**Минобрнауки России**

**Юго-Западный государственный университет**

Кафедра программной инженерии

Лабораторная работа №2

по дисциплине «Компьютерная графика»

по теме:

«Программа для формирования изображений»

Выполнили: студенты группы ПО-32б

Разгуляев Е. О., Норов М. С.

Проверил: ст. преподаватель, зав. лабораториями

Ефремов В. В.

Курск 2025 г.

**Цель работы:**

Совершенствование базовых навыков обработки растровых изображений с помощью алгоритмических языков программирования.

**ЗАДАНИЕ**

1. Разработать алгоритм и программу для чтения из файла, преобразования и записи в файл графической информации по варианту. Программа включает в себя:

* ﻿﻿Создание нового изображения;
* ﻿﻿Чтение изображения из файла;
* ﻿﻿Добавление фрагмента заданной формы из заданной позиции изображения, загруженного из файла в заданную позицию созданного нового изображения попиксельно;
* ﻿﻿Добавление осей системы координат и графика функции;
* ﻿﻿Сохранение созданного изображения в файл на диске.

﻿﻿﻿2. Ответить на контрольные вопросы.

﻿﻿3. Оформить отчёт.

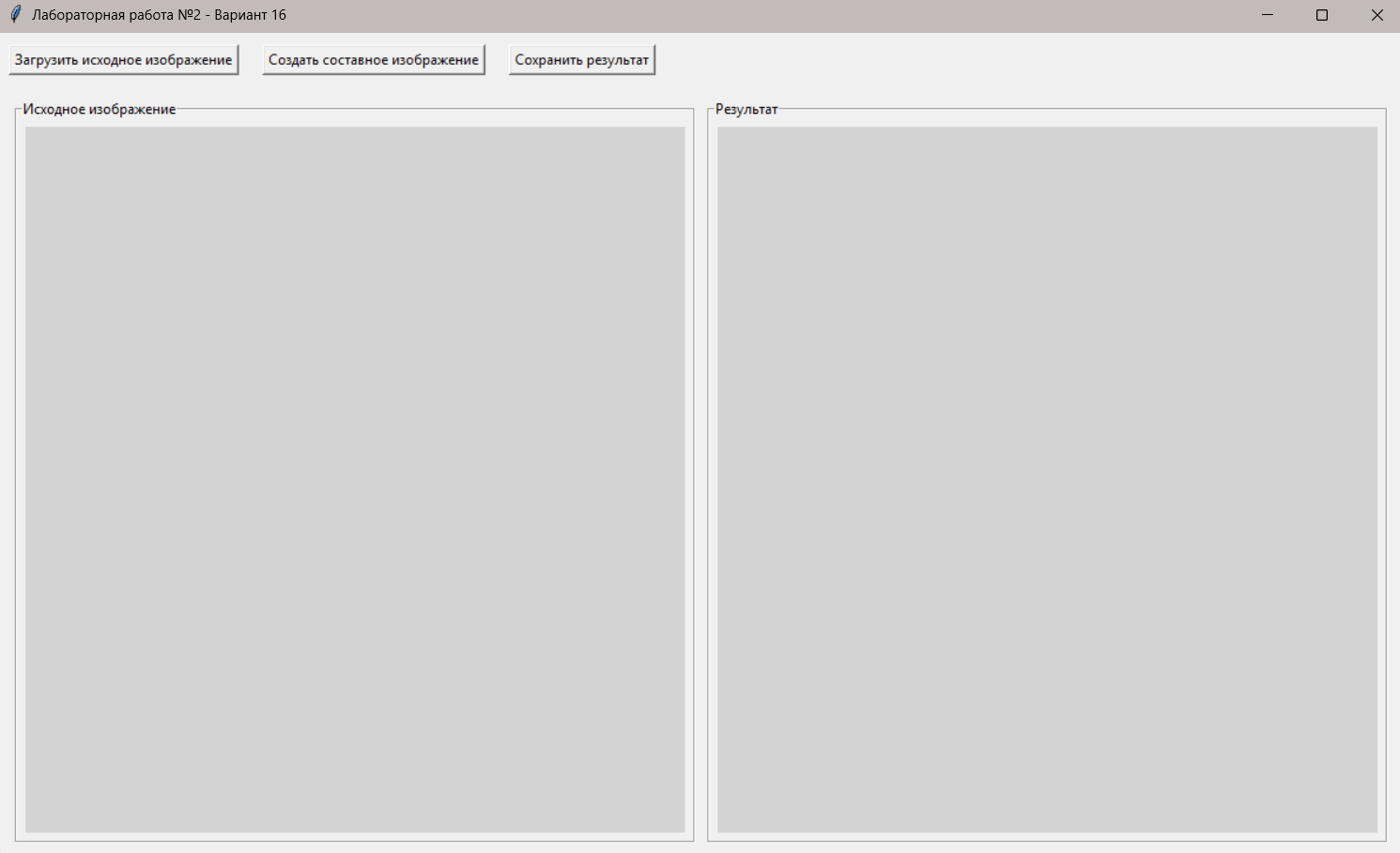
**ХОД РАБОТЫ**

**Вариант 16**

**[isami972/CG-Razgulyaev-Norov-LR-2](https://github.com/isami972/CG-Razgulyaev-Norov-LR-2)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Форма и позиция фрагмента на исходном изображении | Регион размещения на новом изображении | Функция |
| 16 |  | Левый край по центру | |x-1| |

1. Форма:



2. Текст программы (весь модуль):

# --- Блок для автоматической установки модулей ---

import sys

import subprocess

try:

from PIL import Image, ImageDraw, ImageTk

except ImportError:

print("Библиотека Pillow не найдена. Выполняется установка...")

try:

subprocess.check\_call([sys.executable, "-m", "pip", "install", "Pillow"])

print("\nБиблиотека Pillow успешно установлена. Пожалуйста, перезапустите скрипт.")

except Exception as e:

print(f"Ошибка при установке Pillow: {e}")

print("Пожалуйста, установите библиотеку вручную командой: pip install Pillow")

sys.exit()

# --------------------------------------------------

import tkinter as tk

from tkinter import filedialog, messagebox

import math

class Lab2App:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("Лабораторная работа №2 - Вариант 16")

self.root.geometry("1200x700")

self.source\_image = None

self.result\_image = None

self.create\_widgets()

def create\_widgets(self):

top\_frame = tk.Frame(self.root, pady=10)

top\_frame.pack(fill=tk.X)

tk.Button(top\_frame, text="Загрузить исходное изображение", command=self.load\_source\_image).pack(side=tk.LEFT, padx=10)

tk.Button(top\_frame, text="Создать составное изображение", command=self.create\_composite\_image).pack(side=tk.LEFT, padx=10)

tk.Button(top\_frame, text="Сохранить результат", command=self.save\_result).pack(side=tk.LEFT, padx=10)

main\_frame = tk.Frame(self.root)

main\_frame.pack(fill=tk.BOTH, expand=True, padx=10, pady=10)

left\_frame = tk.LabelFrame(main\_frame, text="Исходное изображение", padx=5, pady=5)

left\_frame.pack(side=tk.LEFT, fill=tk.BOTH, expand=True, padx=5)

self.canvas\_source = tk.Canvas(left\_frame, bg="lightgray")

self.canvas\_source.pack(fill=tk.BOTH, expand=True)

right\_frame = tk.LabelFrame(main\_frame, text="Результат", padx=5, pady=5)

right\_frame.pack(side=tk.RIGHT, fill=tk.BOTH, expand=True, padx=5)

self.canvas\_result = tk.Canvas(right\_frame, bg="lightgray")

self.canvas\_result.pack(fill=tk.BOTH, expand=True)

def \_display\_image(self, image, canvas):

if not image: return

canvas.delete("all")

canvas\_width = canvas.winfo\_width()

canvas\_height = canvas.winfo\_height()

if canvas\_width <= 1 or canvas\_height <= 1:

canvas.after(100, lambda: self.\_display\_image(image, canvas))

return

img\_width, img\_height = image.size

ratio = min(canvas\_width / img\_width, canvas\_height / img\_height)

new\_width = int(img\_width \* ratio)

new\_height = int(img\_height \* ratio)

display\_image = image.resize((new\_width, new\_height), Image.ANTIALIAS)

photo = ImageTk.PhotoImage(display\_image)

canvas.create\_image(canvas\_width / 2, canvas\_height / 2, anchor=tk.CENTER, image=photo)

canvas.image = photo

def load\_source\_image(self):

file\_path = filedialog.askopenfilename(filetypes=[("Image files", "\*.png;\*.jpg;\*.jpeg;\*.bmp")])

if not file\_path: return

try:

self.source\_image = Image.open(file\_path).convert("RGB")

self.\_display\_image(self.source\_image, self.canvas\_source)

except Exception as e:

messagebox.showerror("Ошибка", f"Не удалось открыть файл: {e}")

def create\_composite\_image(self):

if not self.source\_image:

messagebox.showerror("Ошибка", "Сначала загрузите исходное изображение!")

return

# --- Шаг 1: Создание нового изображения ---

new\_width, new\_height = 600, 400

self.result\_image = Image.new("RGB", (new\_width, new\_height), "white")

draw = ImageDraw.Draw(self.result\_image)

# --- Шаг 2: Копирование фрагмента ---

src\_w, src\_h = self.source\_image.size

frag\_radius = src\_h // 4

src\_cx, src\_cy = src\_w - 1, src\_h // 2

dst\_cx, dst\_cy = 0, new\_height // 2

for x in range(frag\_radius):

for y in range(-frag\_radius, frag\_radius):

if x\*x + y\*y <= frag\_radius\*frag\_radius:

src\_x = src\_cx - x

src\_y = src\_cy + y

dst\_x = dst\_cx + x

dst\_y = dst\_cy + y

# ИСПРАВЛЕНИЕ: Добавлена проверка границ для нового изображения

if (0 <= src\_x < src\_w and 0 <= src\_y < src\_h) and \

(0 <= dst\_x < new\_width and 0 <= dst\_y < new\_height):

pixel = self.source\_image.getpixel((src\_x, src\_y))

self.result\_image.putpixel((dst\_x, dst\_y), pixel)

# --- Шаг 3: Отрисовка осей координат ---

origin\_x, origin\_y = 100, new\_height - 50

axis\_color = "black"

draw.line((origin\_x, 20, origin\_x, new\_height - 20), fill=axis\_color)

draw.line((origin\_x-5, 25, origin\_x, 20, origin\_x+5, 25), fill=axis\_color)

draw.text((origin\_x - 20, 20), "Y", fill=axis\_color)

draw.line((50, origin\_y, new\_width - 20, origin\_y), fill=axis\_color)

draw.line((new\_width - 25, origin\_y - 5, new\_width - 20, origin\_y, new\_width - 25, origin\_y + 5), fill=axis\_color)

draw.text((new\_width - 20, origin\_y + 10), "X", fill=axis\_color)

# --- Шаг 4: Построение графика функции y = |x - 1| ---

scale = 30

plot\_color = "blue"

prev\_px, prev\_py = None, None

for px in range(50, new\_width - 20):

math\_x = (px - origin\_x) / scale

math\_y = abs(math\_x - 1)

py = origin\_y - (math\_y \* scale)

if prev\_px is not None:

draw.line((prev\_px, prev\_py, px, py), fill=plot\_color, width=2)

prev\_px, prev\_py = px, py

# --- Отображение результата ---

self.\_display\_image(self.result\_image, self.canvas\_result)

messagebox.showinfo("Готово", "Изображение создано!")

def save\_result(self):

if not self.result\_image:

messagebox.showerror("Ошибка", "Сначала создайте составное изображение!")

return

file\_path = filedialog.asksaveasfilename(defaultextension=".png", filetypes=[("PNG file", "\*.png")])

if not file\_path: return

try:

self.result\_image.save(file\_path)

messagebox.showinfo("Успех", f"Изображение сохранено в: {file\_path}")

except Exception as e:

messagebox.showerror("Ошибка", f"Не удалось сохранить: {e}")

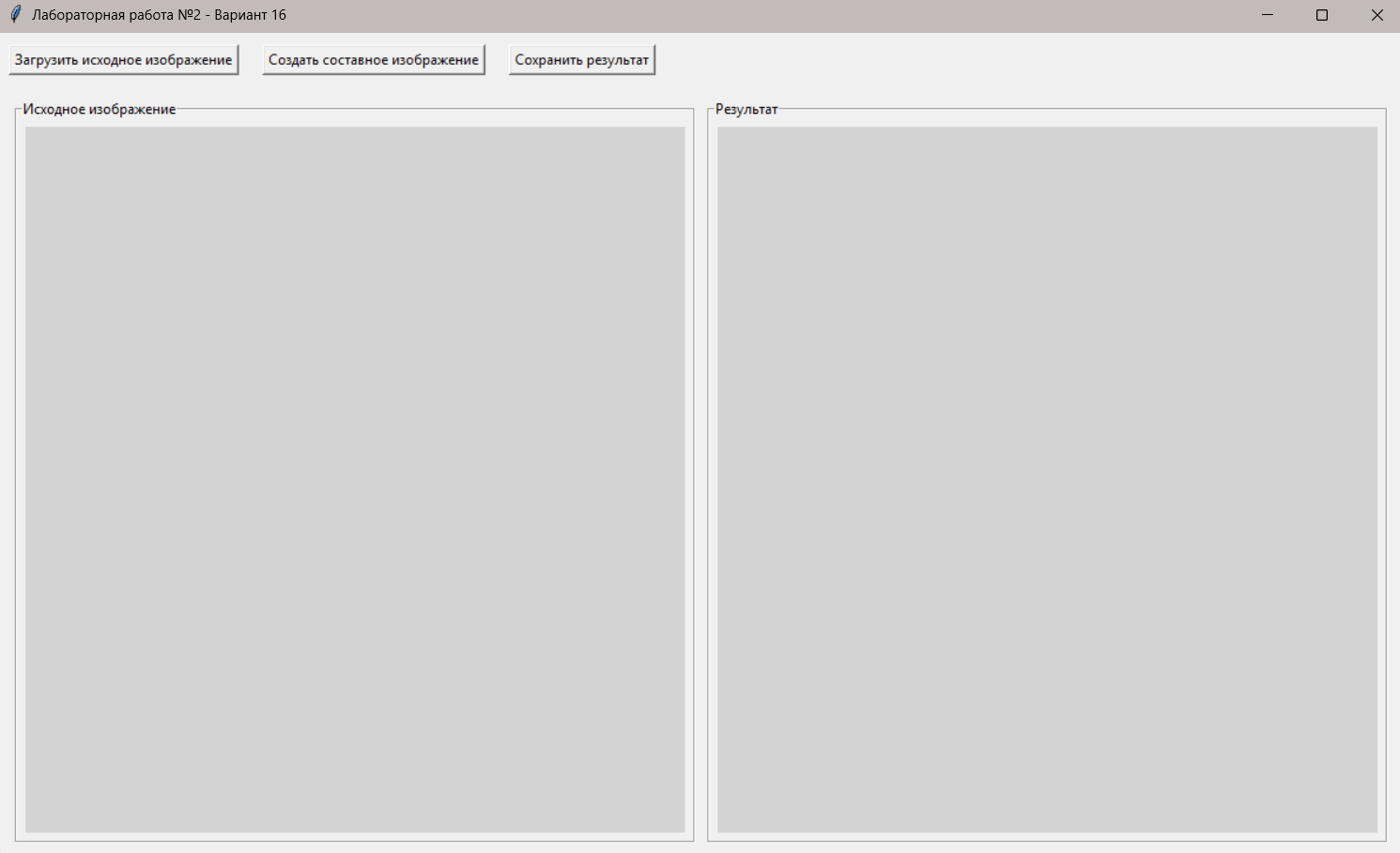
if \_\_name\_\_ == "\_\_main\_\_":

root = tk.Tk()

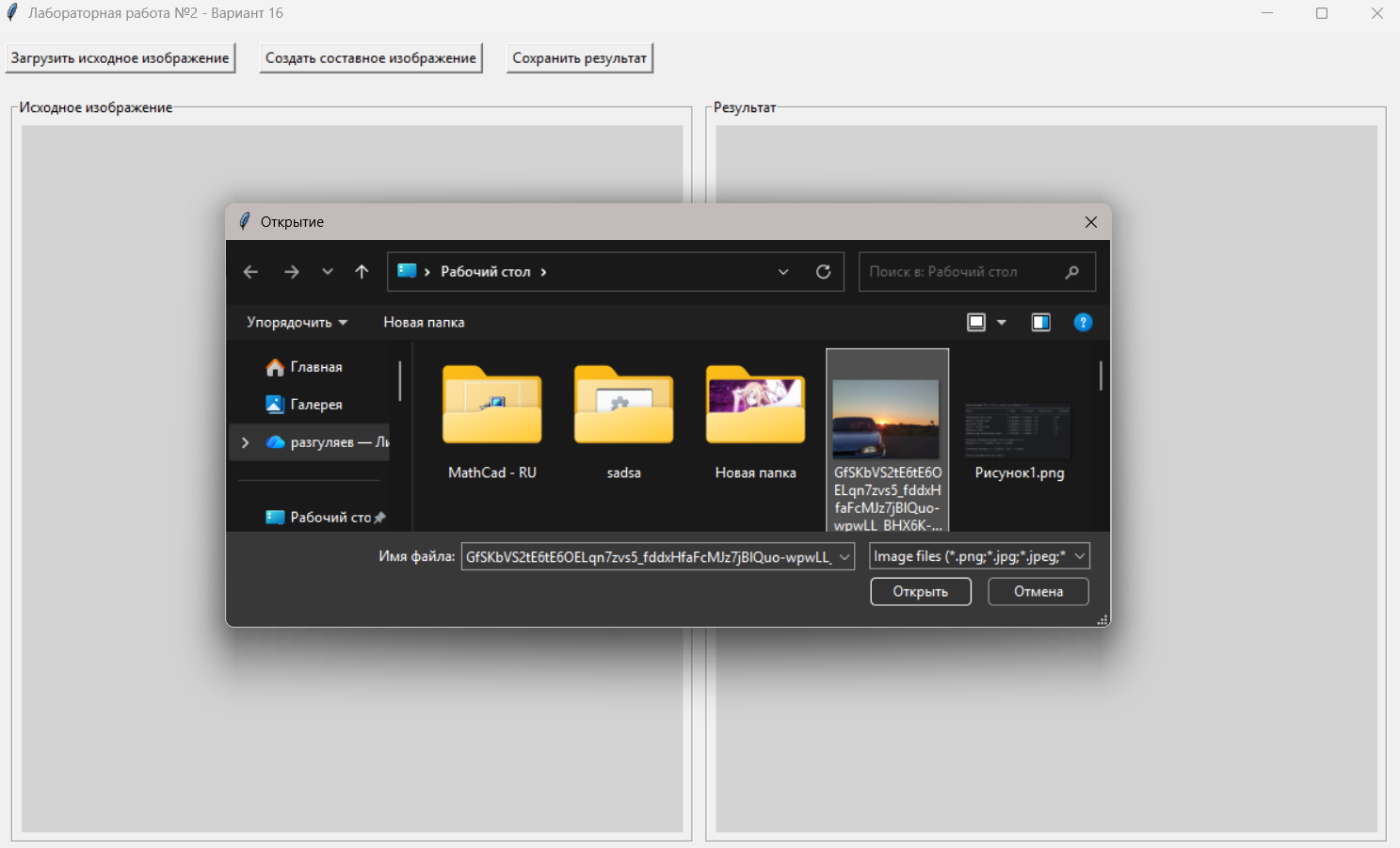
app = Lab2App(root)

root.mainloop().

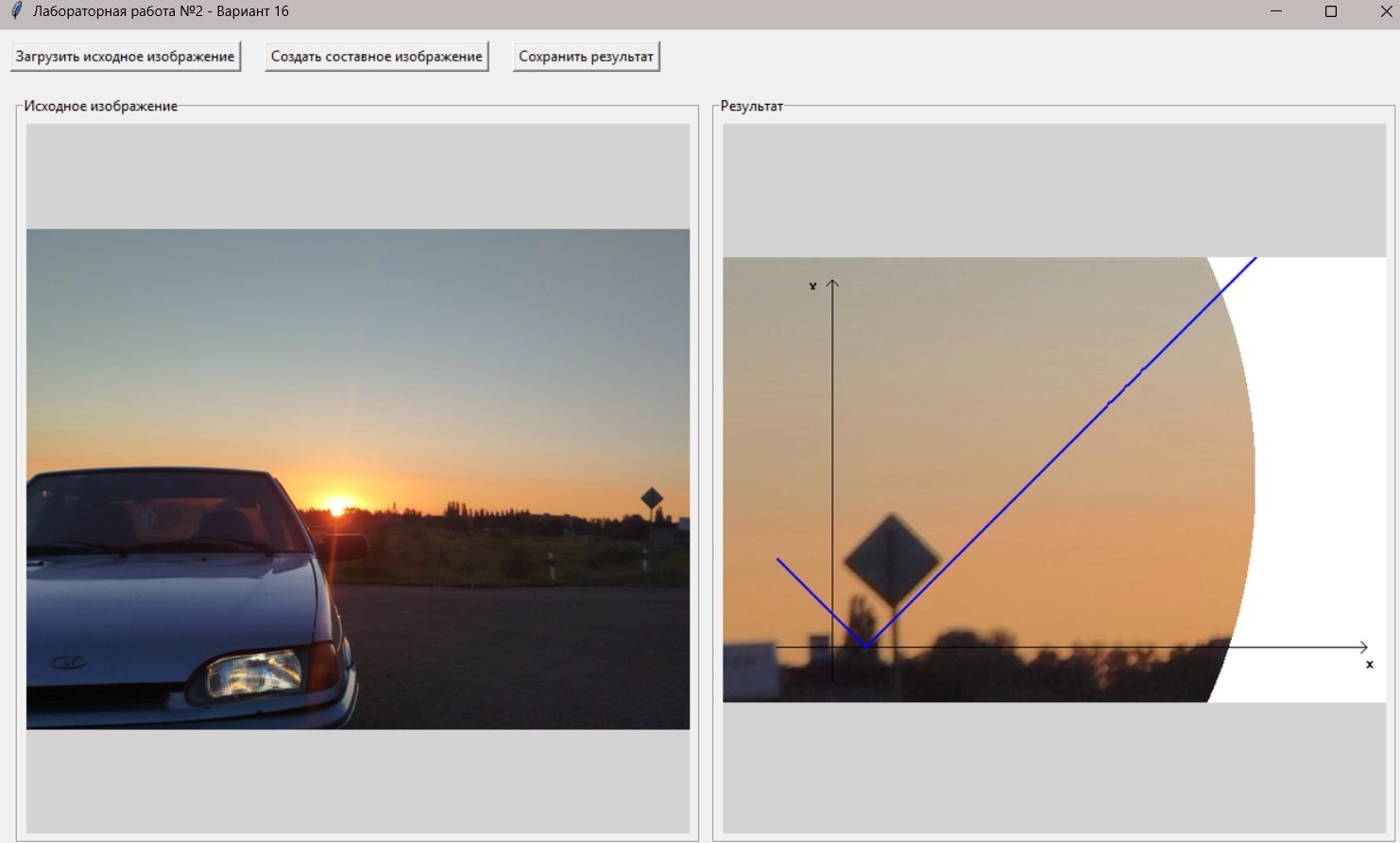
1. Запускаем программу:



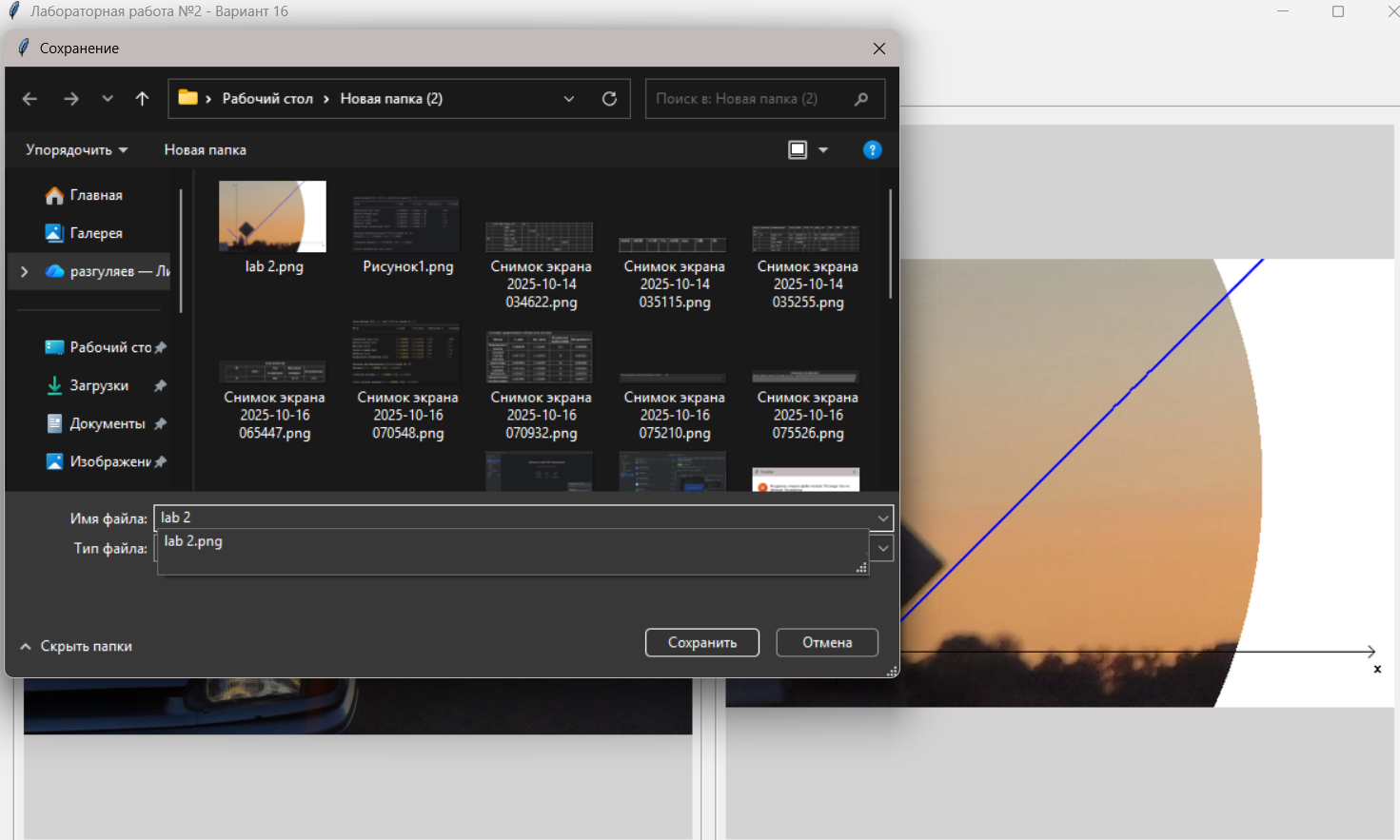
2. Открываем изображение:

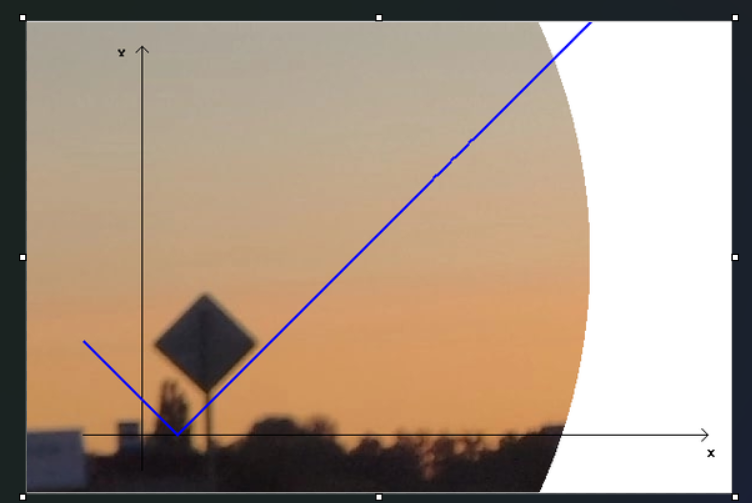


3. Создаём составное изображение:



4. Сохраняем результат:



5. Смотрим на результат:

**ВЫВОД**

В ходе выполненной лабораторной работы мы усовершенствовали базовые навыки обработки растровых изображений с помощью алгоритмических языков программирования.

**ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. Оптическое излучение — электромагнитные волны видимого диапазона (380–780 нм), воспринимаемые глазом. Отличается от других излучений (рентген, радиоволны) длиной волны и способностью напрямую воздействовать на сетчатку.

2. Зрение дает до 80% информации об окружающем мире — форму, цвет, движение. Заменить можно тактильными, звуковыми или электронными системами (например, сонар для слепых), но объем и скорость восприятия будут ниже.

3. Видимый диапазон соответствует прозрачности атмосферы Земли и максимуму излучения Солнца — это результат эволюционной адаптации.

4.

* Лампа накаливания — тепловое излучение, непрерывный спектр.
* Люминесцентная лампа — линейчатый спектр с пиками.
* Солнце — тепловой источник с непрерывным спектром.

5. Коррекция спектра нужна для точной цветопередачи (фотография, кино) или для специальных задач (уличное освещение, выращивание растений).

6. Анизотропный диффузный рефлектор (например, матовая поверхность с микронеровностями) становится изотропным при увеличении рассеивающих свойств (шлифовка, добавление рассеивателя).

7. Яркость — статистическая величина, усредненная по времени и пространству, поэтому она непрерывна, хотя фотоны излучаются дискретно.

8.

* Растр (матрица) — простота изготовления, но возможен муар.
* Случайное размещение (как в глазу) — отсутствие муара, но сложность обработки.

9. Хаотическое размещение эффективнее борется с муаром, но регулярное проще для алгоритмов обработки.

10. Глаз сочетает высокоточную биологическую оптику с мощной постобработкой в мозге — это компенсирует оптические недостатки.

11. Искусственный датчик должен иметь: высокое разрешение, широкий динамический диапазон, чувствительность в видимом спектре и адаптацию к освещению.

12. В первую очередь — количество пикселей и глубина цвета (бит на канал).

13. Система координат холста: начало (0,0) обычно в левом верхнем углу, ось X направлена вправо, ось Y — вниз.

14. Для движения: цикл перерисовки кадров с обновлением координат объектов и очисткой холста между кадрами.