МИНОБРНАУКИ РОССИИ

Санкт-Петербургский государственный

электротехнический университет

«ЛЭТИ» им. В.И. Ульянова (Ленина)

Кафедра систем автоматизированного проектирования (САПР)

отчет

по лабораторной работе № 1

по дисциплине «Компьютерная графика»

Тема: **«Представление и комбинированные преобразования плоских и объемных изображений»**

Аршин А. Д

Баймухамедов Р. Р.

Студенты гр. 3311 Пасечный Л. В.

Преподаватель Колев Г. Ю.

Санкт-Петербург

2025

Цель работы

Исследование математических методов представления и преобразования графических объектов на плоскости и в пространстве

Задание (вариант 3):

Поворот плоского объекта относительно произвольной точки плоскости на заданный угол. Необходимо предусмотреть возможность редактирования положения точки.

Теоретическая основа

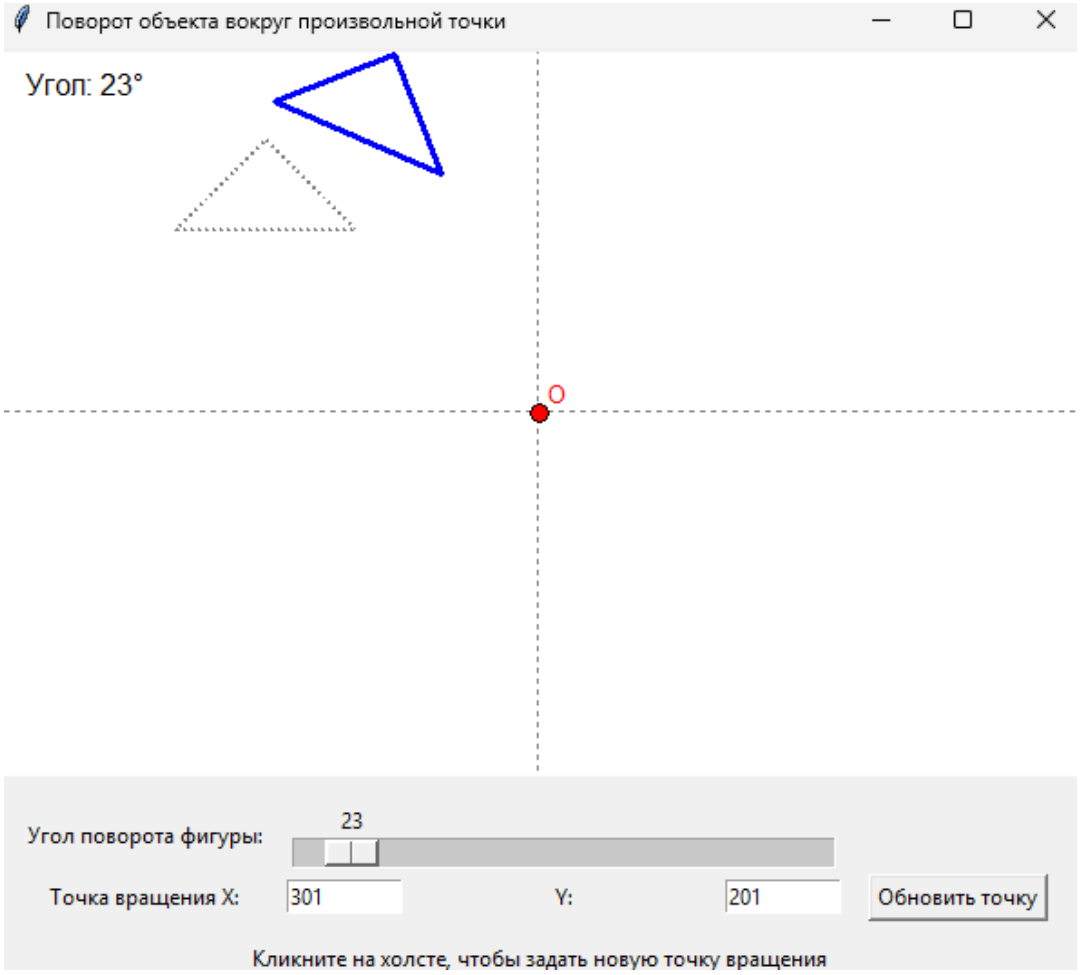
- Поворот вокруг точки начала координат на произвольный угол θ.
- рассмотрим вектор положения от начала координат до точки P (рис. 5). Обозначим r —длину вектора, а ϕ — угол между вектором и осью x.
- Вектор положения поворачивается вокруг начала координат на угол θ и попадает в точку P*. Записав векторы положений для P и P*, получаем:
- $P = \begin{bmatrix} x & y \end{bmatrix} = \begin{bmatrix} r\cos\phi & r\sin\phi \end{bmatrix} \quad P^* = \begin{bmatrix} x^* & y^* \end{bmatrix} = \begin{bmatrix} r\cos(\phi+\theta) & r\sin(\phi+\theta) \end{bmatrix}$
- Используя формулу для cos суммы углов, перепишем выражение для P* следующим образом
- . $P^* = \begin{bmatrix} x^* & y^* \end{bmatrix} = \begin{bmatrix} r(\cos\phi\cos\theta - \sin\phi\sin\theta) & r(\cos\phi\sin\theta + \cos\theta\sin\phi) \end{bmatrix}$
- Используя определения x и y, можно переписать P* как
- . $P^* = \begin{bmatrix} x^* & y^* \end{bmatrix} = \begin{bmatrix} x\cos\theta - y\sin\theta & x\sin\theta + y\cos\theta \end{bmatrix}$
- Таким образом, преобразованная точка имеет координаты
  $x^* = x\cos\theta - y\sin\theta \qquad y^* = x\sin\theta + y\cos\theta$
- **или в матричном виде** $\begin{bmatrix} X^* \end{bmatrix} - [X][T] - \begin{bmatrix} x^* & y^* \end{bmatrix} - \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$
- преобразование поворота вокруг точки начала координат на произвольный угол θ задается матрицей

$$[T] = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

Выполнение лабораторной работы

В качестве языка программирования для выполнения данной лабораторной работы выберем Python. Для быстрого запуска достаточно иметь установленный Python 3 (модули tkinter и math, которые входят в стандартную библиотеку). В качестве фигуры для поворота выберем треугольник.

Интерфейс программы



Демонстрация выполненной лабораторной работы №1

Вывод

В ходе выполнения работы нами были исследованы методы представления и преобразования графических объектов на плоскости и в пространстве. Также получены навыки реализации метода поворота объектов на плоскости с помощью матрицы поворота, а также создания графических приложений.

Код программы

```
import tkinter as tk
import math

class RotationApp:
    # инициализируем окно
    # сохраняем ссылку на главное окно (root) и задаём ему заголовок
    def __init__(self, root):
        self.root = root
        self.root.title("Поворот объекта вокруг произвольной точки")
```

```
        self.canvas_width = 600
        self.canvas_height = 400

        # создаем canvas и область для отрисовки
        self.canvas = tk.Canvas(root, width=self.canvas_width,
height=self.canvas_height, bg="white")
        self.canvas.pack()

        # параметры объекта
        self.original_points = [(100, 100), (150, 50), (200, 100)]  # исходные
координаты треугольник
        self.rotated_points = self.original_points.copy() # копия точек, здесь
будут записываться измененые координаты

        # точка вокруг которой происходит вращения
        self.rotation_point = [300, 200]

        # угол поворота в градусах
        self.angle_deg = 0

        # созадем рамку frame для размещения элементов программы
        control_frame = tk.Frame(root)
        control_frame.pack(pady=10)

        # слайдер для поворота фигуры в градусах вокруг заданной точки
        # при любом движении слайдера вызывается метод update_rotation
        tk.Label(control_frame, text="Угол поворота фигуры:").grid(row=0,
column=0, padx=5)
        self.angle_slider = tk.Scale(control_frame, from_=0, to=360,
orient=tk.HORIZONTAL, length=300,
                                     command=self.update_rotation)
        self.angle_slider.set(self.angle_deg)
        self.angle_slider.grid(row=0, column=1, padx=5)

        # поле ввода для координаты x точки вращения
        # заполняем его начальным значением 300
        tk.Label(control_frame, text="Точка вращения X:").grid(row=1, column=0,
padx=5)
        self.point_x_entry = tk.Entry(control_frame, width=10)
        self.point_x_entry.insert(0, str(self.rotation_point[0]))
        self.point_x_entry.grid(row=1, column=1, sticky="w", padx=5)
        # поле ввода для координаты y
        # расположено справа в той же ячейке (с помощью sticky="e" — прижато к
правому краю)
        tk.Label(control_frame, text="Y:").grid(row=1, column=1)
        self.point_y_entry = tk.Entry(control_frame, width=10)
        self.point_y_entry.insert(0, str(self.rotation_point[1]))
        self.point_y_entry.grid(row=1, column=1, sticky="e", padx=5)

        # кнопка, которая считывает значения из полей ввода и обновляет точку
вращения
```

```
        tk.Button(control_frame, text="Обновить точку",
command=self.update_point_from_entry).grid(row=1, column=2, padx=10)

        # подсказка
        tk.Label(root, text="Кликните на холсте, чтобы задать новую точку
вращения").pack()

        # привязка клика мышки
        self.canvas.bind("<Button-1>", self.set_rotation_point_by_click)

        # рисуем первый кадр
        self.draw_all()

    # поворачивает точку (x,y) вокруг center=(cx,cy) на угол angle_rad
    # сдвигаем систему координат так, чтобы центр вращения стал началом (вычитаем
cx, cy)
    # применяем стандартную матрицу поворота
    def rotate_point(self, point, center, angle_rad):
        x, y = point
        cx, cy = center
        # перенос в начало координат
        x -= cx
        y -= cy
        # поворот
        x_new = x * math.cos(angle_rad) - y * math.sin(angle_rad)
        y_new = x * math.sin(angle_rad) + y * math.cos(angle_rad)
        # обратный перенос
        x_new += cx
        y_new += cy
        return x_new, y_new

    # обновляет повёрнутые координаты объекта и перерисовывает
    # считываем текущий угол со слайдера
    # переводим градусы в радианы
    # для каждой исходной точки вычисляем её новое положение после поворота
    # обновляем список rotated_points
    # перерисовываем всё
    def update_rotation(self, val=None):
        self.angle_deg = self.angle_slider.get()
        angle_rad = math.radians(self.angle_deg)

        self.rotated_points = [
            self.rotate_point(p, self.rotation_point, angle_rad)
            for p in self.original_points
        ]
        self.draw_all()

    # обновляет точку вращения из полей ввода
    # пытаемся прочитать числа из полей
    # если введено не число (например, буквы) ловим ValueError и ничего не делаем
    # иначе обновляем точку и пересчитываем поворот
```

```
    def update_point_from_entry(self):
        try:
            x = float(self.point_x_entry.get())
            y = float(self.point_y_entry.get())
            self.rotation_point = [x, y]
            self.update_rotation()
        except ValueError:
            pass  # игнорируем некорректный ввод

    # устанавливает точку вращения по клику мыши
    def set_rotation_point_by_click(self, event):
        # постоянно обновляем при действии rotation_point
        self.rotation_point = [event.x, event.y]
        self.point_x_entry.delete(0, tk.END)
        self.point_y_entry.delete(0, tk.END)
        self.point_x_entry.insert(0, str(event.x))
        self.point_y_entry.insert(0, str(event.y))
        self.update_rotation()

    # перерисовывает всё на холсте
    def draw_all(self):
        self.canvas.delete("all")

        # рисуем оси
        self.canvas.create_line(0, self.canvas_height//2, self.canvas_width,
self.canvas_height//2, fill="gray", dash=(2,2))
        self.canvas.create_line(self.canvas_width//2, 0, self.canvas_width//2,
self.canvas_height, fill="gray", dash=(2,2))

        # рисуем исходный объект, который не будет менять положение
        if len(self.original_points) >= 2:
            self.canvas.create_polygon(self.original_points, outline="gray",
fill="", dash=(4,4), width=2)

        # рисуем повёрнутый объект который будет менять свое положение вокруг
точки
        if len(self.rotated_points) >= 2:
            self.canvas.create_polygon(self.rotated_points, outline="blue",
fill="", width=3)

        # рисуем точку вращения
        x, y = self.rotation_point
        r = 5
        self.canvas.create_oval(x - r, y - r, x + r, y + r, fill="red")
        self.canvas.create_text(x + 10, y - 10, text="O", fill="red",
font=("Arial", 10))

        # подпись угла в данный момент
        self.canvas.create_text(50, 20, text=f"Угол: {self.angle_deg}°",
fill="black", font=("Arial", 12))
```

```python
# Запуск приложения
if __name__ == "__main__":
    root = tk.Tk()
    app = RotationApp(root)
    root.mainloop()
```