**Минобрнауки России**

**Юго-Западный государственный университет**

Кафедра программной инженерии

Лабораторная работа №6

по дисциплине «Компьютерная графика»

по теме:

«Геометрические преобразования»

Выполнили: студенты группы ПО-32б

Шашков В.А.,

Сазонов М.В.

Проверил: ст. преподаватель, зав. лабораториями

Ефремов В.В.

Курск 2025 г.

**Цель работы** – освоение навыков обработки растровых изображений с помощью алгоритмических языков программирования путём преобразования координат пикселей.

**ЗАДАНИЕ**

1. Разработать алгоритм и программу для реализации индивидуального задания. Программа должна выполнять:

загрузку изображения из файла;

прямые и обратные аффинные преобразования;

функциональное преобразование координат;

сохранение результата в файл.

2. Ответить на контрольные вопросы.

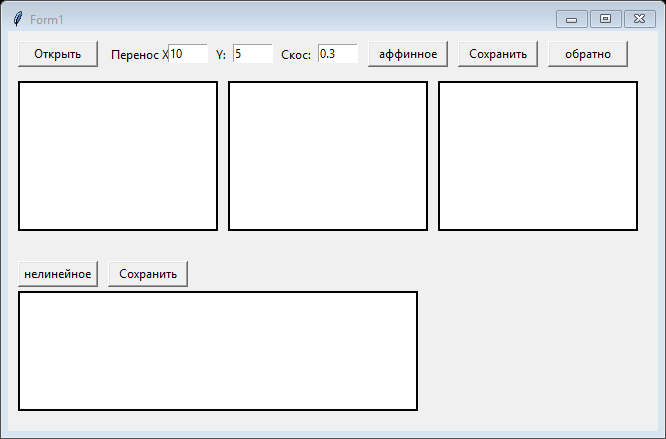
3. Оформить отчёт.

**Ход работы**

**Вариант 13**



1.Форма



2.Текст программы

from tkinter import \*

from tkinter import filedialog, messagebox

from PIL import Image, ImageTk

import math

import os

root = Tk()

root.title("Form1")

root.geometry("650x400")

img\_orig = None

img\_affine = None

img\_inverse = None

img\_nonlinear = None

def show\_image(img, lbl, max\_w=200, max\_h=150):

if img is None:

lbl.config(image='', bg='white')

lbl.image = None

return

w, h = img.size

scale = min(max\_w / w, max\_h / h, 1.0)

im\_r = img.resize((int(w \* scale), int(h \* scale)))

tk\_img = ImageTk.PhotoImage(im\_r)

lbl.config(image=tk\_img)

lbl.image = tk\_img

def open\_image():

global img\_orig

path = filedialog.askopenfilename(filetypes=[("Изображения", "\*.bmp;\*.jpg;\*.png;\*.jpeg;\*.ppm")])

if not path:

return

try:

img\_orig = Image.open(path).convert("RGB")

show\_image(img\_orig, lbl\_orig)

except Exception as e:

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

def save\_image(img):

if img is None:

messagebox.showwarning("Ошибка", "Нет изображения для сохранения")

return

path = filedialog.asksaveasfilename(

defaultextension=".ppm",

filetypes=[("PPM", "\*.ppm"), ("PNG", "\*.png"), ("JPEG", "\*.jpg;\*.jpeg"), ("BMP", "\*.bmp")],

title="Сохранить изображение"

)

if not path:

return

try:

# Сохраняем в формате PPM

img.save(path, "PPM")

messagebox.showinfo("Сохранено", f"Изображение сохранено:\n{os.path.basename(path)}")

except Exception as e:

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

def affine\_transform():

global img\_affine

if img\_orig is None:

messagebox.showwarning("Ошибка", "Сначала загрузите изображение")

return

try:

# Параметры переноса

tx = float(translation\_x\_entry.get())

ty = float(translation\_y\_entry.get())

# Параметр скоса

shear = float(shear\_entry.get())

except:

messagebox.showwarning("Ошибка", "Введите корректные параметры преобразования")

return

w, h = img\_orig.size

src = img\_orig.load()

result = Image.new("RGB", (w, h), (255, 255, 255))

dst = result.load()

for i in range(w):

for j in range(h):

# Обратное преобразование для нахождения исходной точки

# Прямое преобразование: x' = x + tx + shear \* y, y' = y + ty

# Обратное: x = x' - tx - shear \* (y' - ty), y = y' - ty

x\_src = i - tx - shear \* (j - ty)

y\_src = j - ty

xi = int(round(x\_src))

yi = int(round(y\_src))

if 0 <= xi < w and 0 <= yi < h:

dst[i, j] = src[xi, yi]

img\_affine = result

show\_image(img\_affine, lbl\_affine)

def inverse\_button\_action():

global img\_inverse

if img\_orig is None:

messagebox.showwarning("Ошибка", "Сначала загрузите изображение")

return

img\_inverse = img\_orig.copy()

show\_image(img\_inverse, lbl\_inverse)

def nonlinear\_transform():

global img\_nonlinear

if img\_orig is None:

messagebox.showwarning("Ошибка", "Сначала загрузите изображение")

return

w, h = img\_orig.size

cx, cy = w / 2, h / 2

src = img\_orig.load()

result = Image.new("RGB", (w, h), (255, 255, 255))

dst = result.load()

for i in range(w):

for j in range(h):

# Обратное преобразование: находим исходные координаты

# Прямое преобразование: i = cosh(x') - 1, j = y'

# где i, j - координаты в результирующем изображении

# x', y' - координаты в исходном изображении относительно центра

# Сначала преобразуем координаты результата относительно центра

x\_prime = i - cx

y\_prime = j - cy

# Обратное преобразование для i = cosh(x') - 1

# => cosh(x') = i + 1

# => x' = acosh(i + 1)

try:

if (x\_prime + 1) >= 1: # cosh(x) >= 1 для всех x

x\_src = math.acosh(x\_prime + 1)

else:

# Если значение вне области определения, пропускаем пиксель

continue

except (ValueError, OverflowError):

# Если значение вне области определения, пропускаем пиксель

continue

# Для j = y' преобразование простое

y\_src = y\_prime

# Преобразуем обратно в координаты изображения

xi = int(round(x\_src + cx))

yi = int(round(y\_src + cy))

if 0 <= xi < w and 0 <= yi < h:

dst[i, j] = src[xi, yi]

img\_nonlinear = result

show\_image(img\_nonlinear, lbl\_nonlinear, max\_w=400, max\_h=120)

# Создание элементов интерфейса

btn\_open = Button(root, text="Открыть", command=open\_image)

btn\_open.place(x=10, y=10, width=80)

# Параметры для аффинного преобразования (перенос и скос)

Label(root, text="Перенос X:").place(x=100, y=13)

translation\_x\_entry = Entry(root, width=5)

translation\_x\_entry.insert(0, "10")

translation\_x\_entry.place(x=160, y=13, width=40)

Label(root, text="Y:").place(x=205, y=13)

translation\_y\_entry = Entry(root, width=5)

translation\_y\_entry.insert(0, "5")

translation\_y\_entry.place(x=225, y=13, width=40)

Label(root, text="Скос:").place(x=270, y=13)

shear\_entry = Entry(root, width=5)

shear\_entry.insert(0, "0.3")

shear\_entry.place(x=310, y=13, width=40)

btn\_aff = Button(root, text="аффинное", command=affine\_transform)

btn\_aff.place(x=360, y=10, width=80)

btn\_save\_aff = Button(root, text="Сохранить", command=lambda: save\_image(img\_affine))

btn\_save\_aff.place(x=450, y=10, width=80)

btn\_inv = Button(root, text="обратно", command=inverse\_button\_action)

btn\_inv.place(x=540, y=10, width=80)

lbl\_orig = Label(root, bg="white", relief="solid")

lbl\_orig.place(x=10, y=50, width=200, height=150)

lbl\_affine = Label(root, bg="white", relief="solid")

lbl\_affine.place(x=220, y=50, width=200, height=150)

lbl\_inverse = Label(root, bg="white", relief="solid")

lbl\_inverse.place(x=430, y=50, width=200, height=150)

btn\_nl = Button(root, text="нелинейное", command=nonlinear\_transform)

btn\_nl.place(x=10, y=230, width=80)

btn\_save\_nl = Button(root, text="Сохранить", command=lambda: save\_image(img\_nonlinear))

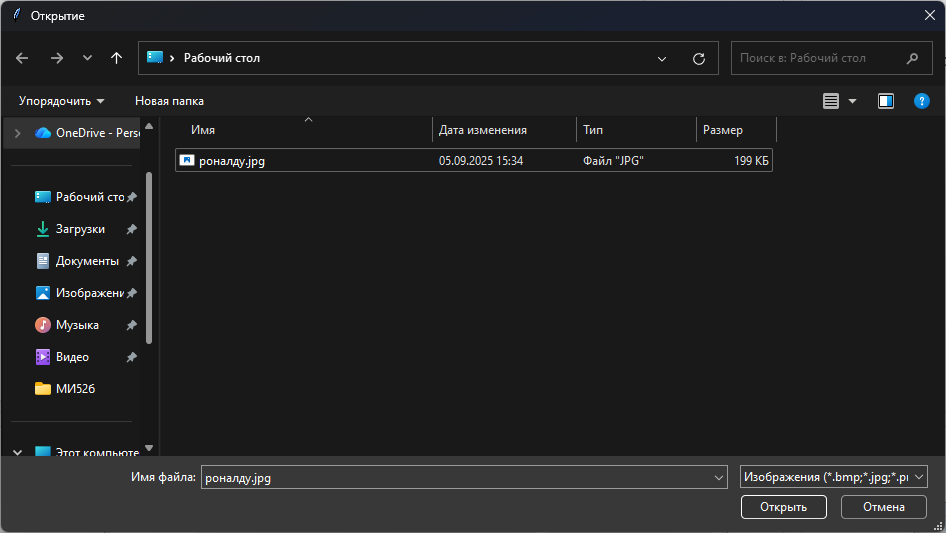
btn\_save\_nl.place(x=100, y=230, width=80)

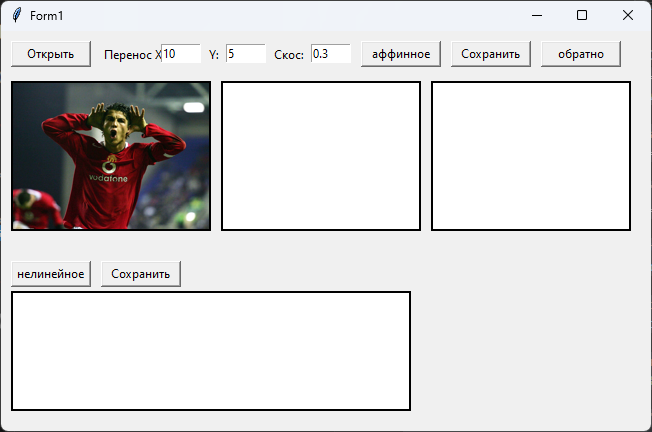
lbl\_nonlinear = Label(root, bg="white", relief="solid")

lbl\_nonlinear.place(x=10, y=260, width=400, height=120)

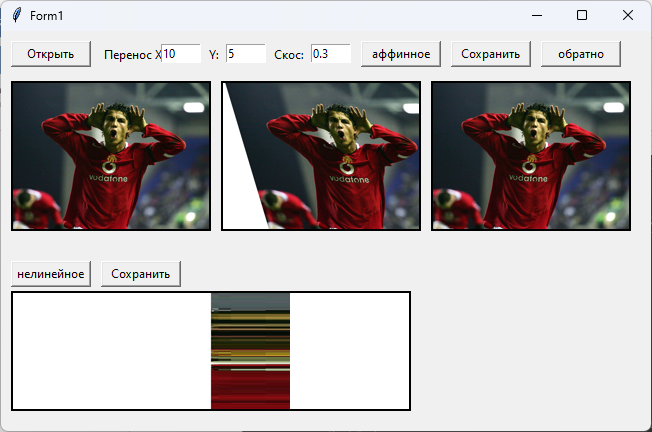
root.mainloop()

1. Загружаем исходное изображение

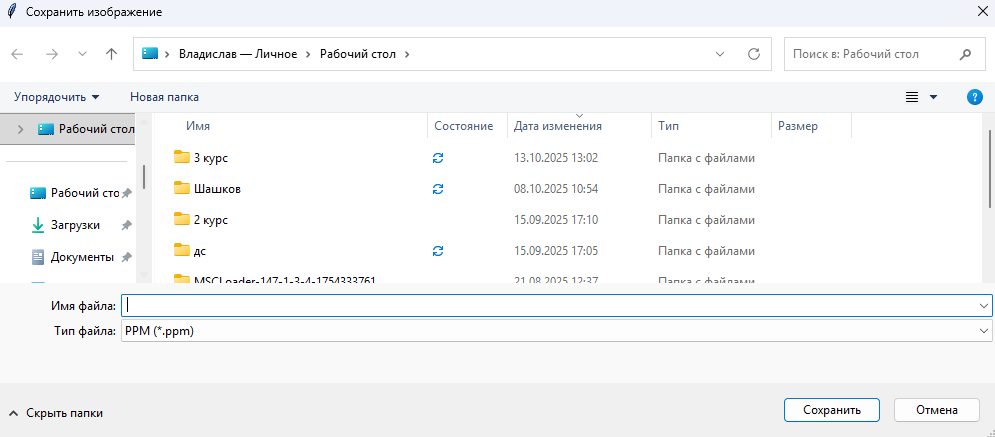




1. Вводим желаемые параметры переноса и скоса и обрабатываем изображение.



1. Сохраняем обработанные изображения



**ВЫВОД**

В ходе работы мы освоили навыки обработки растровых изображений с помощью алгоритмических языков программирования путём преобразования координат пикселей.

Ссылка на гитхаб https://github.com/Str1Ikerz/GraphicsLabs6

**Контрольные вопросы**

1. Формирование изображения включает переход от 3D мировых координат к 2D координатам изображения. Это достигается за счет перспективного преобразования, которое проецирует точки сцены через центр проекции на плоскость фотоприемника, и аффинных преобразований (масштабирование, сдвиг) для согласования с пиксельной сеткой.

2. Физически центр проекции находится в оптическом центре объектива камеры. Именно в этой точке лучи света, идущие от объектов сцены, пересекаются перед тем, как попасть на матрицу фотоприемника.

3. Проецирование 3D сцены на 2D плоскость необратимо, так как приводит к безвозвратной потере информации о глубине. Все точки, лежащие на одном луче зрения, проецируются в одну и ту же точку на изображении, и их невозможно однозначно разделить.

4. Такие инструменты, как «Свободная трансформация» (масштабирование, поворот, наклон), «Коррекция дисторсии» и «Пластика» (Liquify) напрямую работают с координатной сеткой изображения. Они применяют аффинные или нелинейные преобразования к координатам пикселей.

5. При обратном аффинном преобразовании (например, повороте изображения обратно) могут наблюдаться искажения в виде пустых областей (дырок) или ступенчатости. Это происходит потому, что целочисленные координаты пикселей повернутого изображения не всегда точно соответствуют исходной пиксельной сетке, что требует интерполяции.