|  |  |
| --- | --- |
| **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 (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.04 Программная инженерия**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 4 |

**Название:**

Реализация и исследование алгоритмов генерации окружности и эллипса

**Дисциплина:** Компьютерная графика

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ7-43Б |  | 8.04.2020 | С.С. Кононенко |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | А.В. Куров |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2020

**1. Цель работы**

Реализация алгоритмов построения окружности, исследование и сравнение визуальных и временных характеристик алгоритмов.

**2. Техническое задание**

1. Реализовать алгоритмы построения окружности на основе:
   1. Канонического уравнения;
   2. Парметрического уравнения;
   3. Алгоритма Брезенхема;
   4. Алгоритма средней точки;
   5. Библиотечной функции.
2. Реализовать алгоритмы построения эллипса на основе:
   1. Канонического уравнения;
   2. Парметрического уравнения;
   3. Алгоритма Брезенхема;
   4. Алгоритма средней точки;
   5. Библиотечной функции.
3. Сравнить визуальные характеристики разных алгоритмов при рисовании спектра концентрических окружностей и эллипсов.
4. Сравнить временные характеристики разных алгоритмов, построив в одном поле вывода графики зависимости времени работы алгоритма от радиуса (для окружности) и от изменения полуоси (для эллипса).

**3. Теоретический материал**

Эллипс можно описать с помощью канонического уравнения:

**x2/a2 + y2/b2 = 1.**

Так же эллипс можно описать с помощью параметрических уравнений:

**x = a \* cos(t)**

**y = b \* sin(t)**

Данные уравнения приведены для эллипса с центром в (0; 0). Для переноса центра в каноническом уравнении нужно заменить **x → (x — x0), y → (y - y0)***,* а в системе параметрических уравнений - **a \* cos(t) → x0 + a \* cos(t), b \* sin(t) → y0 + b \* sin(t),** где x0, y0 — координаты центра эллипса

Окружность — это эллипс, полуоси которого равны. Таким образом, если домножить каноническое уравнение эллипса на полуось (обе полуоси равны), то можно получить уравнение окружности:

**x2 + y2 = R2**

C параметрическими уравнениями окружности производятся аналогичные преобразования (домножение на полуось):

**x = R \* cos(t)**

**y = R \* cos(t)**

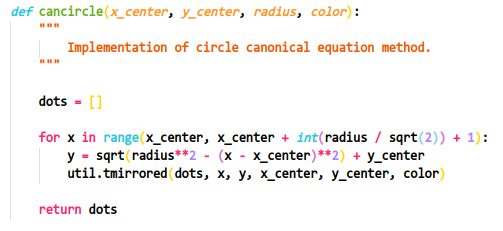
Для переноса центра окружности в произвольную точку требуется сделать преобразования, аналогичные преобразованиям в эллипсе.

**4. Реализация алгоритмов**

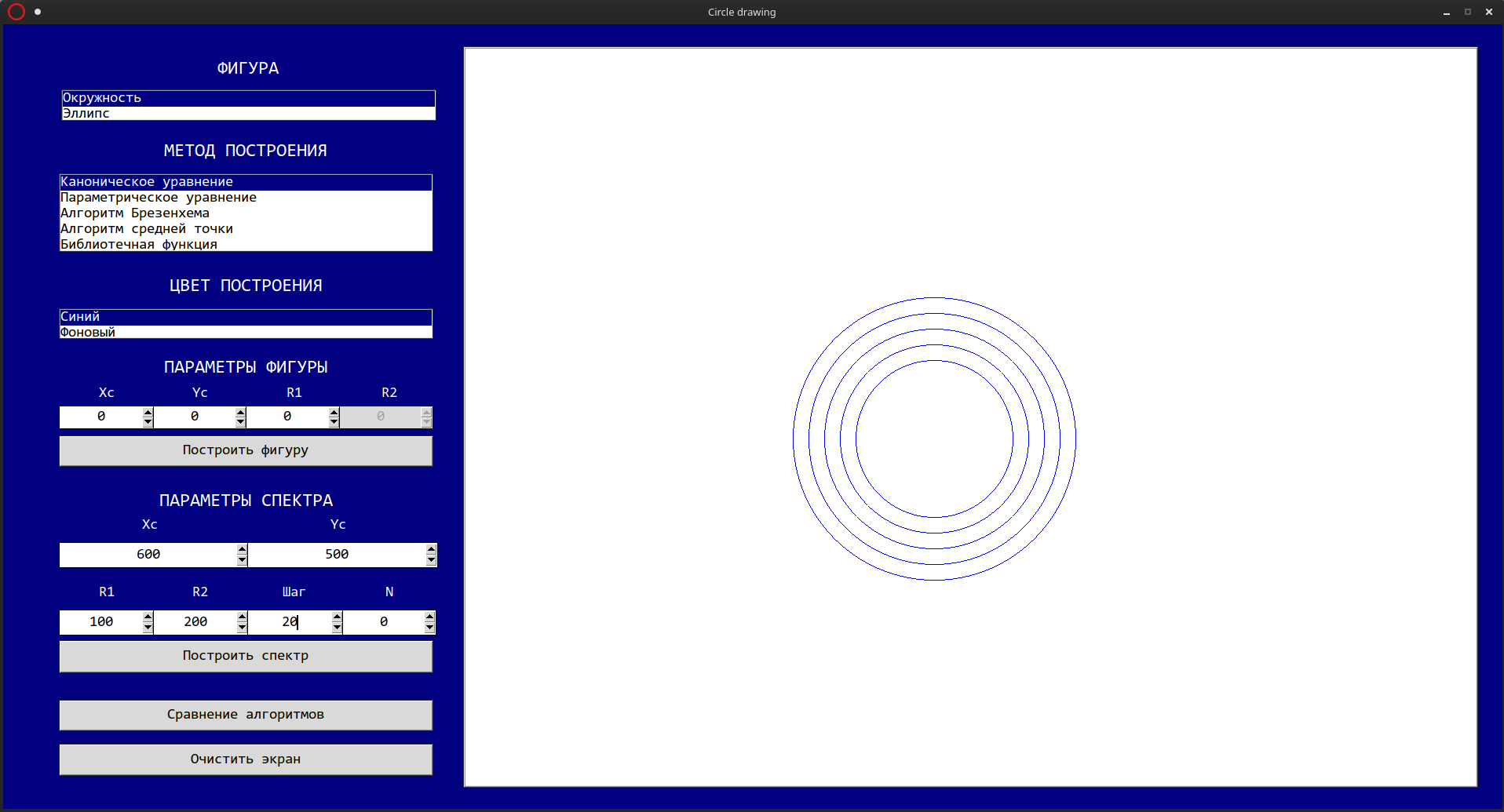
Работа алгоритмов будет демонстрироваться на *спектре кривых*.

**4.1 Алгоритмы построения окружности**

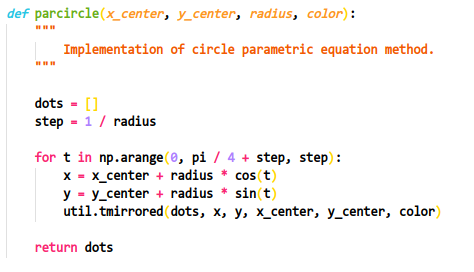
**4.1.1 Каноническое уравнение**

****

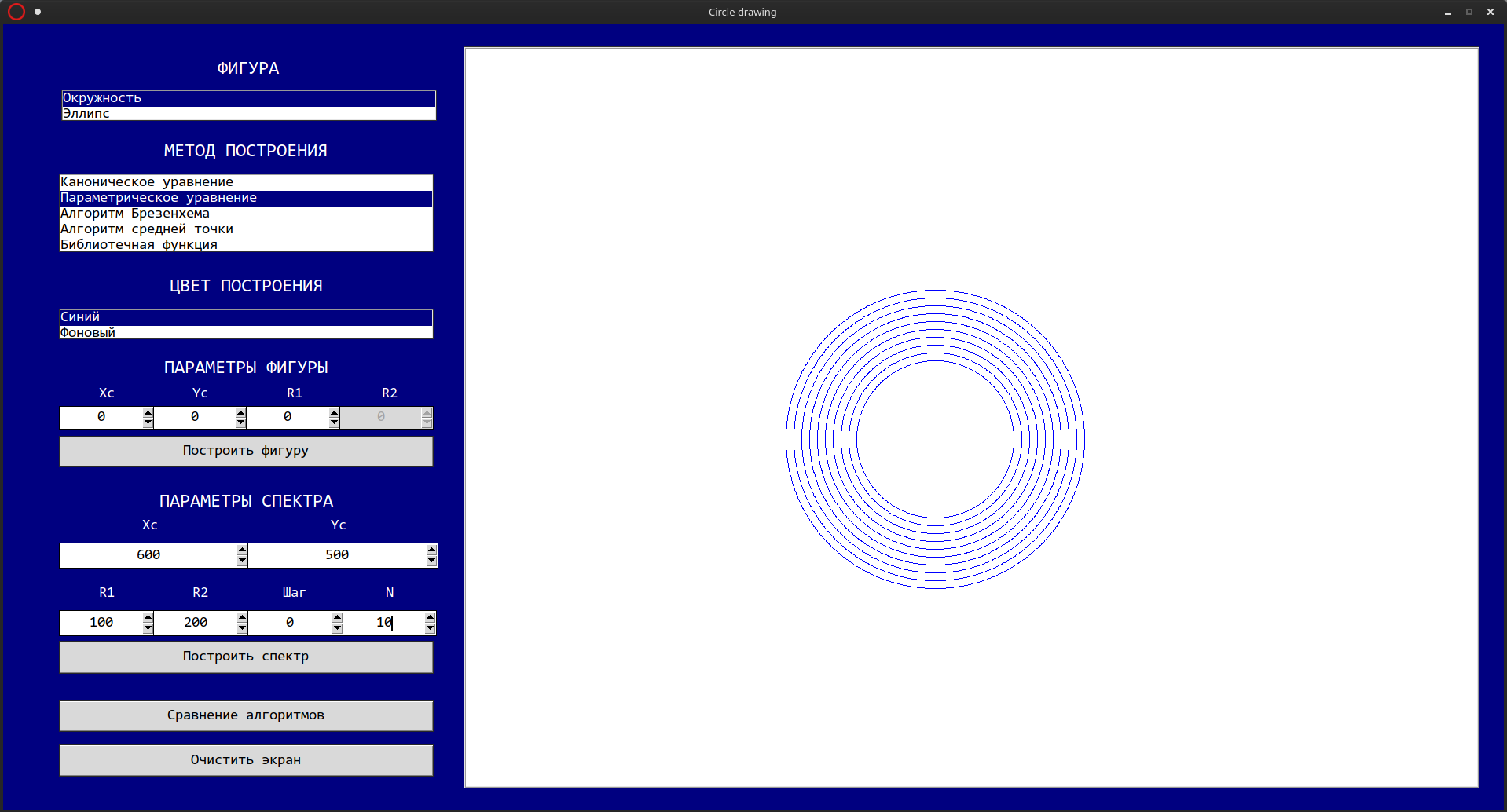
Из канонического уравнения окружности выржается (в данном случае) координата **y**. Устанавливая приращение аргумента **x = 1,** находятся точки только **1/8** части окружности. Затем найденные точки отражаются относительно осей **X** и **Y**, а также относительно **биссектрисы**. Таким образом можно получить окружность целиком.

****

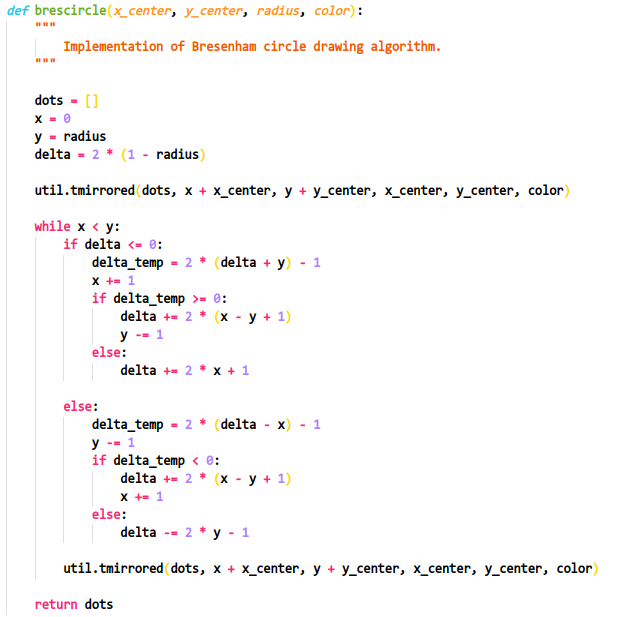
**4.1.2 Параметрическое уравнение**

****

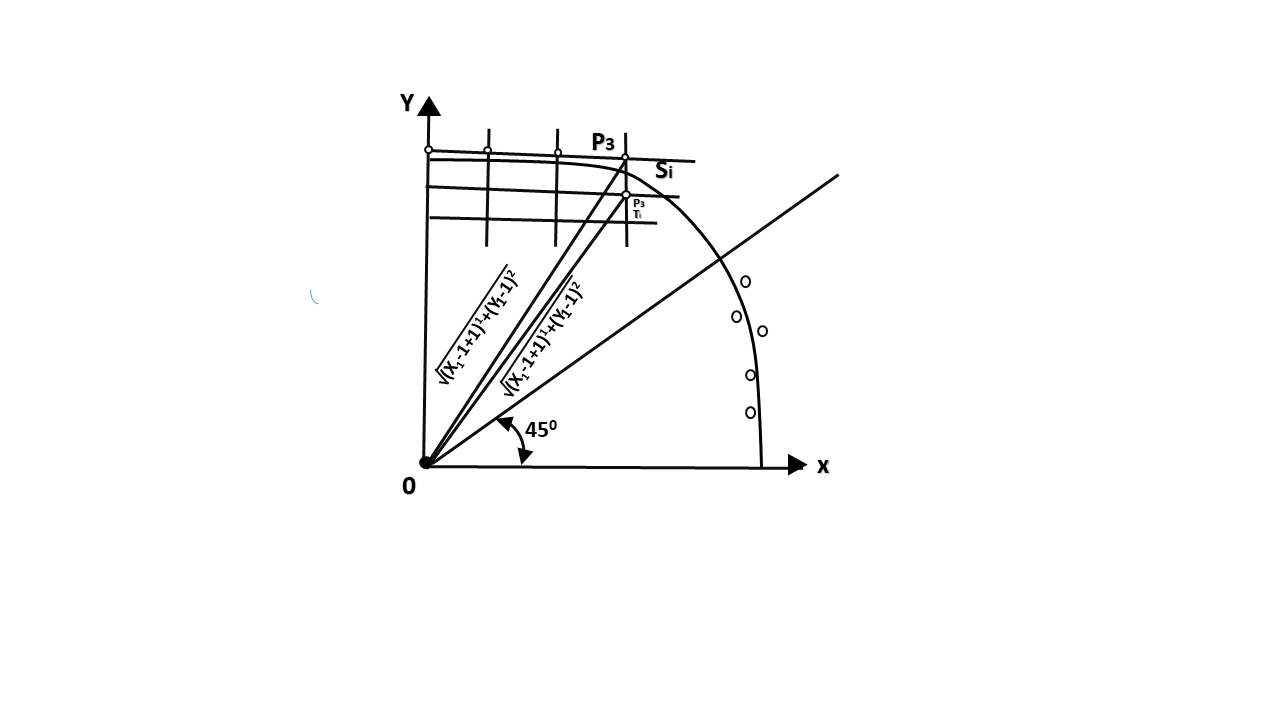
Для изменения угла, фигурирующего в системе параметрических уравнений для окружности, выбирается шаг равный **1/R.** Даннный шаг выбран потому, что расстояние между рисуемыми пикселями пропорционально углу между ними (вершина угла находится в центре окружности). Как и в случае с каноническим уравнением, находятся только точки 1/8 части окружности, затем найденные точки отражаются по трем направлениям.



**4.1.3 Алгоритм Брезенхема**

****

В данном алгоритме окружность строится путем рассмотрения возможных ситуаций прохода прямой и, исходя из этого положения, рассмотрения расстояния до ближайших пикселей.



Так как окружность – симметричная фигура, можнро построить 1/4 или 1/8 окружности, и с помощью отражений и поворотов получить полную фигуру (как делалось в случае параметрического и канонического уравнений).

Рассмотрим первую координатную четверть. Будем начинать построение фигуры из точки с координатами (0, R) до точки (R, 0), тем самым описав окружность в первой четверти. Возможно три перехода из текущего пикселя (xi, yi):

1. (xi + 1, yi) – горизонтальное направление.

2. (xi, yi - 1) – вертикальное направление.

3. (xi + 1, yi - 1) – диаганальное направление.

**1. di < 0**

Посчитаем разность квадратов модулей между диагональным и горизонатнльным пикселем.

**b1 = |(xi + 1)2 + yi2- R2| - |(xi + 1)2 + (yi - 1)2 - R2|**

При **b1 <= 0** растояние до диаганального пикселя больше, чем до горизонтального. Выбираем пиксель (xi + 1, yi).

При **b1 > 0** наооборот, выбираем диаганальный пиксель (xi + 1, yi - 1)

Случай 1:

Первый модуль из **b1 >= 0**, второй модуль **< 0**.

Зная об этом, мы можем раскрыть модуль и в итоге получить: **b1 = 2(di +yi) - 1**

Случай 2:

Оба модуля из **b1 < 0**. Раскроем модули и получим: **b1 = 2yi + 1 < 0**

**2. di > 0**

**b2 = |(xi + 1)2 + (yi - 1) - R2| - |xi2 + (yi - 1)2 - R2|**

При **b2 <= 0** растояние до вертикального пикселя больше, чем до диаганального. Выбираем пиксель (xi + 1, yi - 1).

При **b2 > 0** выбираем вертикальный (xi, yi - 1)

Случай 3:

Первый модуль из **b2 >= 0**, второй модуль **< 0**.

Раскрываем модули и получаем: **b2 = 2(di - xi) - 1**

Случай 4:

Первый модуль из **b2 > 0**, второй **>= 0**.

Раскроем модули и получим: **b2 = 2xi + 1 > 0**

**3. di = 0.** Выбор очевиден, пиксель находится прямо на окружности.

Случай 5:

**b1 = |(xi + 1)2 + yi2 — R2| > 0**

**b2 = - |xi2 + (yi - 1)2 - R2| < 0**

*1. Горизонтальный шаг*

**xi+1 = xi + 1, yi + 1 = yi**

**di + 1 = di + 2xi + 1+ 1**

2. Диагональный шаг

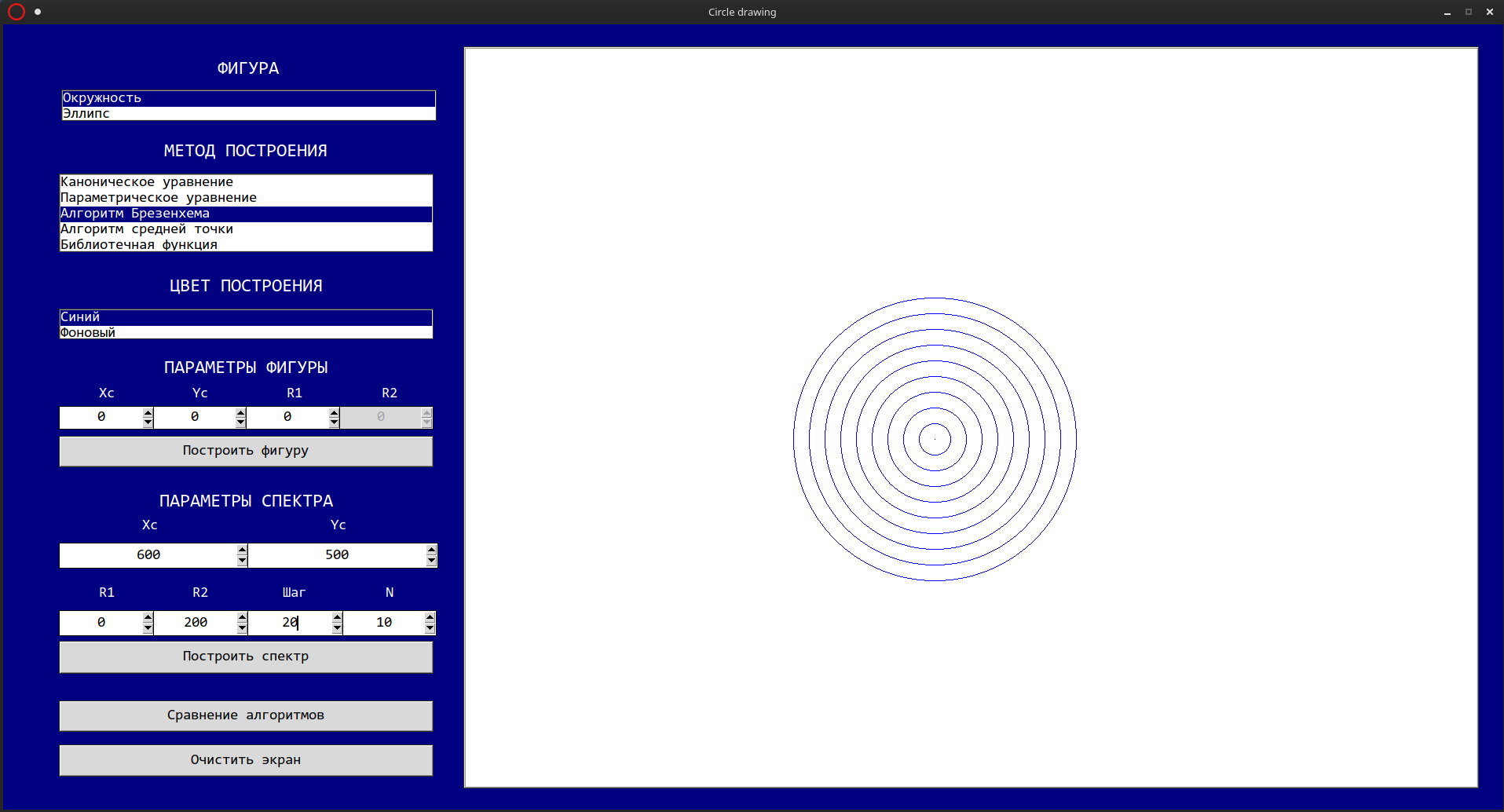
**xi+1 = xi + 1, yi + 1 = yi - 1**

**di + 1 = di + 2xi + 1 - 2yi + 1 + 2**

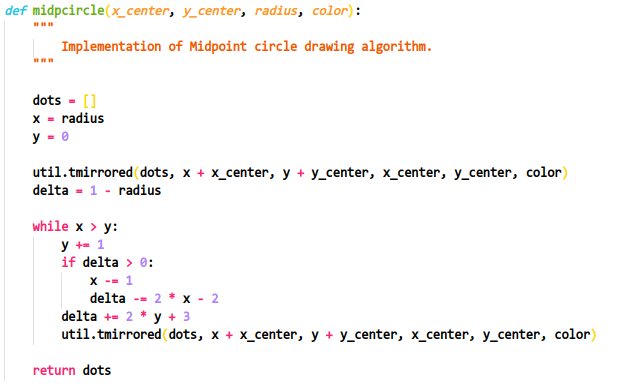
3. Вертикальный шаг

**xi+1 = xi, yi + 1 = yi - 1**

**di + 1 = di + 2yi + 1 + 1**



**4.1.4. Алгоритм средней точки**

****

В данном алгоритме вводится следующая функция:

**bk = (xk  - 0.5)2+ (yk  + 1)2  - R2**

где хk, yk — координаты пикселя, выбранного на последнем шаге. Данная функция содержит разность квадрата расстояния от центра окружности до «средней точки» рассматриваемого на текущий момент пикселя и квадрата расстояния до идеальной окружности. Если расстояние до средней точки будет больше, то функция принимает положительное значение и выбирается диагональный пиксель (средняя точка вне окружности).

Получаем следующее условие:

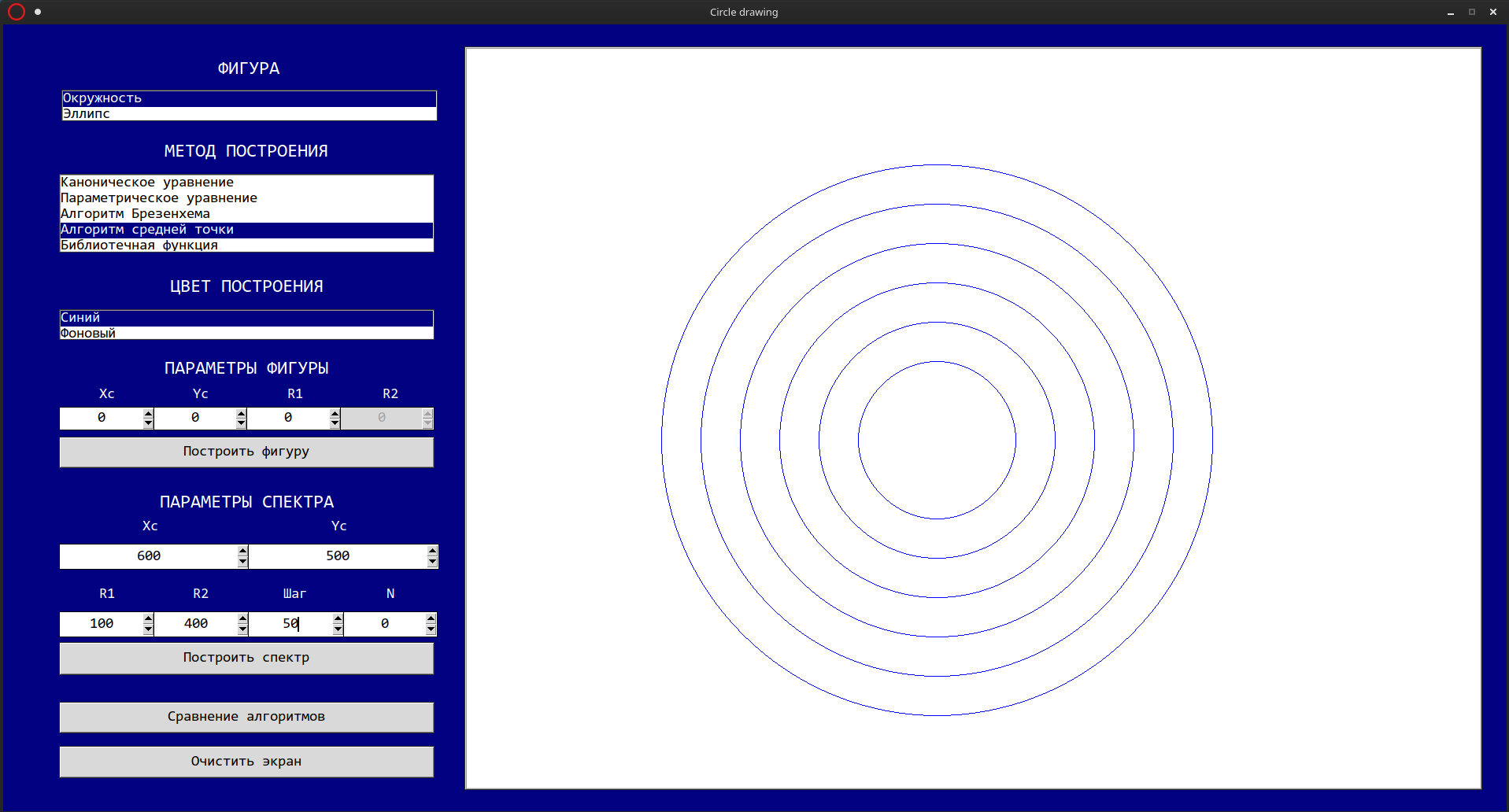
**bk < 0** → верхний, иначе — диагональный (знак может быть и <, и <=)

Расчет каждый раз величины данной функции очень емок в плане ресурсов, поэтому аналогично алгоритму брезенхема была найдена величина «приращения» данной функции:

**d = bk+1  - bk = 2 (yk + 1) + 1** — при вертикальном шаге;

**d = bk+1  - bk = 2 (yk + 1) + 1 - 2 (хk - 1)** — при диагональном шаге.

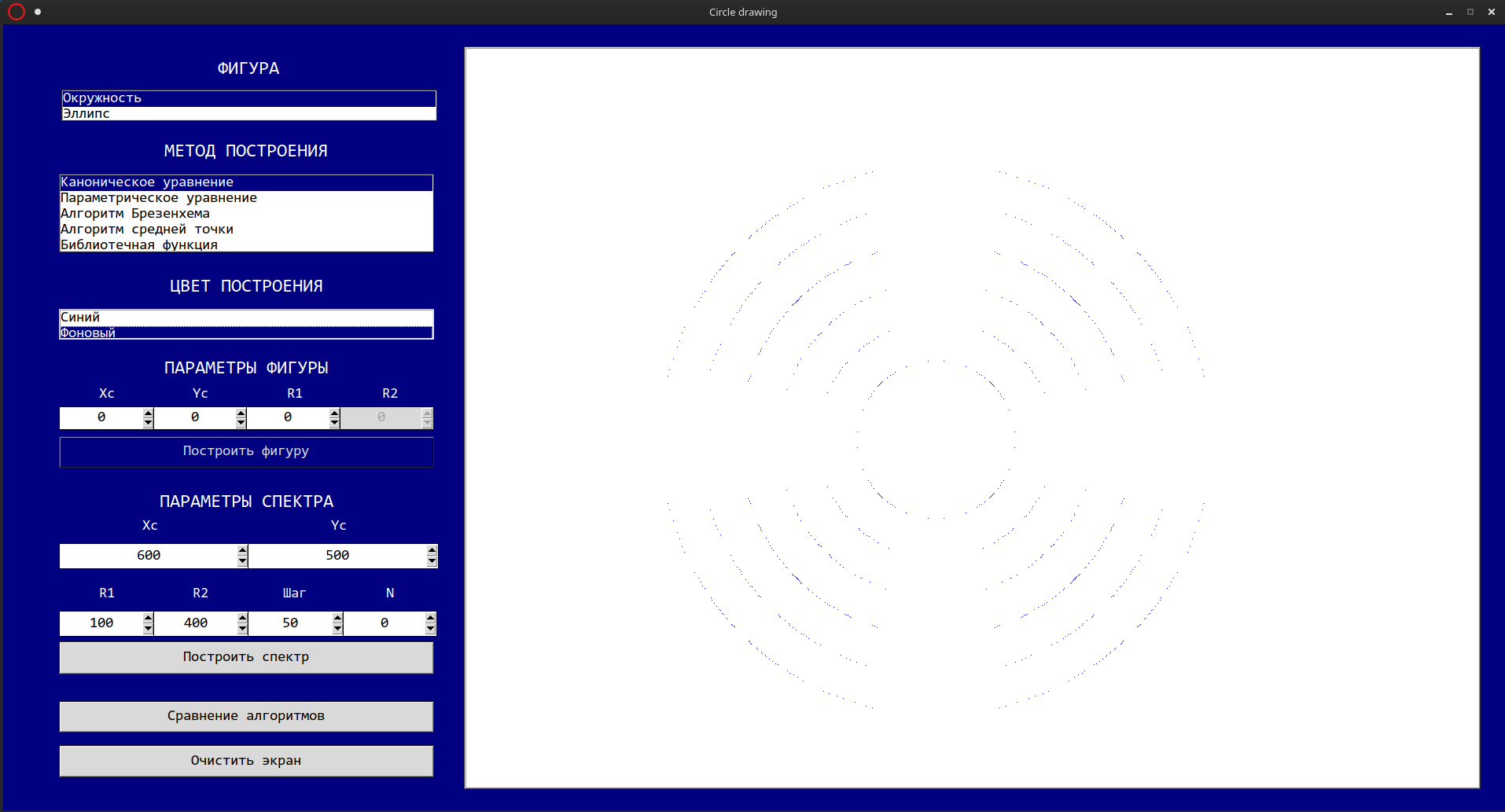
Таким образом: **bk+1  = bk + d**



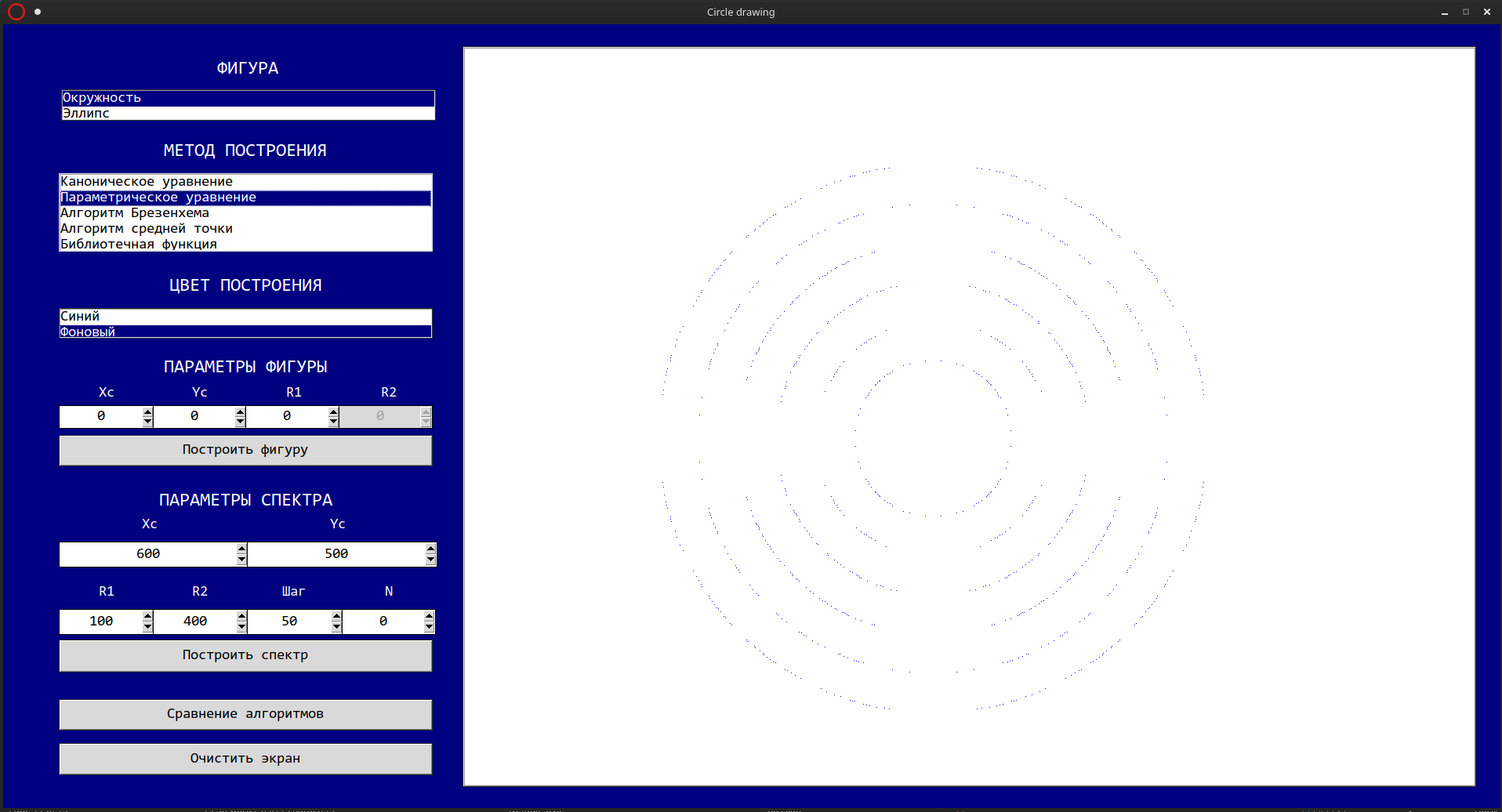
**4.1.5 Библиотечная функция**

Рассмотрим наложения различных алгоритмов на спектр окружностей, построенных библиотечной функцией:

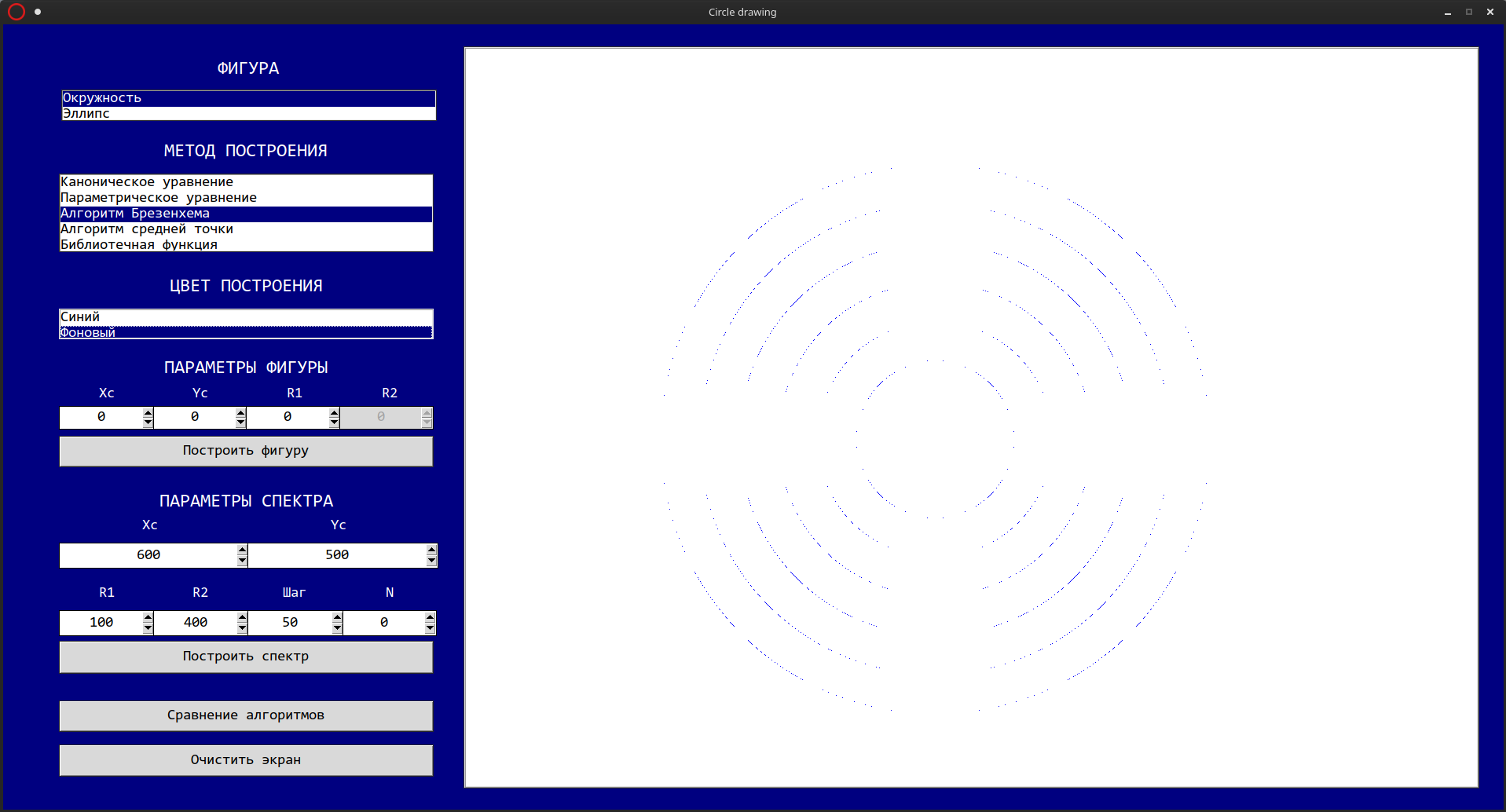
Каноническое уравнение:



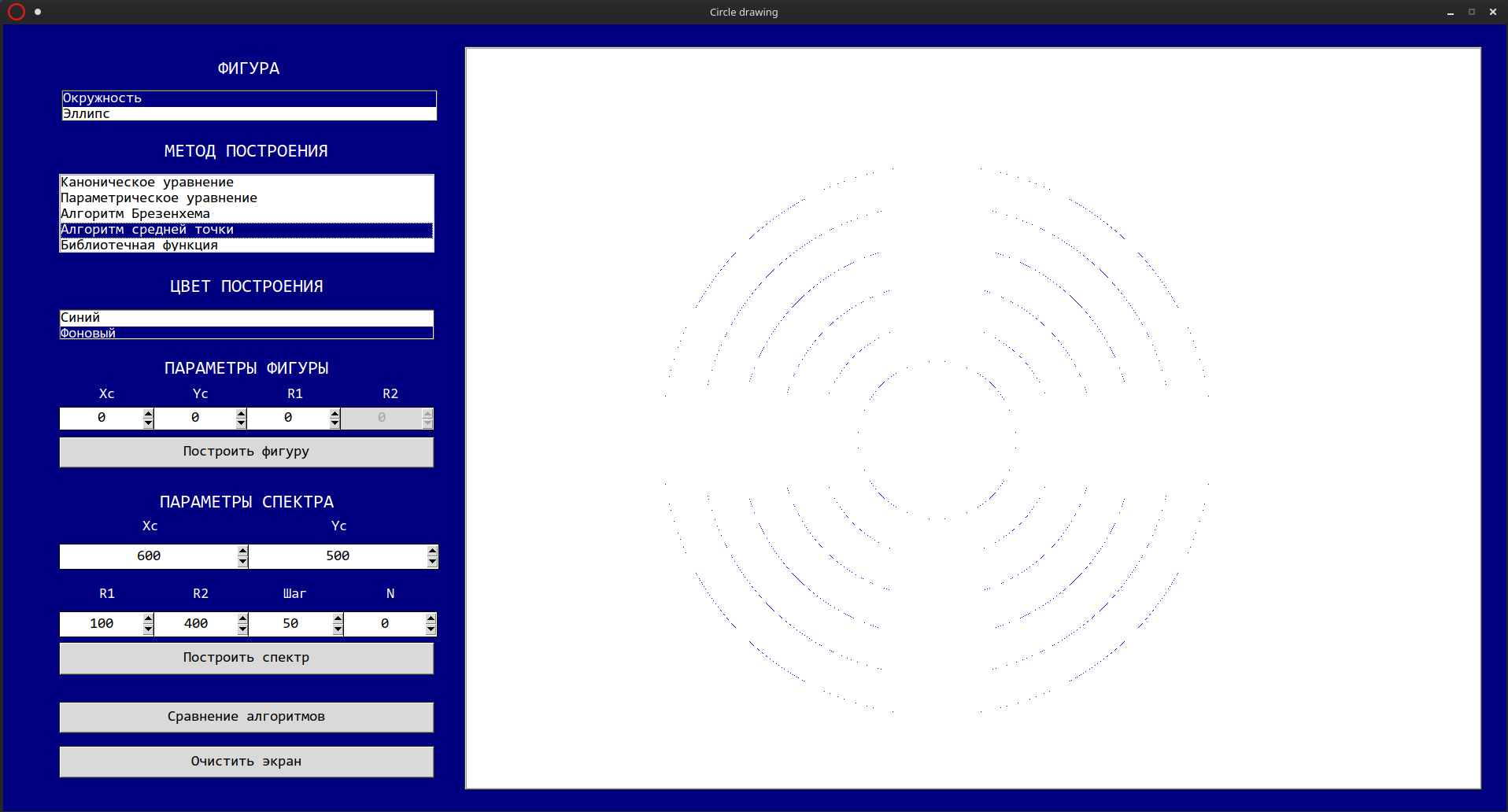
Параметрическое уравнение:



Алгоритм Брезенхема:



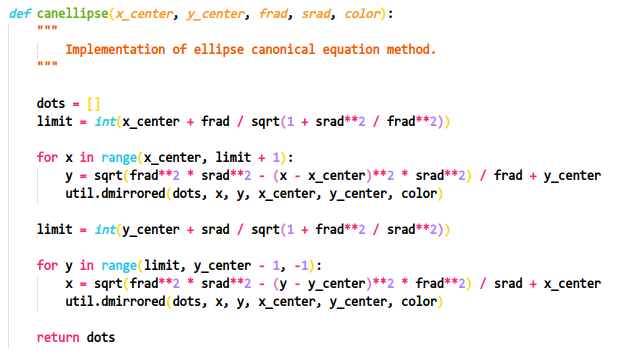
Алгоритм средней точки:



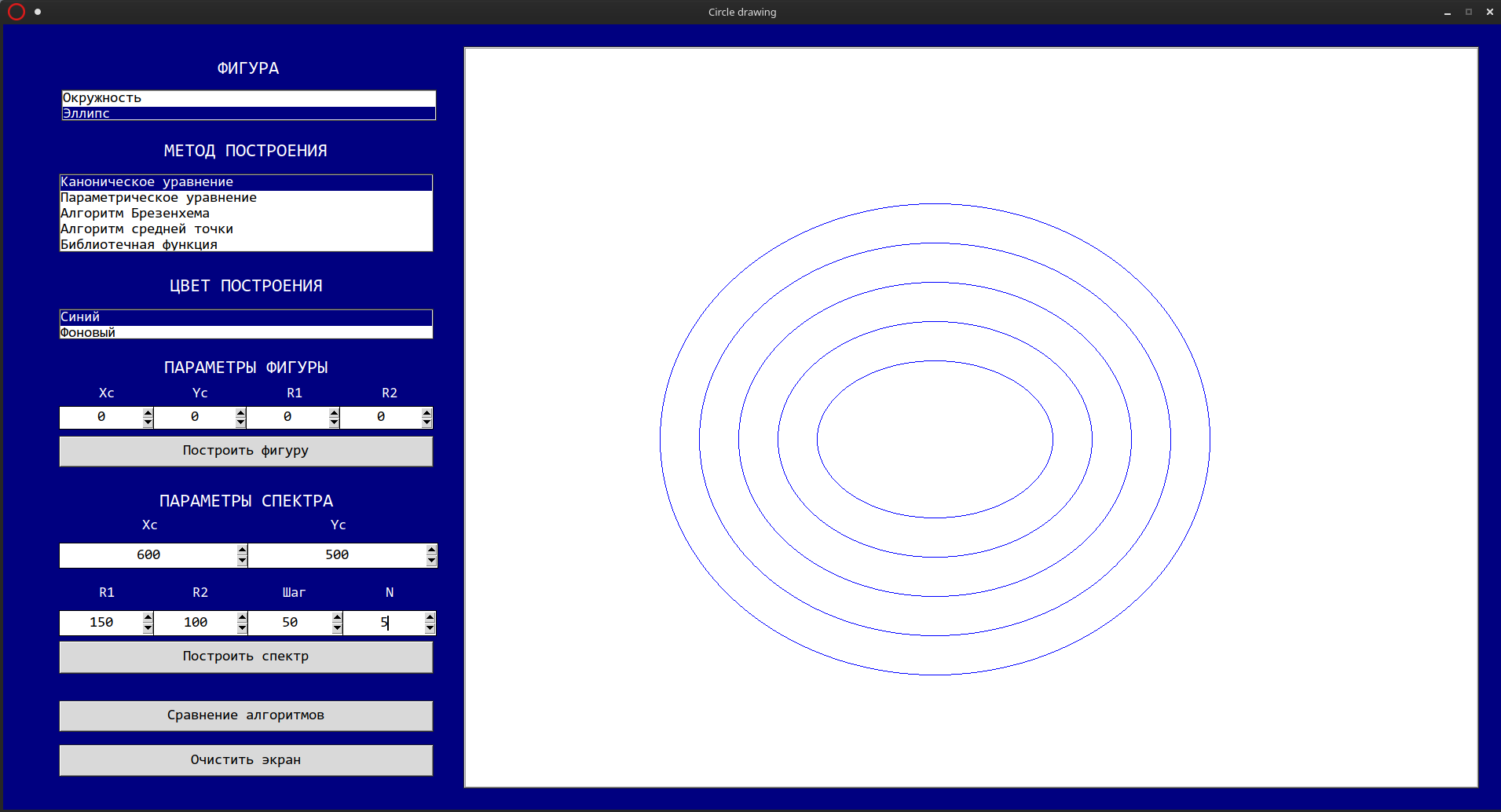
Судя по наложению можно сделать вывод, что библиотечная функция вероятнее всего реализована на основе канонического или параметрического уравнения.

**4.2. Алгоритмы построения эллипсов**

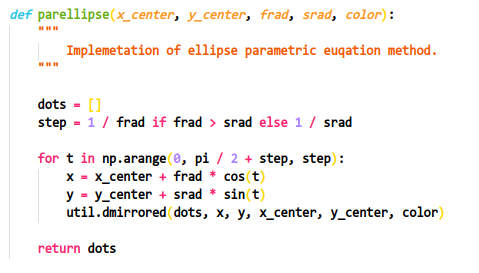
**4.2.1. Каноническое уравнение**

****

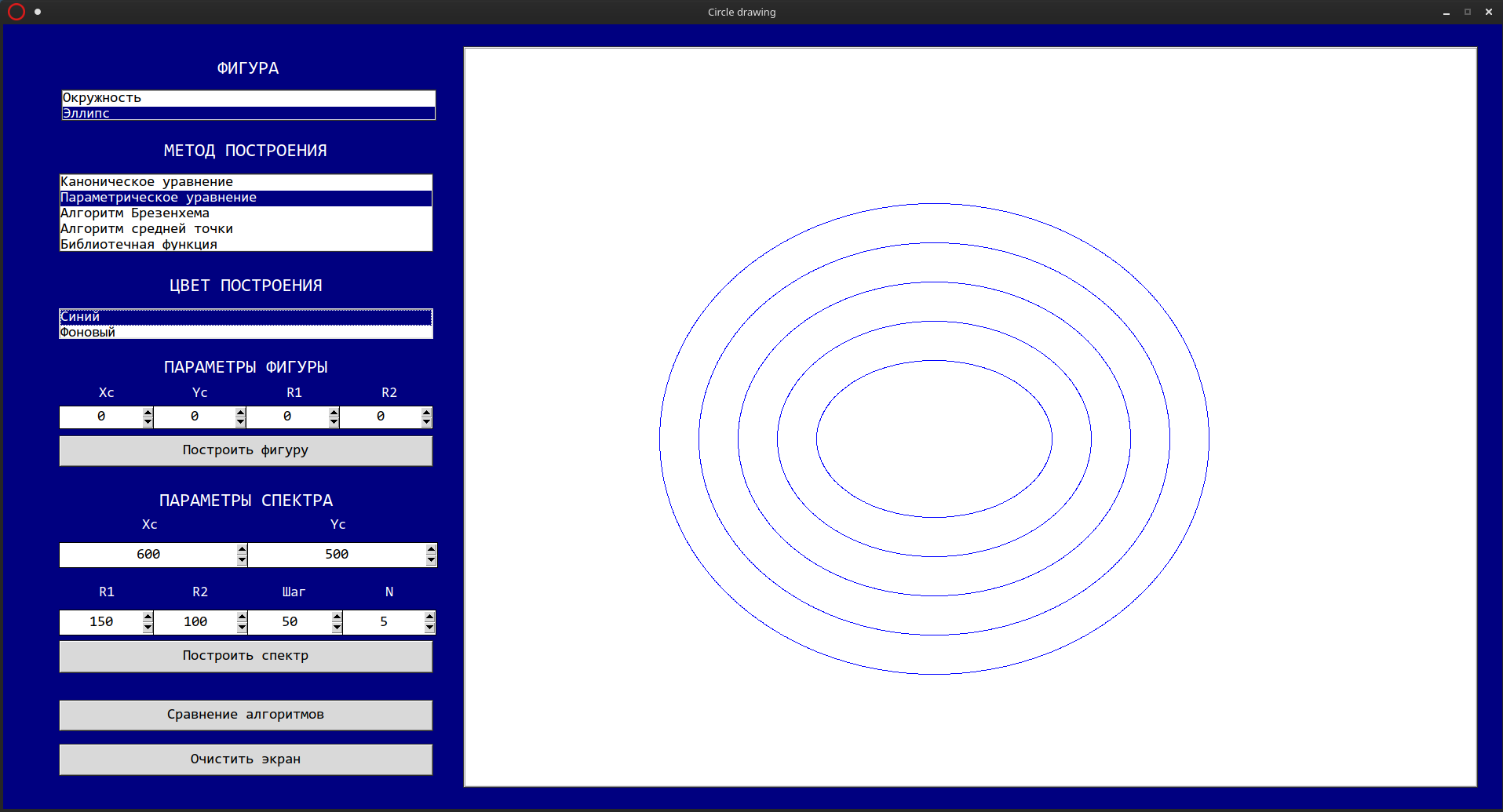
Алгоритм построения эллипса по каноническому уравнению практически аналогичен алгоритму для окружности, за важным исключением, что в данном случае мы строим ¼ часть эллипса, потому как эллипс не симметричен относительно биссектрисы. Кроме того, следует найти точку, где угол наклона касательной к эллипсу становится меньше 45о, так как в этой точке приращения **x** и **y** меняют свою динамику: координата, имевшая большее приращение, теперь получает меньшее и наоборот.



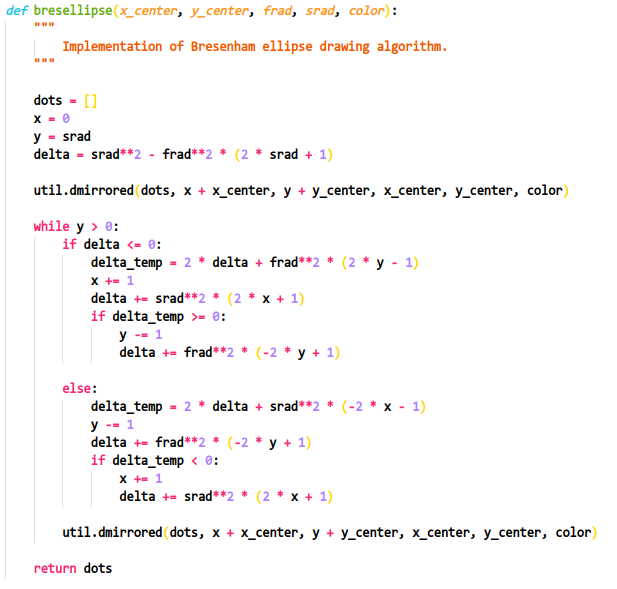
**4.2.2. Параметрическое уравнение**



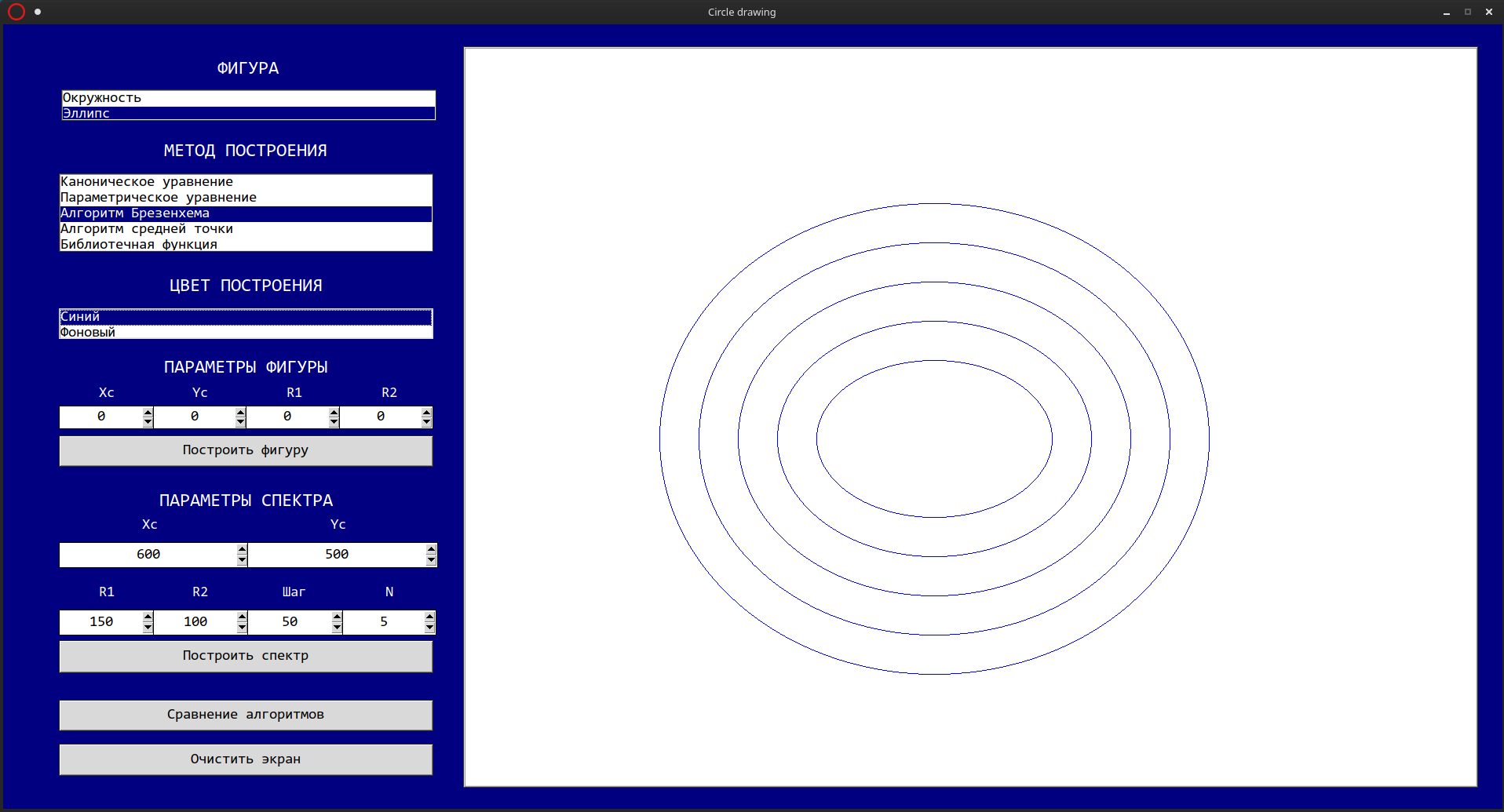
Как и в случае с каноническим уравнением, в данном случае мы строим ¼ часть эллипса, затем производим отражение. В качестве параметра **t** так же выбираем величину **1/R**, где **R** — радиус больше полуоси. Если брать радиус меньшей полуоси, то может произойти так, что часть точек у эллипса будет потеряна (появятся просветы при рисовании).



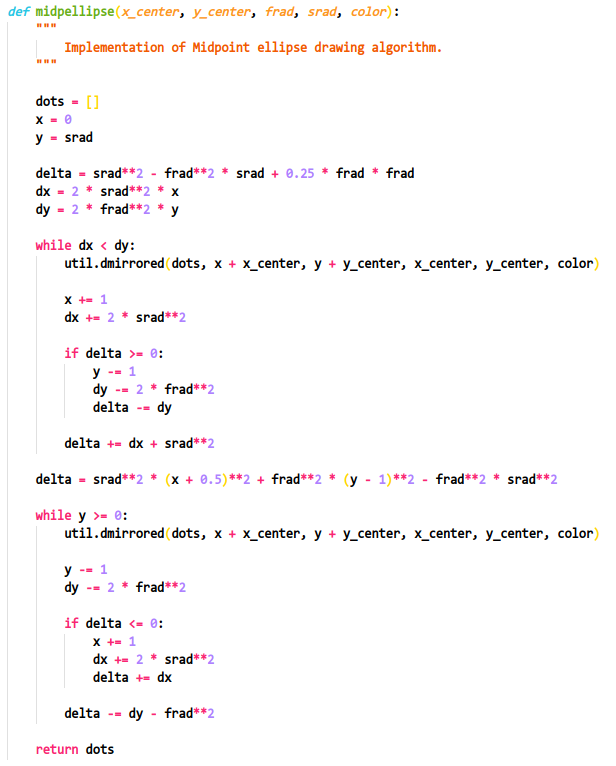
**4.2.3. Алгоритм Брезенхема**



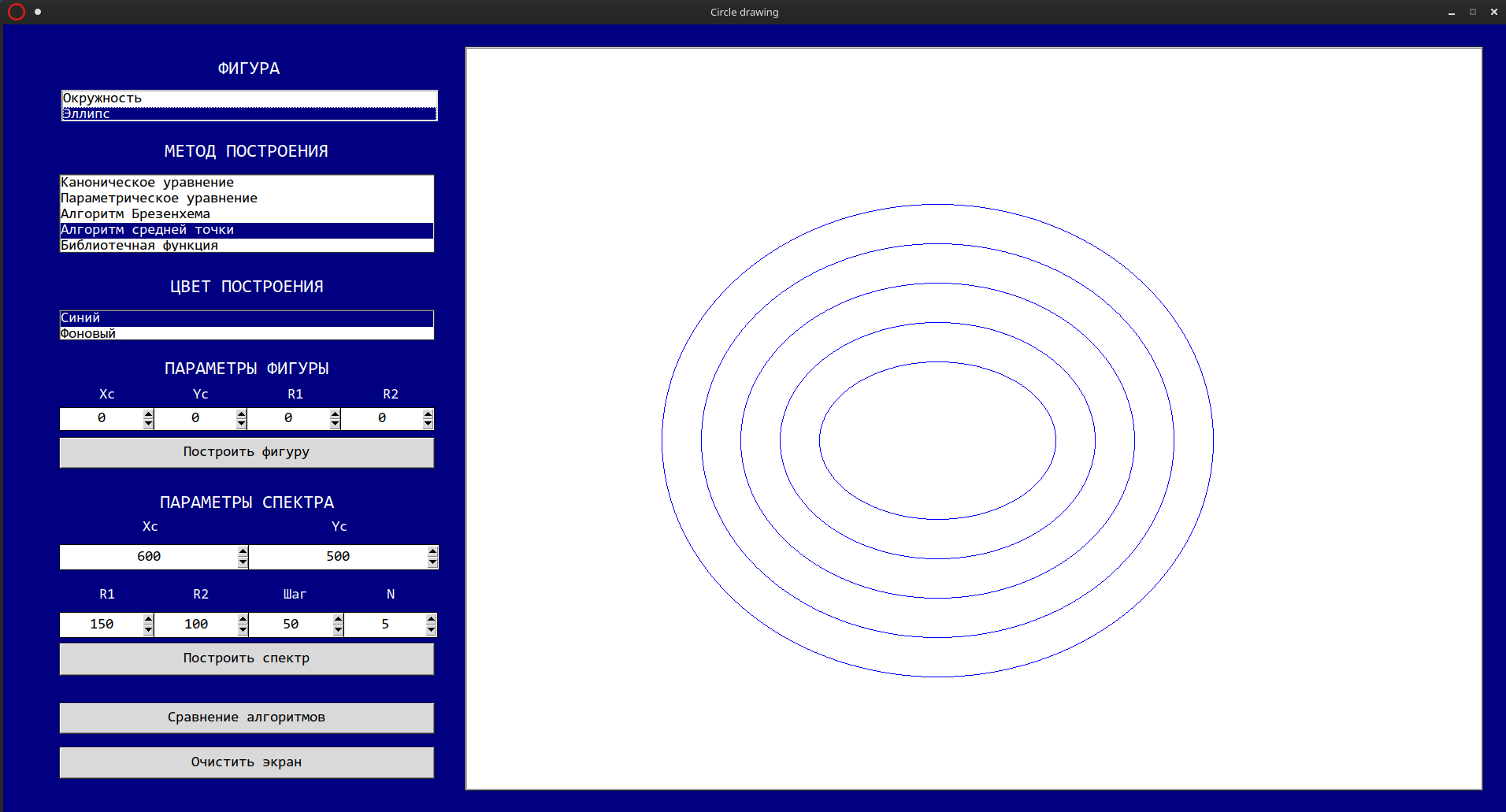
Данный алгоритм представляет собой практически то же самое, что и алгоритм для окружности, за исключением того, что здесь используется каноническое уравнение эллипса и построение происходит ¼ части эллипса.



**4.2.4. Алгоритм средней точки**



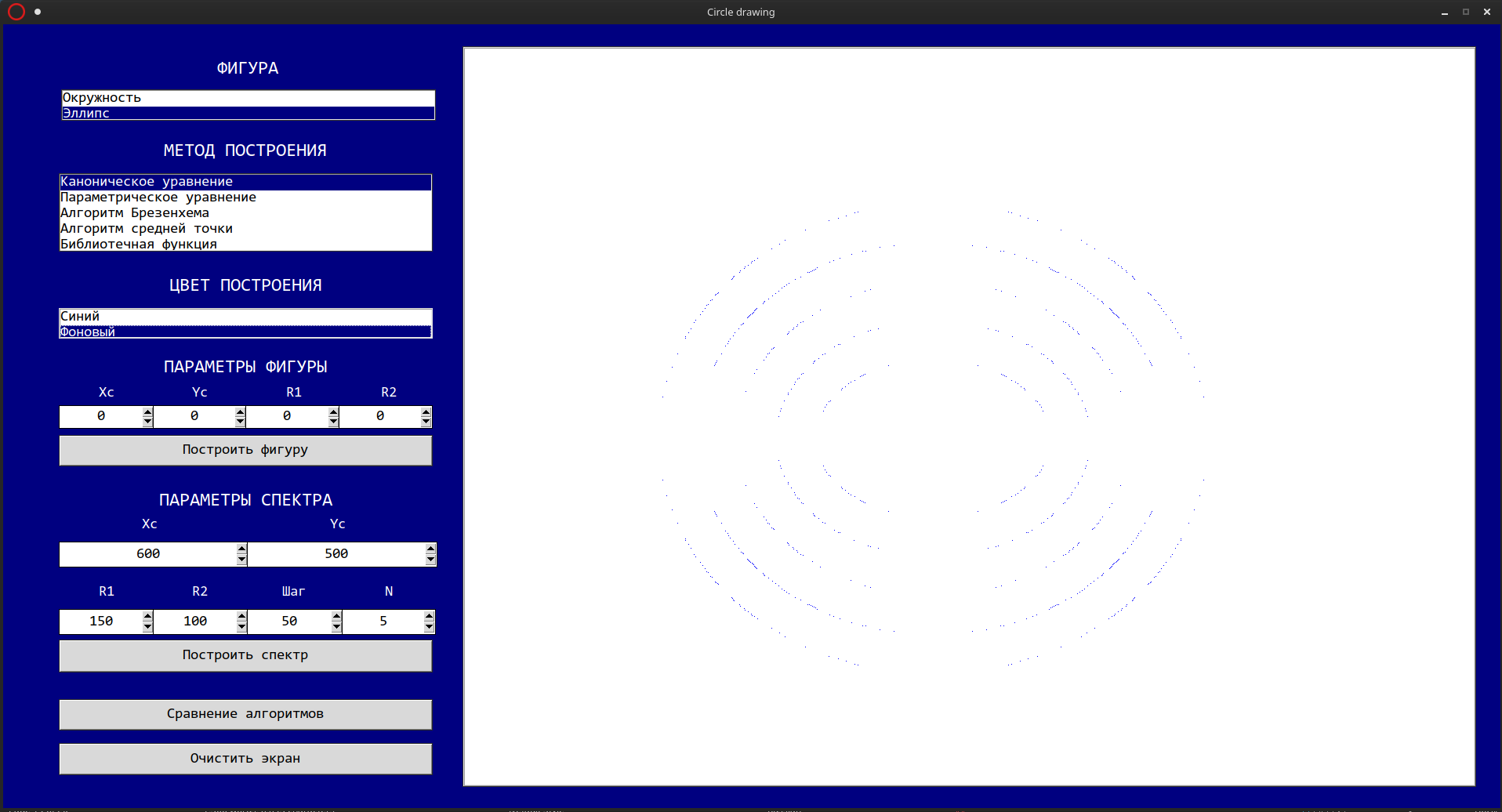
Данный алгоритм представляет собой практически то же самое, что и алгоритм для окружности, за исключением того, что здесь используется каноническое уравнение эллипса и построение происходит ¼ части эллипса. Так же не стоит забывать о том, что необходимо найти точку, в которой меняется динамика приращений координат.



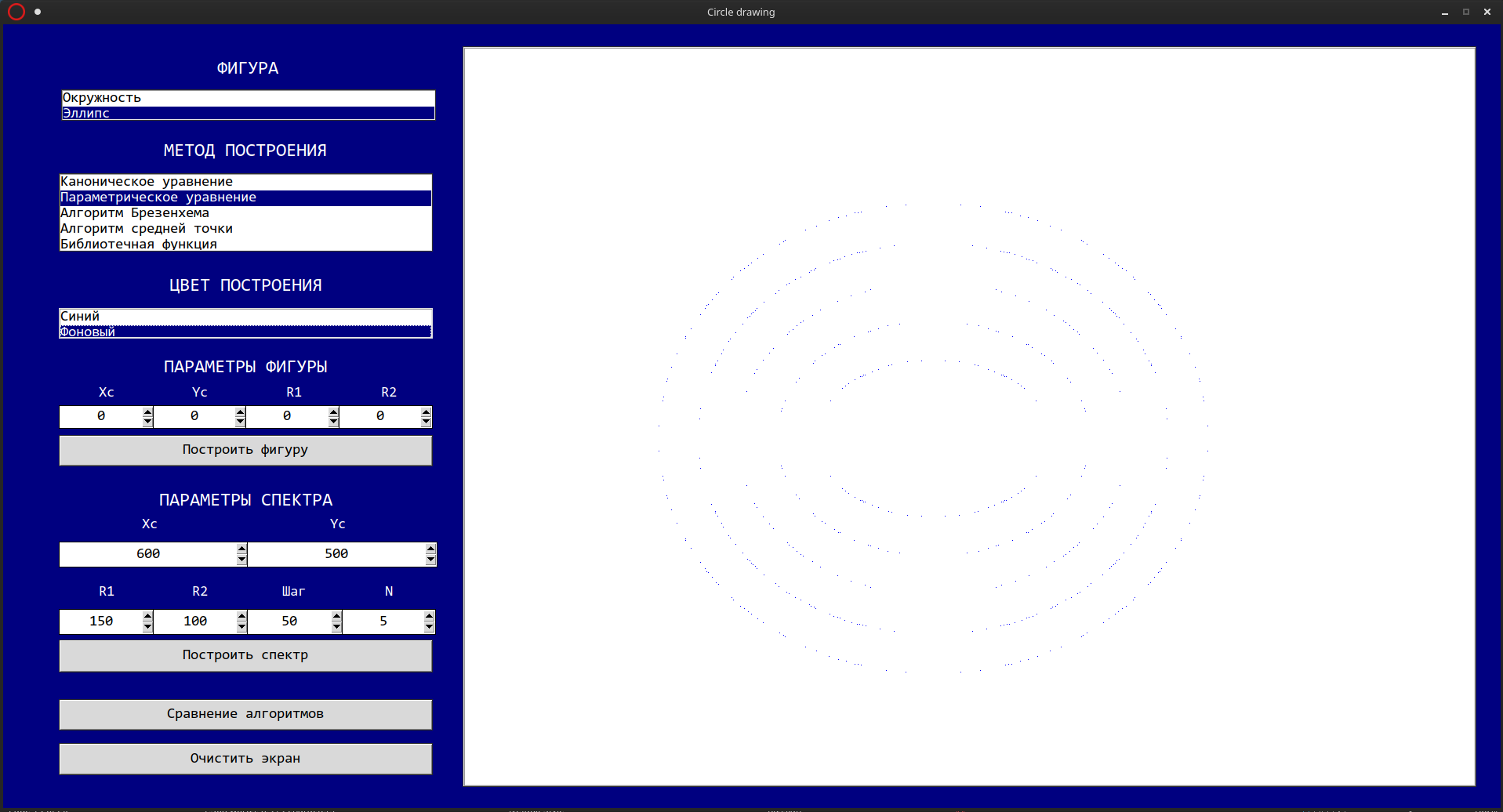
**4.2.5. Библиотечная функция**

Библиотечная функция в библиотеке, которую я использовал, отвечает сразу и за рисование окружностей, и за рисование эллипсов (различие только в передаваемых параметров). Рассмотрим корректность работы представленных алгоритмов.

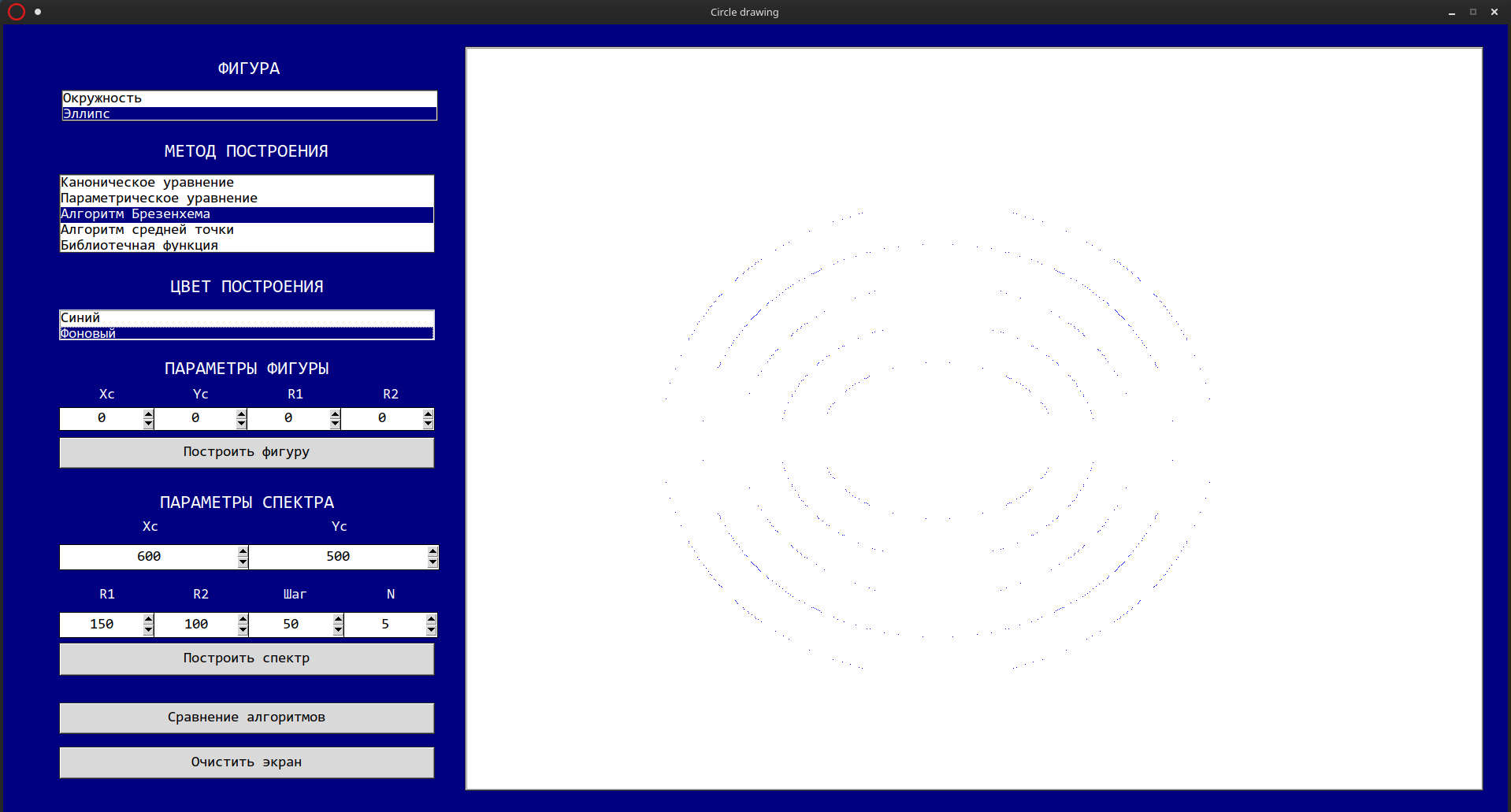
Каноническое уравнение:



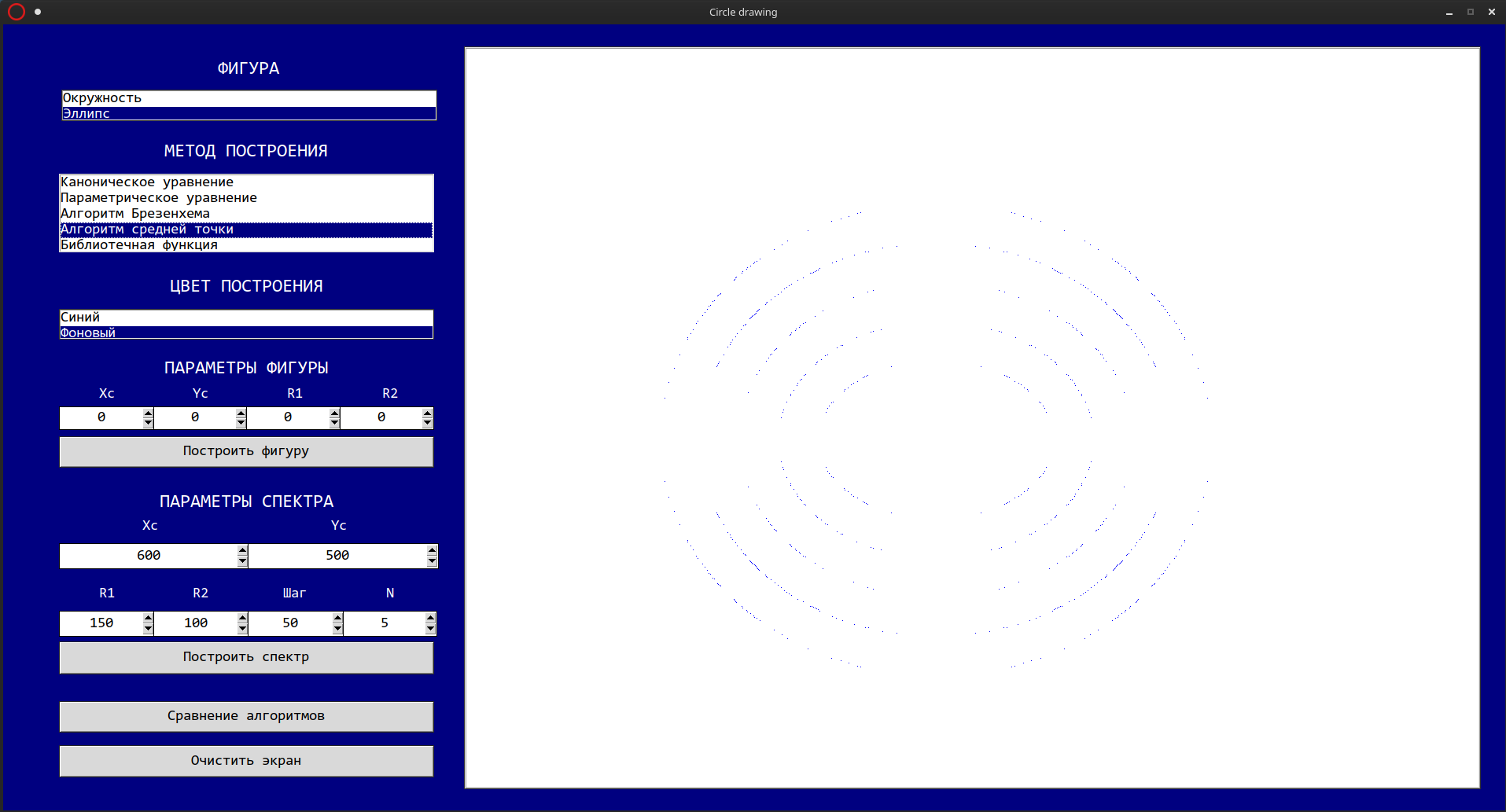
Параметрическое уравнение:



Алгоритм Брезенхема:



Алгоритм средней точки:

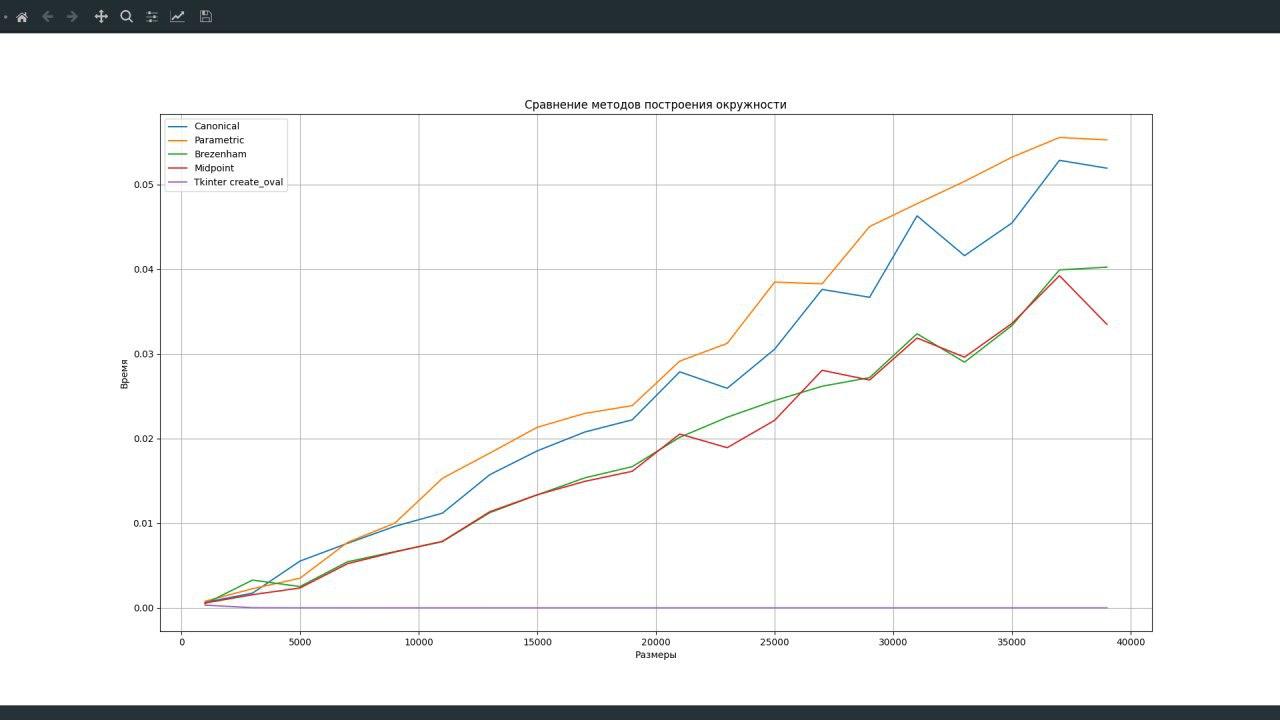


Выводы, сделанные для библиотечной функции при рисовании окружностей, подтвердились.

**5. Сравнение временных характеристик**

Во время измерений для каждого метода 10000 раз строился пучок с параметрами, использованными для демонстрации рабты алгоритмов, приведенных выше. Затем находилось среднее время, затраченное на выполнение повторений. Найденное время и было использовано в качестве демонстрационных данных.

Сравнение временых характеристик для алгоритмов построения окружности:



Сравнение временных характеристик для алгоритмов построения эллипса:

