|  |  |
| --- | --- |
| tit.jpg | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Информатика и системы управления»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Лабораторная работа № 1**

|  |  |
| --- | --- |
| **Тема** Реализация и исследование алгоритмов построения отрезков  **Студент** Саркисов А.С.  **Группа ИУ 7-43**  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |  |

Москва.

2020 г.

**Цель работы:**

Научиться применять знания аналитической геометрии для решения практических задач машинной графики, осуществлять построение изображения (в СКУ) объектов, расположенных в МСК.

**Условие задачи:**

На плоскости дано множество точек. Найти такой треугольник с вершинами в этих точках, для которого разность площадей описанной и вписанной окружности максимальна.

**Техническое задание:**

1. Необходимо определить координаты вершин треугольника, по трем заданным точкам, при этом проверить не лежат ли эти точки на одной прямой.
2. Перебрать все возможные треугольники, найдя треугольник с максимальной разностью площадей описанной и вписанной в него окружностью.
3. Найти координаты центра вписанной и описанной окружности для найденного треугольника.

**Теоретический материал:**

1. Перед расчетом разности площадей, вписанной и описанной окружностей треугольника, необходимо проверить, что три точки, являющиеся вершинами треугольника, не лежат на одной прямой.

*(x3 - x1) / (x2 - x1) == (y3 - y1) / (y2 - y1)*

1. Если условие (п. 1) выполняется, то находим радиусы вписанной и описанной окружностей треугольника. Радиус вписанной в треугольник окружности равен отношению площади треугольника и его полупериметра. Радиус описанной окружности равен отношению произведения всех длин сторон треугольника и корня квадратного, умноженного на 4, произведения полупериметра на разность полупериметра и всех длин сторон треугольника поочередно.
2. Если условие (п. 1) выполняется, то нужно составить два уравнения прямых, которые являются серединными перпендикулярами к двум отрезкам, соединяющим точки (например: 1 и 2, 2 и 3). После этого нужно решить систему с двумя неизвестными (в системе два уравнения).

Полученные x и y являются координатами центра описанной окружности.

1. Если условие (п. 1) выполняется, то используя следствие из теоремы:

*Пусть АВС произвольный треугольник, а, b, с длины сторон, лежащие против вершин А, В и С соответственно, М – точка пересечения его биссектрис. Тогда для любой точки О верно равенство*

*. *

*Следствие. Пусть АВС произвольный треугольник, а, b, с длины сторон, лежащие против вершин А, В и С соответственно, М – точка пересечения его биссектрис, О – начало координат. Тогда:*

*, ,   *.

Полученные x и y являются координатами центра вписанной окружности.

**Алгоритм нахождения треугольника с вершинами в этих точках, для которого разность площадей описанной и вписанной окружности максимальна:**

def showTriangle(canvas, pointsList):

pointsList = []

*for* point in listbox.get(0, END):

pointsList.append([float(point[0]), float(point[1])])

pointsList = pointsListFormation(pointsList)

print("pointsList in showTriangle :", pointsList)

*if* (len(pointsList) < 3):

*return* False

trianglePoints = [[0.0, 0.0], [0.0, 0.0], [0.0, 0.0]]

sidesLength = [0.0, 0.0, 0.0]

maxRadiusDiff = 0.0

triangleReqPoints = [[0.0, 0.0], [0.0, 0.0], [0.0, 0.0]]

*for* pointOneInd in range(len(pointsList) - 2):

*for* pointTwoInd in range(pointOneInd + 1, len(pointsList) - 1):

*for* pointThreeInd in range(pointTwoInd + 1, len(pointsList)):

trianglePoints[0] = pointsList[pointOneInd]

trianglePoints[1] = pointsList[pointTwoInd]

trianglePoints[2] = pointsList[pointThreeInd]

*if* (isPointsCollinear(trianglePoints) == False):

*#print(trianglePoints)*

sidesLength = sidesLengthDef(trianglePoints)

radiusCircum, radiusInscribed = radiusCirclesDef(sidesLength)

*#print(radiusCircum, radiusInscribed)*

*if* (radiusCircum - radiusInscribed >= maxRadiusDiff):

triangleReqPoints[0] = trianglePoints[0]

triangleReqPoints[1] = trianglePoints[1]

triangleReqPoints[2] = trianglePoints[2]

*#print(triangleReqPoints)*

maxRadiusDiff = radiusCircum - radiusInscribed

*if* (maxRadiusDiff != 0.0):

sidesLength = sidesLengthDef(triangleReqPoints)

radiusCircum, radiusInscribed = radiusCirclesDef(sidesLength)

print("triangleReqPoints =", triangleReqPoints)

print("sidesLength =", sidesLength)

interPointInscribedX, interPointInscribedY = interPointsInscribedDef(triangleReqPoints, sidesLength)

print("interPointInscribedX, interPointInscribedY =", interPointInscribedX, interPointInscribedY)

print("radiusInscribed =", radiusInscribed)

interPointCircumX, interPointCircumY = interPointsCircumDef(triangleReqPoints)

print("interPointInscribedX, interPointInscribedY =", interPointInscribedX, interPointInscribedY)

print("radiusCircum =", radiusCircum)

paintTriangle(canvas, triangleReqPoints)

paintCircle(canvas, interPointInscribedX, interPointInscribedY, radiusInscribed)

paintCircle(canvas, interPointCircumX, interPointCircumY, radiusCircum)

*trianglePoints* - список, хранящий по три точки, относящиеся к одному треугольнику.

**Алгоритм проверяющий, что точки не лежат на одной прямой:**

def isPointsCollinear(trianglePoints):

*if* (abs(((trianglePoints[2][1] - trianglePoints[0][1]) \* (trianglePoints[1][0] - trianglePoints[0][0])) -

((trianglePoints[2][0] - trianglePoints[0][0]) \* (trianglePoints[1][1] - trianglePoints[0][1]))) < 0.0001):

*return* True

*return* False

**Алгоритм нахождения радиусов вписанной и описанной окружностей по 3-м точкам:**

def radiusCirclesDef(sidesLength):

semiperimeter = 0.0

semiperimeter = (sidesLength[0] + sidesLength[1] + sidesLength[2]) / 2

sidesProduct = sidesLength[0] \* sidesLength[1] \* sidesLength[2]

radiusCircum = sidesProduct / (4 \* math.sqrt(semiperimeter \*

(semiperimeter - sidesLength[0]) \*

(semiperimeter - sidesLength[1]) \*

(semiperimeter - sidesLength[2])))

radiusInscribed = math.sqrt((semiperimeter - sidesLength[0]) \*

(semiperimeter - sidesLength[1]) \*

(semiperimeter - sidesLength[2]) / semiperimeter)

*return* radiusCircum, radiusInscribed

**Алгоритм нахождения центра описанной окружностей:**

def interPointsCircumDef(trianglePoints):

matrixXOne = [

[trianglePoints[0][0] \* trianglePoints[0][0] + trianglePoints[0][1] \* trianglePoints[0][1], trianglePoints[0][1], 1.0],

[trianglePoints[1][0] \* trianglePoints[1][0] + trianglePoints[1][1] \* trianglePoints[1][1], trianglePoints[1][1], 1.0],

[trianglePoints[2][0] \* trianglePoints[2][0] + trianglePoints[2][1] \* trianglePoints[2][1], trianglePoints[2][1], 1.0]

]

matrixYOne = [

[trianglePoints[0][0], trianglePoints[0][0] \* trianglePoints[0][0] + trianglePoints[0][1] \* trianglePoints[0][1], 1.0],

[trianglePoints[1][0], trianglePoints[1][0] \* trianglePoints[1][0] + trianglePoints[1][1] \* trianglePoints[1][1], 1.0],

[trianglePoints[2][0], trianglePoints[2][0] \* trianglePoints[2][0] + trianglePoints[2][1] \* trianglePoints[2][1], 1.0]

]

matrixXYTwo = [

[trianglePoints[0][0], trianglePoints[0][1], 1.0],

[trianglePoints[1][0], trianglePoints[1][1], 1.0],

[trianglePoints[2][0], trianglePoints[2][1], 1.0]

]

interPointCircumX = matrixDetDef(matrixXOne) / (2 \* matrixDetDef(matrixXYTwo))

interPointCircumY = matrixDetDef(matrixYOne) / (2 \* matrixDetDef(matrixXYTwo))

*return* interPointCircumX, interPointCircumY

**Алгоритм нахождения центра вписанной окружностей:**

def interPointsInscribedDef(trianglePoints, sidesLength):

interPointInscribedX = (trianglePoints[0][0] \* sidesLength[0] +

trianglePoints[1][0] \* sidesLength[1] +

trianglePoints[2][0] \* sidesLength[2]) / (sidesLength[0] + sidesLength[1] + sidesLength[2])

interPointInscribedY = (trianglePoints[0][1] \* sidesLength[0] +

trianglePoints[1][1] \* sidesLength[1] +

trianglePoints[2][1] \* sidesLength[2]) / (sidesLength[0] + sidesLength[1] + sidesLength[2])

*return* interPointInscribedX, interPointInscribedY

**Алгоритм отображения введенных точек на экран (с учетом масштабирования):**

def showPoints(canvas, pointsList):

*if* (pointsList == []):

pointsList = pointsListFormation(pointsList)

*for* point in pointsList:

paintOnePoint(canvas, point[0], point[1], point[2], point[3])

*return* pointsList

def paintOnePoint(canvas, x, y, xOriginal, yOriginal):

canvas.create\_oval(x - 2, y - 2, x + 2, y + 2, fill="black")

pointCoordinatesString = "(" + str(xOriginal) + ";" + str(yOriginal) + ")"

canvas.create\_text(x + 5, y + 10, text=pointCoordinatesString, justify=CENTER, font="Verdana 7", fill="grey")

ef pointsListFormation(pointsList):

*if* (pointsList == []):

listbox.delete(0, END)

pointsString = pointsEntry.get()

coordList = list(map(float, pointsString.split()))

*for* i in range (0, len(coordList), 2):

pointsList.append([coordList[i], coordList[i + 1], coordList[i], coordList[i + 1]])

listbox.insert(0, [coordList[i], coordList[i + 1]])

maxXY, minXY = maxXYDef(pointsList)

*if* (minXY < 0):

*for* i in range(len(pointsList)):

pointsList[i][0] += (abs(minXY))

pointsList[i][1] += (abs(minXY))

maxXY, minXY = maxXYDef(pointsList)

singleSegment = 500 / abs(maxXY)

*for* i in range(len(pointsList)):

pointsList[i][0] \*= singleSegment

pointsList[i][1] \*= singleSegment

pointsList[i][1] -= 550

pointsList[i][1] \*= -1

pointsList[i][0] += 25

pointsList[i][1] -= 25

print("pointsList in pointsListFormation:", pointsList)

*return* pointsList

**Пример работы программы**

Ввод точек и их отображение на плоскости (в программе так же можно изменять, удалять и добавлять точки):

Изображение выглядит как снимок экрана

Автоматически созданное описание

Решение задачи: Изображение выглядит как текст

Автоматически созданное описание  