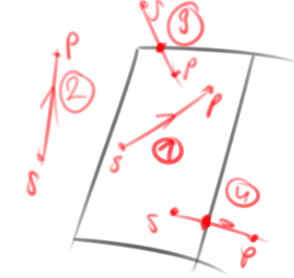
07.05.14-----------------------------------------------------------------------

Алгоритм Сазерленда – Ходжмена. Возможные варианты взаимного расположения.



1 обе вершины отрезка S P видимы, отрезок целиком видим. В результат заносятся точки С и П

2 Обе вершины невидимы, отрезок целиком невидим. В результат заносится 0 точек

3 S невидима, P видима, отрезок частично видим. В результат заносится точка пересечения (надо найти) и П.

4 Наоборот. С и точка пересечения.

Алгоритм:

1. определить видимость точки (вершины рёбер)

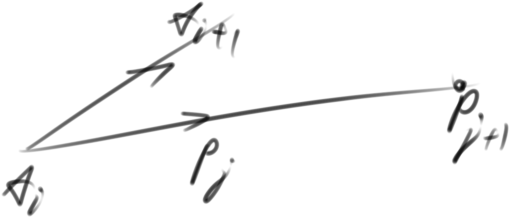
а) использовать скалярное произведение PjAi \* n, >=0 точка видима, <0 невидима



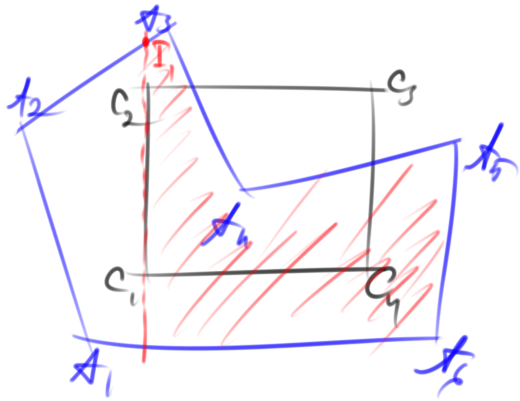
б) использовать пробную функцию (на основе уравнения прямой) F =ax + by + c



в) использование векторного произведения AiPj \* AiAi+1, >=0 точка Pj видима



2. находить точки пересечения



С – отсекатель, А – многоугольник.

В данном алгоритме рассматривается конкретный вариант нахождения: ищется точка пересечения прямой, проходящей через ребро отсекателя, с ребром отсекаемого многоугольника. Поскольку отрезки имеют произвольное расположение, удобно использовать параметрическую форму записи: P(t) = P1 + (P2-P1)t, где 0<=t<=1 – ребро многоугольника, и Q(s) = Q1 + (Q2-Q1)t – ребро отсекателя.

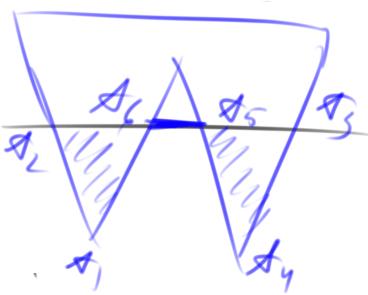
P(t)=Q(s): имеем систему . Предварительно нужно убедиться в непараллельности прямых – точка пересечения должна существовать. Это определяется с помощью видимости концов ребра многоугольника – если видимость разная, то точка пересечения есть.

Начальная вершина очередного ребра является одновременно и конечной вершиной для предыдущего ребра. Эта вершина анализируется (и заносится в результат если видима) на предыдущем шаге.

Данный алгоритм имеет недостаток – можно столкнуться с ситуацией построения «ложных ребер» (I2I3).



Мы работаем с массивом вершин, вершины обходятся последовательно – ложным будет ребро, которое обходится два раза.



Для удобства можно продублировать первую вершину в качестве n+1й для простоты организации цикла отрисовки.

**Алгоритм Вейлера-Азертона**

Наконец рассмотрим алгоритм, позволяющий произвольный многоугольник отсекать произвольным отсекателем. Этот алгоритм позволяет на своей базе построить один из алгоритмов удаления невидимых поверхностей.

Особенности:  
И отсекаемый многоугольник и отсекатель – произвольные многоугольники. Могут быть невыпуклыми и вдобавок могут содержать отверстия.  
Алгоритм позволяет выполнить как внутреннее (рисуем всё что внутри отсекателя) так и внешнее отсечение (рисуем всё что снаружи).  
В результате отсечения получаются многоугольники, ребра которых являются либо ребрами исходного многоугольника, либо ребрами отсекателя – никаких новых рёбер в результате отсечения не получается.

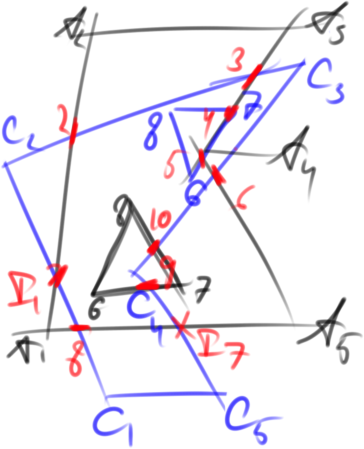
Работа алгоритма сводится к работе с двунаправленными циклическими списками. Решить задачу в общем случае достаточно просто, однако массу неудобств доставляют частные случаи. Проблемы возникают из-за того, что границу отсекателя мы относим к видимой части.

Контуры многоугольников должны задаваться определенным образом: внешняя граница КАЖДОГО многоугольника обходится по часовой стрелке, а внутренние границы – против часовой стрелки, чтобы внутренняя область всегда лежала по правую сторону от направления обхода.

Для реализации отсечения надо найти все точки пересечения границ многоугольников.

1. найти все точки пересечения рёбер отскекаемого с ребрами отсекателя

2. найденные точки разбиваются на две группы: точки входа и точки выхода. Точка входа – если ребро отсекаемого многоугольника входит внутрь отсекателя (с учётом обхода часовой стрелки) и выхода, если выходит из отсекателя. Для получения внутренних многоугольников движение надо начинать с точки входа.



Точки входа: I1 I3 I5 I7 I10

Точки выхода: I2 I4 I6 I8 I9

Построим списки, содержащие и исходные вершины и точки пересечения.

A1 I1 I2 A2 A3 I3 I4 A4 I6 I6 A5 I7 I8 A1 – циклический список обхода внешней грани многоугольника по часовой стрелке.

A6 I9 A7 I 10 A8 A6 – циклический список обхода внутренней грани многоугольника против часовой стрелки.

C1 I8 I1 C2 I2 I3 C3 I6 I10 C4 I9 I7 C5 C1 – обход внешней грани отсекателя по часовой стрелке.

C6 I5 C7 I4 C8 C6 – внутренняя грань отсекателя против часовой.

Одноименные точки пересечения в разных списках лучше соединить связями – можно будет без лишних операций переходить от одного списка к другому.

Для нахождения внутренних многоугольников движение начинаем с очередной точки входа, причём просмотр начинается со списка отсекаемого многоугольника. Просматриваемые вершины заносятся в список вершин результирующего многоугольника.

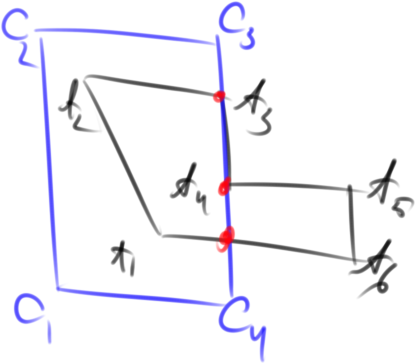
Имеем: I1 I2 (->3) I3 (->1) I4 (->4) C8 C6 I5 I6 I10 A8 A6 I9 I7 I8 I1 //указаны только первые переходы из списка в список!!

Движение заканчивается когда мы вернулись в стартовую точку (И1). Все точки входа вошли в список – внутренний многоугольник один. Если бы остались «лишние» точки входа, то процесс бы повторился для них.

Чтобы находить внешние многоугольники, движение надо начинать с очередной точки выхода. Списки границ отсекателя надо просматривать В ОБРАТНОМ направлении.

I2 A2 A3 I3 (->3) I2 ; I4 A4 I5 C6 C8 I4 ; I6 A5 I7 I9 A7 I10 I6 ; I8 A1 I1 I8

Частные случаи:



А3 – точка касания, не учитывается.

Мы ищем пересечения с границей отсекателя – однако оная относится к видимой части. Истиная граница ^ проходит на пиксель правее от ребра С3С4 – в А3 пересечения не будет.

Точки пересечения делятся на входы и выходы с помощью векторного произведения вектора стороны отсекаемого на вектор стороны отсекателя. Положительное – вход, отрицательное – выход.