|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 № 4**

|  |  |
| --- | --- |
| **Тема** Реализация и исследование алгоритмов построения окружностей и эллипсов  **Студент** Якуба Д. В.  **Группа** ИУ7-43  **Оценка (баллы)** \_ **\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель** Куров А. В. |  |

Москва

2020 г.

Оглавление

[Цель работы 3](#_Toc37206710)

[Техническое задание 3](#_Toc37206711)

[Теоретическая часть 3](#_Toc37206712)

[Описание и реализация алгоритмов 4](#_Toc37206713)

[Алгоритм ЦДА 4](#_Toc37206714)

[Алгоритм, записанный на ЯП Python: 5](#_Toc37206715)

[Алгоритм Брезенхема с действительными коэффициентами 5](#_Toc37206716)

[Алгоритм, записанный на ЯП Python: 6](#_Toc37206717)

[Алгоритм Брезенхема с целыми коэффициентами 7](#_Toc37206718)

[Алгоритм, записанный на ЯП Python: 8](#_Toc37206719)

[Алгоритм Брезенхема с устранением ступенчатости 8](#_Toc37206720)

[Алгоритм, записанный на ЯП Python: 9](#_Toc37206721)

[Алгоритм Ву 10](#_Toc37206722)

[Алгоритм, записанный на ЯП Python: 10](#_Toc37206723)

[Пользовательский интерфейс 11](#_Toc37206724)

[Сравнение визуальных характеристик 19](#_Toc37206725)

[Алгоритм ЦДА 19](#_Toc37206726)

[Алгоритм Брезенхема с действительными коэффициентами 26](#_Toc37206727)

[Алгоритм Брезенхема с целыми коэффициентами 28](#_Toc37206728)

[Алгоритм Брезенхема с устранением ступенчатости 30](#_Toc37206729)

[Алгоритм Ву 32](#_Toc37206730)

[Алгоритм Tkinter.canvas.create\_line 34](#_Toc37206731)

[Все алгоритмы на единой плоскости 36](#_Toc37206732)

[Исследование временных характеристик 37](#_Toc37206733)

# Цель работы

**Р**еализация алгоритмов построения окружности, исследование и сравнение визуальных и временных характеристик алгоритмов.

# Техническое задание

1.Реализовать алгоритмы построения окружности на основе

* Канонического уравнения
* Параметрического уравнения ,
* Алгоритма Брезенхема
* Алгоритма средней точки
* Построение окружности с помощью библиотечной функции

Пользователь выбирает из списка определенный алгоритм, задает координаты центра, радиус, цвет рисования.

Визуальные характеристики исследуются путем рисования той же окружности цветом фона, но с помощью другого алгоритма.

2. Реализовать алгоритмы построения эллипса на основе

* Канонического уравнения
* Параметрического уравнения
* Брезенхема (модифицируется самостоятельно)
* Алгоритма средней точки
* Построение эллипса с помощью библиотечной функции

Пользователь выбирает из списка определенный алгоритм, задает координаты центра, полуоси, цвет рисования.

Визуальные характеристики исследуются путем рисования того же эллипса цветом фона, но с помощью другого алгоритма.

П. 1 и 2 предусматривают рисование одиночных кривых.

3. Сравнение визуальных характеристик разных алгоритмов при рисовании спектра концентрических окружностей.

Пользователь выбирает из списка определенный алгоритм, задает координаты центра, цвет рисования, три из следующих четырех параметров: начальный радиус, конечный радиус, шаг изменения радиуса, количество окружностей.

Визуальные характеристики исследуются путем рисования того же спектра окружностей цветом фона, но с помощью другого алгоритма.

4. Сравнение визуальных характеристик разных алгоритмов при рисовании спектра концентрических эллипсов.

Пользователь выбирает из списка определенный алгоритм, задает координаты центра, цвет рисования, начальные значения полуосей, шаг изменения одной из полуосей, количество эллипсов.

Визуальные характеристики исследуются путем рисования того же спектра эллипсов цветом фона, но с помощью другого алгоритма.

Дополнительное задание.

Сравнить временные характеристики разных алгоритмов, построив в одном поле вывода (в одной системе координат и одном масштабе) графики зависимости времени работы алгоритма от радиуса (для окружности).

Для эллипсов построить аналогичную зависимость (зависимость времени работы алгоритма от изменения полуоси. Имеется в виду, что вторая полуось тоже будет изменяться см. п.4).

# Теоретическая часть

## Генерация окружности

Чтобы построить полную окружность, достаточно сгенерировать ее одну восьмую часть. Остальные части получаются затем путем симметричного отражения относительно определенной прямой. Так, отражая одну восьмую часть, построенную в первом октанте для углов в диапазоне 0°-45°, относительно прямой с уравнением Y=X, получим одну четвертую часть, лежащую в первом квадранте. Отразив эту четверть относительно прямой X=0, получим одну вторую часть, лежащую выше оси абсцисс, наконец, отразив эту полуокружность относительно прямой Y=0, получим полную окружность.

# Описание и реализация алгоритмов

## Алгоритм на основе канонического уравнения

Из курса аналитической геометрии нам известно, что окружность можно описать следующим каноническим уравнением:

, что справедливо для окружности с центром в точке начала координат.

Для окружности, центр которой не совпадает с точкой начала координат:

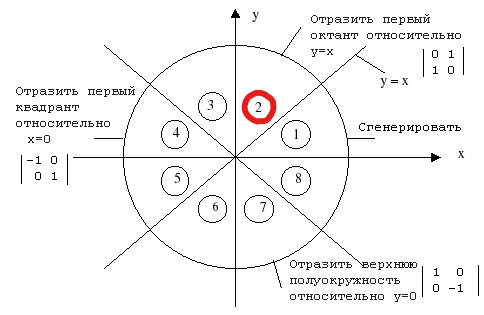
Где *x0, y0 –* координаты центра окружности.

С помощью уравнения (2) имеем возможность через *y* или *x* выразить точки окружности.

Выразим через уравнение (2) *y*:

Так как нам достаточно нарисовать только 1/8 часть окружности, а выше говорилось о том, что строим окружность мы во 2 октанте (красная отметка на рисунке ниже), то при построении мы воспользуемся следующей формулой:

, так как первый октант находится над прямой, проходящей через центр окружности и параллельной оси X.



Выбор 2-го октанта для построения 1/8 окружности обусловлен тем фактом, что в нём приращение значения *y* меньше приращения значения по *x*, что позволит нам построить непрерывную кривую.

### Алгоритм, записанный на ЯП Python:

def canonicalCircleAlg(xCenter, yCenter, radius, colour = "#000000"):  
 pointsArray = []  
 sqrRad = radius \* radius  
 for curX in range(xCenter, round(xCenter + radius / sqrt(2)) + 1):  
 curY = yCenter + sqrt(sqrRad - (curX - xCenter) \* (curX - xCenter))  
 pointsArray.append((curX, curY, colour))

reflectPointsXY(pointsArray, xCenter, yCenter)  
 reflectPointsY(pointsArray, xCenter)  
 reflectPointsX(pointsArray, yCenter)  
 return pointsArray

## Алгоритм на основе канонического уравнения

Из курса аналитической геометрии нам также известно следующее параметрическое уравнение окружности:

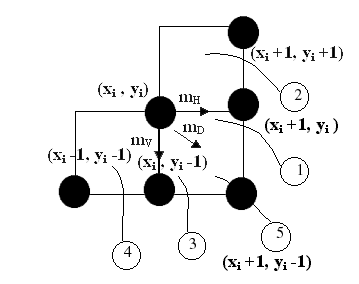
Чтобы определить точки окружности, нужно выбрать некоторый шаг, равный для параметра *t*, и рассчитать для каждого значения этого параметра значения координат соответствующих точек окружности. Значение величины шага приравнивается указанной величине, так как в таком случае угловой шаг будет уменьшаться при увеличении радиуса, что позволит отобразить кривую непрерывной.

### Алгоритм, записанный на ЯП Python:

def parameterCircleAlg(xCenter, yCenter, radius, colour = "#000000"):  
 pointsArray = []  
 angleStep = 1 / radius  
 i = 0  
 while i <= pi / 4:  
 curX = xCenter + radius \* cos(i)  
 curY = yCenter + radius \* sin(i)  
 pointsArray.append((curX, curY, colour))  
 i += degreeStep  
  
 reflectPointsXY(pointsArray, xCenter, yCenter)  
 reflectPointsY(pointsArray, xCenter)  
 reflectPointsX(pointsArray, yCenter)  
 return pointsArray

## Алгоритм Брезенхема построения окружностей

В данный алгоритм строится на том, что для любой заданной точки на окружности при генерации по часовой стрелке существует только три возможности выбрать следующий пиксел, наилучшим образом приближающий окружность: горизонтально вправо, по диагонали вниз и вправо, вертикально вниз.

Таким образом, перед нами стоит выбор между тремя пикселями, показанных квадратами на рисунке.

Разность между квадратами расстояний от центра окружности до диагонального пиксела (*xi, + 1, уi - 1*) и от центра до точки на окружности *R2* равна следующей величине:

При этом, при реализации алгоритма, следует анализировать только знак ошибки. В таком случае, мы будем иметь три следующих случая:

Случай 1.

Расстояние до центра окружности больше, чем до диагонального пикселя, диагональная точка лежит внутри реальной окружности, выбор стоит между горизонтальным и диагональным пикселем.

Введём следующую величину:

Таким образом теперь мы можем проанализировать разницу расстояний от горизонтального и диагонального пиксела до окружности. Если полученная при вычислении величина положительна, то расстояние от диагонального пикселя до окружности меньше, чем от горизонтального. По аналогии для случая, когда полученная величина меньше нуля, расстояние до горизонтального пикселя меньше, следует взять именно его.

Чтобы сократить количество вычислений (а мы видим тут и возведение в квадрат, и модуль, что слишком для нас затратно по времени), рассматриваются два следующих случая:

1)

В таком случае, раскрыв модули, получим:

Дополнение до полного квадрата члена с помощью добавления и вычитания дает:

Таким образом получаем:

И это существенно упрощает вычисления.

2)

Раскрыв модули, получаем:

Заметим, что случай 2) в программной реализации алгоритма не рассматривается отдельно, так как при том же расположении пикселей, что и в рассмотренном 2), случай 1) тоже будет давать отрицательную величину.

Так как для рассмотрения и проверки случая (2) нам понадобятся формулы случая (3), для начала приведём случай (3).

Случай 3.

Расстояние до диагонального пикселя больше, чем до центра окружности, диагональная точка лежит вне окружности, выбираем либо диагональный, либо вертикальный.

Введём величину:

Вновь рассмотрим два случая и раскроем модули.

1)

Имеем:

И таким образом:

2)

Заметим, что случай 2) в программной реализации алгоритма не рассматривается отдельно, так как при том же расположении пикселей, что и в рассмотренном 2), случай 1) тоже будет давать положительную величину.

Случай 2.

Окружность проходит через диагональный пиксел.

Выбор – однозначно диагональный пиксел.

Проверим предыдущие условия:

# Пользовательский интерфейс

Если сильно захочется, то сделаешь.

# Сравнение визуальных характеристик

## Алгоритм …

## Все алгоритмы на единой плоскости

Алгоритмы идут слева-направо сверху-вниз как они упоминались в отчёте.

# Исследование временных характеристик