**Минобрнауки России**

**Юго-Западный государственный университет**

Кафедра программной инженерии

Лабораторная работа №4

по дисциплине «Компьютерная графика»

по теме:

«Растеризация окружностей»

Выполнили: студенты группы ПО-32б

Чаплыгин Т. А.,

Чеховская Ю. М.

Проверил: ст. преподаватель, зав. лабораториями

Ефремов В.В.

Курск 2025 г.

**Цель работы** – изучение алгоритмов растеризации окружностей, создание программы для визуализации работы алгоритмов.

**ЗАДАНИЕ**

1. Написать программу (на языке высокого уровня), реализующую алгоритмы растеризации окружности с их последующей прорисовкой. Координаты центра и радиуса должны задаваться пользователем. Реализовать:

- построение окружности по уравнению окружности;

- построение окружности по параметрическому уравнению;

- алгоритм Брезенхема для построения окружности;

- встроенные средства языка программирования.

2. Проиллюстрировать разницу между результатами или её отсутствие.

3. Ответить на контрольные вопросы.

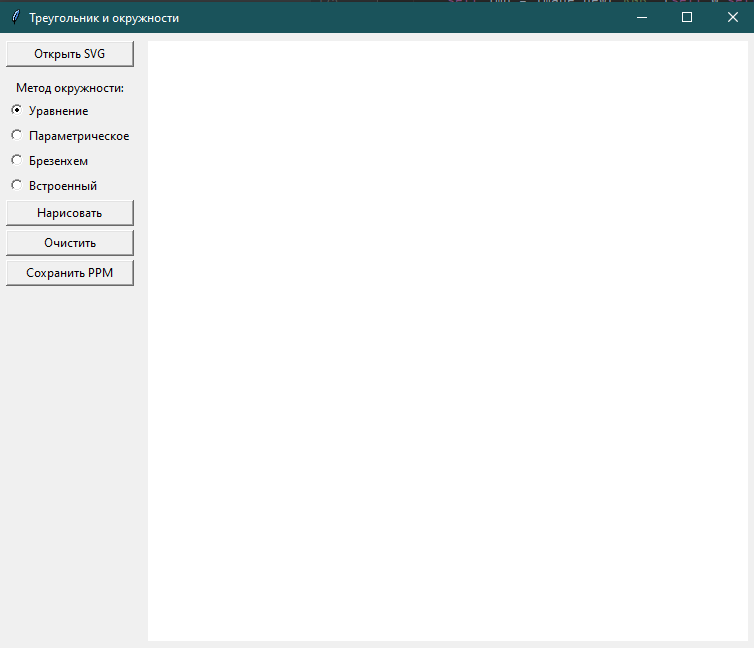
4. Оформить отчёт.

**Ход работы**

**Вариант 11**

По заданным вершинам вывести треугольник и его вписанную и описанную окружности.

1. Форма:



2. Текст программы:

import tkinter as tk

from tkinter import filedialog, messagebox

from PIL import Image, ImageTk, ImageDraw

import math, re

def clamp\_coord(x, w): return max(0, min(w - 1, int(x)))

def sign(v): return 1 if v > 0 else (-1 if v < 0 else 0)

def set\_pixel(img, x, y, color):

w, h = img.size

if 0 <= x < w and 0 <= y < h:

img.putpixel((x, y), color)

# Линии (можно использовать DDA)

def dda\_line(img, x1, y1, x2, y2, color):

dx, dy = x2 - x1, y2 - y1

L = max(abs(dx), abs(dy))

if L == 0: set\_pixel(img, round(x1), round(y1), color); return

dx\_step, dy\_step = dx / L, dy / L

x, y = x1, y1

for \_ in range(int(L)+1):

set\_pixel(img, round(x), round(y), color)

x += dx\_step; y += dy\_step

# --- Окружности ---

def circle\_equation(img, xc, yc, r, color):

for x in range(int(xc-r), int(xc+r)+1):

y\_sq = r\*\*2 - (x-xc)\*\*2

if y\_sq >= 0:

y = int(math.sqrt(y\_sq))

set\_pixel(img, x, yc+y, color)

set\_pixel(img, x, yc-y, color)

def circle\_parametric(img, xc, yc, r, color):

step = 0.01

t = 0

while t <= 2\*math.pi:

x = int(xc + r\*math.cos(t))

y = int(yc + r\*math.sin(t))

set\_pixel(img, x, y, color)

t += step

def circle\_bresenham(img, xc, yc, r, color):

x = 0

y = r

d = 3 - 2\*r

plot\_circle\_points(img, xc, yc, x, y, color)

while y >= x:

x += 1

if d > 0:

y -= 1

d += 4\*(x - y) + 10

else:

d += 4\*x + 6

plot\_circle\_points(img, xc, yc, x, y, color)

def plot\_circle\_points(img, xc, yc, x, y, color):

points = [(xc+x,yc+y),(xc-x,yc+y),(xc+x,yc-y),(xc-x,yc-y),

(xc+y,yc+x),(xc-y,yc+x),(xc+y,yc-x),(xc-y,yc-x)]

for px, py in points:

set\_pixel(img, px, py, color)

def circle\_builtin(img, xc, yc, r, color):

draw = ImageDraw.Draw(img)

draw.ellipse((xc-r, yc-r, xc+r, yc+r), outline=color)

# --- SVG / Треугольник ---

def parse\_svg(filename):

with open(filename, "r", encoding="utf-8") as f:

text = f.read()

m = re.search(r'points="([\d ,.-]+)"', text)

if not m:

return None

nums = list(map(float, re.findall(r'[-]?\d+\.?\d\*', m.group(1))))

if len(nums) < 6: return None

return nums[:6]

def draw\_triangle(img, vertices, color=(0,0,255)):

x1,y1,x2,y2,x3,y3 = vertices

dda\_line(img, x1,y1,x2,y2,color)

dda\_line(img, x2,y2,x3,y3,color)

dda\_line(img, x3,y3,x1,y1,color)

# --- Вписанная и описанная окружности ---

def circumcircle(vertices):

x1,y1,x2,y2,x3,y3 = vertices

D = 2\*(x1\*(y2-y3)+x2\*(y3-y1)+x3\*(y1-y2))

Ux = ((x1\*\*2+y1\*\*2)\*(y2-y3)+(x2\*\*2+y2\*\*2)\*(y3-y1)+(x3\*\*2+y3\*\*2)\*(y1-y2))/D

Uy = ((x1\*\*2+y1\*\*2)\*(x3-x2)+(x2\*\*2+y2\*\*2)\*(x1-x3)+(x3\*\*2+y3\*\*2)\*(x2-x1))/D

r = math.sqrt((Ux-x1)\*\*2 + (Uy-y1)\*\*2)

return int(Ux), int(Uy), int(r)

def incircle(vertices):

x1,y1,x2,y2,x3,y3 = vertices

a = math.hypot(x2-x3,y2-y3)

b = math.hypot(x1-x3,y1-y3)

c = math.hypot(x1-x2,y1-y2)

px = (a\*x1 + b\*x2 + c\*x3)/(a+b+c)

py = (a\*y1 + b\*y2 + c\*y3)/(a+b+c)

s = (a+b+c)/2

r = math.sqrt((s-a)\*(s-b)\*(s-c)/s)

return int(px), int(py), int(r)

# --- PPM ---

def save\_ppm\_ascii(image, path):

w,h = image.size

with open(path,"w") as f:

f.write("P3\n")

f.write(f"{w} {h}\n255\n")

for y in range(h):

row=[]

for x in range(w):

r,g,b = image.getpixel((x,y))

row.append(f"{r} {g} {b}")

f.write(" ".join(row)+"\n")

# --- GUI ---

class App:

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

self.root = root

root.title("Треугольник и окружности")

self.W,self.H=600,600

self.img = Image.new("RGB",(self.W,self.H),(255,255,255))

self.photo = ImageTk.PhotoImage(self.img)

self.vertices = None

ctrl = tk.Frame(root); ctrl.pack(side=tk.LEFT, fill=tk.Y,padx=6,pady=6)

tk.Button(ctrl,text="Открыть SVG",command=self.load\_svg).pack(fill="x",pady=2)

tk.Label(ctrl,text="Метод окружности:").pack(pady=(8,0))

self.method\_var=tk.StringVar(value="equation")

for t,v in [("Уравнение","equation"),("Параметрическое","param"),

("Брезенхем","bresenham"),("Встроенный","builtin")]:

tk.Radiobutton(ctrl,text=t,variable=self.method\_var,value=v).pack(anchor="w")

tk.Button(ctrl,text="Нарисовать",command=self.draw).pack(fill="x",pady=2)

tk.Button(ctrl,text="Очистить",command=self.clear\_canvas).pack(fill="x",pady=2)

tk.Button(ctrl,text="Сохранить PPM",command=self.save\_ppm).pack(fill="x",pady=2)

self.canvas = tk.Label(root,image=self.photo)

self.canvas.pack(side=tk.RIGHT,padx=6,pady=6)

def load\_svg(self):

path = filedialog.askopenfilename(filetypes=[("SVG files","\*.svg")])

if not path: return

vertices = parse\_svg(path)

if not vertices:

messagebox.showerror("Ошибка","Не найден <polygon points=...>")

return

self.vertices = vertices

messagebox.showinfo("SVG",f"Загружены вершины: {self.vertices}")

def draw(self):

if not self.vertices:

messagebox.showerror("Ошибка","Сначала загрузите SVG")

return

self.img = Image.new("RGB",(self.W,self.H),(255,255,255))

draw\_func = dda\_line

draw\_triangle(self.img,self.vertices)

# Описанная окружность

xc,yc,r = circumcircle(self.vertices)

method = self.method\_var.get()

if method=="equation": circle\_equation(self.img,xc,yc,r,(255,0,0))

elif method=="param": circle\_parametric(self.img,xc,yc,r,(0,255,0))

elif method=="bresenham": circle\_bresenham(self.img,xc,yc,r,(0,0,255))

else: circle\_builtin(self.img,xc,yc,r,(0,0,0))

# Вписанная окружность

xi,yi,ri = incircle(self.vertices)

circle\_builtin(self.img,xi,yi,ri,(255,165,0)) # оранжевая

self.update\_canvas()

def clear\_canvas(self):

self.img = Image.new("RGB",(self.W,self.H),(255,255,255))

self.update\_canvas()

def update\_canvas(self):

self.photo = ImageTk.PhotoImage(self.img)

self.canvas.configure(image=self.photo)

self.canvas.image = self.photo

def save\_ppm(self):

path = filedialog.asksaveasfilename(defaultextension=".ppm",

filetypes=[("PPM ASCII","\*.ppm")])

if not path: return

save\_ppm\_ascii(self.img,path)

messagebox.showinfo("Сохранено",f"PPM файл: {path}")

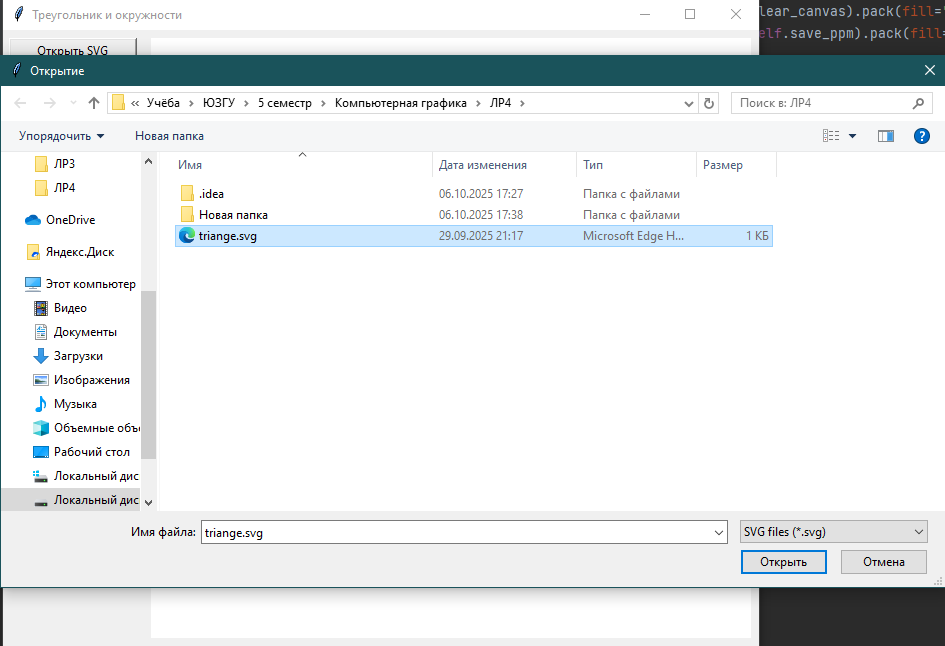
if \_\_name\_\_=="\_\_main\_\_":

root = tk.Tk()

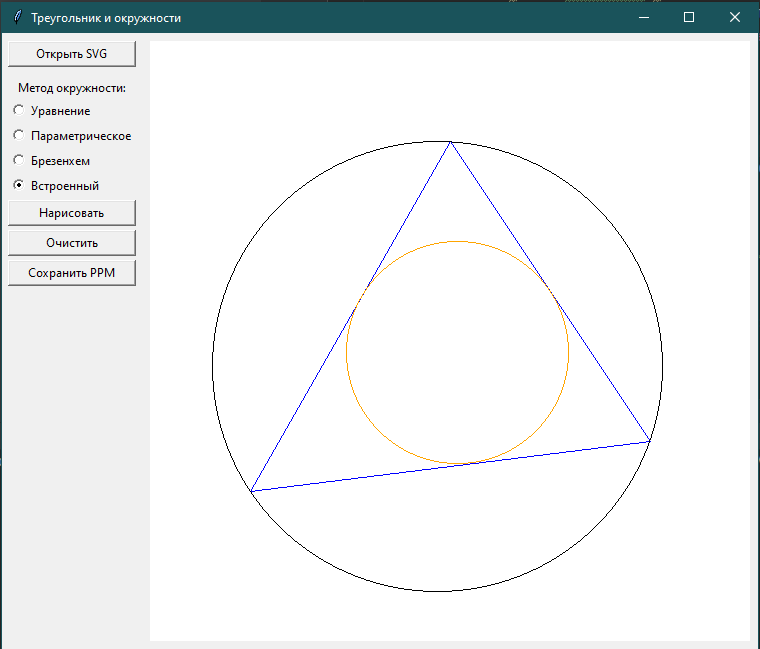
app = App(root)

root.mainloop()

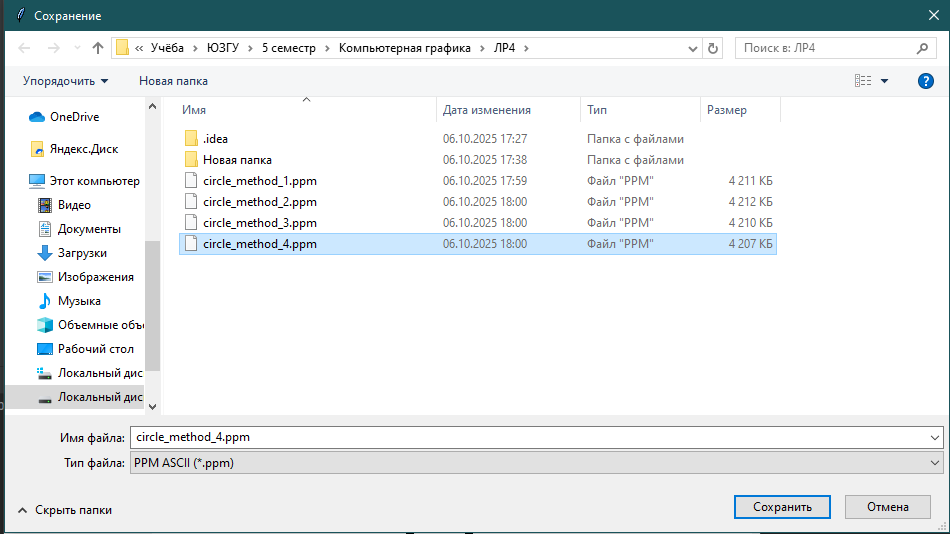
3. Загружаем SVG файл:



4. Выбираем метод и рисуем треугольник с окружностью:



5. Сохраняем результат в формате ppm:



**Контрольные вопросы**

1. Что такое пиксель?

Пиксель (от англ. "picture element" - элемент изображения) - это наименьший логический элемент растрового изображения. В контексте компьютерной графики:

* Представляет собой отдельную точку на экране или в изображении
* Имеет определенные координаты (x, y) и цвет
* В цветных изображениях обычно кодируется комбинацией красного, зеленого и синего компонентов (RGB)
* В программе представлен как точка с координатами, которую мы закрашиваем определенным цветом

2. Что такое растровое изображение?

Растровое изображение - это изображение, состоящее из сетки пикселей (растра), где каждый пиксель имеет свой цвет и положение. Характеристики:

* Состоит из матрицы пикселей фиксированного размера
* Размер определяется шириной и высотой в пикселях
* Качество зависит от разрешения (количества пикселей на единицу длины)
* Примеры форматов: PNG, JPEG, BMP, PPM (которые используются в программе)
* В отличие от векторной графики, при увеличении теряет качество

3. Для чего нужны алгоритмы растеризации окружностей?

Алгоритмы растеризации окружностей необходимы для:

* Отображения окружностей на растровых устройствах (мониторы, принтеры)
* Эффективного вычисления координат пикселей, составляющих окружность
* Минимизации вычислений - использование целочисленной арифметики вместо вещественной
* Создания сглаженных контуров без "ступенчатости"
* Оптимизации скорости отрисовки в реальном времени
* Реализации в аппаратном обеспечении графических ускорителей

4. Какие ещё алгоритмы растеризации окружностей существуют?

Помимо реализованных в программе алгоритмов, существуют:

Алгоритм средней точки (Midpoint Circle Algorithm) - модификация алгоритма Брезенхема

Алгоритм с использованием целочисленной арифметики - избегает операций с плавающей точкой

Алгоритм с использованием симметрии - использует 8-стороннюю симметрию окружности

Алгоритм Xiaolin Wu - для сглаженного (антиалиасинг) рисования окружностей

Алгоритм на основе цепных дробей

Алгоритм с использованием тригонометрических таблиц - для предвычисленных значений

Алгоритм с применением векторных инструкций - для современных процессоров

**ВЫВОД**

В ходе выполнения лабораторной работы мы изучили алгоритмы растеризации окружностей, создали программу для визуализации работы алгоритмов.

Ссылка на гитхаб - https://github.com/Tima2005-rus/ComputerGraphicsLab4