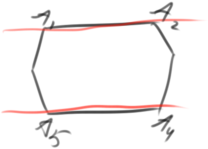
26.03.14-----------------------------------------------------------------------

Вторая проблема – с горизонтальными рёбрами. Ребро лежит на сканирующей строке – огромное число «пересечений. Решение – выкинуть ребро. Тогда сканирующая строка всёравно будет пересекаться с А1.. но уже как с концами других рёбер – это не значит, что они не будут закрашены.

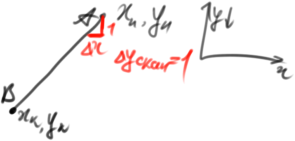
Два тонких момента – горизонтальные ребра (можно не рассматривать) и учет сканирующих строк, проходящих ровно через вершины многоугольника.

Рассмотрим конкретные алгоритмы.

**Алгоритм заполнения с упорядоченным списком ребер**

Активное ребро – ребро, которое в данный момент рассматривается. Ребро, пересекаемое текущей сканирующей строкой. Список активных ребер – список, хранящий информацию об активных рёбрах.

Алгоритм для нахождения точек персечения (прошлая лекция) использовать нельзя – отрезок представлен в виде совокупности ступенек, а значит вместо одной точки пересечения может получиться несколько, и результат заполнения будет определяться четностью\нечетностью количества точек. Необходимо сочетать два подхода – разлагая отрезок в растр надо получать одну точку пересечения для сканирующей строки. – информация об отрезке.



dX – фактически котангенс угла наклона. . . Когда dY <0 – отрезок обработан.

Простой алгоритм:

1. найти пересечение всех отрезков со всеми сканирующими строками (х1,у1), ... (хк,ук).

2. найденные точки персечения упорядочиваем по убыванию координаты У

3. сортировка точек пересечения, принадлежащих одной (каждой) сканирующей строке по возрастанию Х. (х1,у)...(хн,у), х1<=х2<=...хн

4. упорядоченный список для каждой сканирующей строки разбить на пары и закрасить все пиксели, расположенные в интервале, ограниченном очередной парой.

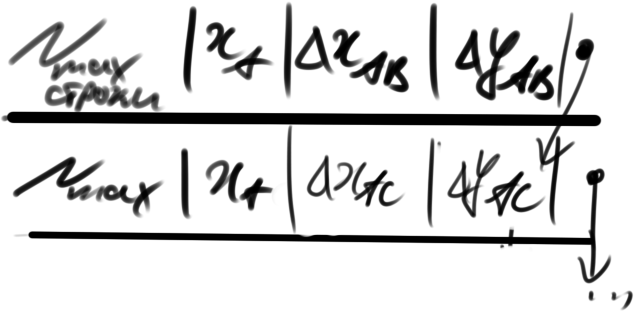
Более эффективный вариант – необходимо избавиться от сортировки всего множества точек пересечения. Для хранения точек в простейшем случае можно использовать массив, и разбить его на n участков, по количеству сканирующих строк.

1. находимые точки пересечения сразу заносятся в нужную группу

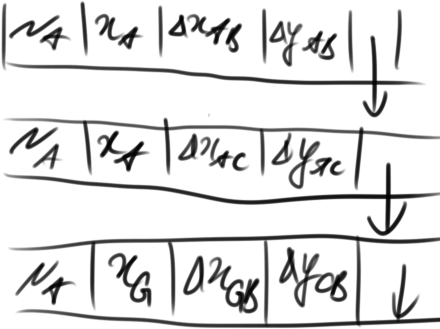
2. упорядочить содержимое каждой у-группе по возрастанию Х

3. образовать пары внутри каждой у-группы и закрасить все пиксели в интервалах.

При этом использовать массивы крайне невыгодно – при статике будет перерасход памяти, при динамике требуется копирование элементов при перевыделении. Можно использовать однонаправленные списки.



Информацию о ребрах в список надо заносить в порядке убывания максимального значения координаты У ребра.

AB, AC, GB, GF, CD, EF, ED. 

Если N текущее > N первого элемента списка, то список активных ребер пуст, многоугольник «ещё не начался». Если N текущее <= N первого элемента списка, то строка пересекает многоугольник (САР не пуст).

Список просматривается до тех пор, пока выполняется условие N тек. <= N очередного элемента списка. При просмотре списка  
1) выбираем Xij абсциссы точек с текущей строкой.  
2) корректируем поля элемента списка: . Если dy<0, то элемент списка удаляется из списка. Удалив обработанное ребро, избавляемся от необходимости просмотра уже просмотренных ребер.  
3) обработка ведется пока список не опустеет

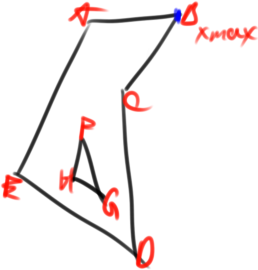
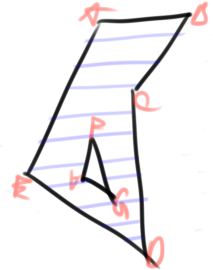
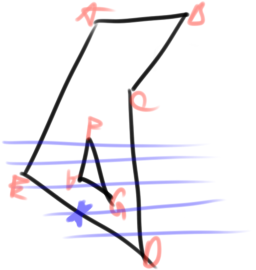
Сформированный список абсцисс точек пересечения ребер с очередной сканирующей строкой надо опять отсортировать по возрастанию и закрасить пиксели, расположенные внутри каждого интервала.

Алгоритм заполнения принято оценивать количеством операций, связанных с изменением цвета пикселя (и получения информации о его цвете), и самим количеством пикселей. В данном алгоритме, обрабатываются только те пиксели, которые лежат внутри области, И каждый пиксель обрабатывается по одному разу. Следовательно, с точки зрения отрисовки данный алгоритм является одним из наиболее быстродействующих. Тем не менее, для некоторых фигур алгоритм становится значительно более накладным («расческа»).

**Алгоритм заполнения по ребрам**

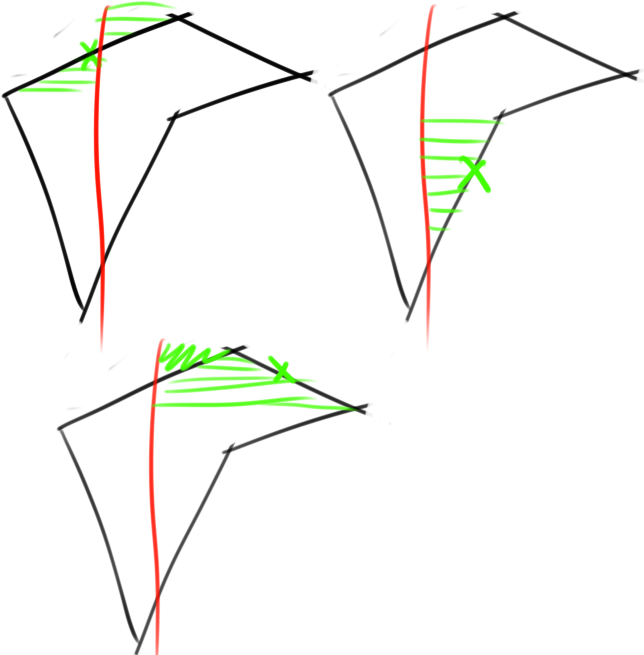
Ребра никак не упорядочиваются, сортировка не происходит. В этом алгоритме ребра могут обрабатываться в произвольном порядке. Алгоритм прост, но требует бОльших временных затрат, поскольку некоторые пиксели могут обрабатываться несколько раз.

Задается цвет фона и цвет закраски; цвет закраски = !(цвет фона), цвет фона = !(цвет закраски) (инверсия/дополнение). Суть алгоритма: для каждой сканирующей строки, пересекающей ребро многоугольника, дополнить (проинвертировать) все пиксели, расположенные правее точки пересечения.



Количество обработок пикселя определяется количеством ребер, расположенных левее этого пикселя.

Алгоритм заполнения с перегородкой: для каждой сканирующей строки пересекающей ребро, дополнить все пиксели, расположенные правее точки пересечения, но левее перегородки (если х < х\_пер). Затем дополнить все пиксели, расположенные левее точки пересечения, но правее перегородки (если х > х\_пер).



**Алгоритм со списком ребер и флагом**

Флаг – признак принадлежности точки внутренней или внешней области многоугольника. =true , если точка расположена внутри. Алгоритм двушаговый:

1) очертить границы закрашиваемого многоугольника

2) закрасить ограниченный многоугольник