Государственное учреждение образования

“БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ”

Кафедра: Интеллектуальных информационных технологий

Дисциплина: Графический интерфейс интеллектуальных систем

Отчет по лабораторной работе №2

**“Алгоритмы построения линий второго порядка”**

Выполнили:

студент гр.121702

Яхья-заде А.

Витковская С. И.

Проверил:

Сальников Д.А.

Минск 2024

**1. Цель работы**

Разработать элементарный графический редактор, реализующий построение линий второго порядка: окружность, эллипс, гипербола, парабола. Выбор кривой задается из пункта меню и доступен через панель инструментов «Линии второго порядка». В редакторе кроме режима генерации отрезков в пользовательском окне должен быть предусмотрен отладочный режим, где отображается пошаговое решение на дискретной сетке.

**2. Ход работы**

1. Реализация алгоритма Брезенхема для генерации окружностей.
2. Реализация алгоритма Брезенхема для генерации эллипсов.
3. Реализация алгоритма построения параболы.
4. Реализация алгоритма построения гиперболы.
5. Реализация режима отладки.

**3. Реализация**

**3.1. Алгоритм Брезенхема для генерации окружностей**

Функция drawCircle(x0, y0, radius, color) принимает четыре параметра: (x0, y0) являются координатами центра окружности, radius - значени радиуса окружности, вычисляемое по теореме Пифагора, color - цвет, выбранный пользователем.

1. Цикл while начинается с точки (radius, 0) и двигаясь внутрь, пока x не станет меньше или равным y. В каждой итерации цикла вызывается функция plotPoints, которая отображает точки на окружности. Эта функция рисует восьмую часть окружности, используя симметрию для отображения остальных секторов.
2. Затем проверяется условие err <= 0. Если оно истинно, y увеличивается на 1, а err увеличивается на 2 \* y + 1. Это перемещает точку вдоль окружности. Если условие err > 0 истинно, x уменьшается на 1, а err уменьшается на 2 \* x + 1. Это также перемещает точку вдоль окружности, но в другом направлении. Этот процесс продолжается, пока x не станет меньше или равным y, что означает, что окружность была полностью нарисована. Функция drawPixel(x0, y0, radius, color) используется для рисования точек на экране в заданном цвете.

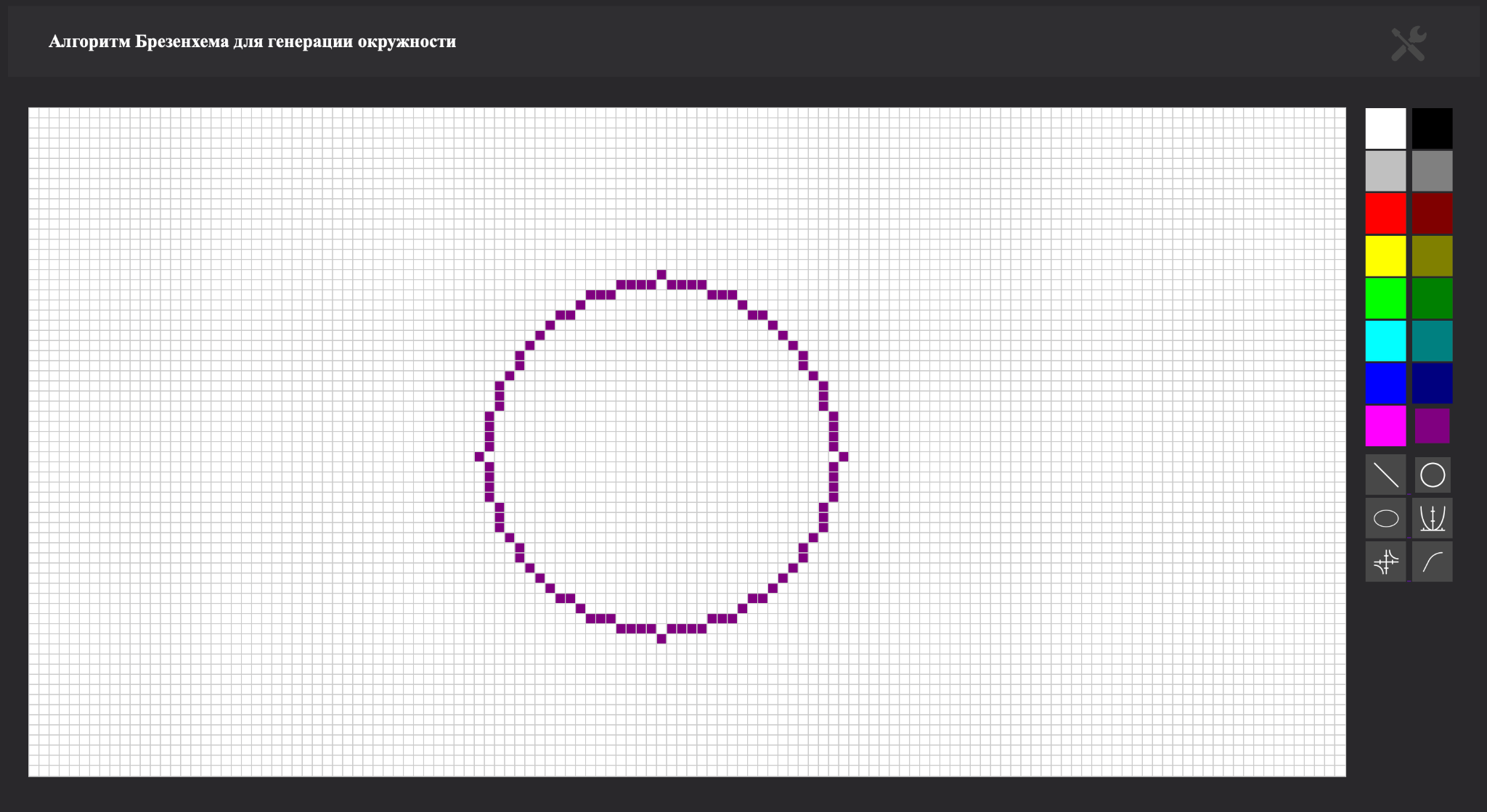
****

Рисунок 3.1 - Пример работы алгоритма генерации окружности

**3.2. Алгоритм Брезенхема для генерации эллипсов**

Алгоритм Брезенхема для генерации эллипса является модификацией алгоритма для генерации окружности. Данный алгоритм позволяет строить эллипсы на основании координат центра фигуры и длин большей и меньшей полуосей.

1. Эллипс описывается уравнением , которое можно переписать в виде f(x, y) ≡ + – = 0.

В отличие от окружности, для которой было достаточно построить одну восьмую ее часть, эллипс имеет только две оси симметрии, поэтому нужно построить четверть фигуры.

1. За основу берется дуга, лежащая в первом квадранте координатной плоскости.
2. Вдоль всей дуги координата Y является монотонно убывающей, но в первой части дуги она убывает медленнее, чем растет X, а во второй быстрее. Поэтому в первой части дуги увеличивается значение X, а во второй – уменьшается Y.
3. При перемещении вдоль первого участка дуги, находясь в точке (x, y), будем увеличивать значение X на единицу, а затем вычислять значение ∆ = f(x+1, y-1/2). Если это значение меньше нуля, то дополнительная точка (x+1, y-1/2) находится внутри эллипса, следовательно, ближайшая точка растра есть (x+1, y), в противном случае это точка (x+1, y-1).
4. На втором участке дуги возможен переход либо по вертикали, либо по диагонали, поэтому здесь сначала уменьшаем значение Y, затем вычисляем ∆ = f(x+1/2, y-1), и направление перехода выбирается аналогично предыдущему случаю.
5. Все оставшиеся дуги эллипса строятся параллельно: после получения очередной точки (x, y), можно инициализировать ещё три точки с координатами (-x, y), (x, -y), (-x, -y).

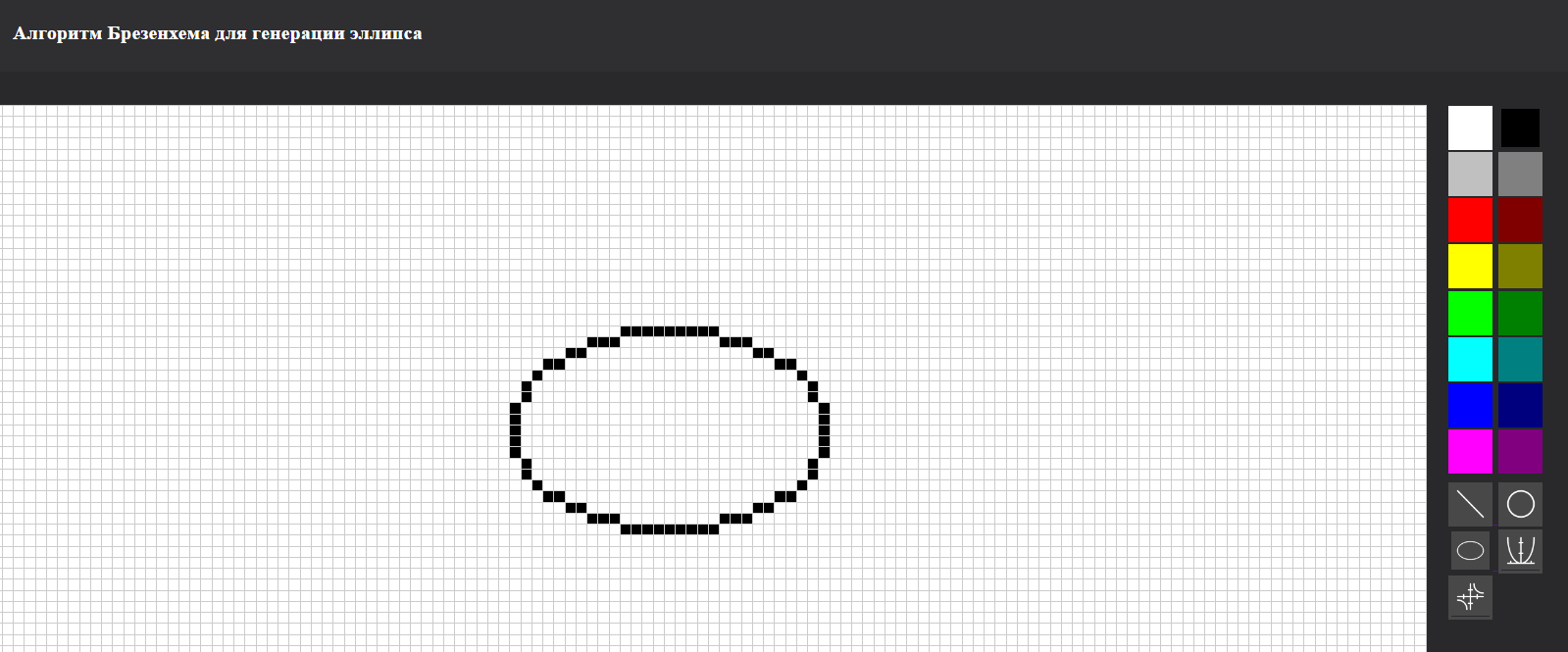


Рисунок 3.2 - Пример работы алгоритма генерации эллипса

**3.3. Алгоритм построения параболы**

Уравнение параболы имеет вид .

1. В зависимости от значения рисуется первая ветвь параболы в первом(если > 0) или в четвертом(если < 0), квадранте.

Так как график параболы имеет одну ось симметрии, достаточно построить лишь одну из ветвей и зеркально отразить ее относительно оси Oy.

1. Центром параболы являются координаты x\_offset и y\_offset, передаваемые функции parabola(x\_offset, y\_offset, p, bound, round, color) в качестве аргументов. Параметр p определяет ширину параболы, а bound - рамки относительно оси Ox, в пределах которых будет строиться парабола. round - переменная, ведущая счетчик ветвей параболы.
2. В начале каждой итерации цикла x и y обозначают текущую точку на параболе. Проверяется значение дискриминанта d. Если d >= 0, то мы двигаемся вправо и вверх по параболе. В этом случае переменные x и y увеличиваются на 1 (x++, y++). Если d < 0, то мы двигаемся только вправо. В этом случае переменная x увеличивается на 1 (x++), а значение дискриминанта d пересчитывается в соответствии с текущими значениями x и y.
3. Затем, в функции drawPixel(x\_offset, y\_offset, y, x, color) используются значения x и y, чтобы определить местоположение точки относительно центра (x\_offset, y\_offset), и эта точка отрисовывается. Этот процесс продолжается, пока x не достигнет ограничивающей координаты bound.

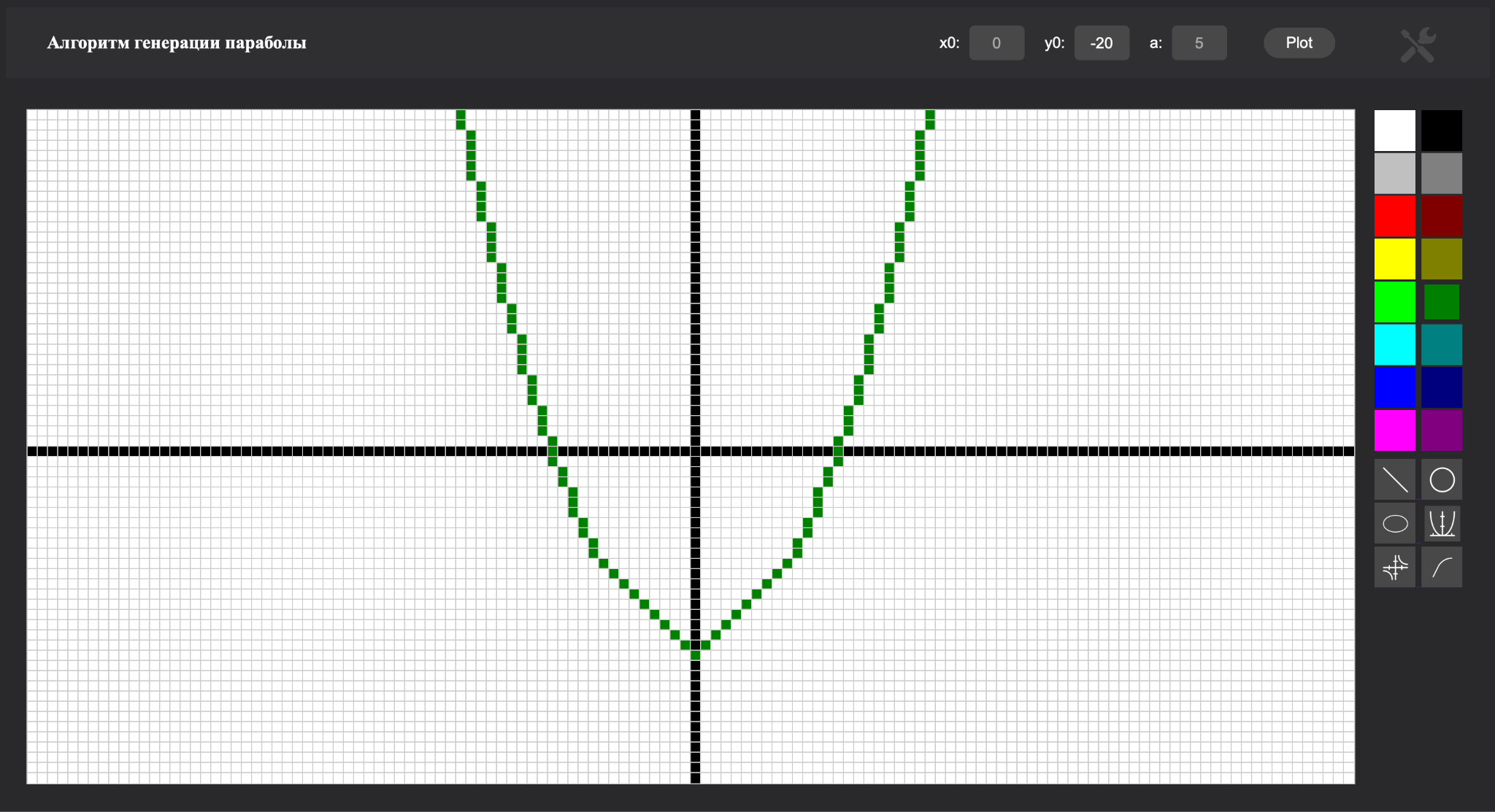
****

Рисунок 3.3 - Пример работы алгоритма генерации параболы

**3.4. Алгоритм построения гиперболы**

Уравнение гиперболы имеет вид .

1. Аналогично алгоритму построения эллипса, график гиперболы, имеет две оси симметрии, соответственно, достаточно построить лишь половину одной из ветвей, а остальные части достроить симметрично.
2. За основу берется четверть первого квадранта координатной плоскости. Она также состоит из двух частей: на первой итерация ведется путем увеличения y, на второй – x.
3. На каждой итерации цикла происходит проверка на то, находится ли точка внутри гиперболы. Если точка не находится внутри гиперболы, она обновляет наклон гиперболы. Этот процесс продолжается до тех пор, пока не будут нанесены все точки гиперболы.

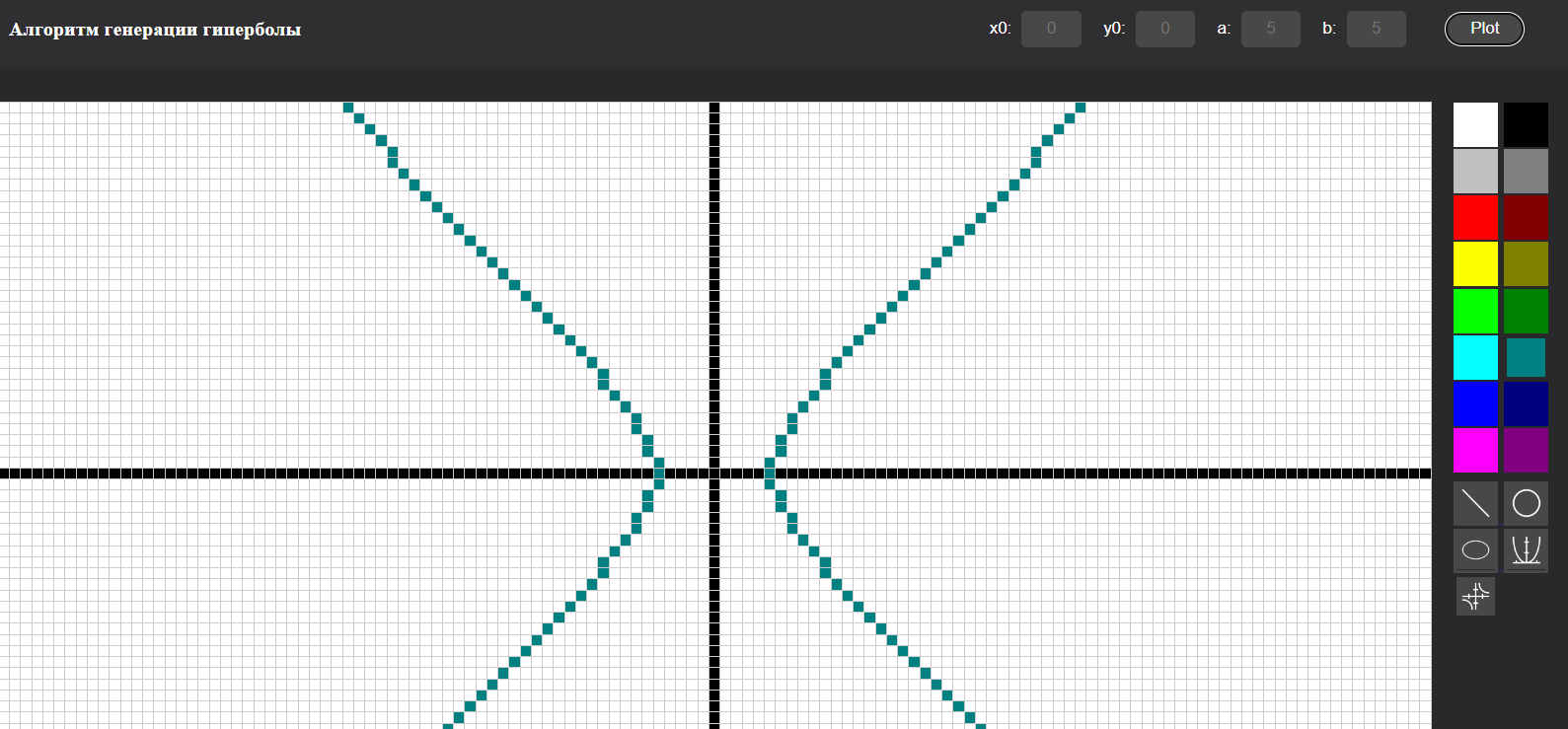


Рисунок 3.4 - Пример работы алгоритма генерации эллипса

**3.5. Отладочный режим**

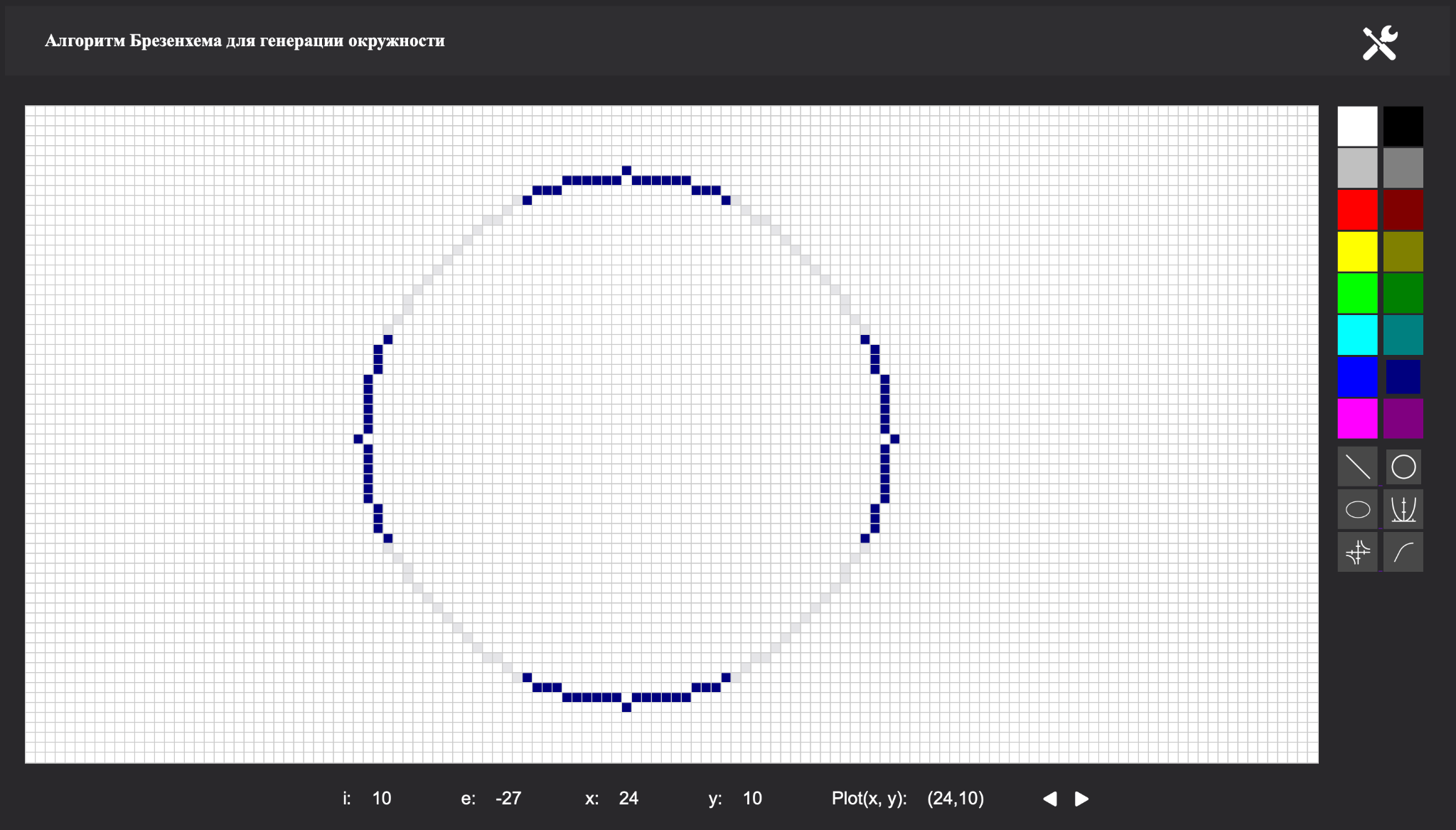


Рисунок 3.5 - Пример работы режима отладки для алгоритма генерации окружности

**Вывод:**

В ходе лабораторной работы были реализованы алгоритмы, генерирующие все виды конического сечения: окружность, эллипс, параболу и гиперболу. Также был реализован отладочный режим, отображающий пошаговое решение на дискретной сетке.