][0],
w // 2 + pnt[6][2]),
            (h // 2 + pnt[7][0], w // 2 + pnt[7][2]), (h // 2 + pnt[2][0],
w // 2 + pnt[2][2]),
            (h // 2 + pnt[3][0], w // 2 + pnt[3][2]), (h // 2 + pnt[5][0],
w // 2 + pnt[5][2]),
            (h // 2 + pnt[4][0], w // 2 + pnt[4][2]), (h // 2 + pnt[1][0],
w // 2 + pnt[1][2]),
            (h // 2 + pnt[0][0], w // 2 + pnt[0][2])
            ])
        pa4 = np.array([
            (h // 2 + pnt[8][0], w // 2 + pnt[8][2]), (h // 2 + pnt[0][0],
w // 2 + pnt[0][2]),
```

```
            (h // 2 + pnt[8][0], w // 2 + pnt[8][2]), (h // 2 + pnt[1][0],
w // 2 + pnt[1][2]),
            (h // 2 + pnt[8][0], w // 2 + pnt[8][2]), (h // 2 + pnt[2][0],
w // 2 + pnt[2][2]),
            (h // 2 + pnt[8][0], w // 2 + pnt[8][2]), (h // 2 + pnt[3][0],
w // 2 + pnt[3][2]),
            (h // 2 + pnt[8][0], w // 2 + pnt[8][2]), (h // 2 + pnt[4][0],
w // 2 + pnt[4][2]),
            (h // 2 + pnt[8][0], w // 2 + pnt[8][2]), (h // 2 + pnt[5][0],
w // 2 + pnt[5][2]),
            (h // 2 + pnt[8][0], w // 2 + pnt[8][2]), (h // 2 + pnt[6][0],
w // 2 + pnt[6][2]),
            (h // 2 + pnt[8][0], w // 2 + pnt[8][2]), (h // 2 + pnt[7][0],
w // 2 + pnt[7][2])
            ])
        cv2.polylines(image, [pa3, pa4], False, (128,128,128), 5)
    elif(vid == 3):
        for zx in gran:
            r = []
            for i in zx:
                r.append(
                (h // 2 + pnt[i][0], w // 2 + pnt[i][1]))
            a3 = np.array( [r],
                 dtype=np.int32 )
            cv2.fillPoly(image, a3, (128, 128,128))
            a3 = np.array( [[
                [h // 2 + pnt[0][0],  w // 2 + pnt[0][1]],
                [h // 2 + pnt[1][0],w // 2 + pnt[1][1]],
                [h // 2 + pnt[4][0],w // 2 + pnt[4][1]],
                [h // 2 + pnt[5][0],w // 2 + pnt[5][1]],
                [h // 2 + pnt[3][0],w // 2 + pnt[3][1]],
                [h // 2 + pnt[2][0],w // 2 + pnt[2][1]],
                [h // 2 + pnt[7][0],w // 2 + pnt[7][1]],
                [h // 2 + pnt[6][0],w // 2 + pnt[6][1]],
                ]],dtype=np.int32 )
            cv2.fillPoly(image, a3, (128, 128, 128))
        cv2.line(image, (h // 2 + pnt[0][0], w // 2 + pnt[0][1]), (h // 2 +
pnt[1][0], w // 2 + pnt[1][1]),(128, 128, 128), 5)
        cv2.line(image, (h // 2 + pnt[2][0], w // 2 + pnt[2][1]), (h // 2 +
pnt[3][0], w // 2 + pnt[3][1]),(128, 128, 128), 5)
        cv2.line(image, (h // 2 + pnt[4][0], w // 2 + pnt[4][1]), (h // 2 +
pnt[5][0], w // 2 + pnt[5][1]),(128, 128, 128), 5)
        cv2.line(image, (h // 2 + pnt[6][0], w // 2 + pnt[6][1]), (h // 2 +
pnt[7][0], w // 2 + pnt[7][1]),(128, 128, 128), 5)



        cv2.line(image, (h // 2 + pnt[0][0], w // 2 + pnt[0][1]), (h // 2 +
pnt[6][0], w // 2 + pnt[6][1]),(128, 128, 128), 5)
        cv2.line(image, (h // 2 + pnt[2][0], w // 2 + pnt[2][1]), (h // 2 +
pnt[7][0], w // 2 + pnt[7][1]),(128, 128, 128), 5)
        cv2.line(image, (h // 2 + pnt[1][0], w // 2 + pnt[1][1]), (h // 2 +
pnt[4][0], w // 2 + pnt[4][1]),(128, 128, 128), 5)
        cv2.line(image, (h // 2 + pnt[3][0], w // 2 + pnt[3][1]), (h // 2 +
pnt[5][0], w // 2 + pnt[5][1]),(128, 128, 128), 5)


        cv2.line(image, (h // 2 + pnt[0][0], w // 2 + pnt[0][1]), (h // 2 +
pnt[8][0], w // 2 + pnt[8][1]),(128, 128, 128), 5)
        cv2.line(image, (h // 2 + pnt[2][0], w // 2 + pnt[2][1]), (h // 2 +
pnt[8][0], w // 2 + pnt[8][1]),(128, 128, 128), 5)
```

```
        cv2.line(image, (h // 2 + pnt[1][0], w // 2 + pnt[1][1]), (h // 2 +
pnt[8][0], w // 2 + pnt[8][1]),(128, 128, 128), 5)
        cv2.line(image, (h // 2 + pnt[3][0], w // 2 + pnt[3][1]), (h // 2 +
pnt[8][0], w // 2 + pnt[8][1]),(128, 128, 128), 5)

        cv2.line(image, (h // 2 + pnt[6][0], w // 2 + pnt[6][1]), (h // 2 +
pnt[8][0], w // 2 + pnt[8][1]),(128, 128, 128), 5)
        cv2.line(image, (h // 2 + pnt[7][0], w // 2 + pnt[7][1]), (h // 2 +
pnt[8][0], w // 2 + pnt[8][1]),(128, 128, 128), 5)
        cv2.line(image, (h // 2 + pnt[4][0], w // 2 + pnt[4][1]), (h // 2 +
pnt[8][0], w // 2 + pnt[8][1]),(128, 128, 128), 5)
        cv2.line(image, (h // 2 + pnt[5][0], w // 2 + pnt[5][1]), (h // 2 +
pnt[8][0], w // 2 + pnt[8][1]),(128, 128, 128), 5)




def update_image(event):
    global image
    global img

    image  =  np.zeros((root.winfo_height()  -  150,  root.winfo_width(),3),
np.uint8)
    white_img(image)
    fig(image)

    fig(image)

    image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
    img = Image.fromarray(image)
    img = ImageTk.PhotoImage(img)

    canvas.configure(width=root.winfo_width(),height=root.winfo_height()  -
150)

    canvas.itemconfigure(imagesprite, image=img)

def setOxy():
    global vid
    vid = 3
    btn1.configure(state=NORMAL)
    btn2.configure(state=DISABLED)
    btn3.configure(state=NORMAL)
    btn4.configure(state=NORMAL)
    label4.configure(text="Oxy")
    update_image(0)
def setOxz():
    global vid
    vid = 2
    btn1.configure(state=NORMAL)
    btn2.configure(state=NORMAL)
    btn3.configure(state=DISABLED)
    btn4.configure(state=NORMAL)
    label4.configure(text="Oxz")
    update_image(0)
def setOyz():
    global vid
    vid = 1
    btn1.configure(state=NORMAL)
    btn2.configure(state=NORMAL)
```

```
    btn3.configure(state=NORMAL)
    btn4.configure(state=DISABLED)
    label4.configure(text="Oyz")
    update_image(0)
def setIz():
    global vid
    vid = 0
    btn1.configure(state=DISABLED)
    btn2.configure(state=NORMAL)
    btn3.configure(state=NORMAL)
    btn4.configure(state=NORMAL)
    label4.configure(text="Изометрия")
    update_image(0)
if __name__ == "__main__":
    ppp = []
    vid = 0
    height = 900
    width  = 900
    root=Tk()
    root.title("Фигура")
    root.minsize(850,400)
    root.maxsize(1800,1000)
    root.wm_geometry("%dx%d+%d+%d" % (width, height, 0, 0))
    image = np.zeros((900,900,3), np.uint8)
    white_img(image)


    sc1 = Scale(root, from_=0.0, to=3.14 * 2, resolution = 0.1,
orient=HORIZONTAL, length=600, command=update_image)
    sc2 = Scale(root, from_=0.0, to=3.14 * 2, resolution = 0.1,
orient=HORIZONTAL, length=600, command=update_image)
    sc3 = Scale(root, from_=0.0, to=3.14 * 2, resolution = 0.1,
orient=HORIZONTAL, length=600, command=update_image)
    sc1.grid(row=1, column=1, columnspan=3)
    sc2.grid(row=2, column=1, columnspan=3)
    sc3.grid(row=3, column=1, columnspan=3)

    canvas=Canvas(root)
    canvas.place(x=0,y=150)
    frame=Frame(canvas)
    canvas.create_window((0,0),window=frame,anchor='nw')
    fig(image)

    image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
    img = Image.fromarray(image)
    img = ImageTk.PhotoImage(img)

    imagesprite = canvas.create_image(0,0, anchor=NW, image=img)


    root.bind("<Configure>",update_image)

    btn1 = Button(root, text="Изометрия",
              width=22,
              background="#555",
              foreground="#ccc",
              padx="20",
              pady="8",
              command=setIz,
              state=DISABLED
          )
```

```
    btn1.grid(row=0, column=0)


    btn2 = Button(root, text="Oxy",
              width=22,
              background="#555",
              foreground="#ccc",
              padx="20",
              pady="8",
              command=setOxy
        )
    btn2.grid(row=0, column=1)


    btn3 = Button(root, text="Oxz",
              width=22,
              background="#555",
              foreground="#ccc",
              padx="20",
              pady="8",
              command=setOxz
        )
    btn3.grid(row=0, column=2)

    btn4 = Button(root, text="Oyz",
              width=22,
              background="#555",
              foreground="#ccc",
              padx="20",
              pady="8",
              command=setOyz
        )
    # btn3.pack()
    btn4.grid(row=0, column=3)
    label1 = Label(root, text="Oy, рад", font="16")
    label1.grid(row = 1, column=0)
    label2 = Label(root, text="Ox, рад", font="16")
    label2.grid(row = 2, column=0)
    label3 = Label(root, text="Oz, рад", font="16")
    label3.grid(row = 3, column=0)

    label4    =    Label(root,    text="Изометрия",    font=("Courier",    20),
background="#FFF")
    label4.place(x=0, y=154)

    root.mainloop()
```