02.04.14-----------------------------------------------------------------------

**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

Цикл по всем сканирующим строкам (по у от Ymax до Ymin)

Flag = false

Цикл по всем пикселям сканирующей строки (по х от 0 до Xmax)

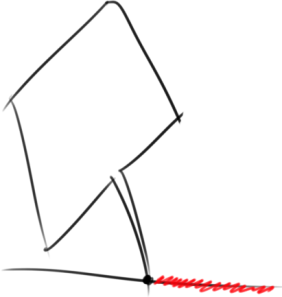
Если цвет(х,у) = цвет границ, то flag=!flag; продолжаем

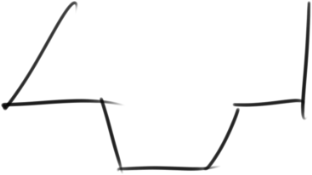
Если флаг=истина, то цвет(х,у) = цвет закраски; иначе цвет(х,у) = цвет фона

Конец цикла по Х

Конец цикла по У

Возникающие проблемы:

На стадии отрисовки границ, два ребра могут сливаться в одно. Как вариант – отрисовывать границу с использованием инверсного цвета пера. Либо же, первое ребро отрисовывается, при отрисовке последующих – если точку надо поставить в ту точку, которая уже подсвечена, то поставить её рядом.



Также извечный вопрос строки, проходящей через вершины.

Также, использовать обычный способ отрисовки отрезков нельзя – может возникнуть разбиение; точек пересечения со сканирующей строкой может оказаться несколько.

Скорость работы алгоритма закраски определяется количеством операций обращений к пикселю. И первый и последний алгоритмы обрабатывают каждый пиксель единожды, НО – нужно и смотреть, сколько обрабатывается пикселей. В некоторых алгоритмах анализируются пиксели, лежащие не только внутри обрабатываемой области.

**Алгоритмы затравочного заполнения**

1.должна быть задана область, подлежащая заполнению (гранично-определённая, или внутренне-определённая)

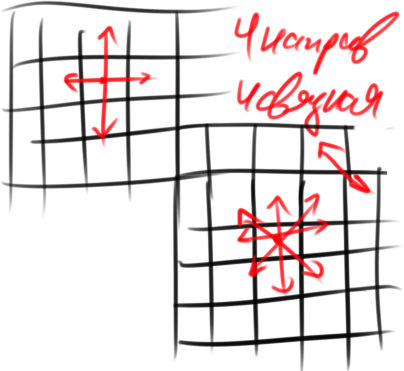
2.должен быть задан затравочный пиксель – точка, заведомо лежащая внутри области

Способы задания области.

1. Гранично-определённая => задана граница области, т.е. известен цвет границы



2. Внутренне-определённая область => все пиксели, принадлежащие области, имеют один и тот же цвет – этот цвет должен быть известен. Закрашиваемые области могут быть четырёхсвязными и восьмисвязными.Связность можно определить количеством направлений движения для достижения точки области.

Граница является 8связной; у 8связной граница является 4связной.

Алгоритмы можно разделить на алгоритмы для гранично-определённых областей, и внутренне-определённых областей. Также для 4связных областей и для 8связных областей, причем 8связный алгоритм справляется и с 4связной.

**Простой алгоритм заполнения с затравкой**

В алгоритмах заполнения очень удобно и эффективно использовать стек для хранения затравочных пикселей.



3(x,y); (x,y+1); (x+1,y); (x,y-1); (x-1; y)

1.Задаются граница, цвет границы, координаты затравочного пикселя

2.занесение затравочного пикселя в стек

3.пока стек не пуст, выполнить следующие действия:

- извлечь пиксель из стека (координаты)

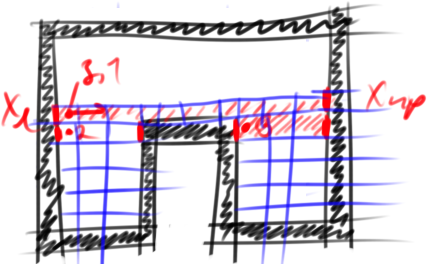
- закрасить извлеченный пиксель (возможно, с проверкой на цвет – нужно ли закрашивать или он уже)

- проанализировать цвет 4-х соседних пикселей ^. Если пиксель не является граничным, то поместить его в стек

Алгоритм достаточно прост в реализации и понимании. Недостатки: перерасход памяти – в стек заносится любой незакрашенный неграничный пиксель, т.е. в стеке побывает каждый пиксель области. В наихудшем случае, пиксель может быть занесён в стек аж четыре раза.

**Построчный алгоритм заполнения с затравкой**

Для улучшения характеристик алгоритма, в стек надо заносить не каждый незакрашенный пиксель, а один на группу незакрашенных. Под группой будем понимать пиксели в одной строке, образующие непрерывный интервал. Это группа примыкающих друг к другу пикселей, ещё не закрашенных, ограниченных уже закрашенными\граничными пикселями. В стек заносится самый левый или самый правый пиксель.

Анализируем пиксель – закрашиваем – анализируем пиксели сверху и снизу. Каждый пиксель анализируется, а затем может быть рассмотрен ещё как и верхняя и нижняя граница – рассматривается три раза.

1. Если стек пуст
2. то конец
3. иначе: извлечение пикселя (х,у) из стека
4. цвет(х,у) = цвет закраски
5. tx = x, запоминаем абсицссу  
   //два однотипных куска, обеспечивают закраску пикселей слева и справа, с запоминанием границы
6. двигаемся влево, x=x-1
7. если цвет(х,у) != цвету границы
8. цвет(х,у) = цвет закраски; гото 6
9. иначе: Xleft = X+1
10. x=tx
11. двигаемся вправо, x=x+1
12. если цвет(х,у) != цвету границы
13. цвет(х,у) = цвет закраски; гото 11
14. иначе: Xright = X-1  
    //два однотипных куска, обеспечивают закраску пикселей слева и справа, с запоминанием границы
15. x = Xleft
16. y = y+1 рассматриваем строку сверху
17. если x <= Xright (иначе гото
18. то Fl=0 //признак нахождения
19. Если цвет(х,у) != цвету закраски (иначе гото 25)
20. То если цвет(х,у) != цвету границы (иначе гото 25)
21. То если х<=хпр (иначе гото 25) (иначе гото 25)
22. То если Fl=0
23. То FL=1
24. Х=Х+1; гото 19

№ по журналу mod 4:

1 -> с упорядоченным списком

2 -> по ребрам

3 -> с перегородкой

0 -> с флагом

Реализовать алгоритм: нарисовать область; закрасить