Инженерная и компьютерная графика
6 семестр (диф.зачет)

Лектор:

Таранцев Игорь Геннадьевич

Доцент ФИТ НГУ, ИАиЭ, «СофтЛаб-НСК»

Создатели курса:

Дебелов Виктор Алексеевич

Валеев Тагир Фаридович

Козлов Дмитрий Сергеевич

Лекция №9

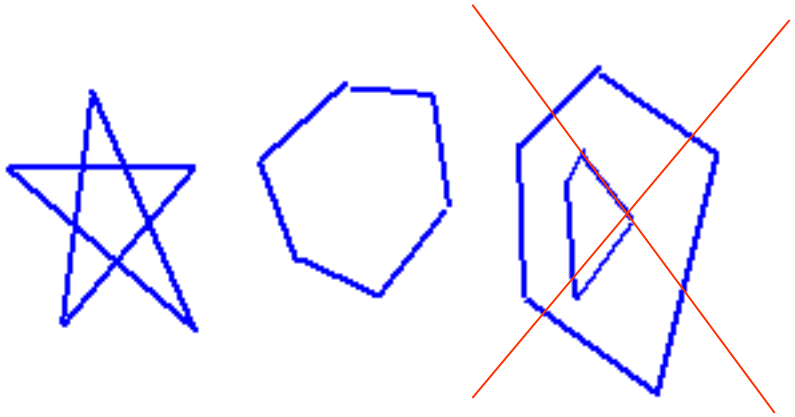
Клиппирование. Алгоритмы
Сазерленда-Ходжмана;
Вейлера-Айзертона

Растеризация многоугольника

Многоугольники

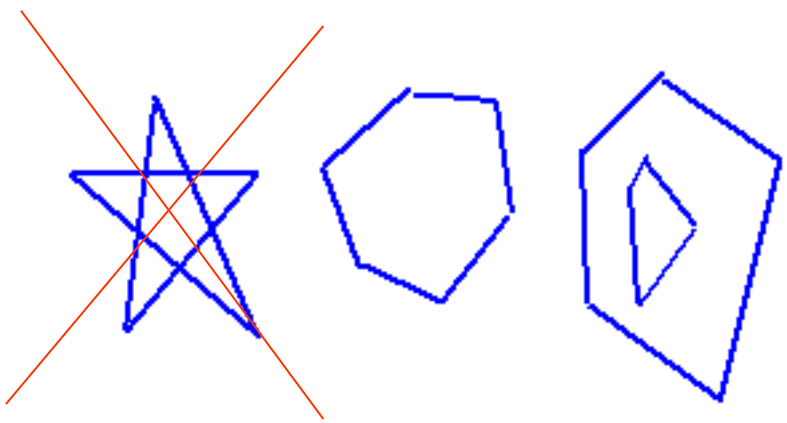
Многоугольники (polygons) в плоскости R^2. Отметим, что ломаная в машинной графике обычно переводится как polyline

- **Определение 1.** M – это замкнутая ломаная линия, т.е. берем n точек A1, …, An и соединяем их отрезками (A1,A2), …, (An,A1). Таким образом определенный M допускает самопересечение границы. Другими словами, M – это некоторый замкнутый путь. А отрезки называются сторонами M.



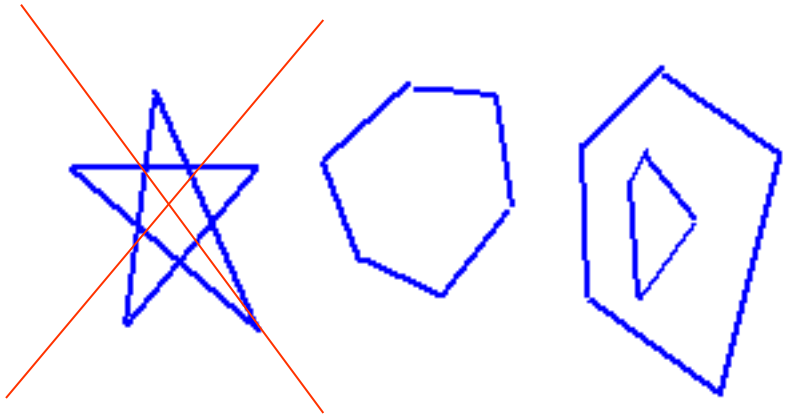
Многоугольники

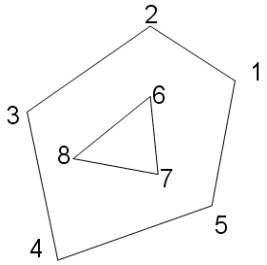
- **Определение 2.** М – это *связная* часть плоскости, вся граница которой состоит из конечного числа отрезков, называемых сторонами М. Могут быть и неограниченные М, т.е. часть плоскости, ограниченная конечным числом отрезков и полупрямых (лучей). Примером неограниченного М является плоскость с многоугольной "дыркой".



Многоугольники

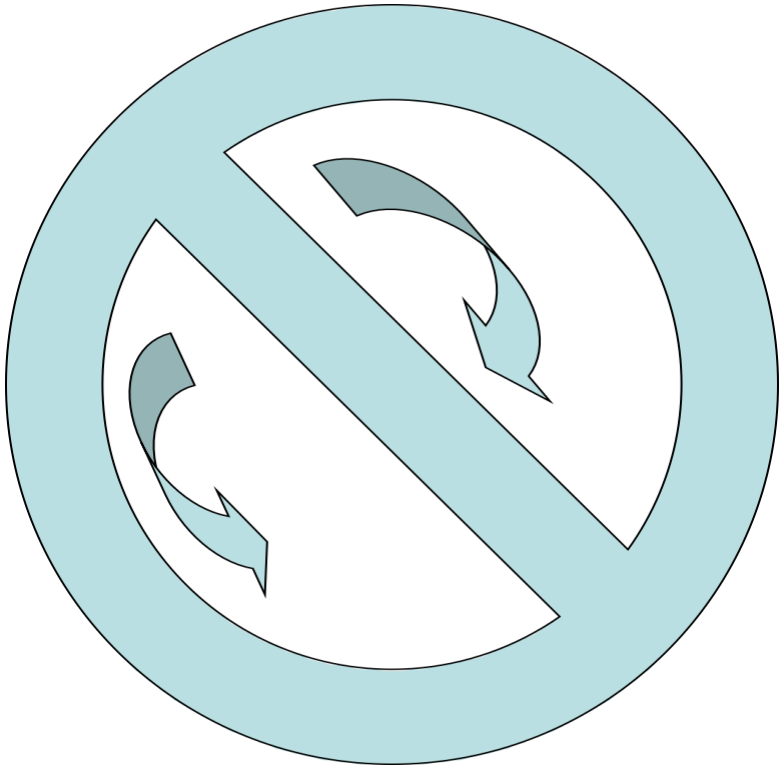
- **Определение 3. Ориентированный М**. Если каждой стороне приписать направление и все стороны упорядочить, т.е. создать из них ориентированный путь, то *положительной* считается ориентация, когда обход границы М (контура) делается против часовой стрелки – область остается слева. Такое определение ориентации верно и для области, не только для М. Возможны многосвязные границы.



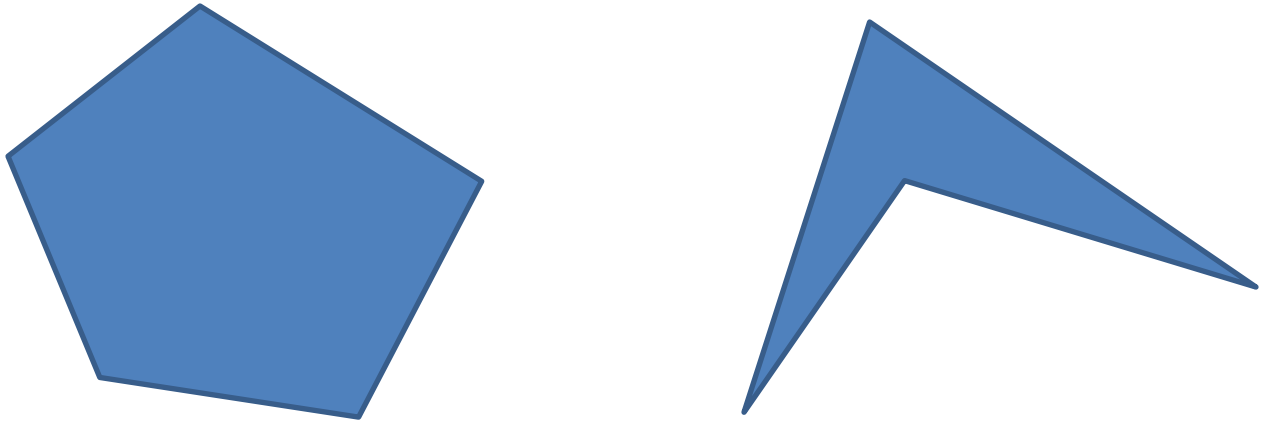


Многоугольники

- При чтении литературы и разборе алгоритмов следует обратить внимание какой обход применяет автор – инженеры часто рассматривают задание области границей, оставляя ее справа при обходе (по нашему – отрицательно ориентируют).



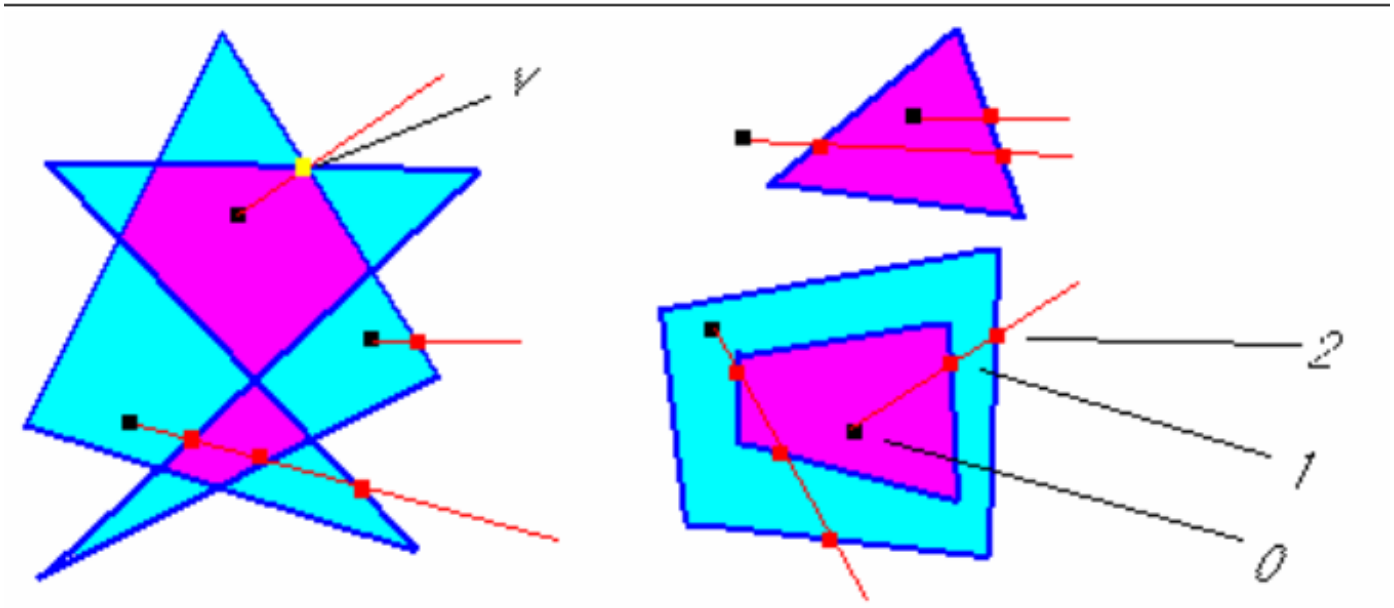
Многоугольники



M считается выпуклым, если никакая сторона M, будучи неограниченно продолженной, не разрезает M на две части.

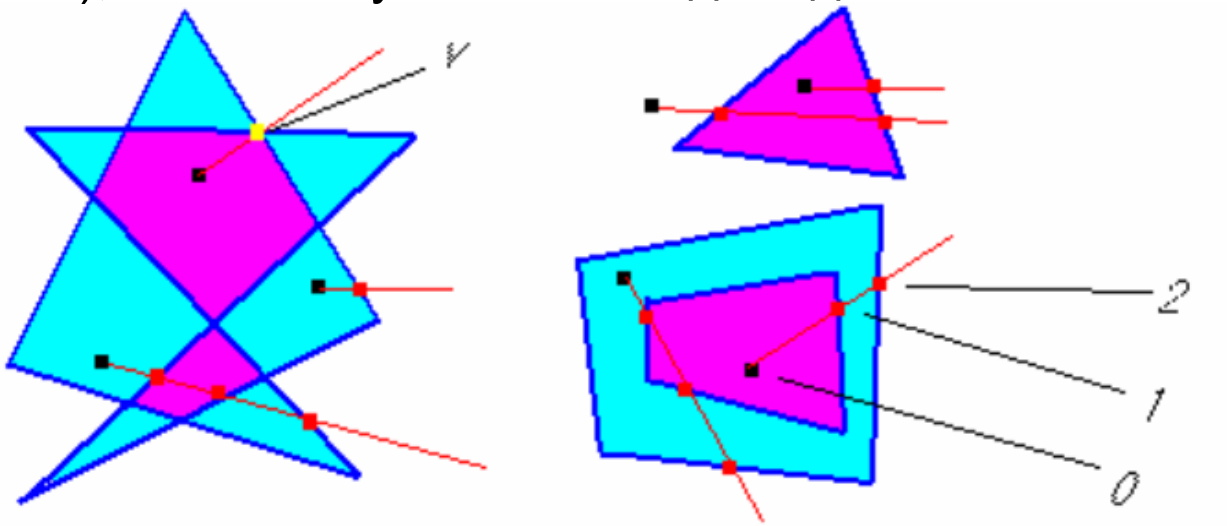
Характеристическая функция

- При помощи определений 1 или 2 задается принадлежность M точки (x,y) из R². И используя любое определение, применяем правило **«чет-нечет» (even-odd)**



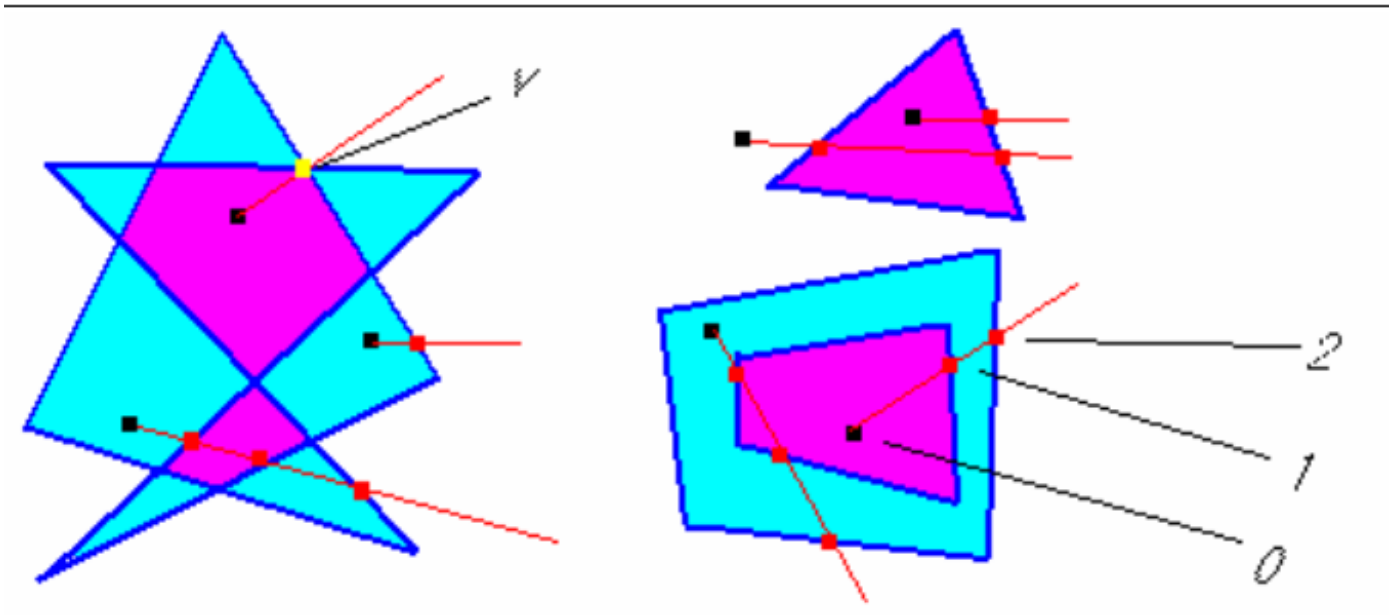
Характеристическая функция

- Выпускаем луч из точки 0 в произвольном направлении. Кстати, вместо луча можно использовать любую кривую заканчивающуюся в бесконечности.
- Вычисляем точки пересечения этого луча (отмечены красным цветом на рисунке) со всеми сторонами M (или с границей области в случае произвольных границ). Следует помнить, что стороны M должны рассматриваться как полуоткрытые отрезки, чтобы в случае пересечения луча с границей в вершине M, не считать пересечение дважды. В случае пересечения луча с точкой пересечения отрезков пути (как на левой части рисунка – помечена желтым и как "v"), эта точка учитывается дважды.



Характеристическая функция

Если число точек пересечения луча с границей нечетно (1, 3, 5, …), то тестируемая точка принадлежит области, с чьей границей анализировались пересечения. Иначе точка находится вне области.



Характеристическая функция

- Часто используется характеристическая функция вида:

$$Fo(x,y)=\begin{cases}1, & (x,y)\in \overset{o}{M}\\ 0, & (x,y)\in \partial(M)\\ -1, & (x,y)\in \overline{M}\end{cases}$$

- Учитывая неточность вычислений вещественной арифметики, очень трудно определить приналежность точки границе области.

Характеристическая функция

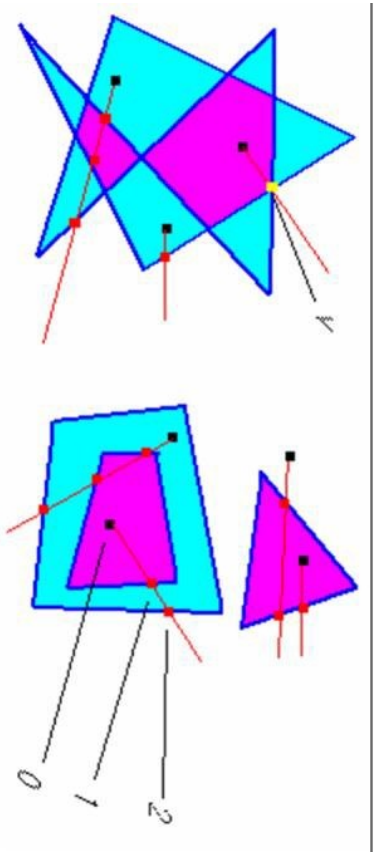
- В связи с этим нередко используется несколько модернизированная функция близости точки к границе, напр., на основе Эвклидового расстояния:

$$Fo(x,y)=\begin{cases}\rho(x,y), & (x,y)\in \overset{o}{M}\\ 0, & (x,y)\in \partial(M)\\ -\rho(x,y), & (x,y)\in \overline{M}\end{cases}$$

- По правде говоря, это достаточно трудоемкая задача.

Многоугольник

- Вернемся еще раз к рисунку. Каждый замкнутый контур может определять две области: ограниченную (внутри контура) и неограниченную – вне его. Чтобы избежать неопределенности и вводилось понятие ориентации границы области (направление обхода). В качестве альтернативы этому используется буквальное понятие ограниченности области Ω, т.е. предикат $(\infty \in \Omega)$.
- Учитывая факт, что понятие многоугольник можно понимать по-разному, то при любом упоминании слова «*многоугольник*» или «*полигон*» при описании алгоритмов или постановок задач нужно обязательно говорить, что под этим понимается. Отметим, что достаточно недвусмысленно следующее определение.



Многоугольник

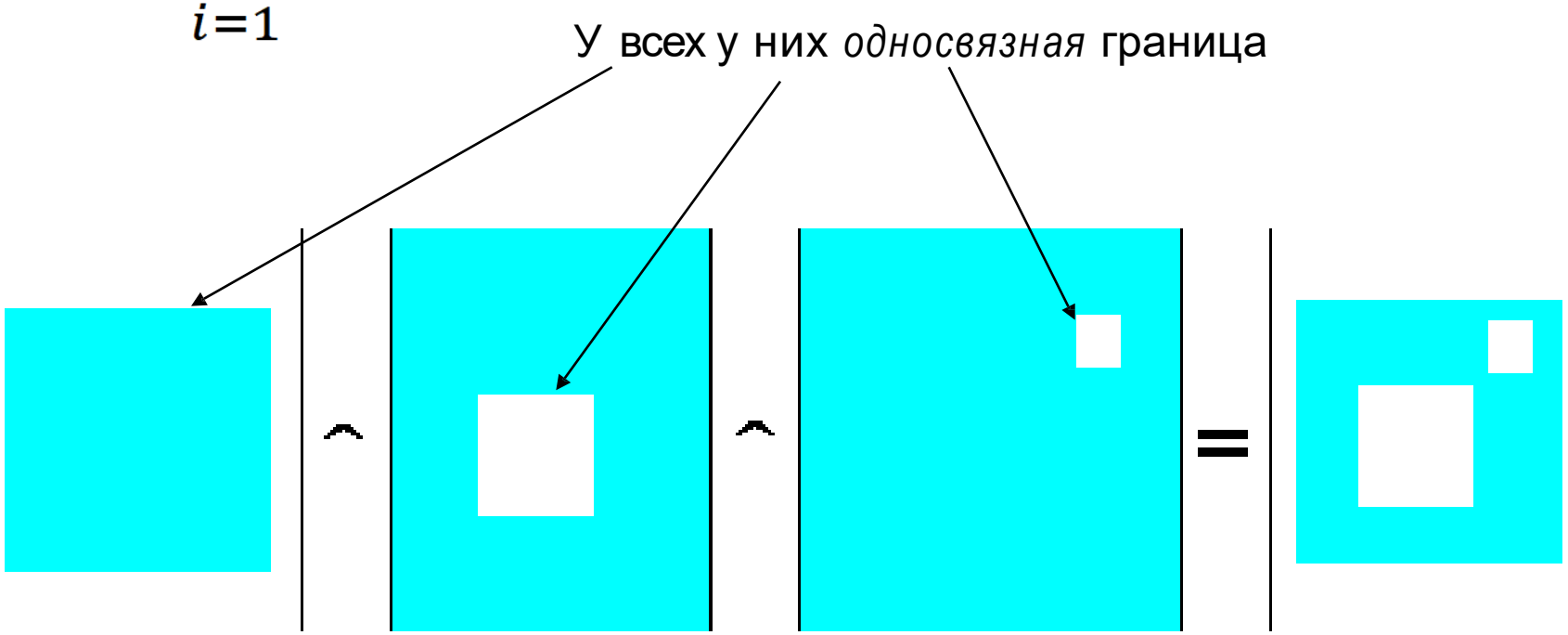
- **Определение 4**. Граница M – это множество отрезков. M может быть представлен в виде пересечения конечного числа *связных* (ограниченных или неограниченных) областей M_i $(i = 1..n)$, имеющих односвязную границу

$$M = \bigcap_{i=1}^{n} M_i$$

- Две области специального вида ø (пустое) и R^2 (вся плоскость) тоже отнесем к M (к многоугольникам)

Многоугольник

$$M = \bigcap_{i=1}^{n} M_i$$



Площадь и периметр

- Многоугольник задан набором вершин (x_i, y_i); $x_0 = x_n$, $y_0 = y_n$:
  - Площадь: $\frac{1}{2}\sum_{i=0}^{n-1}(x_i y_{i+1} - x_{i+1} y_i)$
  - Периметр: $\sum_{i=0}^{n-1}\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}$

Многоугольник

- В машинной графике рассматриваются М как выпуклые, так и невыпуклые. М с самопересечением границ не используются по очевидной причине, т.к. они являются объединением более простых.

Две популярные задачи:

- Отсечение (clipping) многоугольников
- Растровая развертка многоугольников (или в более общей постановке – закраска –shading). См. книгу Роджерса для получения более полной информации.

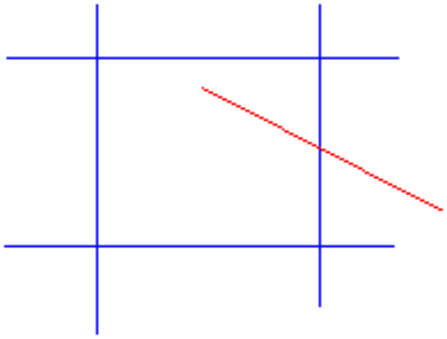
Д. Роджерс. Алгоритмические основы машинной графики. Пер. с англ. – М.: Мир, 1989. (2-ое издание – 2001 год).

Отсечение, клиппирование (clipping)

Обычное применение этой операции – отображение на прямоугольном экране части многоугольника, попадающей в него. На самом деле отсечение рассматривается более широко – отсечение по многоугольнику (по любой фигуре). Но, случай, когда М является прямоугольным окном, наиболее важен.

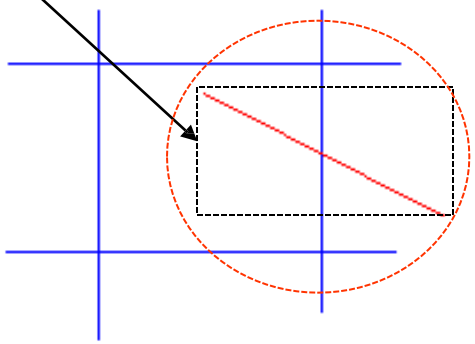
Алгоритмы отсечения можно разделить:

- отсечение отрезков и линий более высокого порядка;
- отсечение пути "пера";
- отсечение многоугольников и других фигур



Отсечение (clipping)
Простейшие алгоритмы

- Осуществляется растеризация плоской фигуры. Пиксели, которые не попадают в экран (окно), отбрасываются. Очень дорогой алгоритм, поскольку вся фигура может оказаться вне экрана.
- Габаритный тест. Вокруг растеризуемой фигуры описывается габаритный **бокс** (*прямоугольник, стороны которого параллельны осям координат*) или круг. Если габарит весь лежит вне экрана, то фигура не обрабатывается.
- Другие (см. книгу Роджерса)



Отсечение (clipping)

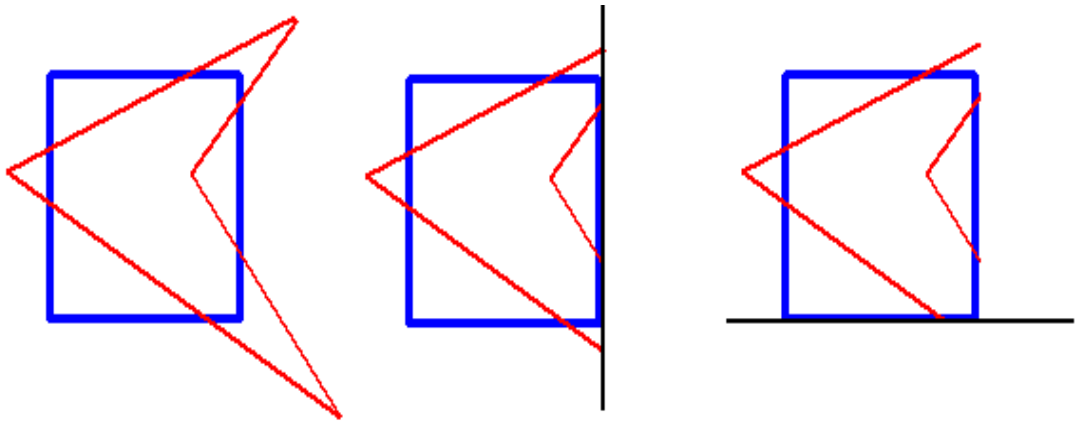
Алгоритм Сазерленда-Ходжмана

- Данный алгоритм выполняет отсечение (*клиппирование*) многоугольника, пути пишущего инструмента, просто отрезков, как на плоскости, так и в пространстве. Область, по которой делается отсечение, должна быть выпуклой: выпуклый многоугольник на плоскости или выпуклый многогранник в пространстве (возможно неограниченный типа пирамиды видимости).
- Проиллюстрируем работу алгоритма отсечения серией картинок.

Отсечение (clipping)

Алгоритм Сазерленда-Ходжмана

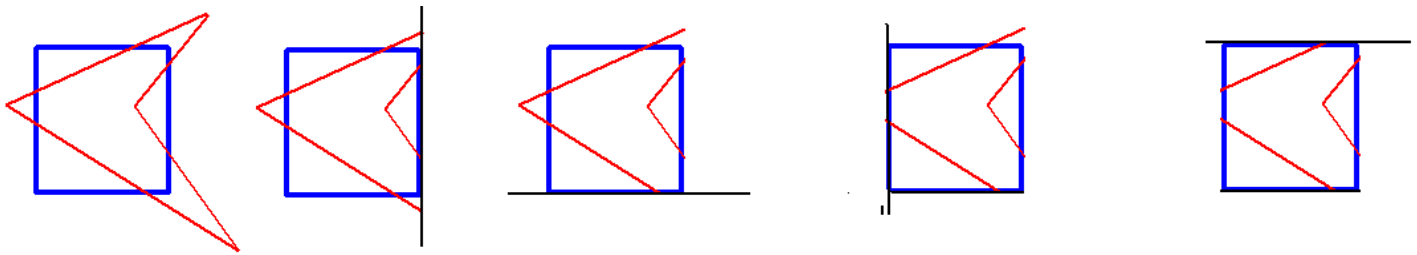
- Слева прямоугольное окно $[a,b] \times [c,d]$ и исходный многоугольник (красный).
- Посередине – первая фаза – отсечение по прямой $x = b$
- Справа – вторая фаза – отсечение по прямой $y = c$



Отсечение (clipping)

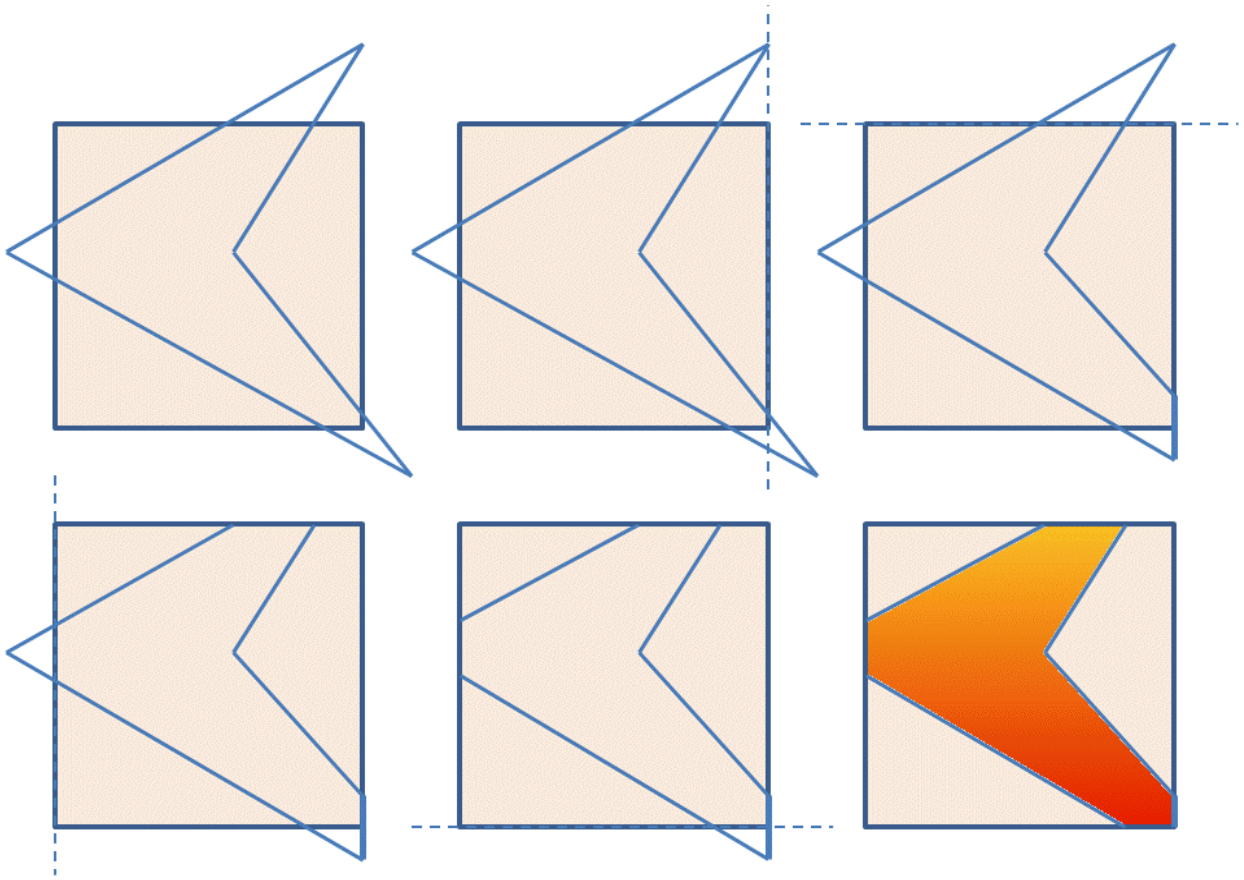
Алгоритм Сазерленда-Ходжмана

- И аналогично последние две фазы: x = a и y = d.
- Таким образом, на каждом этапе производится отсечение только одним ограничением – прямой, на которой лежит отсекающая сторона. На следующий этап передается скорректированная последовательность вершин.



Отсечение (clipping)

Алгоритм Сазерленда-Ходжмана



Отсечение (clipping)

Алгоритм Сазерленда-Ходжмана

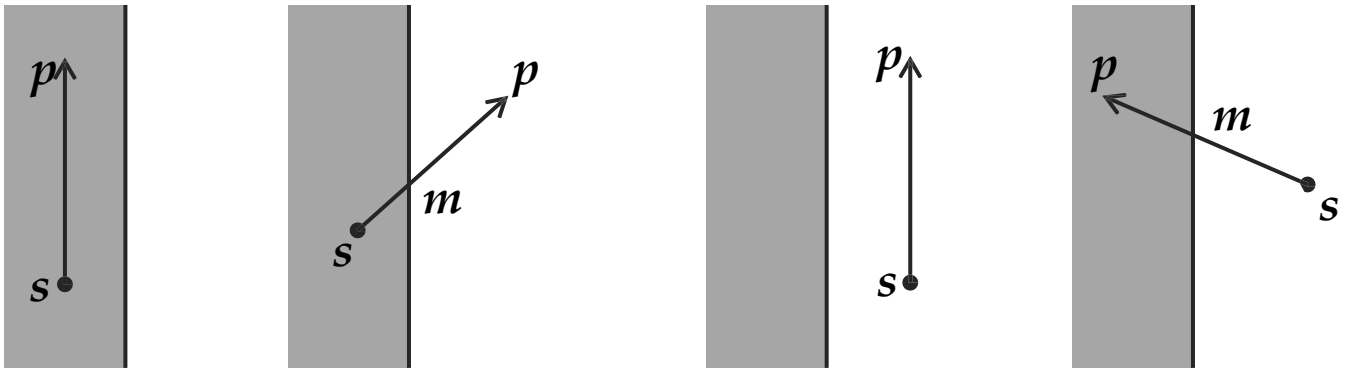
Клиппирование пути пера (или границы многоугольника M) выпуклым многоугольником-окном Э. Пусть путь (polyline) задан вершинами $v(0)\ldots v(n-1)$ и состоит из отрезков $[v(j), v(j+1)], j = 0\ldots(n-2)$ и $[v(n-1), v(0)]$.

Особенность алгоритма в том, что он обрабатывает последовательность вершин. Но на каждом этапе рассматривается отсечение *ориентированного* "текущего" отрезка. Таким образом, все сводится к отсечению отрезка. Рассмотрим отрезок $[s, p]$ - [начало, конец]. Возможны 4 случая (см. рис.). На рисунке точки области (внутренности окна) закрашены.

Отсечение (clipping)

Алгоритм Сазерленда-Ходжмана

- Вершины $v(0) \ldots v(n-1)$
- Отрезки $[v(j), v(j+1)]$, $j = 0 \ldots (n-2)$ и $[v(n-1), v(0)]$.
- Рассмотрим отрезок $[s, p]$ – [начало, конец]. Возможны четыре случая. На рисунке точки области (внутренности окна) закрашены.

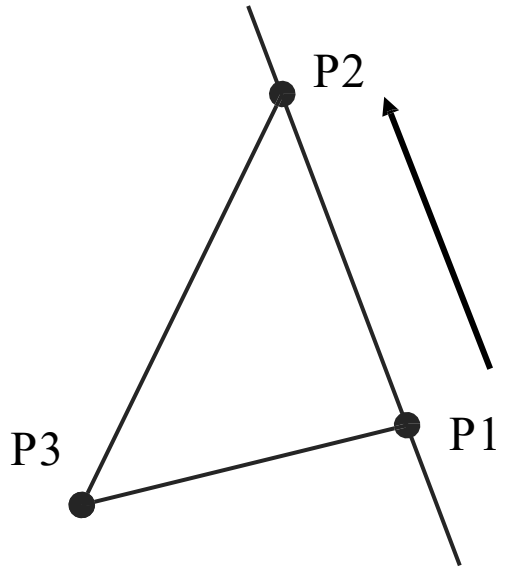


a) b) c) d)

Отсечение (clipping)

Определение взаиморасположения прямой и отрезка

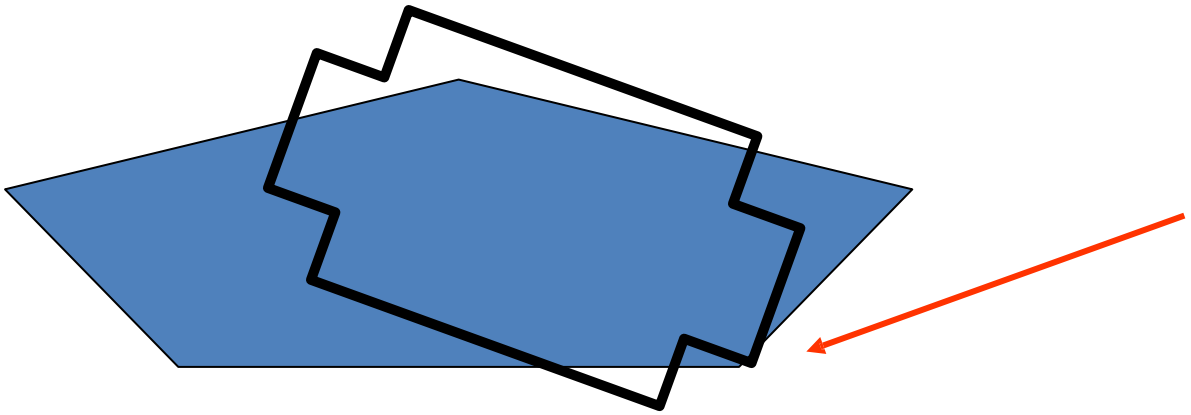
- **Например, подстановкой точки в уравнение прямой**
- Другой способ: на прямой берем две точки P1 и P2 (порядок важен), надо определить, где лежит точка P3. Вычисляем векторное произведение P1P2×P1P3 и смотрим z-координату = $(x3-x1)(y2-y1)-(y3-y1)(x2-x1)$.
- $z > 0$ – P3 лежит справа
- $z = 0$ – P3 лежит на прямой (бывает ли?)
- $z < 0$ – P3 лежит слева



Отсечение (clipping)

Алгоритм Сазерленда-Ходжмана

- Вершины $v(0)\ldots v(n-1)$
- Отрезки $[v(j), v(j+1)]$, $j = 0\ldots(n-2)$ и $[v(n-1), v(0)]$.
- Отсекаемый многоугольник М (не ломаная), отсекающий – выпуклый многоугольник Э
- Делаем цикл по ограничениям – сторонам Э. Пусть это К-ая сторона Э. Для всех $j=0,\ldots, n-1$ выполняем следующие действия
- Шаг 1. Подаем $v(j)$ в качестве s и $v(j+1)$ – p. Добавляем замыкающий отрезок: если $j = n-1$, то $p = v(0)$.



Отсечение (clipping)

Алгоритм Сазерленда-Ходжмана

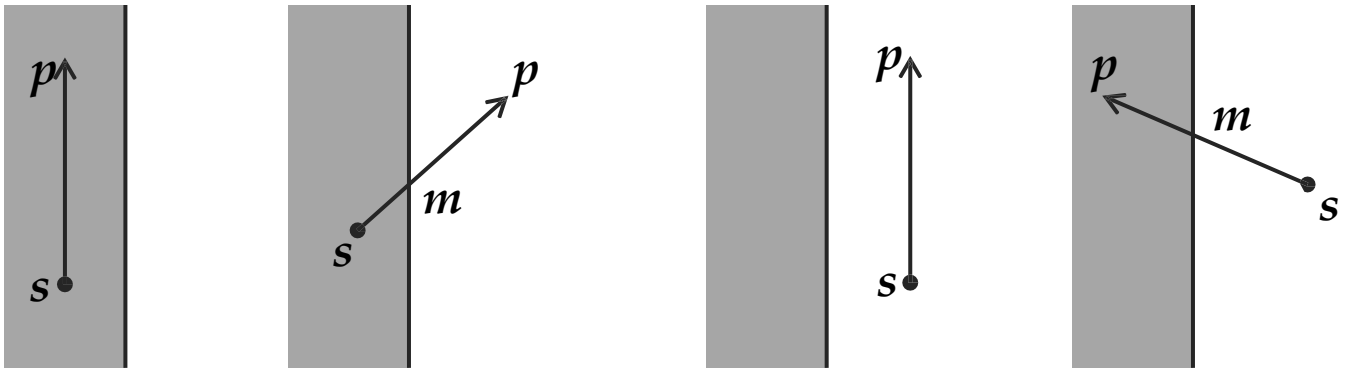
Шаг 2. Определяем случай для текущего ограничения К:

a) выдаем на выход только p, т.к. s была занесена на предыдущем шаге.

b) определяем m, выдаем на выход m.

c) никаких действий (уменьшили число вершин М).

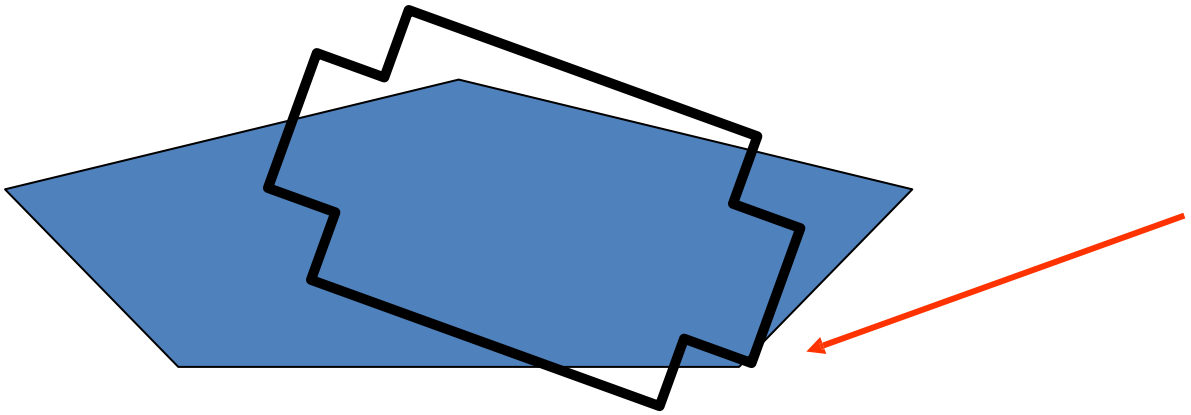
d) определяем m, выдаем на выход $[m, p]$ – 2 вершины. Здесь мы увеличили число вершин М.



Отсечение (clipping)

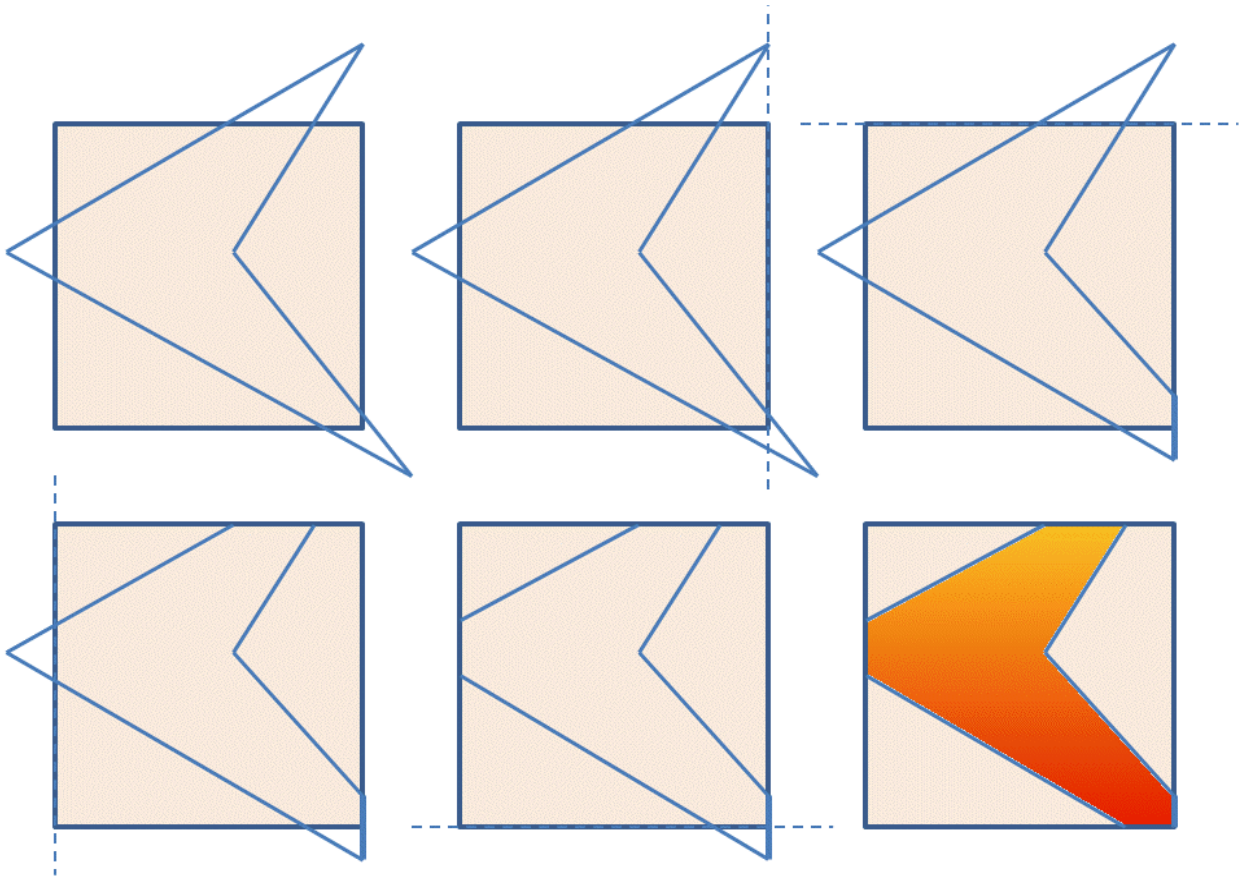
Алгоритм Сазерленда-Ходжмана

- Вершины $v(0)\ldots v(n-1)$
- Отрезки $[v(j), v(j+1)]$, $j = 0\ldots(n-2)$ и $[v(n-1), v(0)]$.
- Шаг 3. $j = j + 1$. Если $j < n$, то идем на шаг 1, иначе идем далее.
- Переходим к следующему ограничению, начиная с шага 1. Здесь на входе в качестве M рассматриваем вершины, полученные на выходе из отсечения по предыдущему ограничению. Если ограничение последнее, то получен результат.
- Конец алгоритма. *Кстати, он появился не раньше 1976 года.*



Отсечение (clipping)

Алгоритм Сазерленда-Ходжмана

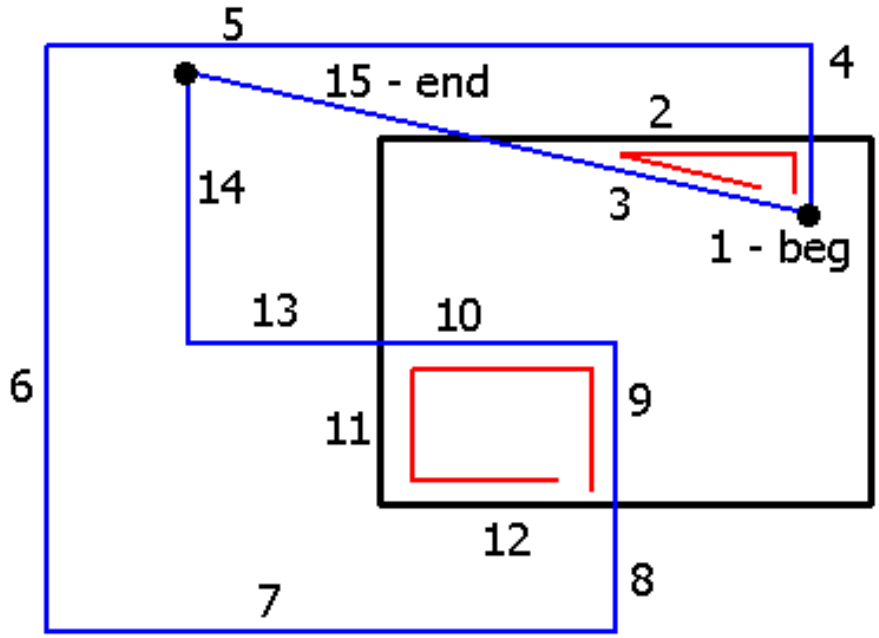


Отсечение (clipping)

Алгоритм Вейлера-Азертона

- Отсечение невыпуклого М по невыпуклому Э. М отсекается по границам Э, путем отслеживания в направлении против часовой стрелки границы М, вплоть до ее пересечения с Э. Правила очень простые:
- Если ребро М входит в Э, то обход идет *вдоль* ребра М.
- Если ребро М выходит из Э, то производим поворот *налево* и идем *вдоль* ребер Э.

Внимание: здесь рассматривается алгоритм для многоугольников без дыр



Хоть на приведенном примере Э выпуклый, мы этим его свойством не воспользовались.

Замечание. В случае другого обхода границы все правила записываются наоборот: *налево* заменяется на *направо*.

Отсечение (clipping)

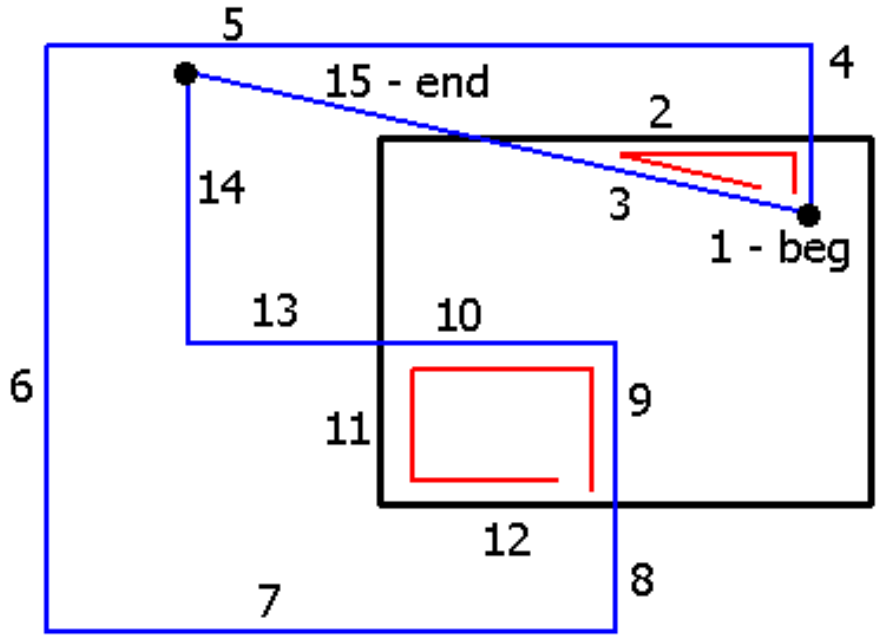
Алгоритм Вейлера-Азертона

Предполагается, что каждый из многоугольников задан списком вершин, причем таким образом, что при движении по списку вершин в порядке их задания внутренняя область многоугольника находится слева от границы.

В случае пересечения границ и отсекаемого многоугольника и окна возникают точки двух типов:

•входные точки, когда ориентированное ребро отсекаемого многоугольника входит в окно,

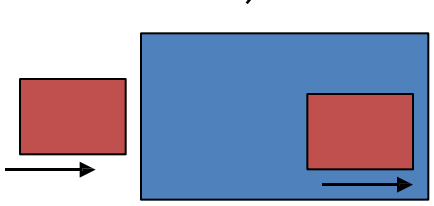
•выходные точки, когда ребро отсекаемого многоугольника идет с внутренней на внешнюю стороны окна.

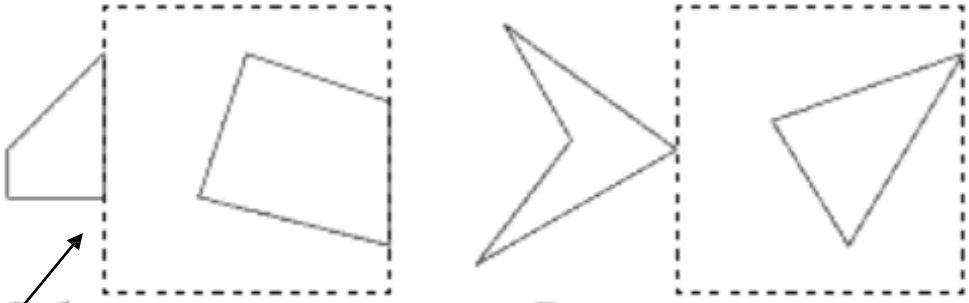


Отсечение (clipping)

Алгоритм Вейлера-Азертона

1. Строятся списки вершин отсекаемого многоугольника и окна.
2. Отыскиваются все точки пересечения. При этом расчете касания не считаются пересечением, т.е. когда вершина или ребро отсекаемого многоугольника инцидентна или совпадает со стороной окна

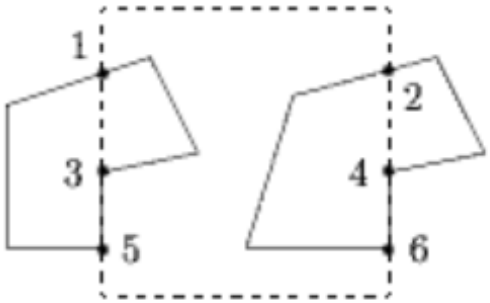




Ребро отсекаемого многоугольника совпадает с окном.

Вершина отсекаемого многоугольника касается окна.

Случаи не считающиеся пересечением.



Точки 1–4 считаются пересечением.

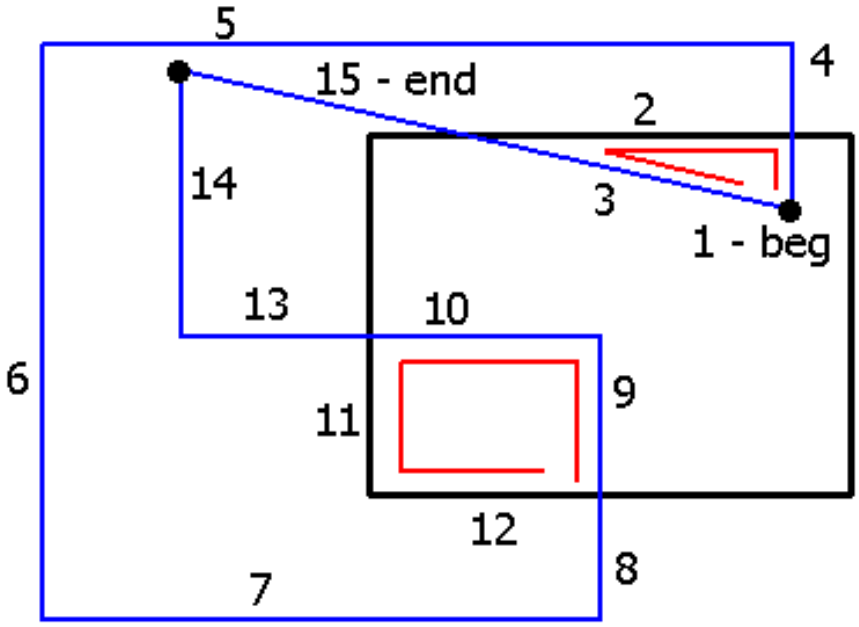
Точки 5,6 не считаются пересечением.

Частные случаи пересечения.

Отсечение (clipping)

Алгоритм Вейлера-Азертона

Списки координат вершин отсекаемого многоугольника и окна дополняются новыми вершинами - координатами точек пересечения. Причем если точка пересечения Pk находится на ребре, соединяющем вершины Vi, Vj, то последовательность точек Vi, Vj превращается в последовательность Vi, Pk, Vj. При этом устанавливаются двухсторонние связи между одноименными точками пересечения в списках вершин отсекаемого многоугольника и окна. Входные и выходные точки пересечения образуют отдельные подсписки входных и выходных точек в списках вершин.



Отсечение (clipping)

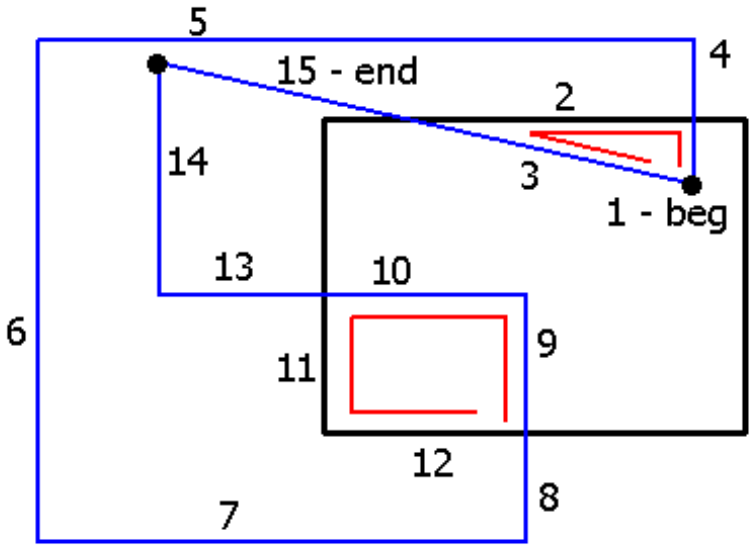
Алгоритм Вейлера-Азертона

Если не исчерпан список входных точек пересечения, то выбираем очередную входную точку.

Двигаемся по вершинам отсекаемого многоугольника пока не обнаружится следующая точка пересечения; все пройденные точки, не включая прервавшую просмотр, заносим в результат; используя двухстороннюю связь точек пересечения, переключаемся на список вершин окна.

Двигаемся по вершинам окна до обнаружения следующей точки пересечения; все пройденные точки, не включая последнюю, прервавшую просмотр, заносим в результат. Используя двухстороннюю связь точек пересечения, переключаемся на список вершин многоугольника.

Эти действия повторяем пока не будет достигнута исходная вершина - очередная часть отсекаемого многоугольника, попавшая в окно, замкнулась. Переходим на выбор следующей входной точки в списке отсекаемого многоугольника.

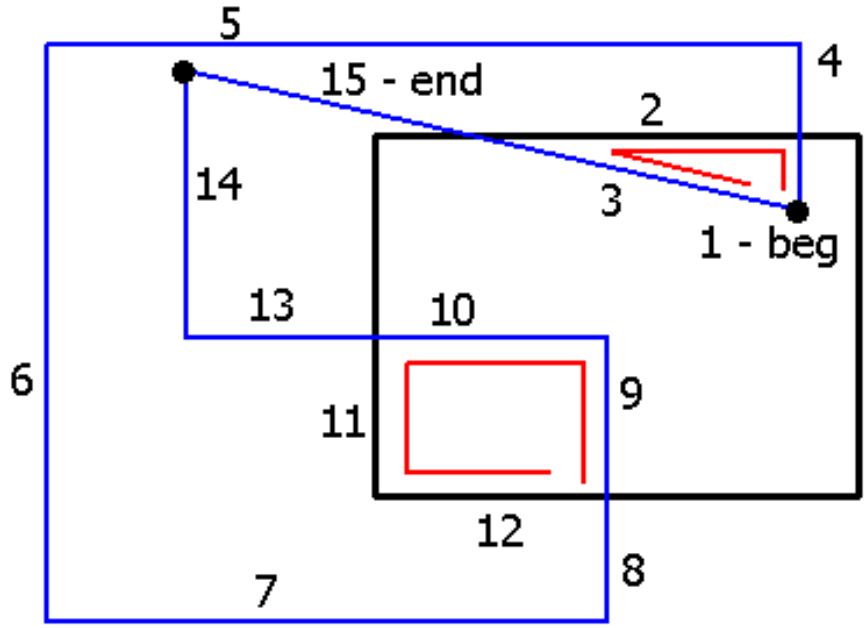


Отсечение (clipping)

Алгоритм Вейлера-Азертона

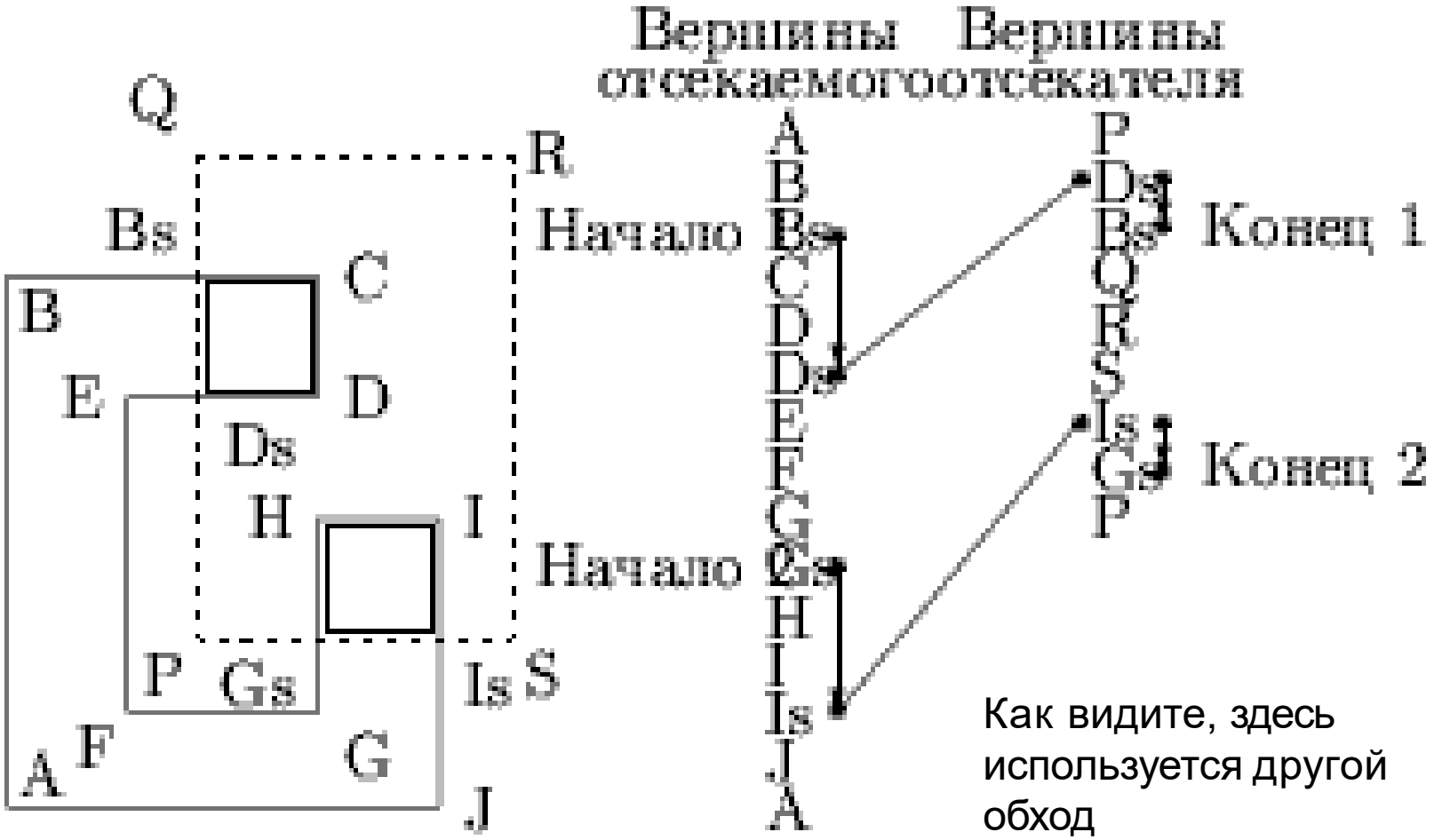
Модификация этого алгоритма для определения части отсекаемого многоугольника, находящейся вне окна, заключается в следующем:

- исходная точка пересечения берется из списка выходных точек;
- движение по списку вершин окна выполняется в обратном порядке, т.е. так чтобы внутренняя часть отсекателя была слева.



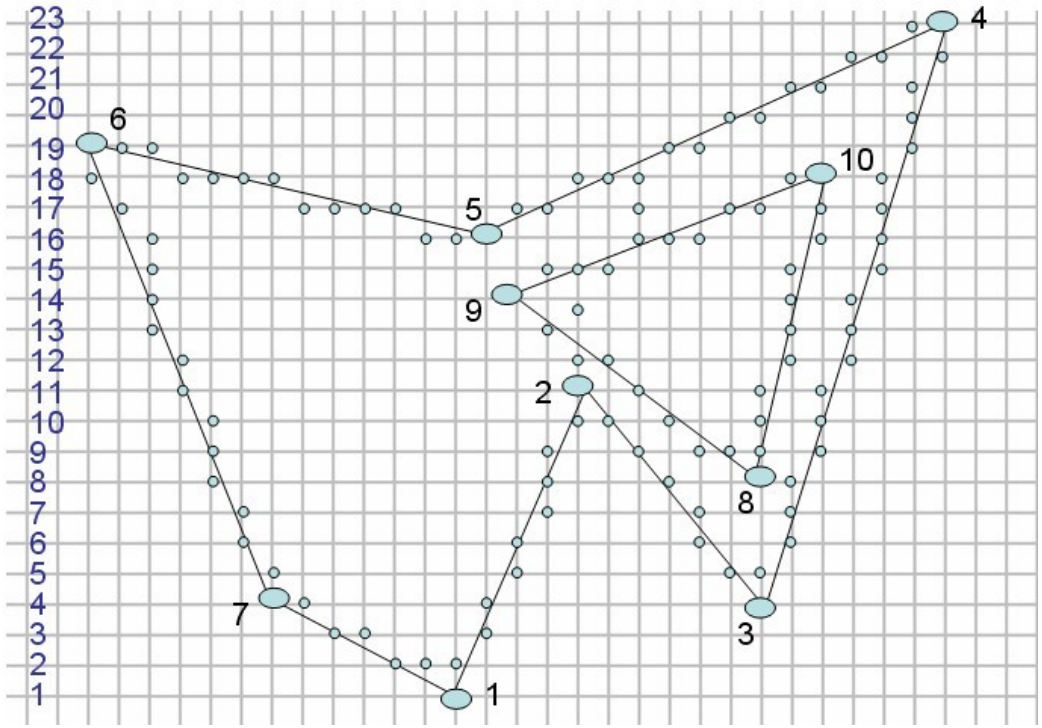
Отсечение (clipping)

Алгоритм Вейлера-Азертона



Как видите, здесь используется другой обход

Растеризация М
Использование заливки

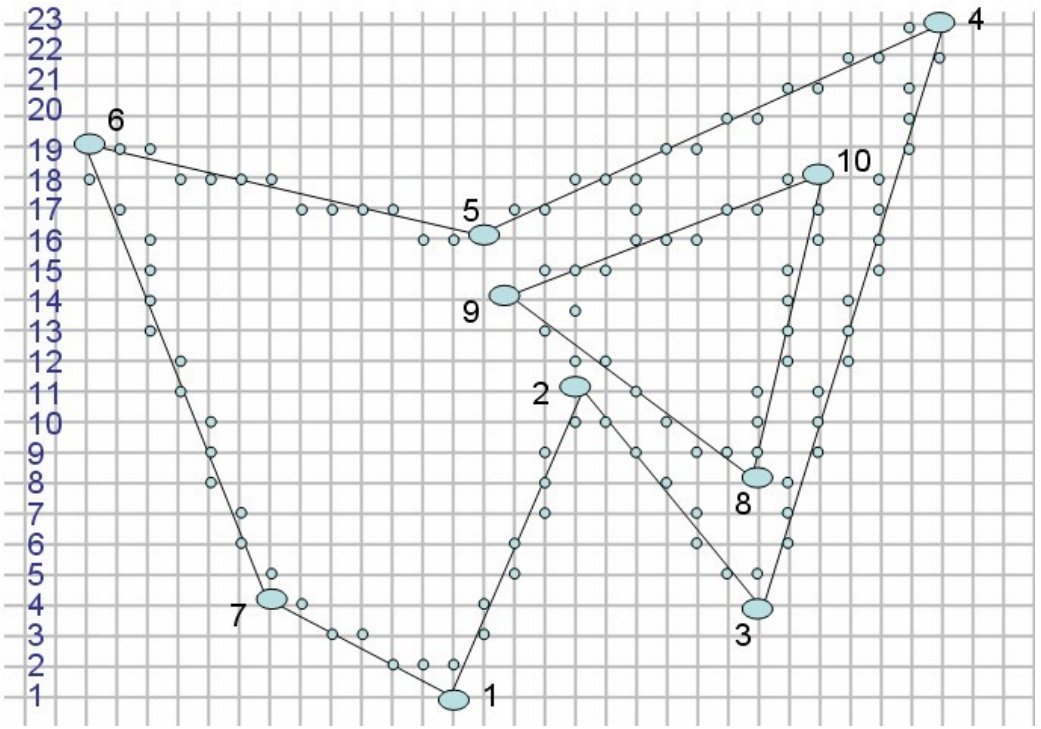


Мелкие точки – это центры пикселей границы.

Слева: номера активных линий растра

1. Растеризуем стороны, например, по Брезенхэму
2. Применяем любой алгоритм заливки
3. Плохо, что нет *затравки*

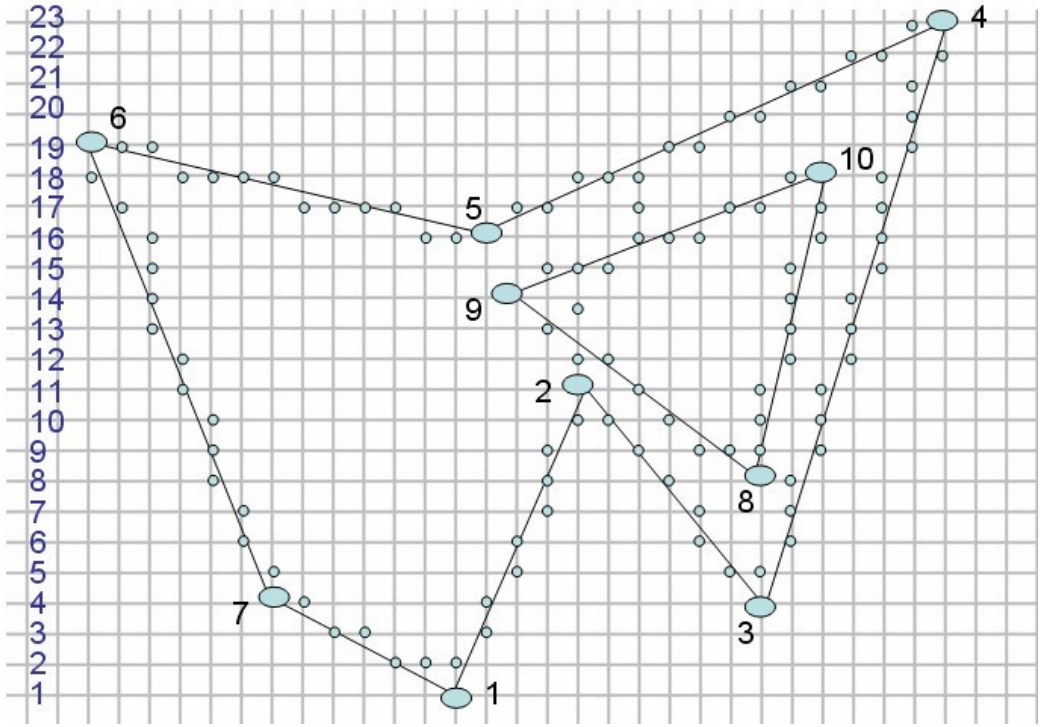
Растеризация M
Алгоритмы, основанные на применении сканирующей линии



1 – 23 – активные линии растра для M

4 – 19 – активные линии растра для ребра (6, 7)

Растеризация М
Список активных линий растра



$$V_1 = (x_1, y_1), \ldots, V_n = (x_n, y_n)$$

порядок сканирования

$$V_i < V_j$$

$$y_i < y_j$$

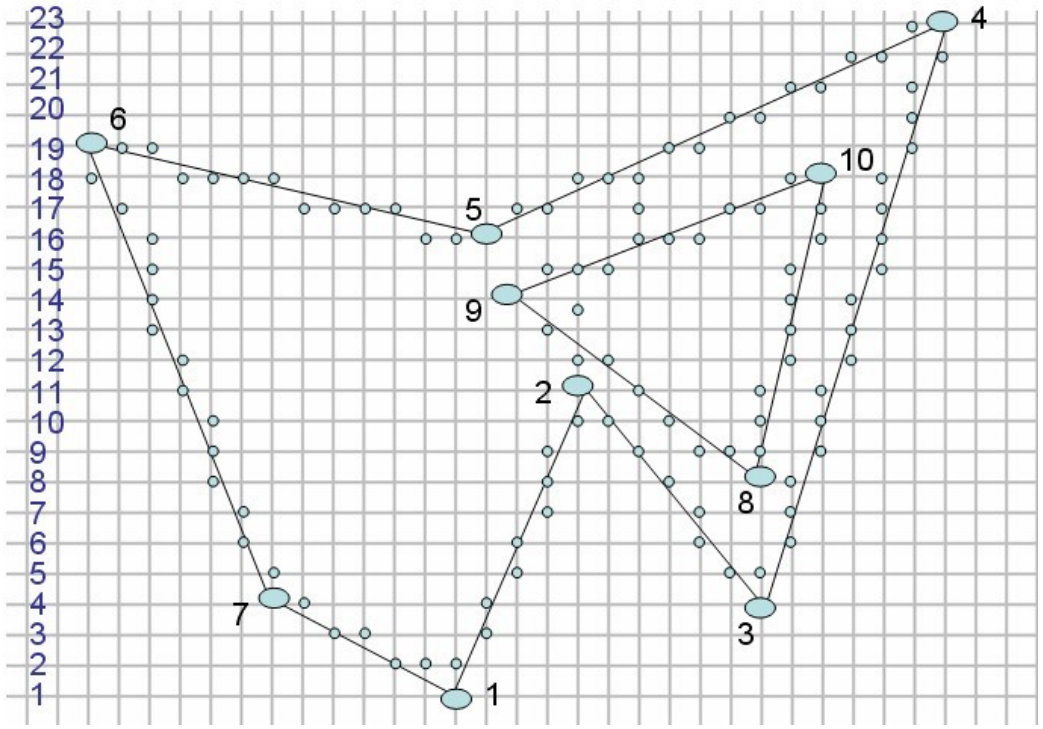
$$y_i = y_j \longrightarrow x_i < x_j$$

Предположения:

Бесконечная прямая пересекает замкнутую ограниченную область *четное* число раз кроме случаев, когда прямая проходит по сторонам или по вершинам.

Мы можем заполнять многоугольник, перебирая последовательные сканирующие линии. При этом заполняются пиксели, расположенные между точками пересечения линии со сторонами.

Растеризация М Алгоритм 1



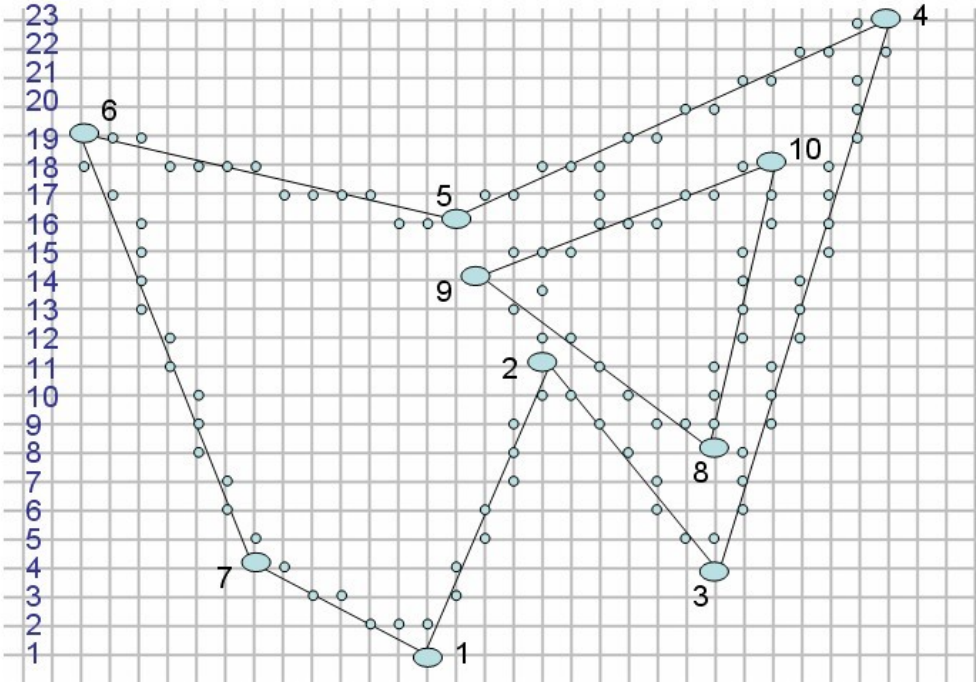
1. Применяем растеризацию сторон и определяем все пиксели, принадлежащие границе .
2. Согласно порядку сканирования сортируем все граничные пиксели и запоминаем их в списке.
3. Выбираем попарно точки из списка и красим каждый пиксель (x, y), удовлетворяющий условию чётности

Исключительные ситуации:

1. Не обрабатывать горизонтальные стороны, $y_1 = y = y_2$ т.к. эти пиксели находятся между двумя вершинами и будут закрашены.
2. Если сканирующая линия попадает на вершину, то: Записываем два пересечения, если эта вершина является локальным максимумом или локальным минимумом;Записываем одно пересечение в противном случае

Если не является локальным минимумом, то производим корректировку: и .

Растеризация М
Список активных рёбер (LAS)



Строится список ключевых пикселей (LKP).

Ключевой пиксель — нижний пиксель всякой негоризонтальной стороны

Для стороны с вершинами

(x_1, y_1) [redacted]

Запоминаем 3 величины:

$$\delta y = y_2 - y_1 + 1 \qquad x = x_1 \qquad \delta x = \frac{x_2 - x_1}{y_2 - y_1}$$

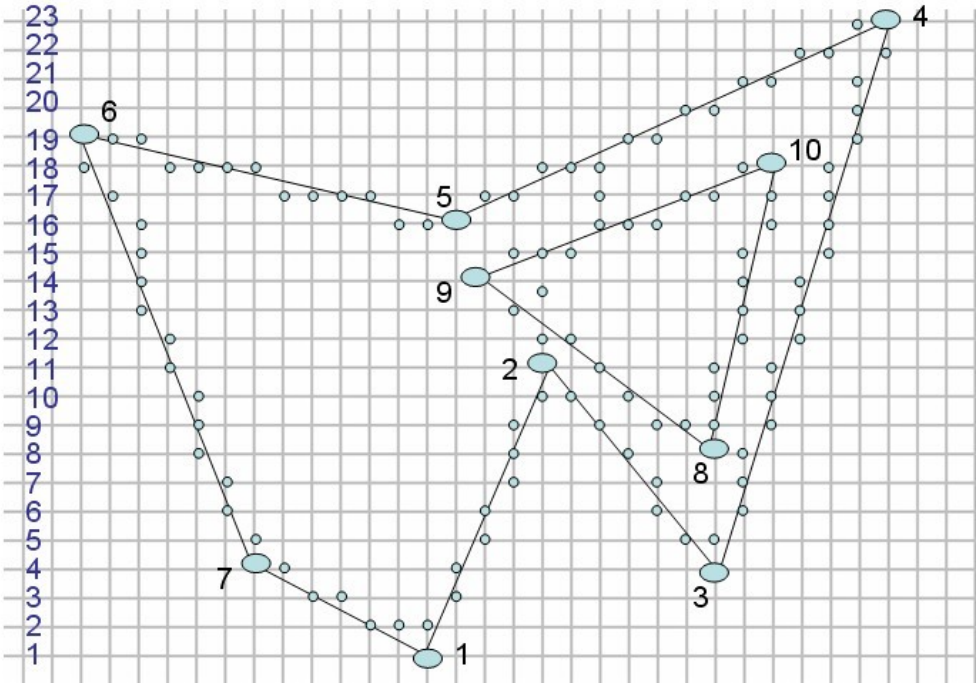
Если (x_1, y_1) не локальный минимум, то $y = y_1 + 1$ и $x = x_1 + \delta x$

Привязываем ключевые рёбра к горизонтальным линиям.

По мере обхода тройки из LKP будут копироваться в LAS

Растеризация М

Список активных ребер, Алгоритм



1. Начинаем с нижней сканирующей линии растра $y = 1$. Выбираем из LKP все тройки запомненных данных для нижней строки растра и помещаем их в LAS.
2. Закрашиваем все точки (пиксели) сканирующей линии между значениями x каждой пары троек из LAS.
3. $\delta y \rightarrow \delta y - 1,\ x \rightarrow x + \delta x$
4. Если $\delta y = 0$, убираем соответствующую тройку данных из списка активных сторон.
5. $y = y + 1$
6. Просматриваем начало LKP и, если для строки в нем имеются тройки, то мы их добавляем в LAS и сортируем его согласно порядку (такая ситуация встретится нам на линиях 4, 8, 14, 16 в примере).
7. Повторяем шаги $2 - 6$ до тех пор, пока список LAS не пуст