МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ
УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И ТЕХНОЛОГИЧЕСКОГО ОБРАЗОВАНИЯ
Кафедра информационных технологий и электронного обучения

ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №2

По дисциплине: Математические основы компьютерной графики

Выполнил студент 3 курса группы 1.1

Щербинин А. В.

Санкт-Петербург
2022

Постановка задачи:

1. **Комплект 1:**
   - **1.1.** Напишите программу, которая по материалам лекции преобразовывает концы отрезка
   $$L = \begin{pmatrix} 0 & 100 \\ 200 & 300 \end{pmatrix}$$
   по заданной матрице преобразования
   $$T = \begin{pmatrix} 1 & 2 \\ 3 & 1 \end{pmatrix}$$
   Затем рассчитайте середину начального отрезка и с помощью матрицы преобразования найдите середину нового отрезка. Отрезки и середины отрезков прорисуйте на экране с помощью библиотеки pygame.
   - **1.2.** Напишите программу, которая по материалам лекции преобразовывает параллельные отрезки, заданные матрицей (первые две строки - координаты первого отрезка, вторые две координаты второго)
   $$L = \begin{pmatrix} 50 & 100 \\ 250 & 200 \\ 50 & 200 \\ 250 & 300 \end{pmatrix}$$
   по заданной матрице преобразования
   $$T = \begin{pmatrix} 1 & 2 \\ 3 & 1 \end{pmatrix}$$
   Затем рассчитайте начальный наклон отрезков и конечный наклон отрезков с дополнительной проверкой по формулам из лекции. Прорисуйте всё на экране с помощью библиотеки pygame.
   - **1.3.** Напишите программу, которая по материалам лекции преобразовывает пересекающиеся отрезки, заданные матрицей (первые две строки - координаты первого отрезка, вторые две координаты второго)
   $$L = \begin{pmatrix} -1/2 & 3/2 \\ 3 & -2 \\ -1 & -1 \\ 3 & 5/3 \end{pmatrix}$$
   по заданной матрице преобразования
   $$T = \begin{pmatrix} 1 & 2 \\ 1 & -3 \end{pmatrix}$$
   Предварительно умножьте L на 100 и перед каждой отрисовкой на экране сместите отрезки на 100–200 пикселей в положительных направлениях по осям OX OY на экране компьютера для лучшего отображения. Прорисуйте всё на экране с помощью библиотеки pygame.
   - **1.4.** Напишите программу, которая по материалам лекции вращает фигуру треугольник на 90 градусов против часовой стрелки относительно начала координат
   $$L = \begin{pmatrix} 3 & -1 \\ 4 & 1 \\ 2 & 1 \end{pmatrix}$$

по заданной матрице преобразования

$$T = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

Предварительно умножьте L на 100 и перед каждой отрисовкой на экране сместите все координаты на 100–200 пикселей в положительных направлениях по осям OX OY на экране компьютера для лучшего отображения. Прорисуйте всё на экране с помощью библиотеки pygame.

Напишите программу, которая по материалам лекции отражает фигуру треугольник относительно прямой $y = x$

$$L = \begin{pmatrix} 8 & 1 \\ 7 & 3 \\ 6 & 2 \end{pmatrix}$$

по заданной матрице преобразования

$$T = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Предварительно умножьте L на 100 и перед каждой отрисовкой на экране сместите все координаты на 100–200 пикселей в положительных направлениях по осям OX OY на экране компьютера для лучшего отображения. Прорисуйте всё на экране с помощью библиотеки pygame.

1.5. Напишите программу, которая по материалам лекции масштабирует со смещением фигуру треугольник, заданный матрицей

$$L = \begin{pmatrix} 5 & 1 \\ 5 & 2 \\ 3 & 2 \end{pmatrix}$$

по заданной матрице преобразования

$$T = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$

Предварительно умножьте L на 100 и перед каждой отрисовкой на экране сместите все координаты на 100–200 пикселей в положительных направлениях по осям OX OY на экране компьютера для лучшего отображения. Прорисуйте всё на экране с помощью библиотеки pygame.

Комплект 1: Задачи для самостоятельной работы

Код программы

```
import numpy as np
import pygame as pg
from pprint import pprint as pp
import sys
from pygame.locals import *


# Переменные цвета
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
RED = (255, 0, 0)
```

```
GREEN = (0, 255, 0)
BLUE = (0, 0, 255)


# Инициализируем игровую библиотеку, игровое окно и один из шрифтов для
надписей в окне
pg.init()
pg.font.init()
window = pg.display.set_mode((0, 0), FULLSCREEN)
window.fill((255, 255, 255))
my_game_font = pg.font.SysFont('Fira Code', 40)
pg.display.update()


def main():
    menu()
    FPS = 30
    clock = pg.time.Clock()
    while True:
        pressed_keys = pg.key.get_pressed()
        for event in pg.event.get():
            if (event.type == QUIT) or (pressed_keys[K_ESCAPE]):
                pg.quit()
                sys.exit()
            if pressed_keys[K_1]:
                window.fill(WHITE)
                lab2_1()
            if pressed_keys[K_2]:
                window.fill(WHITE)
                lab2_2()
            if pressed_keys[K_3]:
                window.fill(WHITE)
                lab2_3()
            if pressed_keys[K_4]:
                window.fill(WHITE)
                lab2_4()
            if pressed_keys[K_5]:
                window.fill(WHITE)
                lab2_5()
            if pressed_keys[K_6]:
                window.fill(WHITE)
                lab2_6()
            if pressed_keys[K_0]:
                window.fill(WHITE)
                menu()
        clock.tick(FPS)
        pg.display.update()
```

```
# "Главное меню"
def menu():
    window.blit(
        my_game_font.render(
            "'1-6' - выбор задания",
            True, BLACK), (5, 5))
    window.blit(
        my_game_font.render(
            "'0' - возвращение в меню",
            True, BLACK), (5, 45))
    window.blit(
        my_game_font.render(
            "'Escape' - выход из программы",
            True, BLACK), (5, 85))
    pg.display.update()


# Самостоятельная работа 1
def lab2_1():
    L = np.array([[0, 100], [200, 300]])
    T = np.array([[1, 2], [3, 1]])
    print("L:")
    mat_out(L)
    print("T:")
    mat_out(T)
    L1 = L @ T
    print("L1:")
    mat_out(L1)
    mid_point = np.array([(L[0, 0] + L[1, 0]) / 2, (L[0, 1] + L[1, 1]) / 2])
    print("Mid point of L:")
    mat_out(mid_point)
    mid_point_1 = mid_point @ T
    print("Mid point of L1:")
    mat_out(mid_point_1)
    pg.draw.lines(window, BLUE, True, L, 3)
    pg.draw.circle(window, BLUE, mid_point, 4, 0)
    pg.draw.lines(window, RED, True, L1, 3)
    pg.draw.circle(window, RED, mid_point_1, 4, 0)
    pg.display.update()


# Самостоятельная работа 2
def lab2_2():
    L = np.array([[50, 100], [250, 200], [50, 200], [250, 300]])
    T = np.array([[1, 2], [3, 1]])
    print("L:")
    mat_out(L)
    print("T:")
    mat_out(T)
```

```
    L1 = L @ T
    print("L1:")
    mat_out(L1)
    m1_1 = (L[1][1] - L[0][1]) / (L[1][0] - L[0][0])
    m1_2 = (L[3][1] - L[2][1]) / (L[3][0] - L[2][0])
    m2_1 = (T[0][1] + T[1][1] * m1_1) / (T[0][0] + T[1][0] * m1_1)
    m2_2 = (T[0][1] + T[1][1] * m1_2) / (T[0][0] + T[1][0] * m1_2)
    print(
        f"For the first line m1 = {m1_1}, for the second line m1 = {m1_2} =>
These lines are parallel"
    )
    print(
        f"For the first line m2 = {m2_1}, for the second line m2 = {m2_2} =>
These lines are parallel"
    )
    for i in range(0, 4, 2):
        pg.draw.lines(window, BLUE, True, (L[i], L[i + 1]), 3)
        pg.draw.lines(window, RED, True, (L1[i], L1[i + 1]), 3)
    pg.display.update()


# Самостоятельная работа 3
def lab2_3():
    L = np.array([[-0.5, 1.5], [3, -2], [-1, -1], [3, 5 / 3]])
    T = np.array([[1, 2], [1, -3]])
    print("L:")
    mat_out(L)
    print("T:")
    mat_out(T)
    L1 = L @ T
    print("L1:")
    mat_out(L1)
    Lex = np.copy(L) * 100 + 200
    L1ex = np.copy(L1) * 100 + 200
    for i in range(0, 4, 2):
        pg.draw.lines(window, BLUE, True, (Lex[i], Lex[i + 1]), 3)
        pg.draw.lines(window, RED, True, (L1ex[i], L1ex[i + 1]), 3)
    pg.display.update()


# Самостоятельная работа 4
def lab2_4():
    L = np.array([[3, -1], [4, 1], [2, 1]])
    T = np.array([[0, 1], [-1, 0]])
    print("L:")
    mat_out(L)
    print("T:")
    mat_out(T)
    L1 = L @ T
```

```
    print("L1:")
    mat_out(L1)
    Lex = np.copy(L) * 100 + 200
    L1ex = np.copy(L1) * 100 + 200
    pg.draw.lines(window, BLUE, True, Lex, 3)
    pg.draw.lines(window, RED, True, L1ex, 3)
    pg.display.update()


# Самостоятельная работа 5
def lab2_5():
    L = np.array([[8, 1], [7, 3], [6, 2]])
    T = np.array([[0, 1], [1, 0]])
    print("L:")
    mat_out(L)
    print("T:")
    mat_out(T)
    L1 = L @ T
    print("L1:")
    mat_out(L1)
    Lex = np.copy(L) * 100 + 50
    L1ex = np.copy(L1) * 100 + 50
    mat_out(L1ex)
    pg.draw.lines(window, BLUE, True, Lex, 3)
    pg.draw.lines(window, RED, True, L1ex, 3)
    pg.display.update()


# Самостоятельная работа 6
def lab2_6():
    L = np.array([[5, 1], [5, 2], [3, 2]])
    T = np.array([[2, 0], [0, 2]])
    print("L:")
    mat_out(L)
    print("T:")
    mat_out(T)
    L1 = L @ T
    print("L1:")
    mat_out(L1)
    Lex = np.copy(L) * 100 + 200
    L1ex = np.copy(L1) * 100 + 200
    pg.draw.lines(window, BLUE, True, Lex, 3)
    pg.draw.lines(window, RED, True, L1ex, 3)
    pg.display.update()


# Функция вывода матрицы
def mat_out(n):
    print("-------------")
```

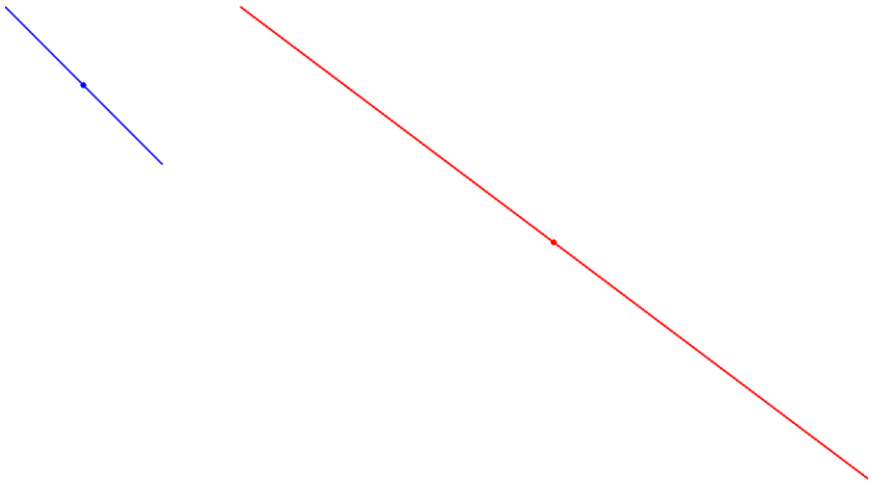
```
    pp(n)
    print("-------------\n")


main()
```

Результаты выполнения программы

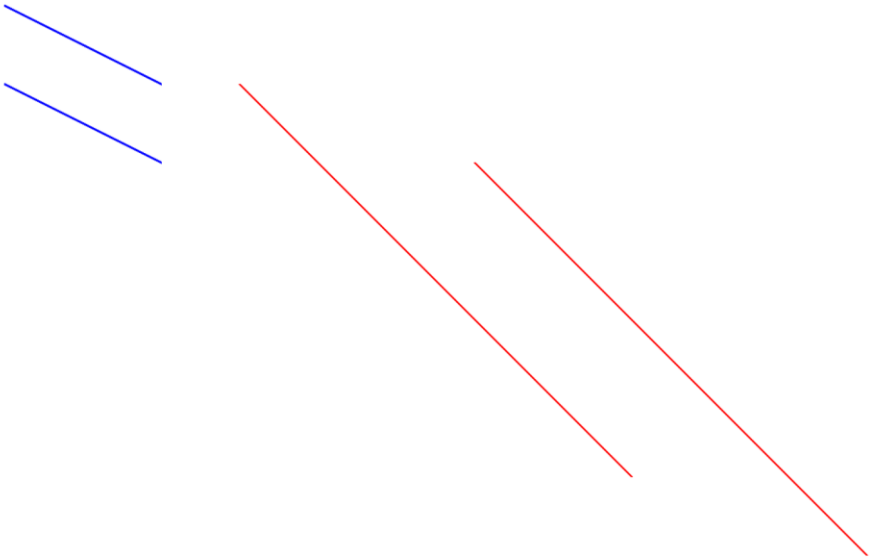
Меню программы (можно выбрать задание, нажав на соответствующую цифру):

'1-6' - выбор задания
'0' - возвращение в меню
'Escape' - выход из программы

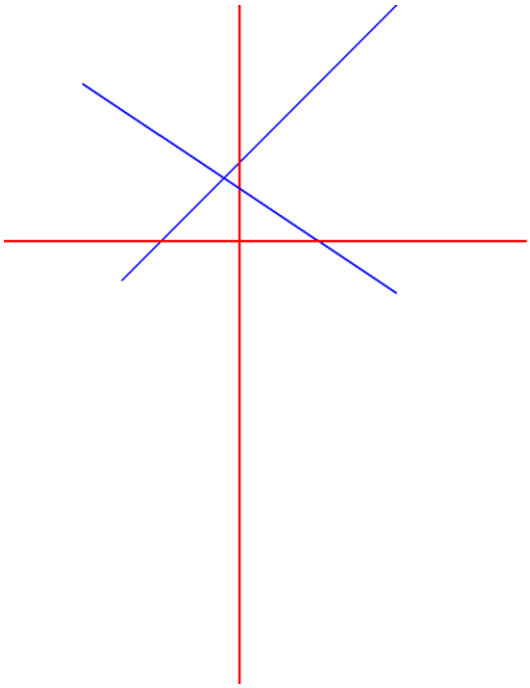
1.1 Середины отрезков:



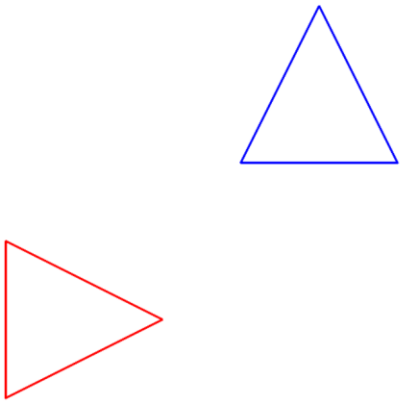
1.2 Параллельные линии:



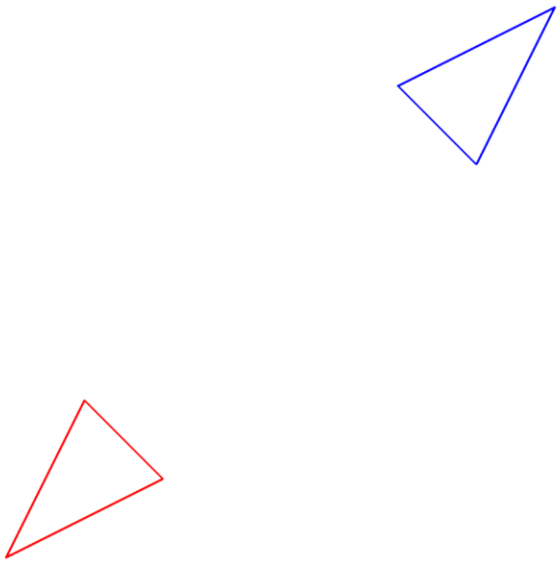
1.3 Пересекающиеся линии:



1.4 Вращение:



1.5 Отражение:



1.6 Масштабирование: