

## Лабораторная работа №4

### Алгоритмы растеризации отрезков и окружностей

Лабораторная работа №4 посвящена изучению и практическому применению базовых алгоритмов растеризации.

**Цель работы:** закрепление теоретического материала и практическое освоение основных возможностей по использованию базовых алгоритмов растеризации отрезков и кривых. Работа направлена на изучение фундаментальных методов растровой графики, которые лежат в основе построения изображений в компьютерных системах.

**Задача:** разработать веб-приложение, иллюстрирующее работу базовых растровых алгоритмов, включающее реализацию четырех основных алгоритмов и двух дополнительных для получения дополнительных баллов. Приложение должно предоставлять визуализацию работы алгоритмов на координатной сетке с возможностью интерактивного управления параметрами.

### Алгоритмы растеризации

*Растеризация* — это процесс преобразования математического описания графических примитивов (отрезков, окружностей, полигонов) в растровое изображение, состоящее из дискретных пикселей. Эта задача является фундаментальной в компьютерной графике, поскольку все визуальные объекты на экране в конечном итоге представлены в виде массива пикселей.

*Проблемы растеризации:*

- Дискретность пространства. Непрерывные математические объекты должны быть представлены в дискретной пиксельной сетке;
- Алиасинг (ступенчатость). Наклонные линии выглядят как "лесенка" из-за ограниченного разрешения;
- Эффективность вычислений. Необходимость быстрого вычисления координат пикселей
- Точность. Минимизация отклонения от идеальной геометрической формы

*Виды алгоритмов растеризации:*

*По типу примитивов:* алгоритмы для отрезков, алгоритмы для окружностей и эллипсов, алгоритмы для полигонов, алгоритмы для кривых Безье и сплайнов;

*По принципу работы:* прямые методы (использующие уравнение примитива), инкрементальные методы (с пошаговым вычислением), целочисленные методы (без операций с плавающей точкой), антиалиасинговые методы (со сглаживанием);

*По историческому развитию:* ранние простые алгоритмы (пошаговый, DDA), оптимизированные алгоритмы (Брезенхема), алгоритмы с улучшенным качеством (B<sub>y</sub>), современные гибридные алгоритмы (Кастла-Питвея).

### **Пошаговый алгоритм**

Алгоритм основан на прямом использовании уравнения прямой  $y = kx + b$ . Для отрезка с координатами  $(x_1, y_1)$  и  $(x_2, y_2)$  вычисляется угловой коэффициент  $k = \Delta y / \Delta x$  и свободный член  $b = y_1 - k \cdot x_1$ . Затем, начиная от  $x_1$  до  $x_2$  с шагом 1, вычисляются соответствующие значения  $y$ , которые округляются до ближайшего целого для определения позиции пикселя.

*Особенности реализации:*

- Для почти вертикальных линий ( $|\Delta y| > |\Delta x|$ ) алгоритм меняет роли  $x$  и  $y$
- Использует вещественную арифметику с операциями деления и умножения
- Требуется отдельной обработки вертикальных линий ( $\Delta x = 0$ )

*Преимущества:*

- Простота понимания и реализации
- Прямое следование математическому определению прямой
- Хорошо подходит для образовательных целей

*Недостатки:*

- Использование операций с плавающей точкой замедляет выполнение
- Накопление ошибок округления при больших  $\Delta x$
- Неэффективен для встраиваемых систем без FPU

*Области применения:*

- Обучение основам компьютерной графики
- Прототипирование графических алгоритмов
- Системы, где простота важнее производительности

### **Алгоритм ЦДА (Цифровой дифференциальный анализатор, DDA)**

DDA является инкрементальным улучшением пошагового алгоритма. Вместо вычисления  $y$  для каждого  $x$  через умножение, алгоритм вычисляет приращения и накапливает их. Определяется количество шагов как  $\max(|\Delta x|, |\Delta y|)$ , вычисляются инкременты  $dx = \Delta x / \text{steps}$  и  $dy = \Delta y / \text{steps}$ , затем на каждом шаге текущие координаты увеличиваются на эти инкременты.

*Особенности реализации:*

- Автоматически обрабатывает все направления линий
- Всегда делает ровно  $\text{steps}$  шагов, где  $\text{steps} = \max(|\Delta x|, |\Delta y|)$
- Требуется одной операции деления для вычисления инкрементов

*Преимущества:*

- Быстрее пошагового алгоритма за счет замены умножения на сложение
- Более равномерное распределение пикселей
- Меньше ошибок округления по сравнению с пошаговым алгоритмом

*Недостатки:*

- Все еще использует вещественную арифметику
- Накопление ошибок при большом количестве шагов
- Требуется операций округления на каждом шаге

*Области применения:*

- Системы с поддержкой аппаратных операций с плавающей точкой
- Приложения, где требуется баланс между простотой и производительностью
- Как промежуточный этап при изучении оптимизированных алгоритмов

### **Алгоритм Брезенхема для отрезков**

Алгоритм Джека Брезенхема (1965) — революционный метод, использующий только целочисленную арифметику. В основе алгоритма лежит управление ошибкой — величиной, характеризующей отклонение текущей позиции от идеальной прямой. На каждом шаге алгоритм решает, нужно ли изменять координату  $y$ , основываясь на накопленной ошибке. Ключевая инновация алгоритма: вместо вещественных вычислений алгоритм использует только целочисленные операции сложения, вычитания и умножения на 2 (что эквивалентно битовому сдвигу). Решение об изменении  $y$  принимается на основе знака переменной ошибки.

*Особенности реализации:*

- Алгоритм для всех октантов:
  1. Определяются знаки приращений  $sx$  и  $sy$  (+1 или -1)
  2. Используются абсолютные значения  $\Delta x$  и  $\Delta y$
  3. Основной цикл выполняется  $\max(\Delta x, \Delta y)$  раз
  4. В зависимости от того, что больше ( $\Delta x$  или  $\Delta y$ ), изменяется либо  $x$ , либо  $y$

*Преимущества:*

- Только целочисленные операции (очень быстрые)
- Не требует операций деления или умножения (кроме на 2)
- Высокая точность без накопления ошибок
- Стандарт де-факто в графических библиотеках

*Недостатки:*

- Более сложная логика по сравнению с DDA
- Требуется отдельной реализации для разных случаев ( $|\Delta x| > |\Delta y|$  и наоборот)
- Не обеспечивает сглаживания (антиалиасинга)

*Области применения:*

- Графические процессоры и аппаратное ускорение
- Игровые движки и системы реального времени
- Встроенные системы с ограниченными вычислительными ресурсами
- Все основные графические API (OpenGL, DirectX, Vulkan)

### **Алгоритм Брезенхема для окружностей**

Алгоритм Джека Брезенхема (1965) — революционный метод, использующий только целочисленную арифметику. В основе алгоритма лежит управление ошибкой — величиной, характеризующей отклонение текущей позиции от идеальной прямой. На каждом шаге алгоритм решает, нужно ли изменять координату  $y$ , основываясь на накопленной ошибке. Ключевая инновация алгоритма: вместо вещественных вычислений алгоритм использует только целочисленные операции сложения, вычитания и умножения на 2 (что эквивалентно битовому сдвигу). Решение об изменении  $y$  принимается на основе знака переменной ошибки.

*Особенности реализации:*

- Окружность имеет 8-лучевую симметрию:
  1. Отражение относительно осей:  $(x, y) \rightarrow (-x, y), (x, -y)$
  2. Отражение относительно диагоналей:  $(x, y) \rightarrow (y, x)$
  3. Это позволяет сократить вычисления в 8 раз

*Преимущества:*

- Только целочисленные операции
- Использование симметрии значительно ускоряет вычисления
- Равномерное распределение пикселей по окружности
- Отсутствие разрывов в контуре

*Недостатки:*

- Только для окружностей (не подходит для эллипсов)
- Требуется отдельного алгоритма для заполненных окружностей
- Не поддерживает сглаживание

*Области применения:*

- Рисование окружностей в графических интерфейсах
- Генерация круговых диаграмм и индикаторов
- Игровая графика (планеты, траектории, эффекты)
- Системы ЧПУ и графопостроители

## **Антиалиасинговые алгоритмы**

### **Алгоритм Брезенхема для окружностей**

Алгоритм Сяолиня Ву (1991) решает проблему алиасинга (ступенчатости) путем сглаживания. Вместо рисования одного пикселя на каждом шаге, алгоритм рисует два пикселя с интенсивностями, пропорциональными расстоянию до идеальной линии. Это создает иллюзию более гладкой линии.

*Основная идея:*

Для каждой позиции  $x$  вычисляется точное значение  $y_{ideal}$ . Алгоритм рисует два пикселя: основной пиксель с интенсивностью  $(1 - \text{дробная\_часть}(y_{ideal}))$ , соседний пиксель с интенсивностью  $\text{дробная\_часть}(y_{ideal})$

*Особенности реализации:*

- Обработка конечных точек требует особого внимания
- Для почти вертикальных линий меняются роли  $x$  и  $y$
- Интенсивность пикселей обычно квантуется в 256 уровней (8 бит на канал)

*Преимущества:*

- Высокое визуальное качество (сглаженные линии)
- Постепенное изменение толщины при изменении угла
- Сохранение общей яркости линии
- Меньше видимых артефактов при анимации

*Недостатки:*

- В 2 раза больше операций рисования пикселей
- Требуется вычислений с плавающей точкой
- Сложнее реализация по сравнению с Брезенхемом
- Может требовать поддержки альфа-канала или смешивания цветов

*Области применения:*

- Полиграфия и DTP-системы
- CAD/CAM системы с высокими требованиями к качеству
- Медицинская визуализация
- Профессиональные графические редакторы
- Интерфейсы с субпиксельным рендерингом

### **Алгоритм Кастла-Питвея**

Алгоритм Кастла-Питвея представляет собой гибридный подход, сочетающий эффективность Брезенхема с качеством Ву. Основная идея — использовать целочисленную логику Брезенхема для определения основного пикселя, а затем распределять интенсивность между основным и соседним пикселями на основе накопленной ошибки.

Алгоритм модифицирует стандартный алгоритм Брезенхема:

Вычисляется ошибка как в алгоритме Брезенхема, основной пиксель рисуется с интенсивностью, пропорциональной  $(1 - \text{norm})$ , соседний пиксель (выше или ниже) рисуется с интенсивностью, пропорциональной  $\text{norm}$  (где  $\text{norm}$  — нормализованное значение ошибки)

*На практике алгоритм Кастла-Питвея часто реализуется как модификация алгоритма Ву, где вместо точного вычисления расстояния до идеальной линии используется ошибка из алгоритма Брезенхема.*

*Преимущества:*

- Более высокая скорость, чем у чистого алгоритма Ву
- Лучшее качество, чем у чистого алгоритма Брезенхема
- Сохранение целочисленной арифметики для основной логики
- Постепенное перераспределение интенсивности создает плавные переходы

*Недостатки:*

- Все еще требует вычислений интенсивности для двух пикселей
- Не такое высокое качество, как у оптимизированного алгоритма Ву
- Более сложная реализация, чем у Брезенхема
- Может требовать дополнительной памяти для хранения интенсивностей

*Области применения:*

- Игры и интерактивные приложения, где важен баланс качества и производительности
- Мобильные устройства с ограниченными ресурсами
- Системы реального времени с умеренными требованиями к качеству
- Встроенные графические интерфейсы

### **Сравнение производительности и качества работы алгоритмов**

Все алгоритмы имеют временную сложность  $O(N)$ , поэтому основным фактором скорости работы алгоритмов является константный множитель.

Получим рейтинг алгоритмов по скорости:

1. Брезенхема (только целочисленные операции)
2. Кастла-Питвея (целочисленные + немного плавающих)
3. DDA (вещественные, но только сложение)
4. Пошаговый (вещественные с умножением)
5. Ву (вещественные + рисование двух пикселей)

По качеству растеризации:

1. Ву (качественное сглаживание)
2. Кастла-Питвея (умеренное сглаживание)
3. Брезенхема (четкие, но с алиасингом)

4. DDA (похоже на пошаговый, но равномернее)
5. Пошаговый (базовое, с алиасингом)

### **Вывод**

Данная лабораторная работа позволяет получить практические навыки реализации основных алгоритмов растеризации, понять их преимущества и недостатки, а также научиться выбирать подходящий алгоритм для конкретной задачи. Приложение демонстрирует эволюцию алгоритмов от простейших методов до современных оптимизированных решений, включая технику сглаживания для повышения визуального качества. Полученные знания могут быть применены в разработке графических приложений, игровых движков, систем автоматизированного проектирования и других областях компьютерной графики.