1. Реализованный функционал

Веб-приложение реализует визуализацию работы четырёх базовых алгоритмов растеризации:

1. Растеризация отрезка по пошаговому алгоритму (на основе уравнения прямой).
2. Растеризация отрезка методом ЦДА (Digital Differential Analyzer).
3. Растеризация отрезка по алгоритму Брезенхема.
4. Растеризация окружности по алгоритму Брезенхема.

В логическом (декартовом) пространстве задаются:

- для отрезка — координаты начальной и конечной точек;
- для окружности — координаты центра и целочисленный радиус.

Для каждого набора входных данных автоматически выбирается диапазон по осям, на основе которого формируется прямоугольная сетка с целочисленными логическими координатами. Масштаб сетки и положение осей подстраиваются под введённые значения так, чтобы:

- начало координат (0, 0) попадало в видимую часть;
- оси и линии сетки проходили строго по целочисленным делениям.

На основе выбранного алгоритма вычисляется набор целочисленных точек (пикселей), которые визуально приближают идеальный геометрический объект (отрезок или окружность). Время вычисления соответствующего алгоритма выводится в миллисекундах. Реализована очистка рисунка, при которой сетка и оси остаются неизменными, а ранее отрисованные пиксели удаляются.

---

2. Растеризация отрезка: пошаговый алгоритм

Проблема растеризации отрезка заключается в том, что идеальный отрезок задан в непрерывном пространстве, а результат необходимо представить в виде набора дискретных пикселей с целочисленными координатами. Пошаговый алгоритм опирается на аналитическое уравнение прямой.

1. Сначала по двум заданным точкам (x_0, y_0) и (x_1, y_1) вычисляются коэффициенты наклона и сдвига:
   - $k = \dfrac{y_1 - y_0}{x_1 - x_0}$ — наклон;
   - $b = y_0 - kx_0$ — свободный член.
2. Далее выбирается «основная» координата, по которой удобно идти с шагом 1.
   На практике обычно предполагается, что $|x_1 - x_0| \geq |y_1 - y_0|$ и шаг делается по оси x.
3. Для каждого целого значения x на отрезке $[x_0, x_1]$ вычисляется вещественное значение:

$$y = kx + b$$

4. Полученное y округляется до ближайшего целого и полученная целочисленная пара $(x, \mathrm{round}(y))$ добавляется в набор пикселей.

Плюсы:

- простая и прозрачная вычислительная логика;

- легко связывается с аналитической геометрией и уравнением прямой.

Минусы:

- на каждом шаге выполняются операции с вещественными числами;
- происходит повторный пересчёт выражения $kx + b$, что неэффективно при большой длине отрезка;
- накопление ошибок округления может приводить к местам с «рваными» участками.

---

3. Растеризация отрезка методом ЦДА (DDA)

Алгоритм ЦДА также основан на представлении отрезка как функции в непрерывном пространстве, но формулируется в инкрементной форме. Вместо повторного вычисления формулы для каждого пикселя, он использует постоянные приращения координат.

1. Вычисляются:
   - $dx = x_1 - x_0$;
   - $dy = y_1 - y_0$.
2. Определяется число шагов:

   $$\text{steps} = \max(|dx|, |dy|)$$

   Это гарантирует, что на каждом шаге изменение по одной из координат не превышает 1 по модулю.
3. Вычисляются приращения:

   $$x_{\text{step}} = \frac{dx}{\text{steps}}, \quad y_{\text{step}} = \frac{dy}{\text{steps}}.$$
4. Начальная точка инициализируется как $x = x_0, y = y_0$.
5. Далее повторяется цикл из $\text{steps} + 1$ итераций:
   - текущие x, y округляются до ближайших целых;
   - соответствующий пиксель добавляется в результат;
   - к x и y прибавляются соответственно x_{step} и y_{step}.

Плюсы:

- вычисления организованы в виде простого инкрементного процесса;
- нет повторного использования уравнения прямой, только сложения;
- легко реализуется и хорошо иллюстрирует идею дискретизации.

Минусы:

- по-прежнему использует вещественные числа, что замедляет выполнение и может приводить к накоплению ошибок округления;
- не такой быстрый, как полностью целочисленные алгоритмы на той же аппаратной платформе.

---

4. Растеризация отрезка по алгоритму Брезенхема

Алгоритм Брезенхема для отрезка строится как целочисленная аппроксимация идеальной прямой. В отличие от предыдущих методов, он избегает вещественных операций и формулируется через «решающую» величину (decision variable), которая измеряет отклонение текущей дискретной линии от идеальной.

Для простоты можно рассмотреть случай, когда $|x_1 - x_0| \geq |y_1 - y_0|$, и отрезок идёт слева направо.

1. Вычисляются:
   - $dx = |x_1 - x_0|$;
   - $dy = |y_1 - y_0|$.
2. Определяются знаки изменения по осям:
   - $sx = \text{sign}(x_1 - x_0)$;
   - $sy = \text{sign}(y_1 - y_0)$.
3. Вводится целочисленная «ошибка»:

   $$\text{err} = dx - dy$$

   Эта величина косвенно отражает положение текущего пикселя относительно идеальной прямой.
4. Начальная точка считается рисунком, после чего выполняется цикл, пока не будет достигнута конечная точка:
   - текущий пиксель (x, y) заносится в результат;
   - вычисляется вспомогательное значение $e2 = 2 \cdot \text{err}$;
   - если $e2 > -dy$, то:
     - $\text{err} = \text{err} - dy$;
     - $x = x + sx$;
   - если $e2 < dx$, то:
     - $\text{err} = \text{err} + dx$;
     - $y = y + sy$.

Таким образом, на каждом шаге алгоритм принимает решение, нужно ли сместиться по вертикали (изменить y), исходя из накопленной ошибки. Все вычисления выполняются в целых числах.

Плюсы:

- операции только с целыми числами: сложение, вычитание, сравнения;
- высокая производительность и предсказуемость;
- качество аппроксимации отрезка близко к идеальной прямой на растровой сетке.

Минусы:

- математически и алгоритмически сложнее для понимания, чем простые методы на основе уравнения прямой;
- для обработки всех возможных направлений (разные октанты) требуется аккуратная работа с знаками и модулями.

---

5. Растеризация окружности по алгоритму Брезенхема

Алгоритм Брезенхема для окружности решает задачу аппроксимации идеальной окружности с уравнением:

$$x^2 + y^2 = r^2$$

на целочисленной сетке. Ключевую роль играет использование симметрии окружности и целочисленной решающей функции.

1. Рассматривается только один сегмент окружности (обычно первый октант: $x \geq 0$, $y \geq 0$, $x \leq y$).
   Оставшиеся семь сегментов восстанавливаются за счёт симметрий относительно осей и диагонали.
2. Начальные значения:
   - $x = 0$;
   - $y = r$;
   - решающая функция (ошибка) $d = 3 - 2r$.
3. На каждом шаге координаты (x, y) используются для генерации восьми симметричных точек относительно центра окружности (x_c, y_c):
   - $(x_c \pm x, y_c \pm y)$;
   - $(x_c \pm y, y_c \pm x)$.
4. Далее выбирается направление шага:
   - если $d \leq 0$, окружность ещё «слишком далеко» от решётки; выполняется смещение по x:
     $d = d + 4x + 6$;
   - если $d > 0$, необходимо одновременно уменьшить y и увеличить x:
     $d = d + 4(x - y) + 10$;
     $y = y - 1$.
5. После каждой корректировки d выполняется $x = x + 1$.
   Цикл продолжается, пока $y \geq x$.

Плюсы:

- использование восьмикратной симметрии существенно снижает количество вычислений;
- алгоритм полностью целочисленный, быстрый и хорошо подходит для графических систем низкого уровня;
- полученная окружность визуально близка к идеальному контуру.

Минусы:

- логика заметно сложнее по сравнению с простым параметрическим заданием окружности;
- для понимания требуется интерпретация решающей функции и переход от непрерывного уравнения к целочисленной форме.