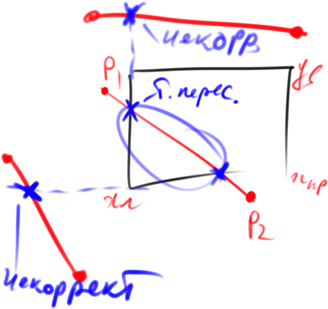
16.04.14-----------------------------------------------------------------------

Простой алгоритм отсечения:



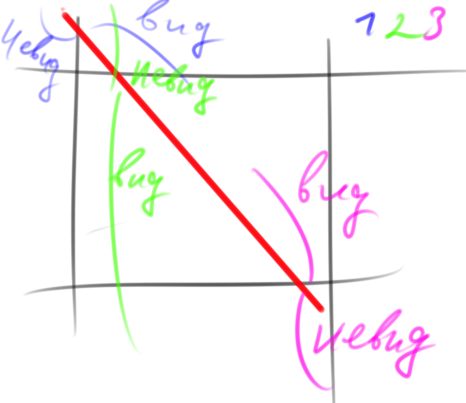
Точка пересечения проверяется на корректность.  
Изначально флаг установлен Flag=0;

1. Ввод исходных данных: хл, хпр, ун, ув; р1,р2
2. Вычисление Т1,Т2 (кодов концов отрезка)
3. Проверка полной видимости (S1=0 и S2=0; )
4. Проверка полей невидимости (  
   //к 5 пункту приходим с частично видимым отрезком
5. Проверка видимости 1й вершины: Если С1=0, то Р1 заносится в результат; i=1 (остается найти вторую вершину – будь то Р2 или точка пересечения отрезка с границей)
6. Проверка видимости 2й вершины: если С2=0, то Р2 заносится в результат (аналогично)
7. Поиск очередной точки пересечения (i-й)
   1. Определение наличия пересечения отрезка с левой границей (Pi.x < Xл)
   2. Нахождение точки пересечения (ординаты) yt= m\*(Pi.x – Xл) + Pi.y
   3. Проверка корректности пересечения Yн<= yt <=Yк (точка пересечения отрезка с границей лежит за пределами отсекателя); занесение точки в результат. i=i+1; повторяем пункт 7 если i<3
   4. Определение наличия пересечения отрезка с правой границей (Pi.x > Xпр)
   5. Нахождения точки пересечения yt = m\*(Pi.x – Xпр) + Pi.y
   6. Проверка корректности; занесение точки в результат.
   7. Определение наличия пересечения отрезка с нижней границей (Pi.y < Yн)
   8. Нахождение точки пересечения xt = (Pi.y-Yн)/m + Pi.x
   9. Проверка корректности пересечения Xл<= xt <=Xпр; занесение точки в результат при корректности; i=i+1; поиск i-го пересечения (i<3)
   10. Определение наличия пересечения с верхней границей
   11. Нахождение пересечения
   12. Проверка пересечения на корректность; точка заносится в результат.  
       //в 7.13 приходим если ни одного пересечения не найдено
   13. Признак видимости отрезка устанавливается Flag=-1.
8. Если Flag==0 то высвечиваем видимую часть отрезка по результату. Если Flag==-1, то отрезок является невидимым.

Недостаток алгоритма – мы ищем точку пересечения, потом проверяем на корректность; если точка некорректна то вычисления выполнялись зря.

**Алгоритм Сазерленда-Коэна.**

Алгоритм отсечения, основанный на разбиении отрезка сторонами отсекателя. При работе с отрезок, оный разбивается на два – на отрезок, лежащий в невидимой области, и отрезок, лежащий в видимой.



Найдя точку пересечения торезка со стороной, необходимо отбросить невидимую часть. Невидимые отрезки обычно определяются с помощью логического произведения кодов концов. Однако если мы искали точку пересечения с очередной границей, то мы её искали для случая когда она действительно существует;отрезок может закончиться раньше чем дойдёт до пересечения. Значит есть вершина, лежащая в пределах отскателя.

Чтобы легко можно было распознать невидимую часть отрезка, нужно при определении пересечения определить, какая из двух вершин невидимая. Когда точка пересечения найдена, то невидимая вершина де-факто перемещается в эту точку.

Алгоритм схемой:

1. начало //заводим "окно" с четыремя элементами для T#i - левое правое нижнее верхнее; O(Хл, Хпр, Yн, Yв)

2. ввод исходных данных P1,P2

3. формирование признака расположения отрезка: Fl=-1 вертикальный, 0 горизонтальный, 1 общего положения

4. вычисление тангенса угла наклона m (для невертикальных отрезков); m= (P2.y - P1.y)/(P2.x - P1.x)

5. цикл отсечения отрезка по четырём границам; i=1..4

6. определение Т1,Т2, S1,S2, произведений Pr (логическое произведение кодов концов: Т11Т21 + Т12Т22 + Т13Т23 + Т14Т24) и признака видимости Vis; Vis= -1 отрезок невидимый, 1 видимый, 0 частично видимый

7. Если

Vid=-1 :

8. Конец

Vid=1 :

9. Высвечивание отрезка; гото 8

Vid=0 :

10. Если

T1i = T2i : //в этой точке алгоритма, одноименные разряды кодов могут равняться только 0 для того чтобы алгоритм продолжался

11. Конец цикла отсечения (по i) 5; гото 9.

иначе :

12. Если T2i=1 :

13. t = P1; P1 = P2; P2 = t; //меняем вершины местами

14. Если Fl=-1 : //в данной точке алгоритма, если отрезок вертикальный, то он как минимум частично выдимый.

17. P1.y = Oi; гото А //"О"кно ^

Иначе:

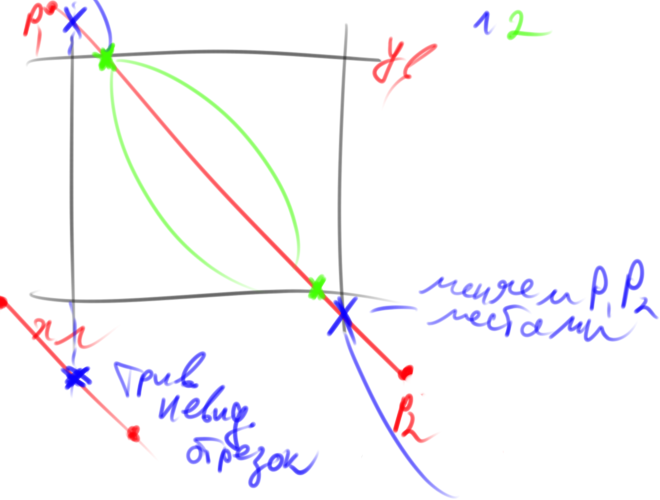
15. Если i<=2

16. P1.y = m\*(Oi - Pi.x) + Pi.y; P1.x = Oi;

(A) гото 11 //к хвосту цикла - проверить нужно ли искать остальные точки, или выйти из цикла

иначе : //на 3 или 4 шагах, и отрезок невертикальный - ищем нижнюю\верхнюю границу

18. Р1.х = (Oi - Pi.y)/m + P1.x; гото 17



**Алгоритм отсечения средней точкой**

Основное отличие – точка пересечения находится не аналитически. Уравнение, фактически, решается численно – методом деления отрезка пополам. Деление на 2 – сдвиг, выполняется достаточно быстро, посему алгоритм имеет право на существование. Точность или . Нахождение точек пересечения отрезка со стороной отсекателя.