МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

по дисциплине Компьютерная графика

Тема: «Исследование математических методов представления и преобразования графических объектов на плоскости»

Студенты гр. 1307 Грунская Н.Д.
Тростин М.Ю.
Голубев М.А.

Преподаватель _____ Матвеева И.В.

Санкт-Петербург
2024

Цель работы:
Практическое закрепление теоретических знаний о представлении и преобразований графических объектов на плоскости

Постановка задачи:
Поворот плоского объекта (треугольника) относительно произвольной точки плоскости на заданный угол. Необходимо предусмотреть возможность редактирования положения точки

Краткая теоретическая информация
Для представления треугольниками с вершинами A, B, C используется матрица
$\begin{bmatrix} x_A & y_A \\ x_B & y_B \\ x_C & y_C \end{bmatrix}$

Для того, чтобы выполнить поворот треугольника вокруг некоторой точки d, необходимо для начала сместить все точки так, чтобы точка d лежала в начале координат. Выполнить это можно при помощи следующего преобразования:

$$\begin{bmatrix} x_A & y_A & 1 \\ x_B & y_B & 1 \\ x_C & y_C & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_d & -y_d & 1 \end{bmatrix} = \begin{bmatrix} x_A - x_d & y_A - y_d & 1 \\ x_B - x_d & y_B - y_d & 1 \\ x_C - x_d & y_C - y_d & 1 \end{bmatrix}$$

После чего можно осуществить поворот на произвольный угол α при помощи матрицы поворота:

$$\begin{bmatrix} x_A & y_A & 1 \\ x_B & y_B & 1 \\ x_C & y_C & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_d & -y_d & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Теперь преобразованный треугольник необходимо вернуть в исходную систему координат:

$$\begin{bmatrix} x_A & y_A & 1 \\ x_B & y_B & 1 \\ x_C & y_C & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_d & -y_d & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_d & y_d & 1 \end{bmatrix}$$

Реализация алгоритма
Для реализации алгоритма был использован Python, а для визуализации результата был использован Tkinter
Функция поворота треугольника вокруг точки:

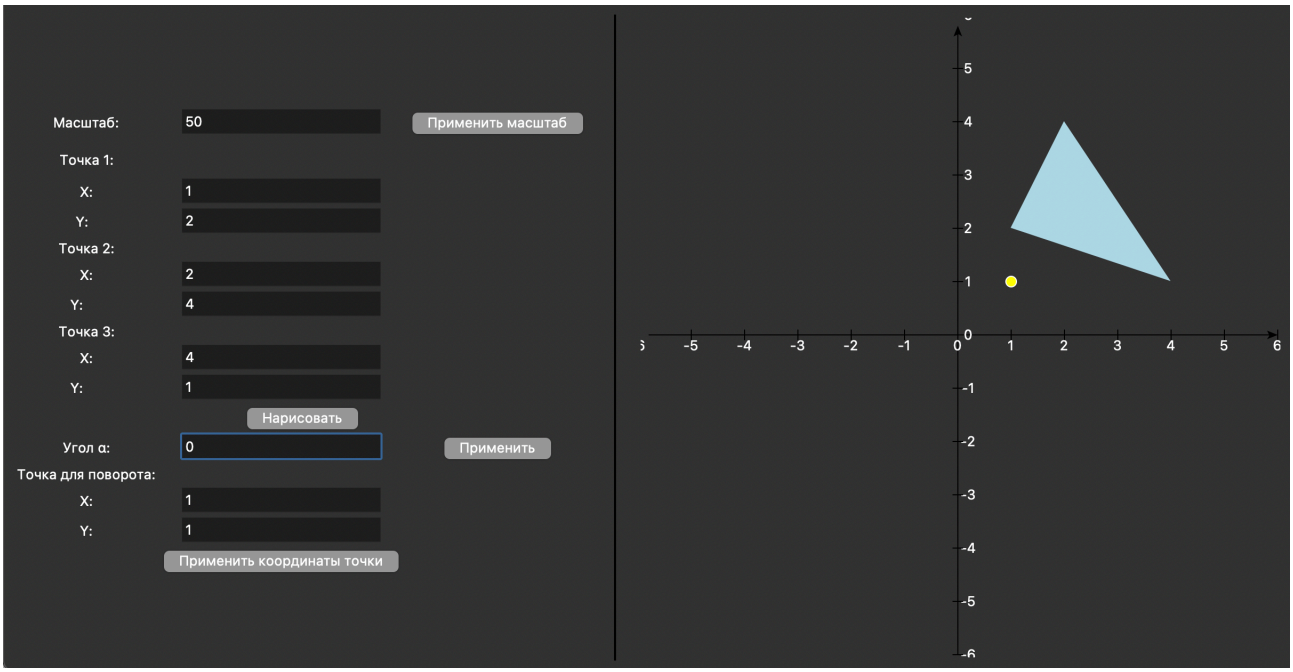
```
def roll_triangle():
```

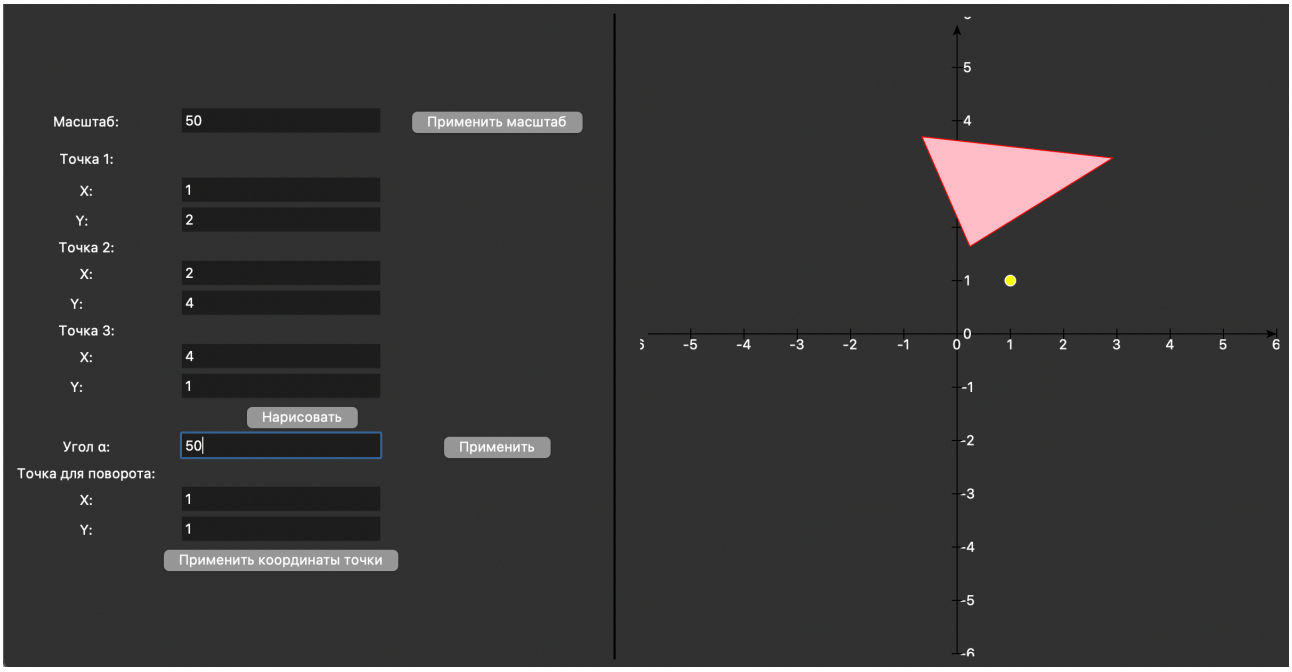
```
    # Получаем координаты точки для поворота
    set_dot()
    # Получаем координаты вершин треугольника
    get_coordinates()
    draw_coord_lines()
    # Получаем угол поворота
    angle = float(entry_angle.get())
    angle = math.radians(angle)


    mat1 = [[x1, y1, 1], [x2, y2, 1], [x3, y3, 1]]
    mat2 = [[1, 0, 0], [0, 1, 0], [dot_x * (-1), dot_y * (-1), 1]]
    mat3 = [[math.cos(angle), math.sin(angle), 0],
[math.sin(angle) * (-1), math.cos(angle), 0], [0, 0, 1]]
    mat4 = [[1, 0, 0], [0, 1, 0], [dot_x * 1, dot_y * 1, 1]]


    result =
multiply_matrices(multiply_matrices(multiply_matrices(mat1, mat2),
mat3), mat4)
    x1_new = result[0][0]
    y1_new = result[0][1]
    x2_new = result[1][0]
    y2_new = result[1][1]
    x3_new = result[2][0]
    y3_new = result[2][1]


    # Масштабируем координаты после вращения
    x1_new_scaled = canvas_x / 2 + x1_new * scale
    y1_new_scaled = canvas_y / 2 - y1_new * scale
    x2_new_scaled = canvas_x / 2 + x2_new * scale
    y2_new_scaled = canvas_y / 2 - y2_new * scale
    x3_new_scaled = canvas_x / 2 + x3_new * scale
    y3_new_scaled = canvas_y / 2 - y3_new * scale
```

```
    # Рисуем преобразованный треугольник
    canvas.create_polygon(x1_new_scaled, y1_new_scaled,
x2_new_scaled, y2_new_scaled, x3_new_scaled, y3_new_scaled,
                          outline="red", fill="pink")
```

Полный исходный код программы представлен в приложении

Пример работы приложения:





Выводы:
Были практически закреплены теоретические знания о представлении и преобразований графических объектов на плоскости

Приложение:

Ссылка на видео:

https://youtu.be/VsRlVSNx7eg

Приложение

```
import tkinter as tk
import math


def set_scale():
    global scale
    scale = int(entry_scale_entry.get())
    draw_coord_lines()


def set_dot():
    global dot_x
    global dot_y
    dot_x = int(entry_dot_x.get())
    dot_y = int(entry_dot_y.get())
    radius = 5
    canvas.create_oval(canvas_x / 2 + dot_x * scale - radius,
canvas_y / 2 - dot_y * scale - radius,
                       canvas_y / 2 + int(entry_dot_x.get()) *
scale + radius, canvas_y / 2 - dot_y * scale + radius,
                       fill="yellow")


def multiply_matrices(matrix1, matrix2):
    result = [[0, 0, 0], [0, 0, 0], [0, 0, 0]]

    for i in range(3):
        for j in range(3):
            for k in range(3):
                result[i][j] += matrix1[i][k] * matrix2[k][j]

    return result
```

```
def draw_coord_lines():
    canvas.delete("all")
    canvas.create_line(10, canvas_y / 2, canvas_x, canvas_y / 2, fill="black", arrow=tk.LAST)  # Ось x
    canvas.create_line(canvas_x / 2, canvas_y, canvas_y / 2, 10, fill="black", arrow=tk.LAST)  # Ось y

    # Рисуем деления на оси
    for i in range(-15, 15):
        canvas.create_line(canvas_x / 2 + i * scale, canvas_y / 2 - 5, canvas_x / 2 + i * scale, canvas_y / 2 + 5,
                           fill="black")  # Деления на оси x
        canvas.create_line(canvas_x / 2 - 5, canvas_y / 2 + i * scale, canvas_x / 2 + 5, canvas_y / 2 + i * scale,
                           fill="black")  # Деления на оси y
        canvas.create_text(canvas_x / 2 + i * scale, canvas_y / 2 + 10, text=str(i))  # Подписи к делениям на оси x
        canvas.create_text(canvas_x / 2 + 10, canvas_y / 2 - i * scale, text=str(i))  # Подписи к делениям на оси y


def get_coordinates():
    global x1, x2, x3, y1, y2, y3
    x1 = float(entry1_x.get())
    y1 = float(entry1_y.get())
    x2 = float(entry2_x.get())
    y2 = float(entry2_y.get())
    x3 = float(entry3_x.get())
    y3 = float(entry3_y.get())


def draw_triangle():
```

```
    get_coordinates()
    draw_coord_lines()
    # Масштабируем координаты
    x1_scaled = canvas_x / 2 + x1 * scale
    y1_scaled = canvas_y / 2 - y1 * scale
    x2_scaled = canvas_x / 2 + x2 * scale
    y2_scaled = canvas_y / 2 - y2 * scale
    x3_scaled = canvas_x / 2 + x3 * scale
    y3_scaled = canvas_y / 2 - y3 * scale
    canvas.create_polygon(x1_scaled, y1_scaled, x2_scaled,
y2_scaled, x3_scaled, y3_scaled, fill="lightblue")



def roll_triangle():
    # Получаем координаты точки для поворота
    set_dot()
    # Получаем координаты вершин треугольника
    get_coordinates()
    draw_coord_lines()
    # Получаем угол поворота
    angle = float(entry_angle.get())
    angle = math.radians(angle)


    mat1 = [[x1, y1, 1], [x2, y2, 1], [x3, y3, 1]]
    mat2 = [[1, 0, 0], [0, 1, 0], [dot_x * (-1), dot_y * (-1), 1]]
    mat3 = [[math.cos(angle), math.sin(angle), 0],
[math.sin(angle) * (-1), math.cos(angle), 0], [0, 0, 1]]
    mat4 = [[1, 0, 0], [0, 1, 0], [dot_x * 1, dot_y * 1, 1]]


    result =
multiply_matrices(multiply_matrices(multiply_matrices(mat1, mat2),
mat3), mat4)


    x1_new = result[0][0]
```

```
    y1_new = result[0][1]
    x2_new = result[1][0]
    y2_new = result[1][1]
    x3_new = result[2][0]
    y3_new = result[2][1]

    # Масштабируем координаты после вращения
    x1_new_scaled = canvas_x / 2 + x1_new * scale
    y1_new_scaled = canvas_y / 2 - y1_new * scale
    x2_new_scaled = canvas_x / 2 + x2_new * scale
    y2_new_scaled = canvas_y / 2 - y2_new * scale
    x3_new_scaled = canvas_x / 2 + x3_new * scale
    y3_new_scaled = canvas_y / 2 - y3_new * scale

    # Рисуем преобразованный треугольник
    canvas.create_polygon(x1_new_scaled, y1_new_scaled,
x2_new_scaled, y2_new_scaled, x3_new_scaled, y3_new_scaled,
                          outline="red", fill="pink")



root = tk.Tk()
root.title("Треугольник")


input_frame = tk.Frame(root)
input_frame.grid(row=0, column=0, padx=10, pady=10)


canvas_x = 600
canvas_y = 600
canvas = tk.Canvas(root, width=canvas_x, height=canvas_y)
canvas.grid(row=0, column=2, padx=10, pady=10, columnspan=2,
sticky="w")


# Разделительная линия
```

```
separator = tk.Frame(root, height=canvas_y, width=2, bg="black")
separator.grid(row=0, column=1, padx=10, pady=10, sticky="ns")


entry_scale = tk.Label(input_frame, text="Масштаб:")
entry_scale.grid(row=0, column=0, padx=5, pady=5)
entry_scale_entry = tk.Entry(input_frame)
entry_scale_entry.grid(row=0, column=1, padx=5, pady=5)
button_scale = tk.Button(input_frame, text="Применить масштаб",
command=set_scale)
button_scale.grid(row=0, column=2, padx=5, pady=5)


label1 = tk.Label(input_frame, text="Точка 1:")
label1.grid(row=1, column=0, padx=5, pady=5)

label1_x = tk.Label(input_frame, text="X:")
label1_x.grid(row=2, column=0)

entry1_x = tk.Entry(input_frame)
entry1_x.grid(row=2, column=1)

label1_y = tk.Label(input_frame, text="Y:  ")
label1_y.grid(row=3, column=0)
entry1_y = tk.Entry(input_frame)
entry1_y.grid(row=3, column=1)

label2 = tk.Label(input_frame, text="Точка 2:")
label2.grid(row=4, column=0)

label2_x = tk.Label(input_frame, text="X:")
label2_x.grid(row=5, column=0)
entry2_x = tk.Entry(input_frame)
entry2_x.grid(row=5, column=1)
label2_y = tk.Label(input_frame, text="Y:     ")
```

```
label2_y.grid(row=6, column=0)
entry2_y = tk.Entry(input_frame)
entry2_y.grid(row=6, column=1)

label3 = tk.Label(input_frame, text="Точка 3:")
label3.grid(row=7, column=0)

label3_x = tk.Label(input_frame, text="X:")
label3_x.grid(row=8, column=0)
entry3_x = tk.Entry(input_frame)
entry3_x.grid(row=8, column=1)
label3_y = tk.Label(input_frame,text="Y:     ")
label3_y.grid(row=9, column=0)
entry3_y = tk.Entry(input_frame)
entry3_y.grid(row=9, column=1)

button_draw = tk.Button(input_frame, text="Нарисовать",
command=draw_triangle)

button_draw.grid(row=10, columnspan=3)

label_angle = tk.Label(input_frame, text="Угол α:")
label_angle.grid(row=11, column=0)  # Поле для ввода угла альфа
entry_angle = tk.Entry(input_frame)
entry_angle.grid(row=11, column=1)
button_scale = tk.Button(input_frame, text="Применить",
command=roll_triangle)
button_scale.grid(row=11, column=2)

global dot_x
global dot_y
label_dot = tk.Label(input_frame, text="Точка для поворота:")
```

```
label_dot.grid(row=12, column=0)
label_dot_x = tk.Label(input_frame, text="X:")
label_dot_x.grid(row=13, column=0)
entry_dot_x = tk.Entry(input_frame)
entry_dot_x.grid(row=13, column=1)
label_dot_y = tk.Label(input_frame, text="Y:")
label_dot_y.grid(row=14, column=0)
entry_dot_y = tk.Entry(input_frame)
entry_dot_y.grid(row=14, column=1)
button_dot = tk.Button(input_frame, text="Применить координаты точки", command=set_dot)
button_dot.grid(row=15, column=1)

# Рисуем координатную ось
canvas.create_line(10, canvas_y / 2, canvas_x - 10, canvas_y / 2, fill="black", arrow=tk.LAST)  # Ось x
canvas.create_line(canvas_x / 2, canvas_y - 10, canvas_x / 2, 10, fill="black", arrow=tk.LAST)  # Ось y

scale = 20  # Масштаб для координат

# Рисуем деления на оси
for i in range(-20, 20):
    canvas.create_line(canvas_x / 2 + i * scale, canvas_y / 2 - 5, canvas_x / 2 + i * scale, canvas_y / 2 + 5,
                       fill="black")  # Деления на оси x
    canvas.create_line(canvas_x / 2 - 5, canvas_y / 2 + i * scale, canvas_x / 2 + 5, canvas_y / 2 + i * scale,
                       fill="black")  # Деления на оси y
    canvas.create_text(canvas_x / 2 + i * scale, canvas_y / 2 + 10, text=str(i))  # Подписи к делениям на оси x
```

```
    canvas.create_text(canvas_x / 2 + 10, canvas_y / 2 - i *
scale, text=str(i))  # Подписи к делениям на оси y


root.mainloop()
```