Компьютерная графика

Курс лекций

Тема №6. Алгоритмы растровой графики.

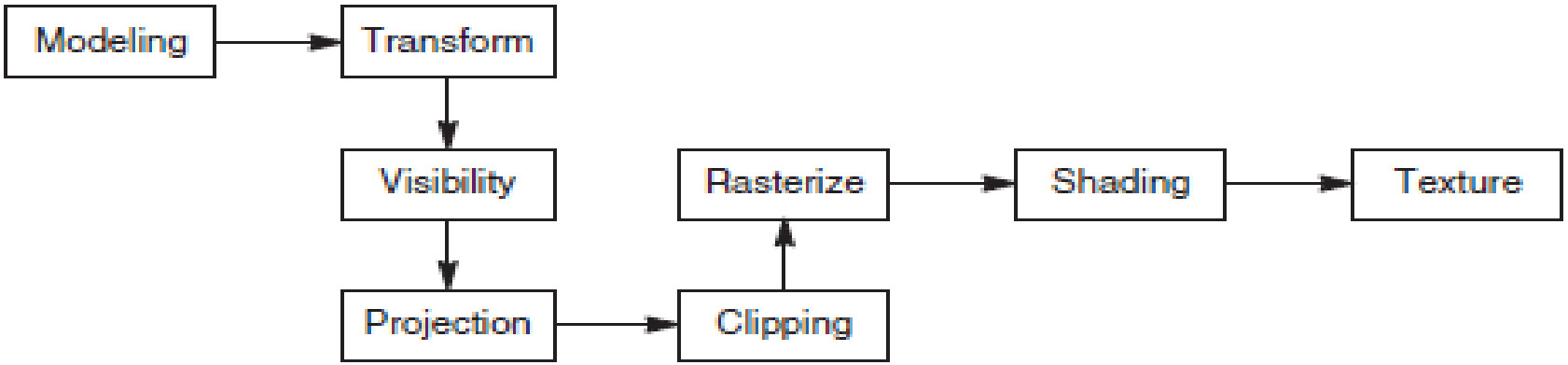
Представление алгоритмов

- назначение, условия применения
- алгоритм
  - инициализация
  - итерации алгоритма
  - условия завершения
- анализ алгоритма
  - достоинства
  - недостатки
- модификации алгоритма / оптимизация

Растровые графические системы

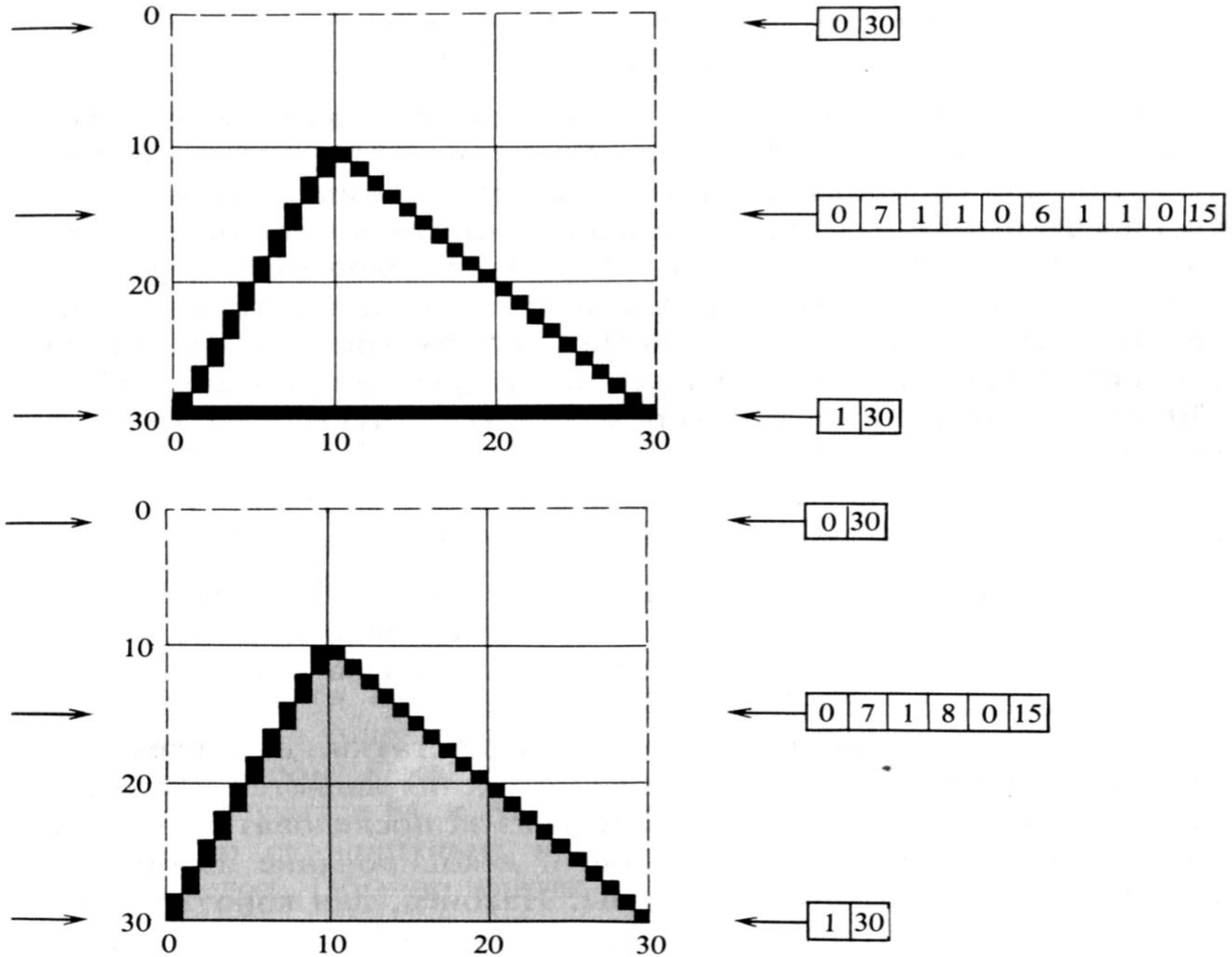
- *принцип записи изображения*: построчное сканирование луча
- *примитив*: точка (пиксель, pixel = picture element)
- необходима *процедура растеризации* геометрических примитивов (*растровая развертка* примитивов):
  - растровая развертка в реальном времени;
  - групповое кодирование (интенсивность + длина участка);
  - клеточное кодирование (шаблоны, например, кодирование литер в алфавитно-цифровом терминале);
  - использование буфера кадра (обеспечивает промежуточное хранение изображения - растеризованных графических примитивов).

Графический конвейер



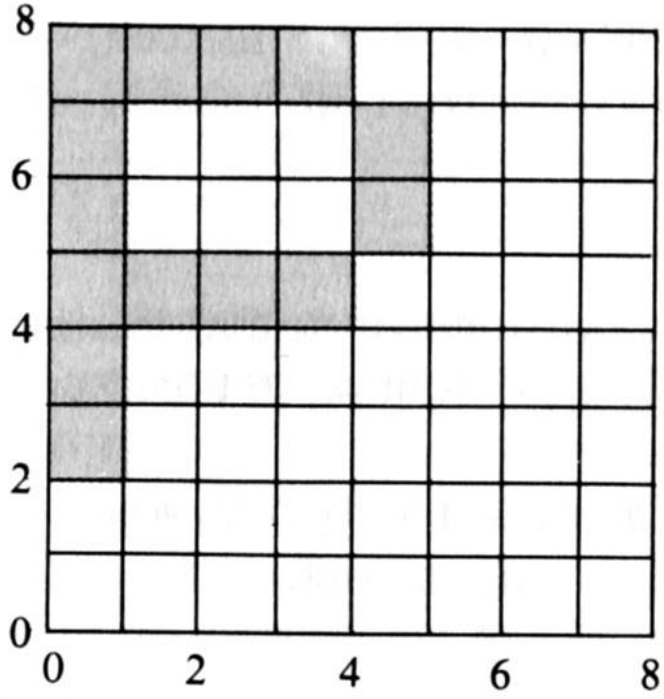
- моделирование (modeling) – математическое описание объектов, всей сцены, источников света, с учётом расположения
- отображение (rendering):
  - преобразования (transformation) – задание местоположения;
  - определение видимости (visibility) – область видимости (field of view) + нелицевые поверхности → отсечение (clipping);
  - проекция на картинную плоскость (projection);
  - растеризация (rasterization);
  - закраска (shading);
  - текстурирование (texturing).

Групповое кодирование (Run-length encoding, RLE)

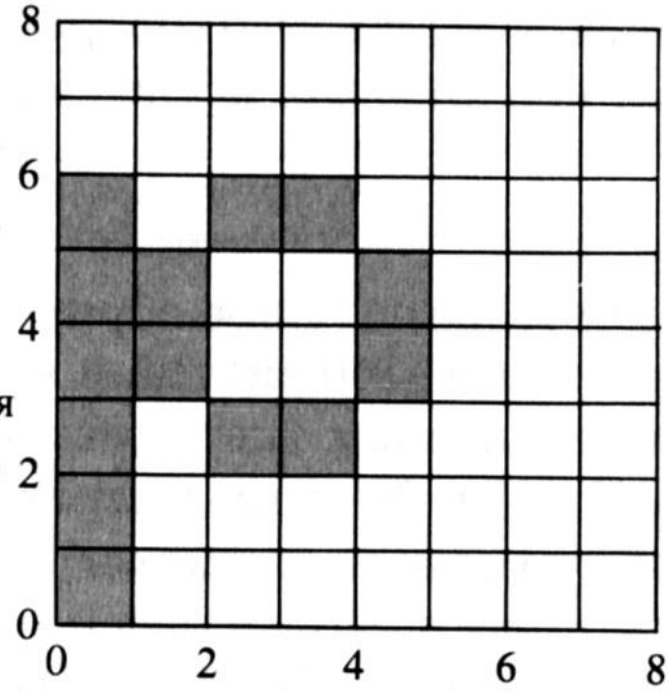


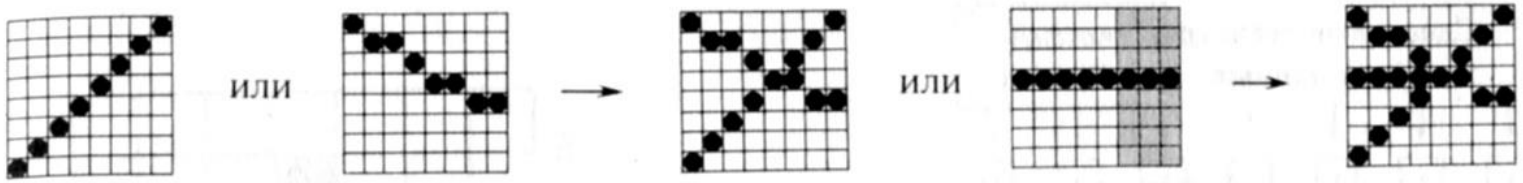
интенсивность + длина участка

Клеточное кодирование

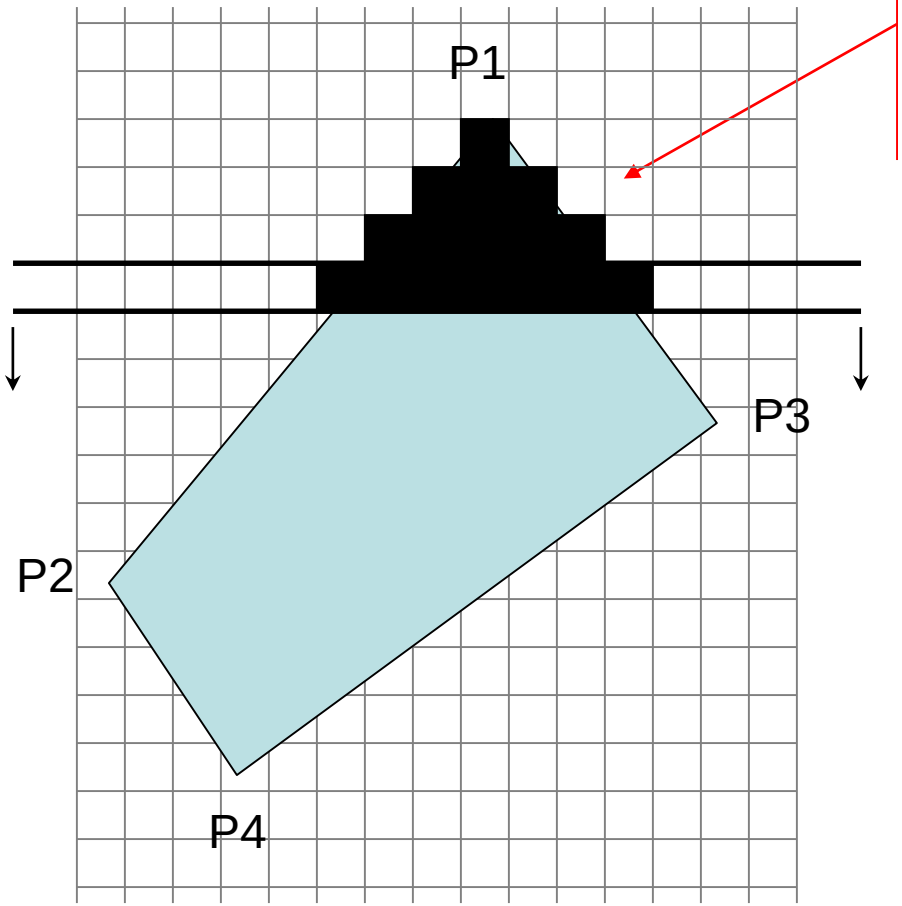


Базовая линия





Растровая развертка примитивов

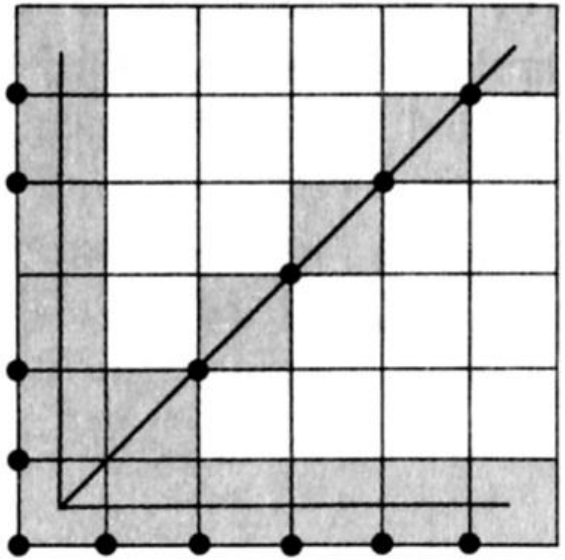


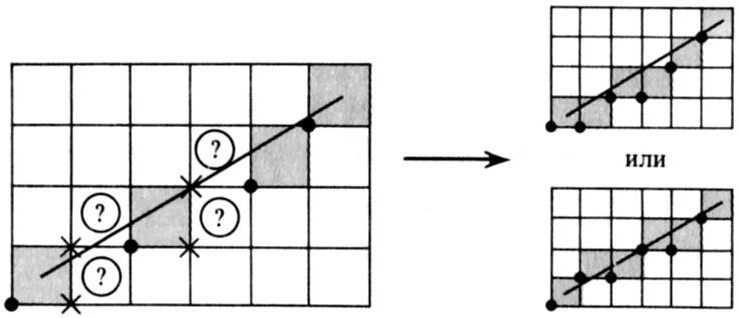
1. растеризовать примитив
2. вычислить цвет каждого пикселя
3. скомбинировать с цветом фона

линия развертки,
строка сканирования
(scanline)

Требования к алгоритмам растровой развертки кривых

- совпадение начальной и конечной точки с заданными;
- соблюдение формы кривой;
- постоянная яркость вдоль кривой (для отрезка: независимо от длины и наклона);
- высокая производительность.





Цифровой дифференциальный анализатор (DDA, Digital Differential Analyzer)

$$\frac{dy}{dx} = \text{const} \quad \text{или} \quad \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$$

$$y_{i+1} = y_i + \Delta y$$

$$y_{i+1} = y_i + \frac{y_2 - y_1}{x_2 - x_1} \Delta x$$

- недостатки алгоритма:
  - накопление ошибки округления;
  - использование вещественной арифметики.

Вещественный алгоритм Брезенхема (Bresenham, 1962)

- рассматривается случай построения отрезка в первом октанте (следовательно, задача сводится к выбору следующей точки отрезка из пары (x_k+1, y_k) и (x_k+1, y_k+1));
- вводится понятие ошибки – расстояния от фактической точки отрезка до точки растра (x_k+1, y_k) при x = x_k+1;
- инициализация:

  $\Delta e = m = \Delta y / \Delta x$

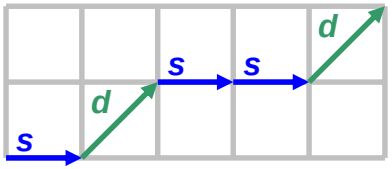
  e = -1/2 (чтобы анализировать только знак)

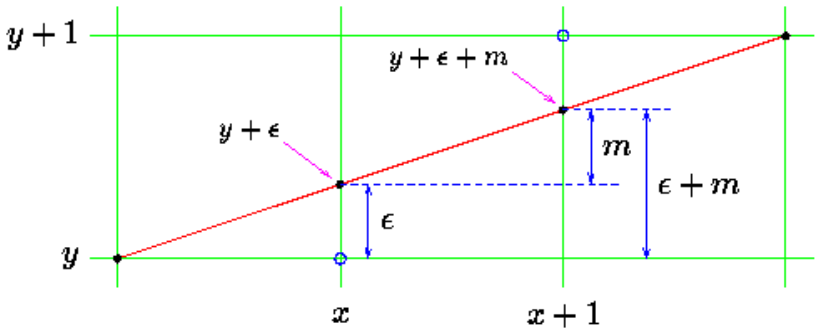
  $x = x_0$, $y = y_0$
- на каждой итерации алгоритма:

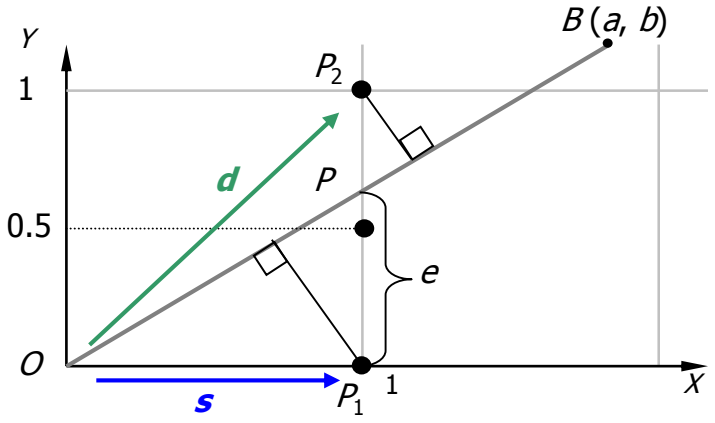
  e += Δe,

  x += 1,

  y += 1, e -=1, если e $\geq$ 0
- недостаток: использование вещественной арифметики







Целочисленный алгоритм Брезенхема

- $e' = 2e\Delta x$ (так как важен только знак ошибки);
- инициализация:

  $\Delta e = 2\Delta y$

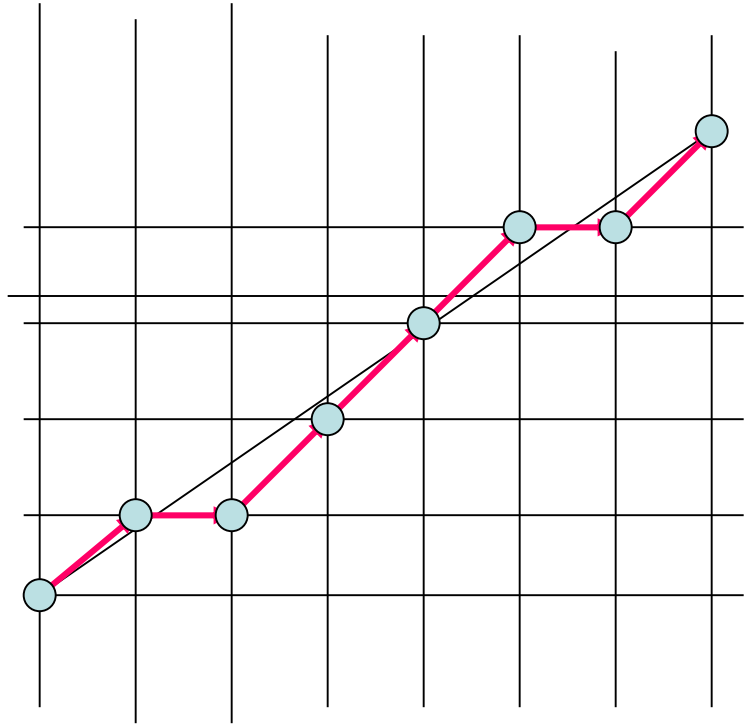
  $e_0 = -\Delta x$

  $x = x_0, y = y_0$
- на каждой итерации алгоритма:

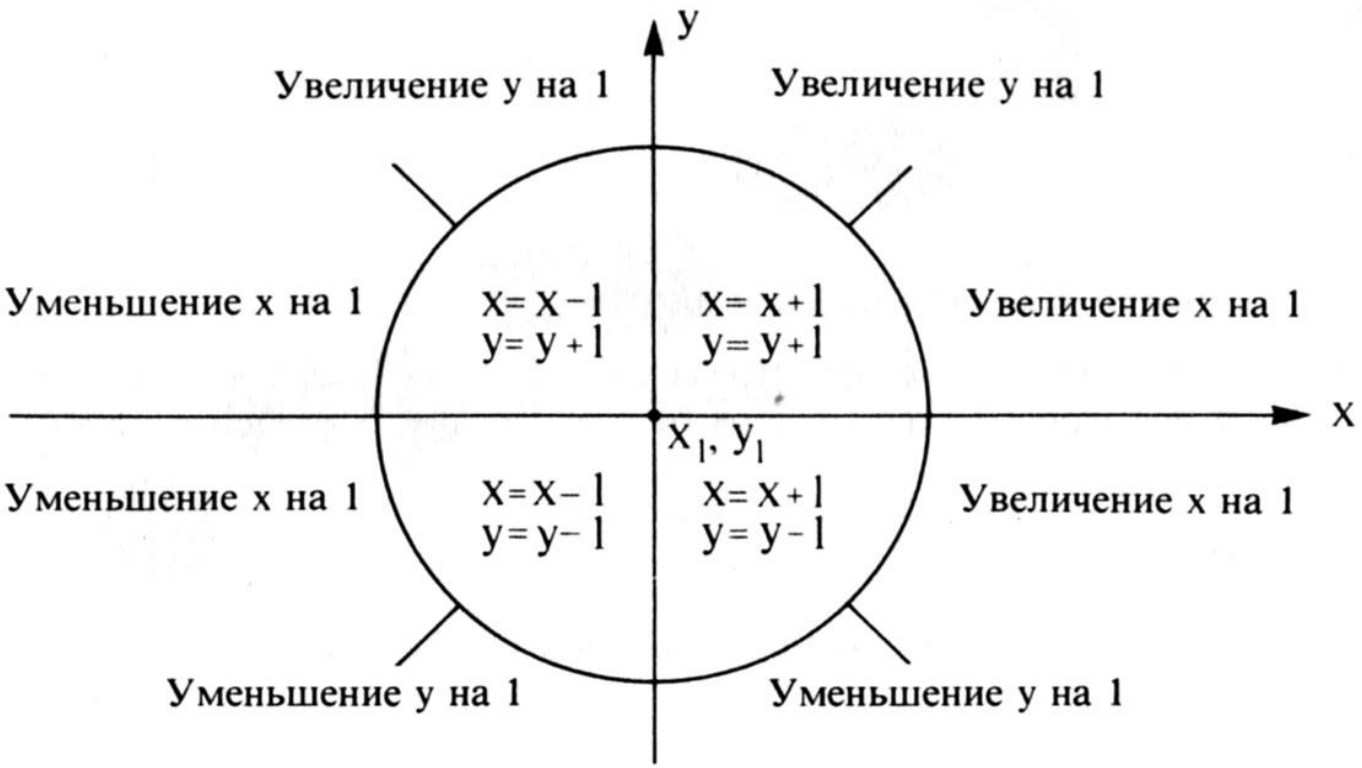
  $e \mathrel{+}= \Delta e$

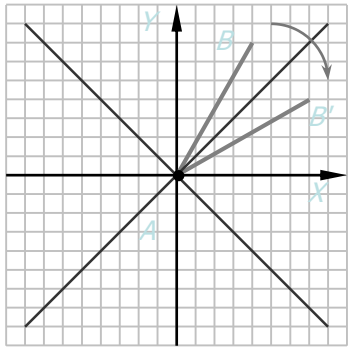
  $x \mathrel{+}= 1,$

  $y \mathrel{+}= 1, e \mathrel{-}= 2\Delta x,$ если $e \geq 0$



Обобщение алгоритма Брезенхема для произвольных октантов





- возможные изменения с учетом октанта:
  - изменение знака приращения x или y;
  - x и y могут поменяться местами.

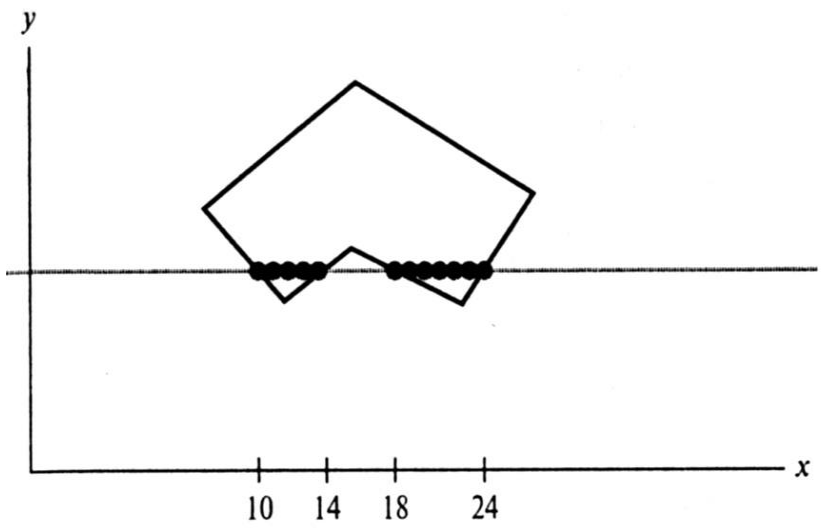
Растровая развертка сплошных областей

- растровая развертка с построчным сканированием:
  - алгоритм с упорядоченным списком ребер;
  - алгоритм со списком активных ребер;
  - алгоритм заполнения по ребрам (с перегородкой);
  - со списком ребер и флагом;
- с затравкой:
  - простой алгоритм;
  - построчный алгоритм.

Построчное сканирование

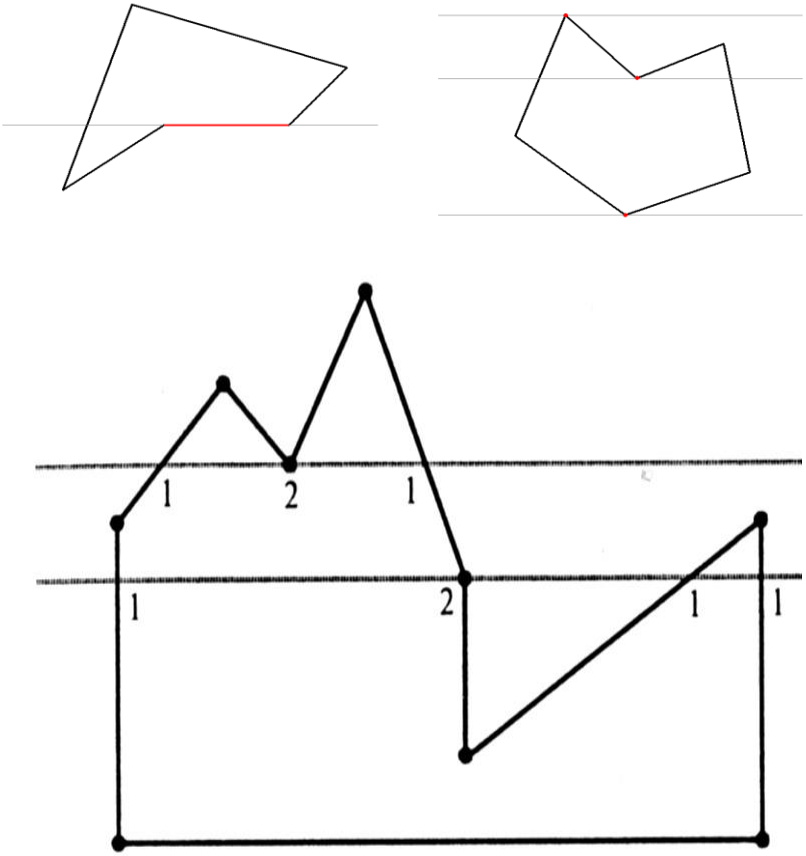
- ребра многоугольника задаются координатами ($x1$, $y1$) и ($x2$, $y2$) его концов;
- соглашение: координата x отсчитывается слева направо, а y – сверху вниз;
- алгоритмы построчного сканирования основаны на том факте, что любое горизонтальное сечение контура многоугольника состоит из четного числа точек, следовательно, для каждой строки сканирования необходимо:
  - определить ее пересечения с ребрами многоугольника;
  - упорядочить их вдоль строки сканирования;
  - заполнить интервалы между нечетными и четными точками пересечения (использование свойств пространственной *когерентности*).

- достоинства алгоритма: операции вывода на экран для каждого пикселя выполняются не более одного раза;
- недостатки алгоритма: использование динамических структур данных (списков), что сильно усложняет код и требует дополнительной памяти.



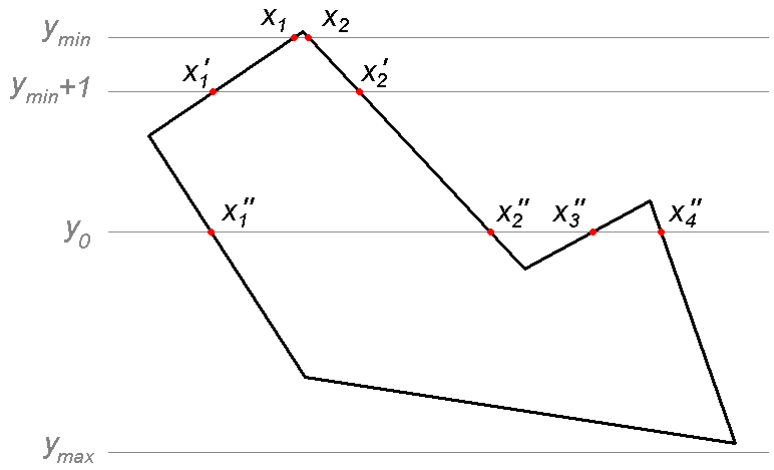
Построчное сканирование: особые случаи определения точек пересечения строки сканирования с ребрами многоугольника

- горизонтальные ребра многоугольника - не учитываются;
- строка сканирования проходит через вершину многоугольника (можно использовать один из двух подходов):
  - смещение всех строк сканирования на ½ пикселя (таким образом, вершины заведомо не будут лежать на строке сканирования);
  - вершина не учитывается (учитывается дважды), если оба смежных ребра лежат по одну сторону от строки сканирования, и учитывается один раз, если смежные ребра лежат по разные стороны от строки сканирования.



Построчное сканирование с упорядоченным списком ребер

- растеризация всех негоризонтальных ребер многоугольника и помещение полученных точек в списки *x*-координат для каждой строки сканирования;
- для каждой строки сканирования:
  - список точек пересечения с ребрами упорядочивается по возрастанию x;
  - заполняются все промежутки вида $[x_{2i-1}; x_{2i})$ – соглашения по подсчету точек пересечения строк сканирования с ребрами многоугольника гарантируют, что в каждом списке содержится четное число элементов.



y_{min}:	x_1, x_2
$y_{min}+1$:	x_1', x_2'
…	…
y_0:	$x_1'', x_2'', x_3'', x_4''$
…	…
y_{max}:	–

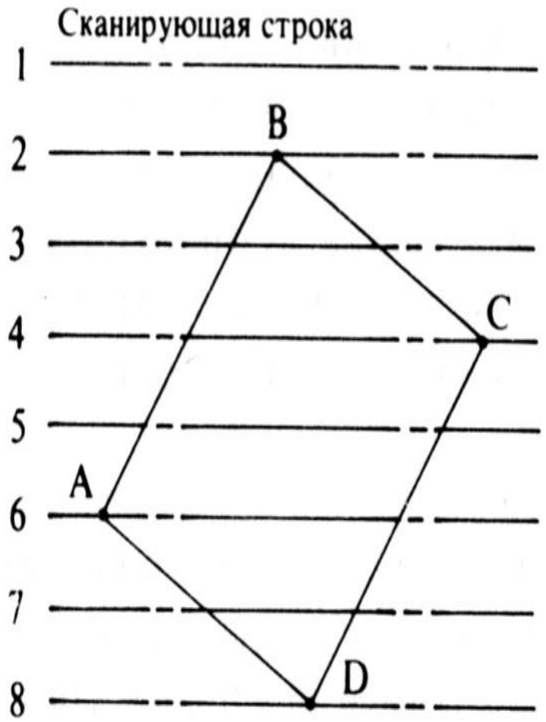
Построчное сканирование со списком активных ребер

- создается *y*-список ребер: ребра многоугольника упорядочиваются по возрастанию по координате *y* начальной вершины (начальной считается вершина с наименьшей координатой *y*);
- на каждой итерации рассматривается только текущая строка сканирования, для которой формируется список «активных» ребер (САР), в котором хранится информация только о ребрах многоугольника, пересекаемых текущей строкой;
- при переходе к очередной строке сканирования:
  - из САР удаляются ребра, чья нижняя вершина оказалась выше текущей строки сканирования;
  - из *y*-списка добавляются ребра, начинающиеся на данной строке сканирования;
  - вычисляются новые точки пересечения строки сканирования с ребрами (используя свойство пространственной когерентности, выводятся рекуррентные соотношения):

$$\Delta x = \frac{x_2 - x_1}{y_2 - y_1}; x_{i+1} = x_i + \Delta x$$

- для каждой строки сканирования:
  - список точек пересечения с ребрами упорядочивается по возрастанию x;
  - заполняются все промежутки вида $[x_{2i-1}; x_{2i})$.

Построчное сканирование со списком активных ребер: пример



a

y-группа

1	пусто
2	$x_{BA}, \Delta x_{BA}, \Delta y_{BA}, x_{BC}, \Delta x_{BC}, \Delta y_{BC}$
3	пусто
4	$x_{CD}, \Delta x_{CD}, \Delta y_{CD}$
5	пусто
6	$x_{AD}, \Delta x_{AD}, \Delta y_{AD}$
7	пусто
8	пусто

b

Список активных ребер

Сканирующая строка 3: $x_{BA} + \Delta x_{BA}, \Delta x_{BA}, \Delta y_{BA} - 1, x_{BC} + \Delta x_{BC}, \Delta x_{BC}, \Delta y_{BC} - 1$

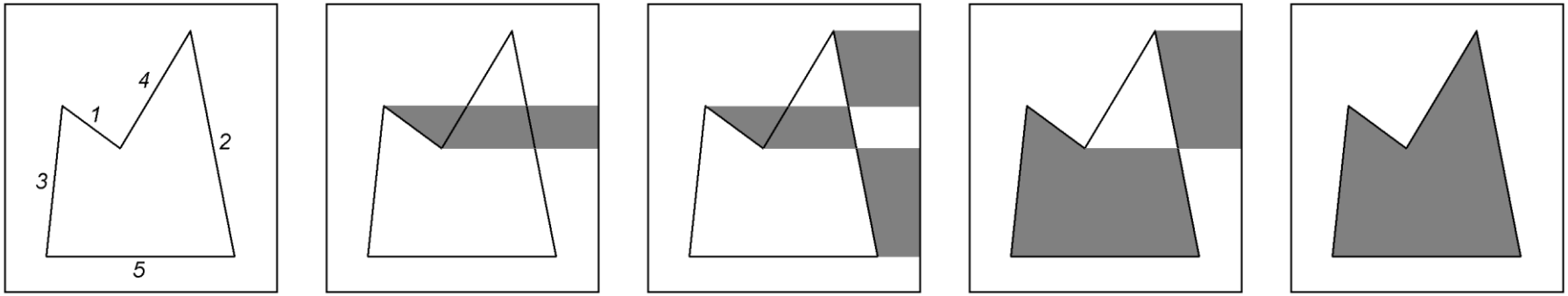
Сканирующая строка 5: $x_{BA} + 3\Delta x_{BA}, \Delta x_{BA}; \Delta y_{BA} - 3, x_{CD} + \Delta x_{CD}, \Delta x_{CD}, \Delta y_{CD} - 1$

Сканирующая строка 7: $x_{CD} + 3\Delta x_{CD}, \Delta x_{CD}, \Delta y_{CD} - 3, x_{AD} + \Delta x_{AD}, \Delta x_{AD}, \Delta y_{AD} - 1$

c

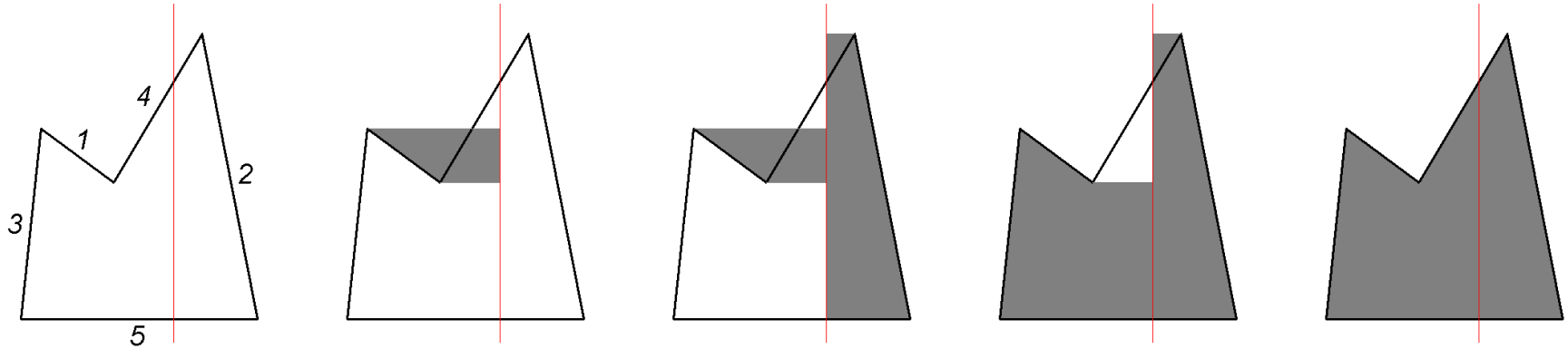
Алгоритм заполнения многоугольника по ребрам

- для каждой строки сканирования инвертируются все пиксели справа от точки пересечения данной строки сканирования с ребром многоугольника;
- порядок обработки ребер многоугольника не важен;
- каждый пиксель может обрабатываться многократно.



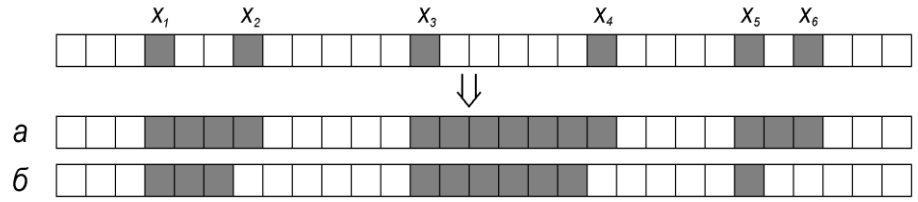
Алгоритм заполнения многоугольника по ребрам с перегородкой

- инвертирование области не между ребром и правой границей экрана, а между ребром и вертикальной прямой («перегородкой»), построенной в любом удобном месте – например, пересекающей многоугольник;
- сокращается количество повторных обработок одного пикселя.



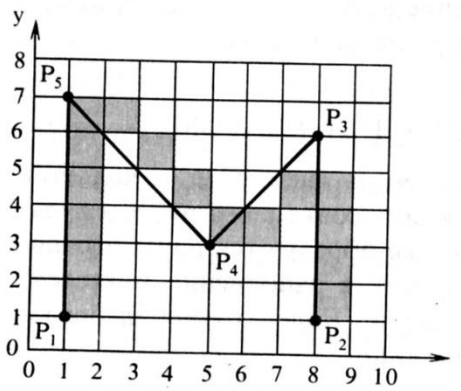
Алгоритм заполнения многоугольника со списком ребер и флагом

- алгоритм является двухшаговым:
  - на первом шаге выводится контур многоугольника, в результате чего на каждой сканирующей строке формируются пары ограничивающих пикселей;
  - на втором шаге производится заполнение интервалов на каждой сканирующей строке (на основании значения флага «внутри-снаружи», который инвертируется каждый раз, когда встречается пиксель с цветом границы);
- алгоритм не требует обработки списков;
- алгоритм допускает простую аппаратную реализацию;
- каждый пиксель обрабатывается один раз;
- алгоритм чувствителен к посторонним изображениям.



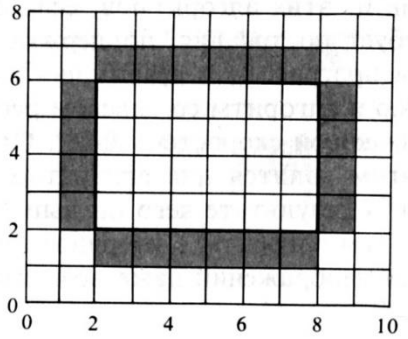
$$I(x+1, y) = I(x+1, y) \,\mathrm{XOR}\, I(x, y)$$

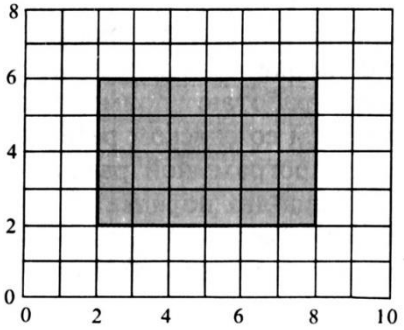
a	b	a XOR b
0	0	0
0	1	1
1	0	1
1	1	0



Алгоритмы заполнения с затравкой

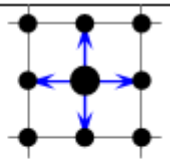
- предполагается, что известен хотя бы один пиксель из заполняемой (внутренней) области - *затравочный* пиксель;
- задание областей:
  - внутренне-определенные (все пикселы области имеют один цвет (интенсивность), а все внешние по отношению к области пиксели – другой цвет);
  - гранично-определенные (все пикселы на границе области имеют определенное значение интенсивности, отличное от цвета внутренних пикселов);
- связность областей
  - 4-связные: каждый пиксель области достижим с помощью комбинации перемещений по 4 направлениям;
  - 8-связные: каждый пиксель области достижим с помощью комбинации перемещений по 8 направлениям.





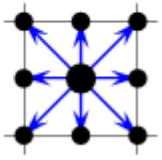
4-связность

$$|x - x_0| + |y - y_0| \le 1$$

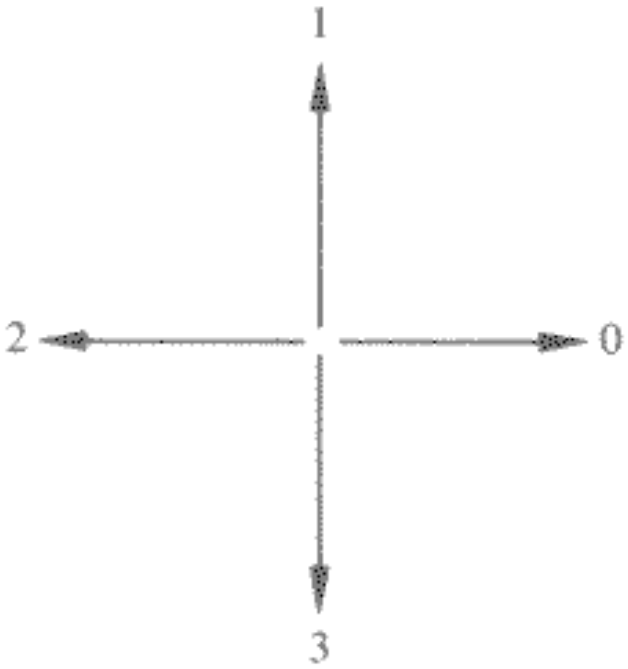


8-связность

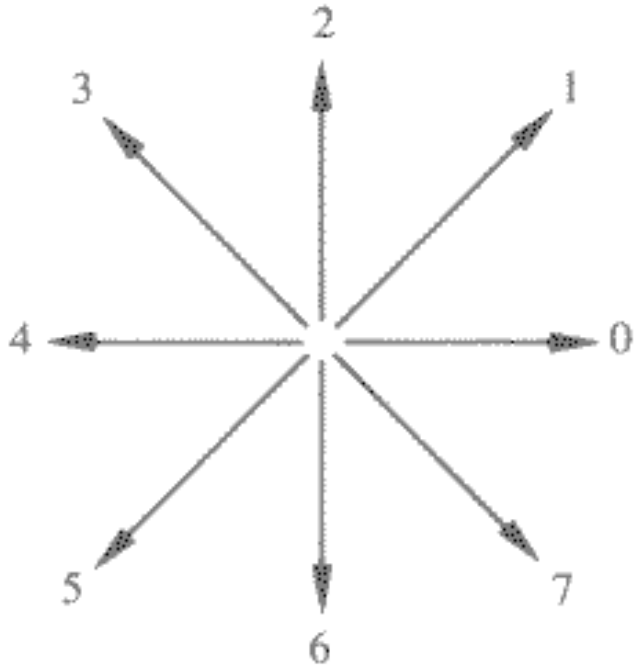
$$\begin{cases} |x - x_0| \le 1 \\ |y - y_0| \le 1 \end{cases}$$



Цепное кодирование (Freeman's chain code)

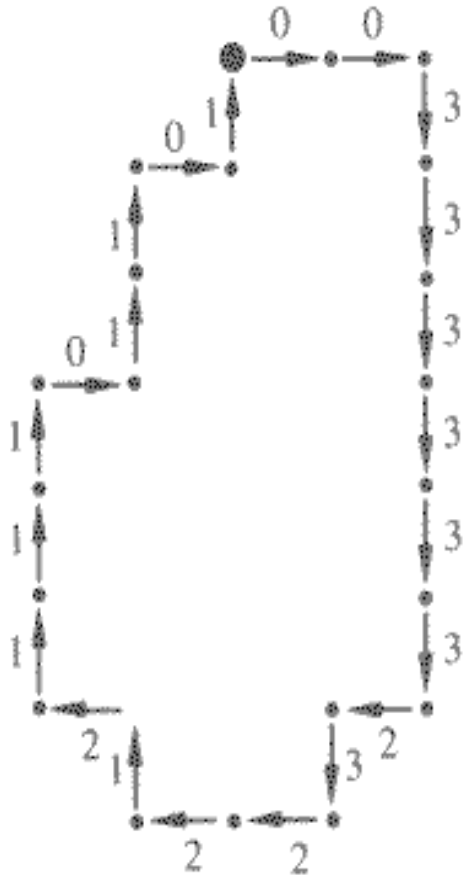


(a)

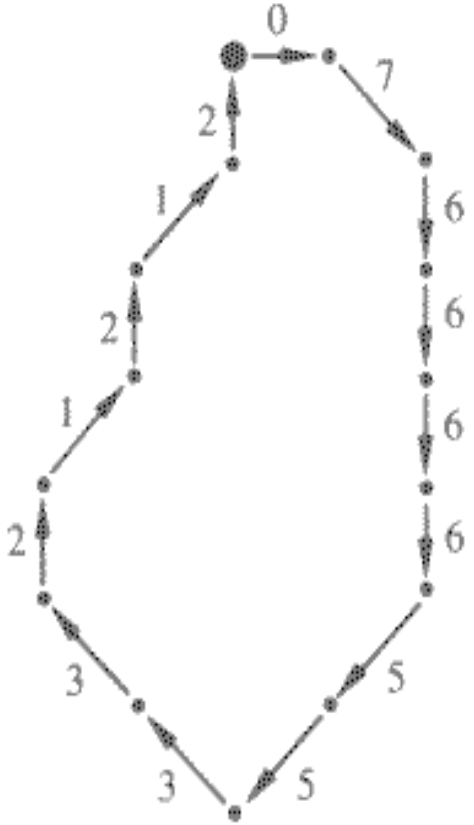


(b)

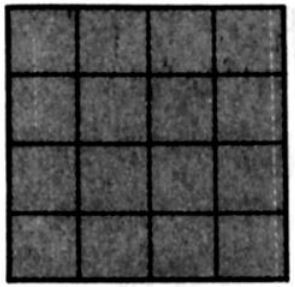
4-х и 8-ми связное кодирование

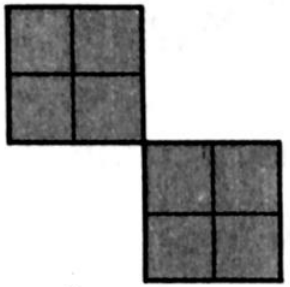


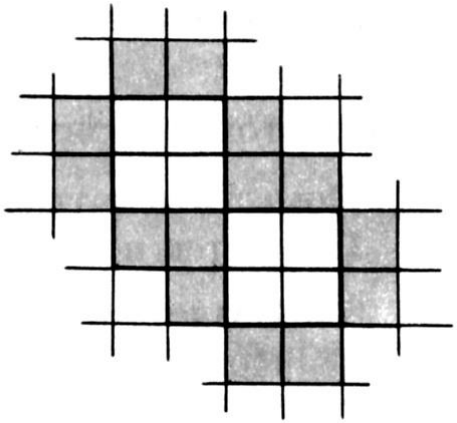
(c)



(d)


4-связная

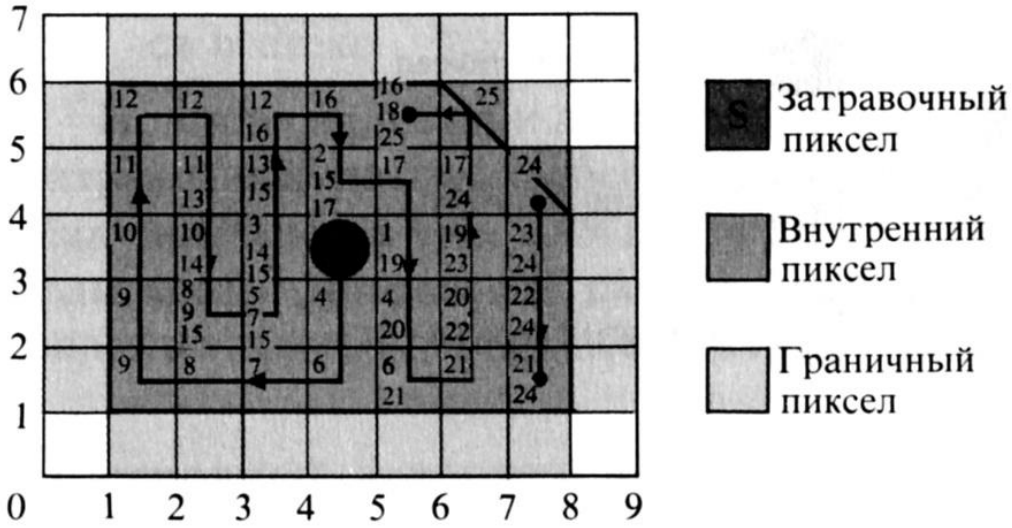

8-связная



Простой алгоритм заполнения с затравкой

- затравочный пиксель помещается в стек;
- пока стек не пуст:
  - извлечь пиксель из стека;
  - присвоить пикселю значение интенсивности заполнения;
  - для каждого из соседних пикселей (4- или 8- связных) проверить значение интенсивности для данного пикселя:
    - цвет заполнения или цвет границы – проигнорировать пиксель;
    - иначе поместить пиксель в стек.

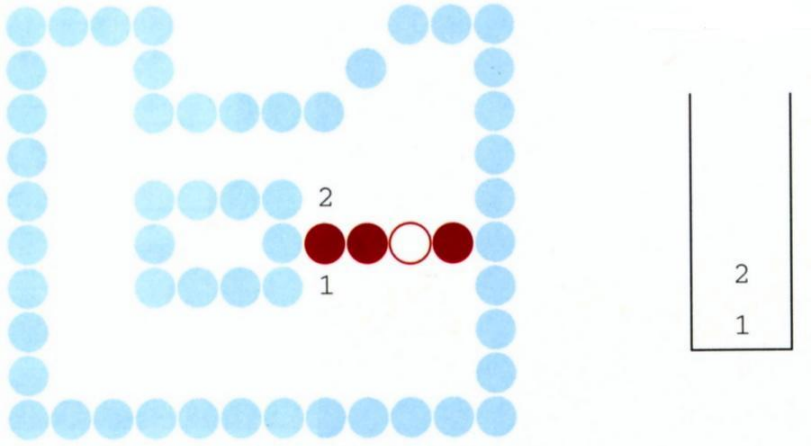
- недостатки:
  - большой размер стека;
  - в стеке содержится дублирующая и ненужная информация.

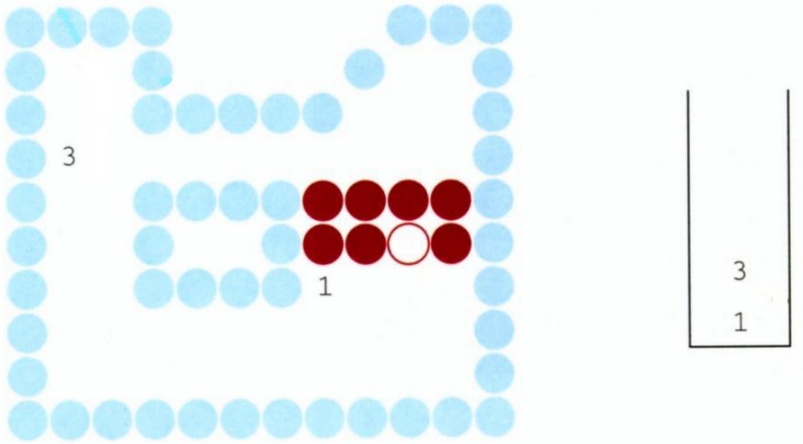


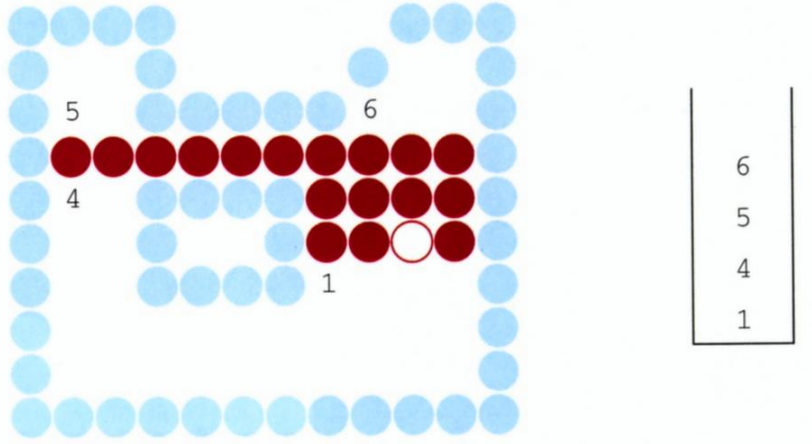
Построчный алгоритм заполнения с затравкой

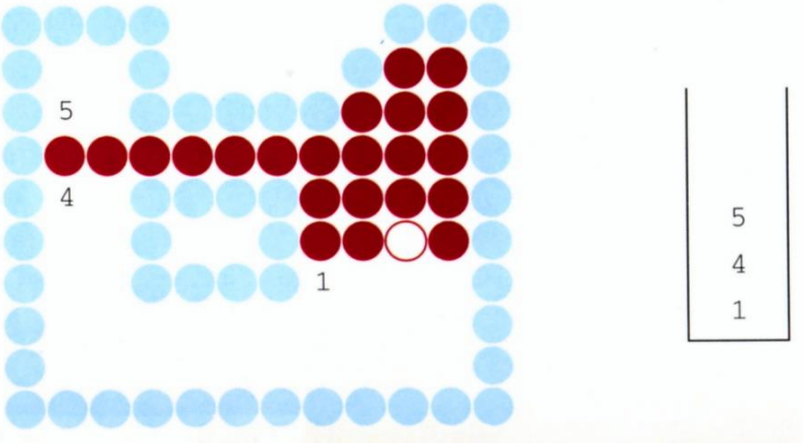
- идея: минимизация стека за счет хранения единственного затравочного пикселя для любого непрерывного интервала вдоль сканирующей строки;
- алгоритм:
  - затравочный пиксель помещается в стек;
  - пока стек не пуст:
    - интервал с затравочным пикселем заполняется влево и вправо вдоль сканирующей строки, пока не достигнут граничный пиксель (интервал $[x_{left}; x_{right}]$);
    - в диапазоне $[x_{left}; x_{right}]$ анализируются сканирующие строки непосредственно ниже и выше текущей: если обнаруживаются незаполненные интервалы в рассматриваемом диапазоне, то для каждого интервала в стек помещается крайний правый (крайний левый) пиксель.

Построчный алгоритм заполнения с затравкой: пример



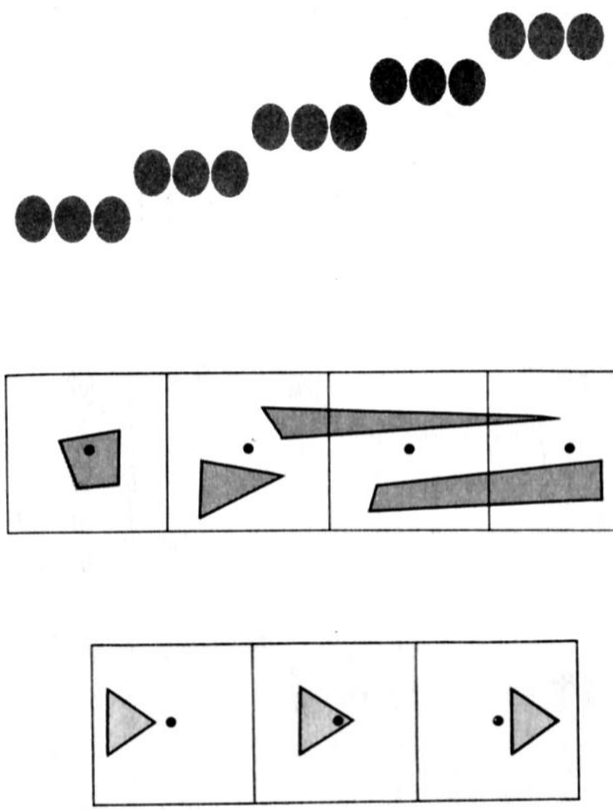






Ступенчатость (aliasing, лестничный эффект)

- типы искажений, связанных с ошибкой дискретизации при разложении в растр:
  - ступенчатость ребер, границ, кривых;
  - некорректная визуализация тонких деталей и фактур;
  - визуализация мелких объектов (проблема «мерцания» при анимации);
- причина: ошибка дискретизации при разложении в растр (недостаточная выборка для высокопериодичных изображений).



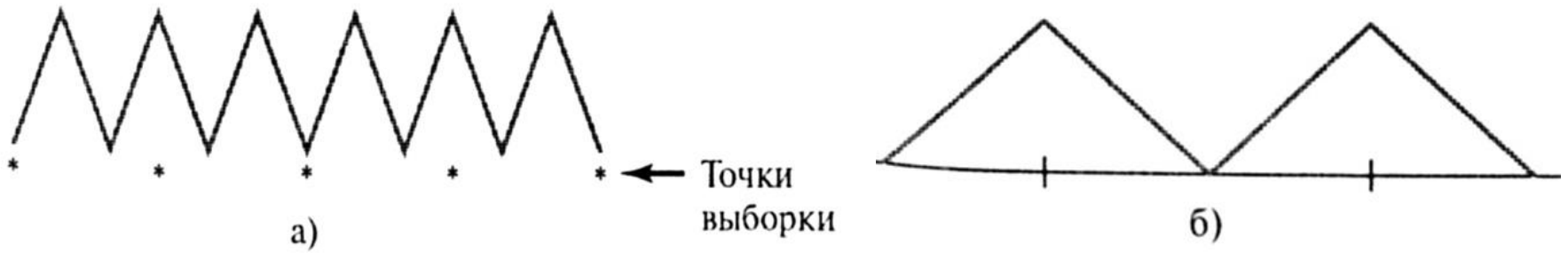
Возникновение ступенчатости

- изображение – двумерный сигнал, а сигнал раскладывается на составляющие с разной частотой и амплитудой (преобразование Фурье, Fourier transform);
- при дискретизации рассматривается только некоторый набор отсчётов (samples);
- теорема Котельникова – Найквиста (Nyquist) – Шэннона (Shannon): во избежание потерь информации о периодических объектах необходимо выбирать частоту дискретизации по меньшей мере в два раза больше наибольшей частоты, встречающейся в объекте (*частоты Найквиста*);
- альтернативная формулировка: интервал дискретизации (*интервал дискретизации Найквиста*) должен составлять не больше половины интервала цикла.

$$f_s = 2f_{\max}.$$

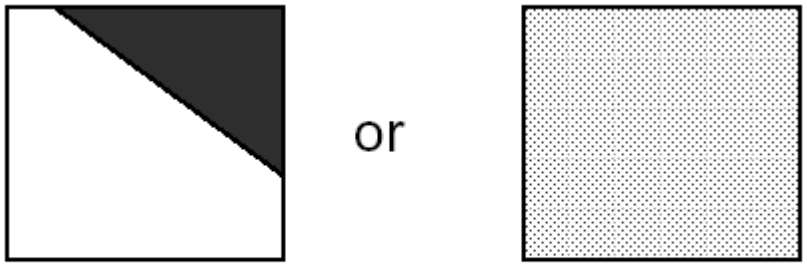
$$\Delta x_s = \frac{\Delta x_{\text{цикла}}}{2}$$

$$\Delta x_{\text{цикла}} = 1/f_{\max}.$$



Методы устранения ступенчатости (antialiasing)

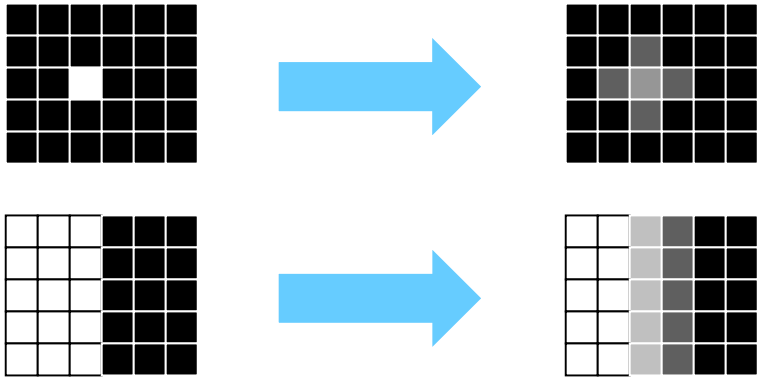
- увеличение частоты дискретизации (увеличение разрешения растра): вычисление интенсивности подпикселей с последующим усреднением результатов – *постфильтрация* (сверхдискретизация, multisampling / supersampling);
- определение интенсивности пиксела путем вычисления площади его перекрывания с изображаемым объектом - префильтрация / свёртка (convolution).



Фильтрация

- фильтрация представляет собой свертку (convolution) сигнала (изображения) с ядром свертки (функцией фильтра) – усреднение сигнала в некоторой области;
- примеры фильтров:
  - размытие (простейшее усреднение, константное, гауссово);
  - повышение четкости;
  - нахождение границ;
  - тиснение;
  - медианный фильтр.

$$Ker[k,p] = \frac{1}{6} \cdot \begin{Vmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{Vmatrix}$$

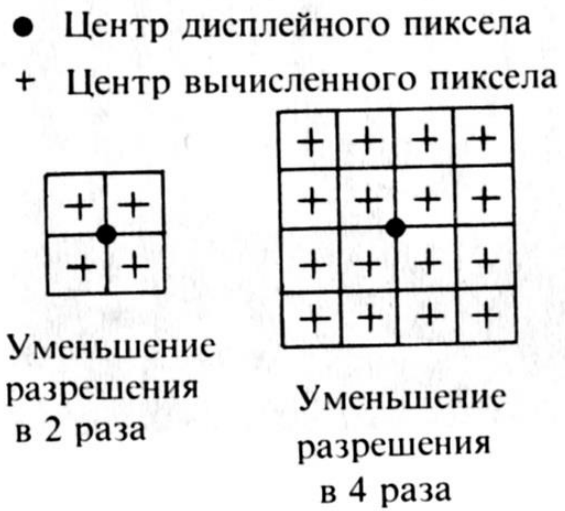


$$< f * g > (x,y) = \iint\limits_{(\lambda,\mu)\in\Omega} f(x-\lambda)(y-\mu) \cdot g(\lambda,\mu) d\lambda d\mu$$

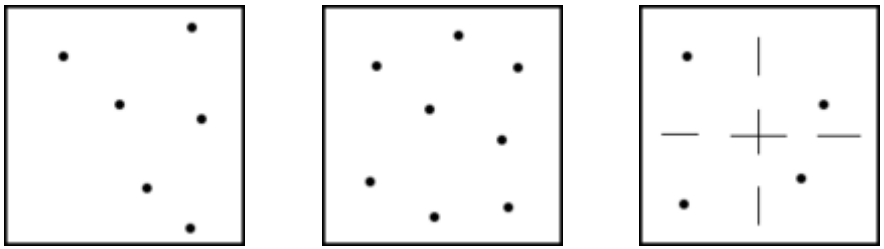
$$< f * g > (i,j) = \sum_{l=n_0}^{n_1} \sum_{k=m_0}^{m_1} f(i-l)(j-k) \cdot g(l,k)$$

Постфильтрация

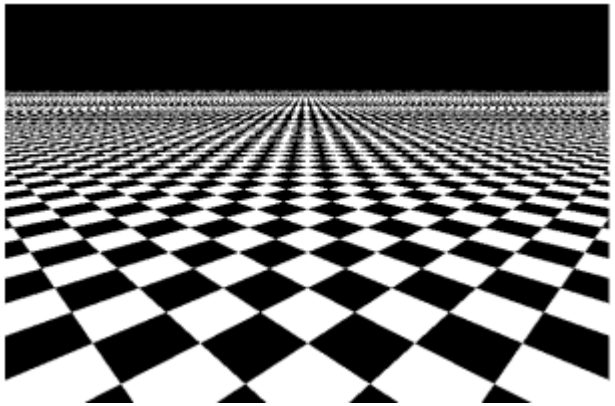
- усреднение характеристик пикселя:
  - равномерное;
  - взвешенное;

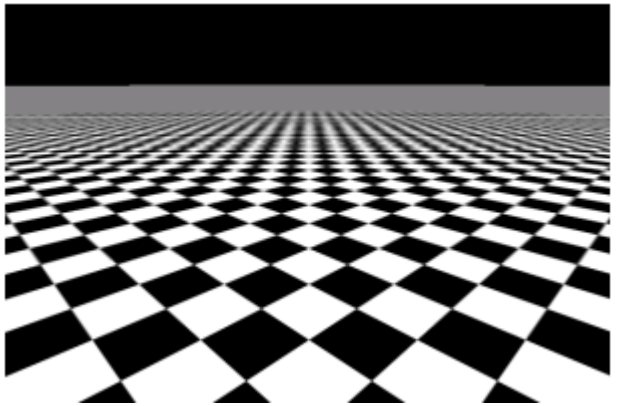


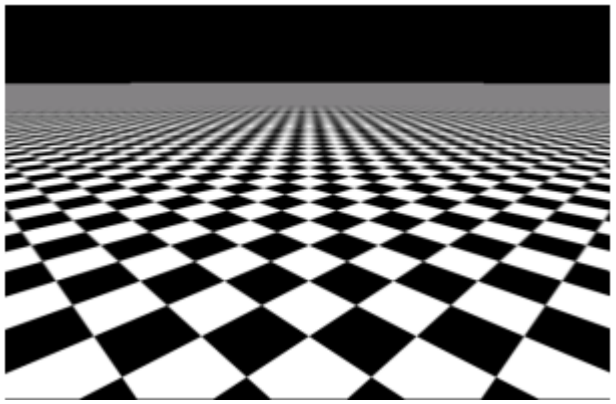
• существуют альтернативные способы выбора подпикселей

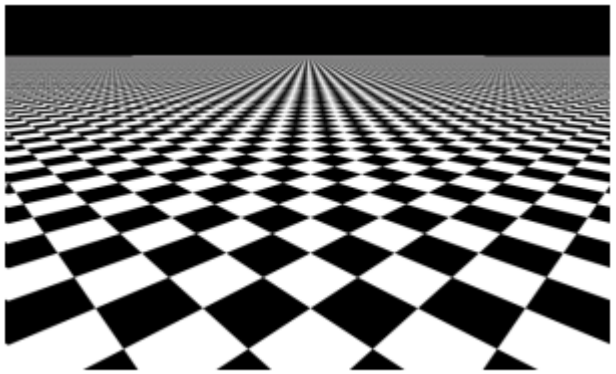


Сравнение фильтраций (при текстурировании)









Алгоритм Брезенхема с устранением ступенчатости

- доля интенсивности зависит от площади пикселя, попадающего «внутрь» многоугольника
- инициализация

  $m = I*(\Delta y / \Delta x)$

  $e_0 = m / 2 \quad (e_0 = I / 2)$

  $\Delta e = m$

  $w = I-m$

  $x = x_0,\ y = y_0$
- на каждой итерации алгоритма:

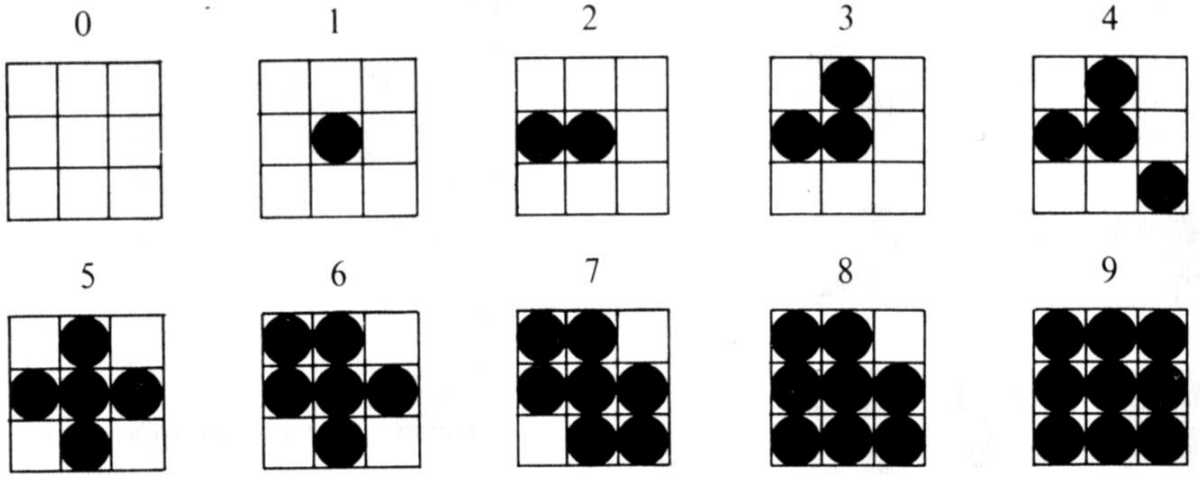
  $x \mathrel{+}= 1$,

  $y \mathrel{+}= 1,\ e \mathrel{-}= w$, если $e \geq w$

  $e \mathrel{+}= \Delta e$, если $e < w$

Аппроксимация полутонами (псевдотонирование)

- метод, в котором используется минимальное число уровней интенсивности (обычно черный и белый) для улучшения визуального разрешения, т.е. получения нескольких полутонов серого или уровней интенсивности;
- *конфигурирование* – объединение нескольких пикселей в конфигурации (улучшение визуального разрешения за счет пространственного);
- применение: полутоновая печать.



OpenGL: операции над пикселями

- определение режимов чтения/записи пикселей при передаче между буфером кадра и программным буфером:
  glPixelStore (GLenum pname, GLtype param)
- определение режимов преобразования пикселей при передаче между буфером кадра и программным буфером:
  glPixelTransfer (GLenum pname, GLtype param)
- задание текущей позиции в растре (подвергается преобразованиям):
  glRasterPos[2 3 4][s i f d] [v] () //GL_CURRENT_RASTER_POSITION_VALID
- определение коэффициента масштабирования для пиксельных операций:
  glPixelZoom (GLfloat xfactor, GLfloat yfactor)
- определение буфера кадра для операций над пикселями:
  glReadBuffer(GLenum mode **)** – для чтения
  glDrawBuffer(GLenum mode **)** – для записи
  // GL_NONE, GL_FRONT_LEFT, GL_FRONT_RIGHT, GL_BACK_LEFT, GL_BACK_RIGHT, **GL_FRONT**, **GL_BACK**, GL_LEFT, GL_RIGHT, GL_FRONT_AND_BACK, GL_AUXi
- операции над пикселями:
  – чтение из буфера кадра:
    glReadPixels(GLint x, GLint y, GLsizei width, GLsizei height, GLenum format, GLenum type, GLvoid *pixels)
  – запись в буфер кадра:
    glDrawPixels(GLsizei width, GLsizei height, GLenum format, GLenum type, const GLvoid *pixels **)**
  – копирование в текущую позицию растра:
    glCopyPixels(GLint x, GLint y, GLsizei width, GLsizei height, GLenum type **)**

Лабораторная работа № 4

- Реализовать алгоритм растровой развертки многоугольника (в соответствии с вариантом):
  - построчного сканирования многоугольника с упорядоченным списком ребер;
  - построчного сканирования многоугольника со списком активных ребер;
  - заполнения многоугольника по ребрам;
  - заполнения многоугольника по ребрам с перегородкой;
  - заполнения многоугольника со списком ребер и флагом;
  - построчного заполнения с затравкой для четырехсвязной гранично-определенной области;
  - построчного заполнения с затравкой для восьмисвязной гранично-определенной области;
- Реализовать алгоритм фильтрации (в соответствии с вариантом):
  - целочисленный алгоритм Брезенхема с устранением ступенчатости;
  - постфильтрация с взвешенным усреднением области NxN (с использованием аккумулирующего буфера);
  - постфильтрация с равномерным усреднением области NxN (с использованием аккумулирующего буфера);
  - постфильтрация с взвешенным усреднением области NxN (без использования аккумулирующего буфера);
  - постфильтрация с равномерным усреднением области NxN (без использования аккумулирующего буфера);
- Реализовать необходимые вспомогательные алгоритмы (растеризации отрезка) с модификациями, обеспечивающими корректную работу основного алгоритма.
- Ввод исходных данных каждого из алгоритмов производится интерактивно с помощью клавиатуры и/или мыши. Предусмотреть также возможность очистки области вывода (отмены ввода).
- Растеризацию производить в специально выделенном для этого буфере в памяти с последующим копированием результата в буфер кадра OpenGL. Предусмотреть возможность изменения размеров окна.

Вопросы к экзамену

- Требования, предъявляемые к алгоритмам разложения отрезка в растр. Алгоритм цифрового дифференциального анализатора.
- Целочисленный и вещественный алгоритмы Брезенхема разложения отрезков в растр.
- Типы искажений, связанные с ошибкой дискретизации при разложении в растр. Основы методов устранения ступенчатости: префильтрация и постфильтрация. Алгоритм Брезенхема с устранением ступенчатости.
- Алгоритмы построчного сканирования с упорядоченным списком ребер и со списком активных ребер.
- Алгоритмы заполнения многоугольников по ребрам, с перегородкой, со списком ребер и флагом.
- Алгоритмы заполнения многоугольников с затравкой. Задание границ и областей. Простой и построчный алгоритмы заполнения с затравкой.