Машинная графика
Computer Graphics

Лекция 7

«Отсечение по произвольному окну»

План лекции

- Параметрическое задание отрезка
- Алгоритм Кируса-Бека
- Внешнее отсечение
- Вычисление внутренних нормалей многоугольника
- Определение факта внутренней ориентации нормали
- Разбиение невыпуклых многоугольников
- Отсечение многоугольников

Параметрическое представление отрезка

Обычный вид задания прямой: $y=kx+c$

Вопрос:

Как с помощью данного уравнения задать концы отрезка?

$$x \in [x_0, x_1];$$

Параметрическое задание прямой: $x=x_0+dx*t$

$y=y_0+dy*t$

где $dx = x_1-x_0$; $dy = y_1-y_0$; $0 \le t \le 1$

В векторном виде: $V(t) = V_0 + (V_1-V_0)*t$

Параметрическое представление отрезка

Параметрическое задание прямой:

$x = x_0 + dx * t$

$y = y_0 + dy * t$

$0 \le t \le 1$

где $dx = x_1 - x_0$; $dy = y_1 - y_0$;

В векторном виде: $V(t) = V_0 + (V_1 - V_0) * t$

Точки пересечения отрезка
со сторонами окна:

левой - $t_л = (x_л - x_0)/(x_1 - x_0)$

правой - $t_п = (x_п - x_0)/(x_1 - x_0)$

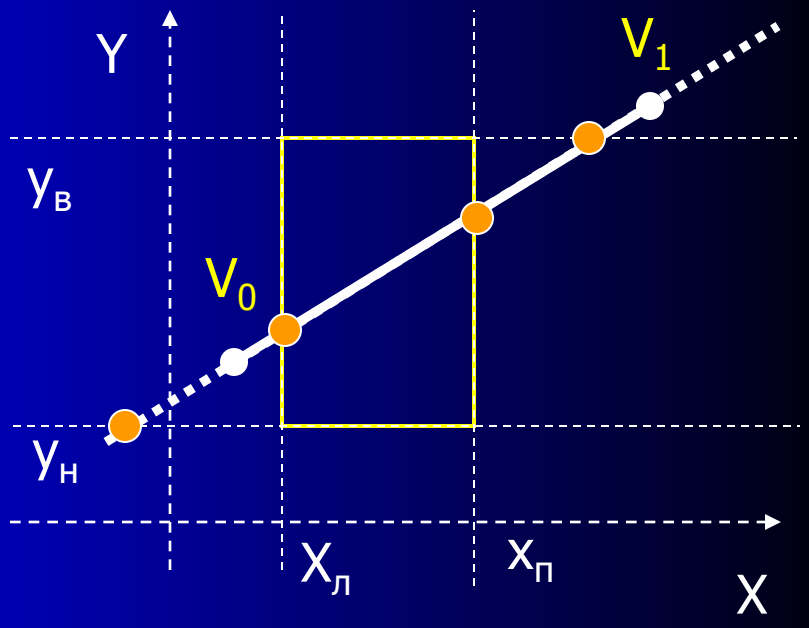
верхней - $t_в = (y_в - y_0)/(y_1 - y_0)$

нижней - $t_н = (y_н - y_0)/(y_1 - y_0)$

Координаты пересечения отрезка с
левой стороной окна –

$x = x_0 + dx * t_л$

$y = y_0 + dy * t_л$

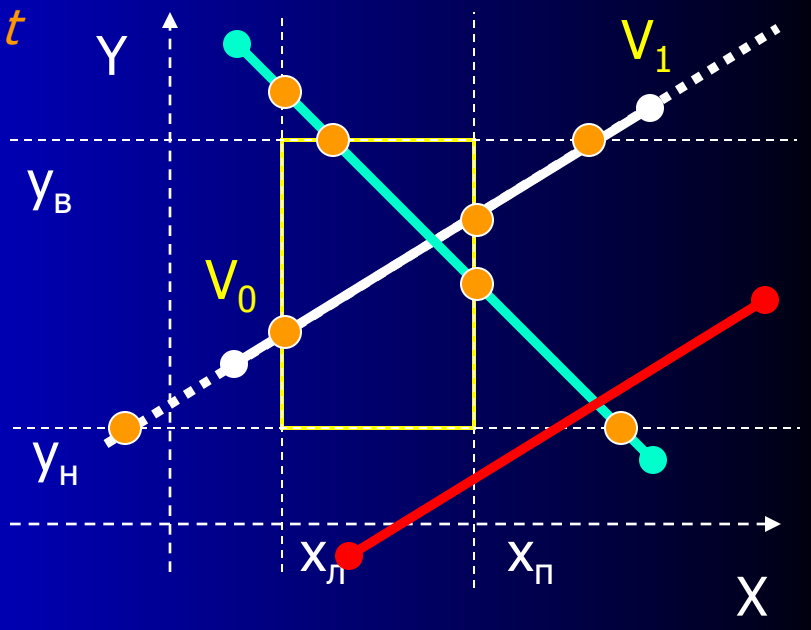


Параметрическое представление отрезка

Вопрос: Как выбрать из $t_л$, $t_п$, $t_в$, $t_н$ два нужных параметра, реально соответствующих точкам пересечения?

Во первых: – следует проверить значения $t_л$, $t_п$, $t_в$, $t_н$ на их принадлежность интервалу $0 \le t \le 1$

Во вторых: – если более двух значений t принадлежат интервалу [0,1], то необходимо их упорядочить по возрастанию и вычислить значения:
t_{minmax} – минимальное среди двух максимальных
и t_{maxmin} – максимальное среди двух минимальных,
т.е. значения, стоящие в середине упорядоченного ряда.



Алгоритм Кируса-Бека (Cyrus & Beck)

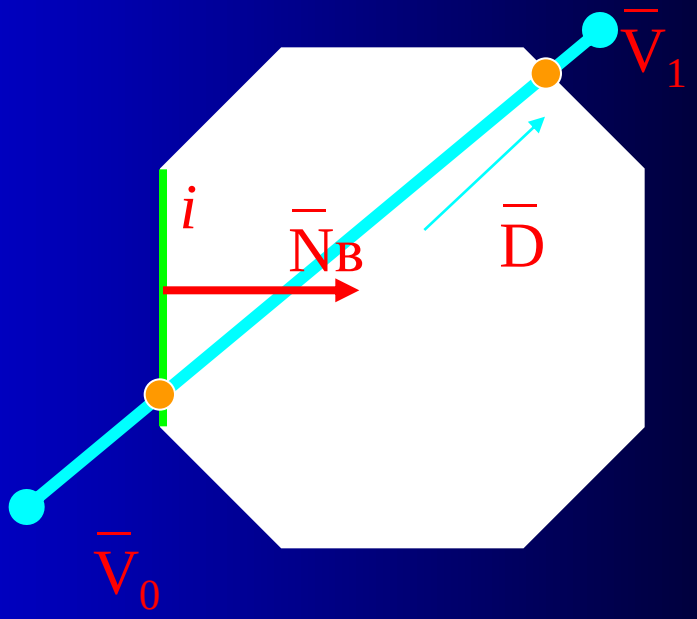
Алгоритм предложен Кирусом и Беком в 1978 году, корректно работает для любого выпуклого многоугольника.
Может отсекать как 2D, так и 3D линии по выпуклым многоугольникам и многогранникам.
В 1984 году Лианг (Liang) и Барски (Barsky) усовершенствовали данный алгоритм.
В алгоритме Кируса-Бека вычисляется значение параметра t для нахождения точки пересечения линии и ребра (грани) многоугольника (или многогранника в 3D варианте). В алгоритме используются элементарные операции сравнения для определения реальных точек пересечения.
В усовершенствованной версии алгоритма Лианг и Барски нашли способ отбраковывать параметр t на стадии его вычисления и таким образом отрисовывать или отбрасывать части отрезка немедленно после вычисления очередного значения t.

Алгоритм Кируса-Бека (Cyrus & Beck)

Алгоритм использует понятие нормали к стороне многоугольника – окна отсечения. Существует две версии алгоритма – с использованием внутренней и наружной нормали к стороне многоугольника. Независимо от варианта суть алгоритма не меняется.

Так как многоугольник выпуклый, то у него может быть только две точки пересечения с отрезком.
Обозначим через $\overline{N}_i$ внутреннюю нормаль к i-той стороне окна, а через $\overline{D}$ – вектор, задающий ориентацию отсекаемого отрезка.

$\overline{D} = \overline{V}_1 - \overline{V}_0$



Алгоритм Кируса-Бека (Cyrus & Beck)

Скалярное произведение двух векторов:

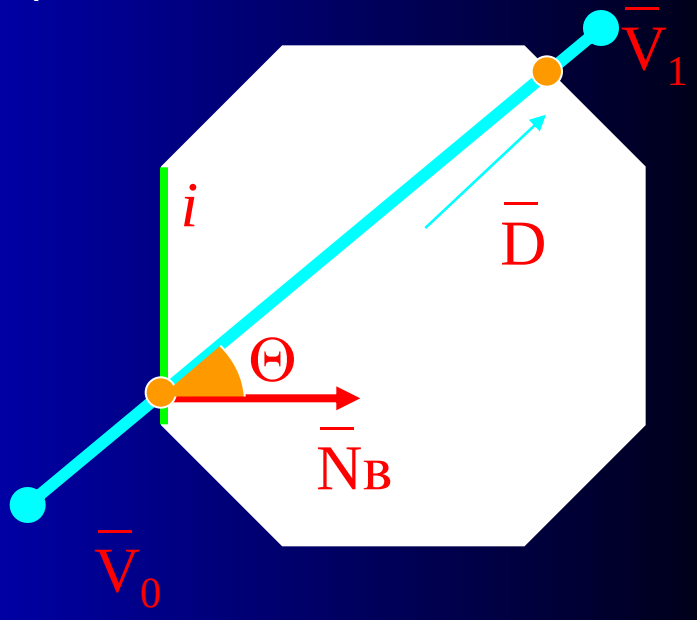
$P_i = \overline{N_i} \cdot \overline{D_i} = \overline{N_i} \cdot (\overline{V_1} - \overline{V_0}) = |\overline{N}| * |\overline{D}| * \cos\Theta$

С помощью данного выражения можно различать три случая взаимной ориентации стороны окна отсечения и рассматриваемого отрезка:

1) Отрезок входит в окно через i-тое ребро многоугольника отсечения $P_i>0$

2) Отрезок параллелен i-той стороне окна $P_i=0$

3) Отрезок выходит из окна отсечения, пересекая i-тое ребро многоугольника $P_i<0$



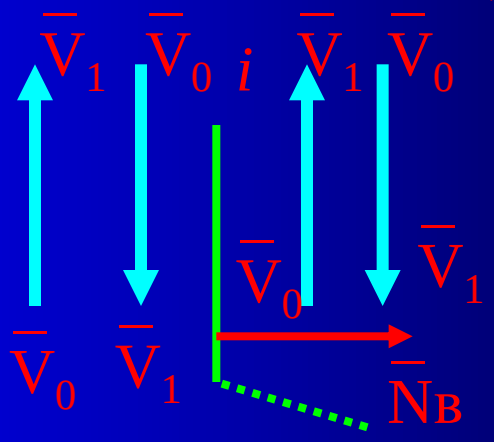
Алгоритм Кируса-Бека (Cyrus & Beck)

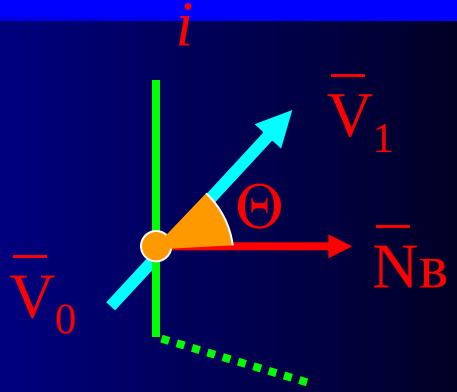
Иллюстрации к указанным случаям:

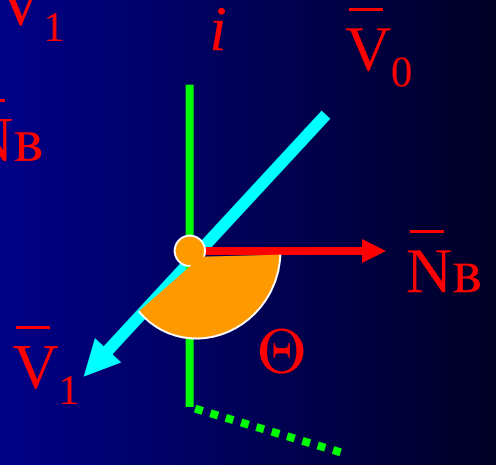
1) Отрезок входит в окно через i-тое ребро многоугольника отсечения
cos (Θ) >0 --> P_i>0

2) Отрезок параллелен i-той стороне окна
Θ = 90 °, cos (90 °)=0
--> P_i=0 ,

3) Отрезок выходит из окна отсечения, пересекая i-тое ребро многоугольника
Θ > 90 ° --> cos (Θ) <0 --> P_i<0





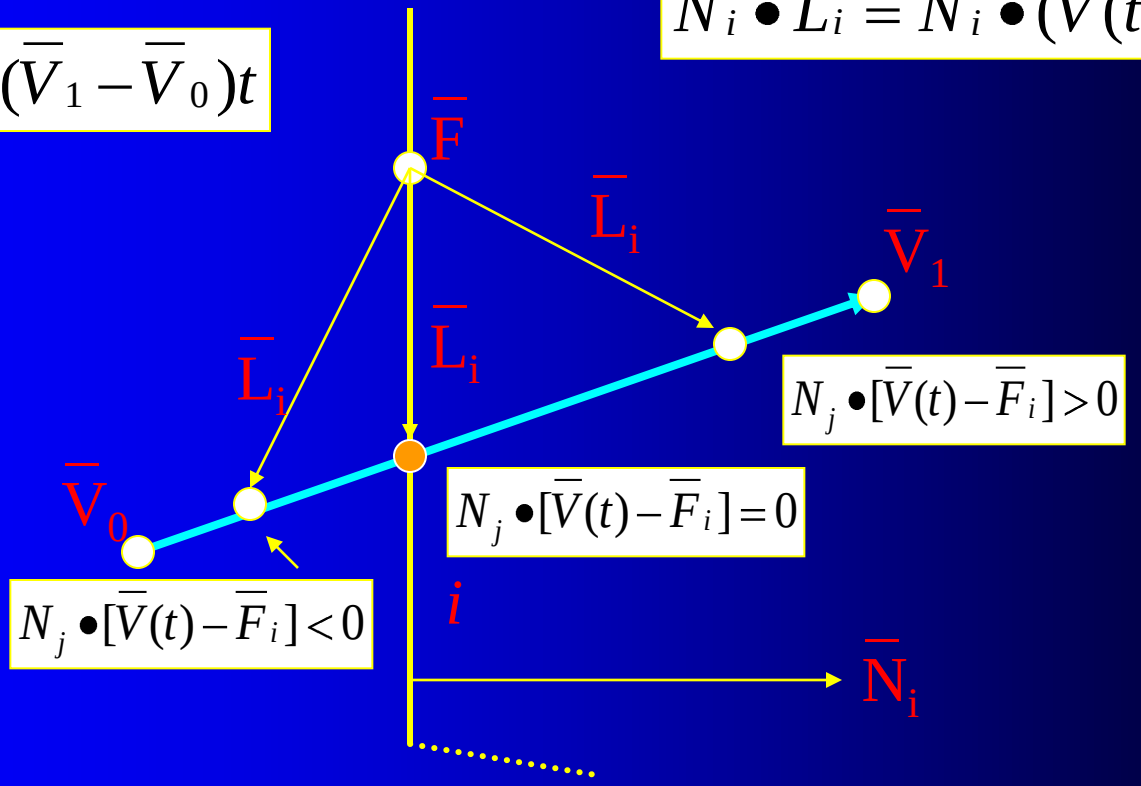


Алгоритм Кируса-Бека (Cyrus & Beck)

Для определения факта принадлежности точки пересечения *i* – тому ребру, используем вектор $\overline{L}_i = \overline{V}(t) - \overline{F}_i$ и его скалярное произведение с внутренней нормалью:

$$\overline{N}_i \bullet \overline{L}_i = \overline{N}_i \bullet (\overline{V}(t) - \overline{F}_i)$$

$$\overline{V}(t) = \overline{V}_0 + (\overline{V}_1 - \overline{V}_0)t$$



Алгоритм Кируса-Бека (Cyrus & Beck)

$$\overline{V}(t) = \overline{V}_0 + (\overline{V}_1 - \overline{V}_0)t$$

$$\overline{N}_i \bullet [\overline{V}(t) - \overline{F}_i] = 0$$

$$\overline{N}_i \bullet [\overline{V}_0 + (\overline{V}_1 - \overline{V}_0)t - \overline{F}_i] = 0$$

$$\overline{N}_i \bullet [\overline{V}_0 - \overline{F}_i] + \overline{N}_i \bullet [\overline{V}_1 - \overline{V}_0]t = 0$$

$$\overline{D}_i = (\overline{V}_1 - \overline{V}_0),\ \overline{w}_i = (\overline{V}_0 - \overline{F}_i)$$

выразим t :

$$t = \frac{\overline{N}_i \bullet (\overline{V}_0 - \overline{F}_i)}{-\overline{N}_i \bullet (\overline{V}_1 - \overline{V}_0)} = -\frac{\overline{N}_i \bullet \overline{w}_i}{\overline{N}_i \bullet \overline{D}_i} = -\frac{Q_i}{P_i};$$

$$Q_i = \overline{N}_i \bullet \overline{w}_i$$

$\overline{D}_i \neq 0$ ($\overline{D}_i = 0$ только в случае, если $\overline{V}_1 = \overline{V}_0$, т.е. при вырождении отрезка в точку).

Знак $\overline{D}_i$ – имеет существенное значение

$\overline{D}_i > 0 \rightarrow$ точка входа отрезка в многоугольник

$\overline{D}_i < 0 \rightarrow$ точка выхода отрезка из многоугольника

Алгоритм Кируса-Бека (Cyrus & Beck)

Формальная запись алгоритма:

Инициализация $t_0=0$, $t_1=1$

Цикл для каждой i-той стороны окна отсечения:

{ Вычисления скалярных произведений P_i и Q_i

Если $P_i = 0$, то отрезок либо вырожден в точку либо параллелен данной стороне окна. В этом случае следует проанализировать знак Q_i, и принять решение отбрасывать или не отбрасывать отрезок целиком.

Если $Q_i < 0$ – то отрезок вне окна, отсечение закончено.

Иначе берем следующую сторону.

Если $P_i \neq 0$, то вычисляем значение t. Если t не попадает в интервал [0,1] то оно отбрасывается. Иначе анализируем знак P_i.

Алгоритм Кируса-Бека (Cyrus & Beck)

Формальная запись алгоритма:

Инициализация $t_0=0$, $t_1=1$

Цикл для каждой i-той стороны окна отсечения:

{ …

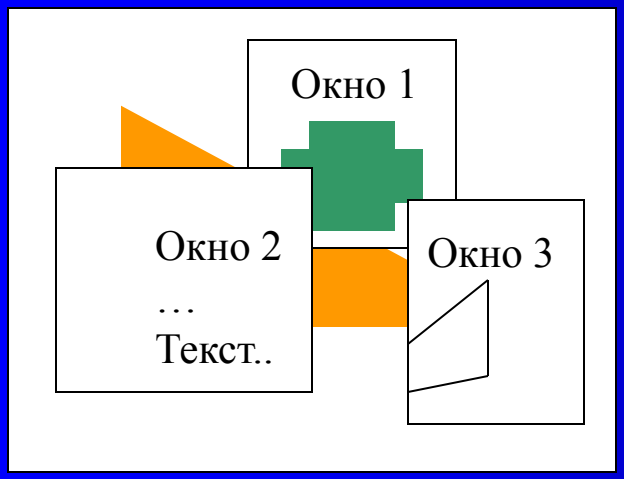
Если $P_i < 0 \rightarrow$ точка выхода отрезка из многоугольника, т.е. мы ищем значение t_1 (верхний предел отрезка). Проверяем $t>t_0$, если нет, то отбрасываем (переход к следующей стороне). Если $t<t_1$, то $t_1 = t$.

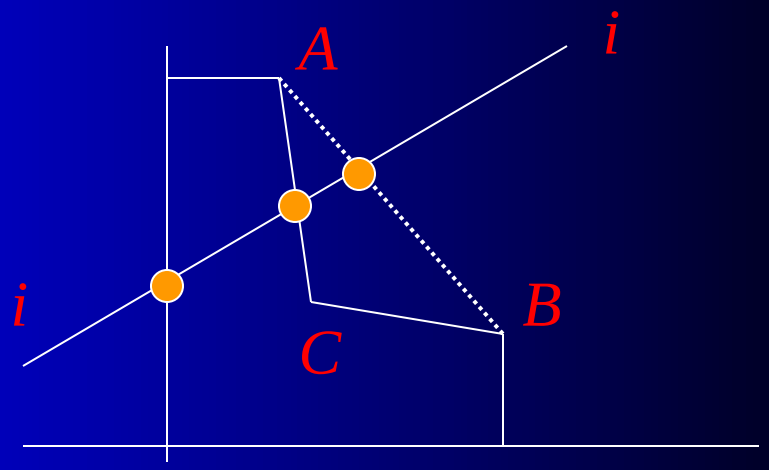
Если $P_i > 0 \rightarrow$ точка входа отрезка в многоугольник, т.е. мы ищем значение t_0 (нижний предел отрезка). Проверяем $t<t_1$, если нет, то отбрасываем (переход к следующей стороне). Если $t>t_0$, то $t_0= t$.

}// Конец по циклу сторон многоугольника

Внутреннее и внешнее отсечение

При внешнем отсечении требуется определить части отрезка, лежащие вне окна. Данный тип отсечения важен для работы с несколькими перекрывающимися окнами (фигурами), кроме того может использоваться при работе с вогнутым полигональным окном.





Определение факта выпуклости многоугольника и вычисление его внутренних нормалей

Для работы с алгоритмом Кируса-Бека следует убедиться, что многоугольник выпуклый. Факт выпуклости многоугольника можно определить по знаку векторного произведения его смежных сторон. Все вычисленные знаки анализируются:

Все знаки равны нулю – многоугольник вырожден в отрезок.

Есть как положительные, так и отрицательные знаки – многоугольник невыпуклый.

Все знаки неотрицательные – многоугольник выпуклый, его внутренние нормали ориентированы влево от направления обхода его контура.

Все знаки отрицательные – многоугольник выпуклый, его внутренние нормали ориентированы вправо от направления обхода его контура.

Векторное произведение

- Вычисление векторного произведения в матричной форме:

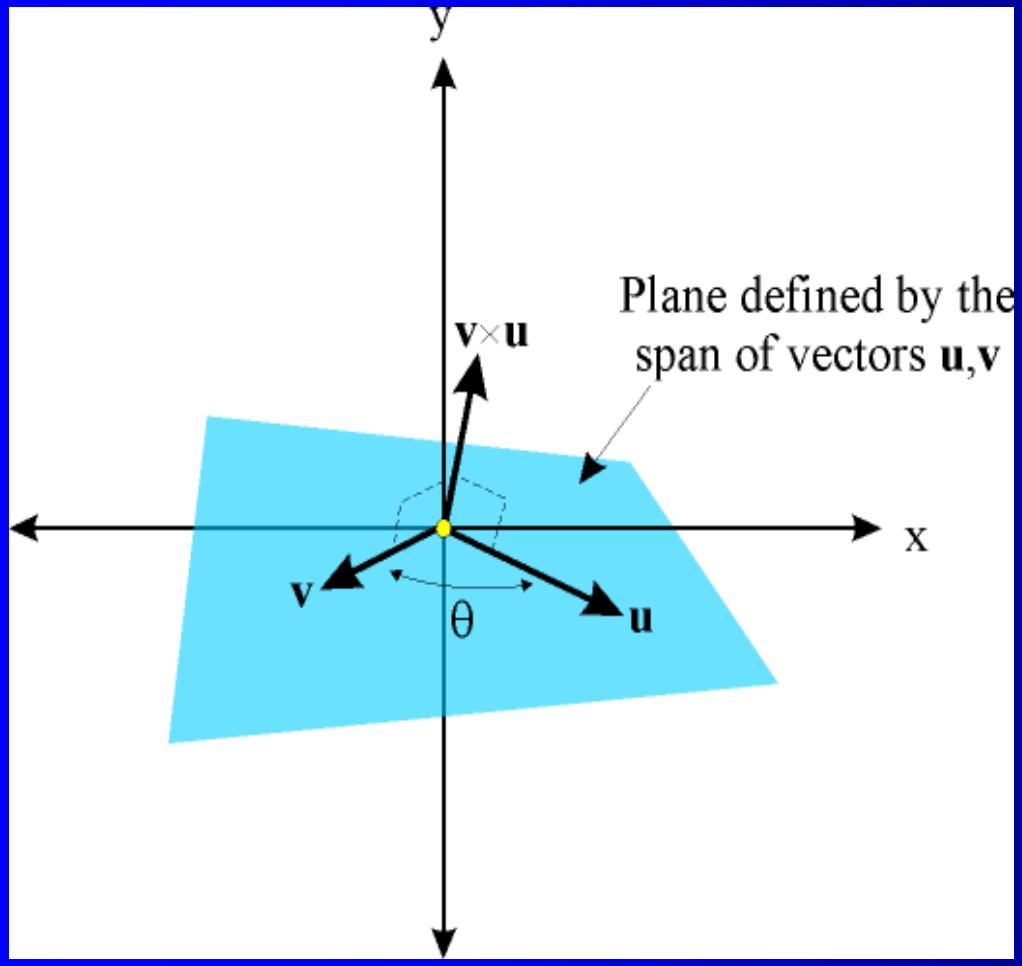
$$\mathbf{u}\times\mathbf{v}=\begin{bmatrix}u_1\\u_2\\u_3\end{bmatrix}\times\begin{bmatrix}v_1\\v_2\\v_3\end{bmatrix}=\begin{bmatrix}u_2v_3-u_3v_2\\u_3v_1-u_1v_3\\u_1v_2-u_2v_1\end{bmatrix}$$

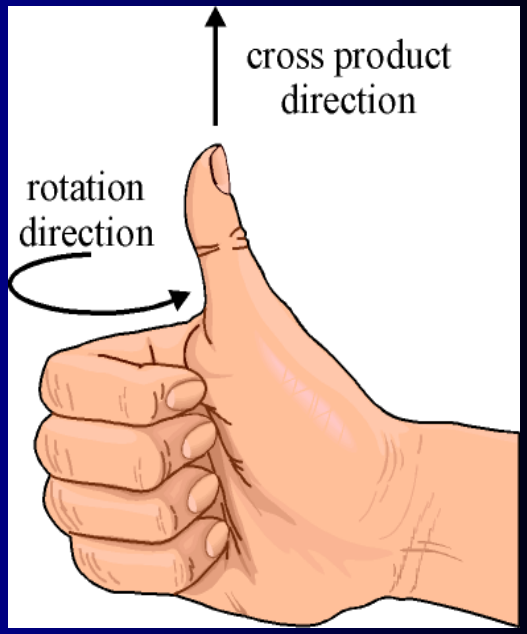
- Результатом является *вектор*, перпендикулярный плоскости исходных векторов **u** и **v**:

$$\mathbf{u}\times\mathbf{v}=\mathbf{w}\left\|\mathbf{u}\right\|\left\|\mathbf{v}\right\|\sin\theta$$

$$\left\|\mathbf{u}\times\mathbf{v}\right\|=\left\|\mathbf{u}\right\|\left\|\mathbf{v}\right\|\sin\theta$$

Векторное произведение (Cross Product)

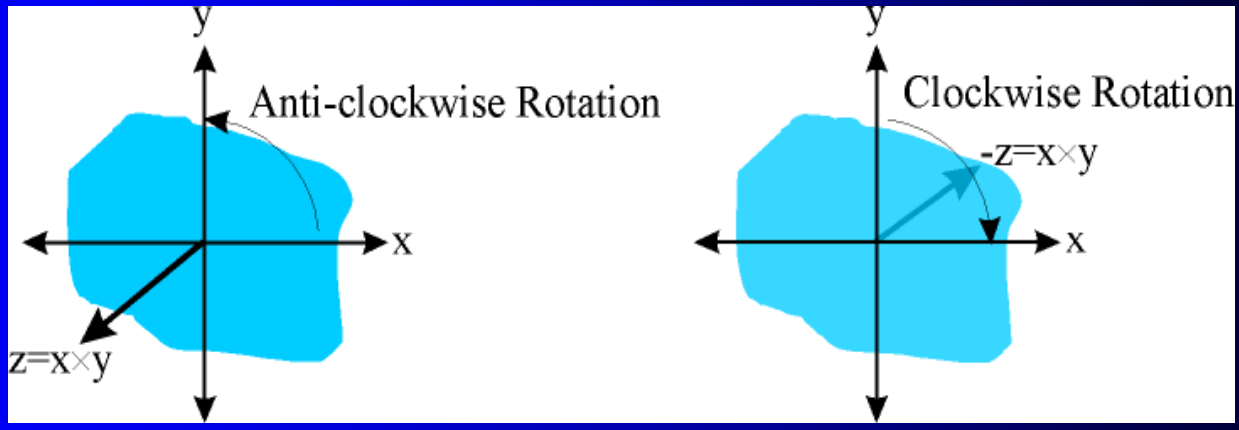




Правая координатная система

Векторное произведение (Cross Product)

- Анти-коммутативно: $\mathbf{u} \times \mathbf{v} = -(\mathbf{v} \times \mathbf{u})$
- Не ассоциативно: $\mathbf{u} \times (\mathbf{v} \times \mathbf{w}) \neq (\mathbf{u} \times \mathbf{v}) \times \mathbf{w}$
- Направление результирующего вектора зависит от порядка операндов (от угла между ними):



R.H.S. – правая координатная система
(Right Handed Coordinate System)

Вычисление внутренних нормалей многоугольника

Если скалярное произведение двух векторов равно 0, то они перпендикулярны.

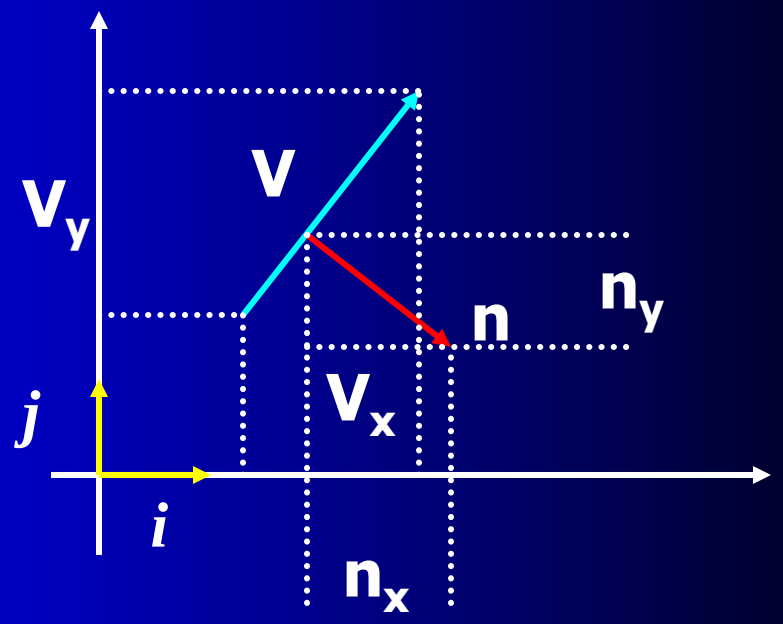
$(\mathbf{n}_x \boldsymbol{i} + \mathbf{n}_y \boldsymbol{j})(\mathbf{V}_x \boldsymbol{i} + \mathbf{V}_y \boldsymbol{j}) = 0$

$\mathbf{n}_x \mathbf{V}_x + \mathbf{n}_y \mathbf{V}_y = 0$

$\mathbf{n}_x \mathbf{V}_x = -\mathbf{n}_y \mathbf{V}_y$

Примем $\mathbf{n}_y = 1$, тогда

$\mathbf{n}_x = -\mathbf{V}_y / \mathbf{V}_x$



Вектор нормали можно вычислить: $\mathbf{n} = -(\mathbf{V}_y / \mathbf{V}_x)\boldsymbol{i} + \boldsymbol{j}$

Вычисление внутренних нормалей многоугольника

Пример:

Вычисление нормали для стороны V_1V_2.

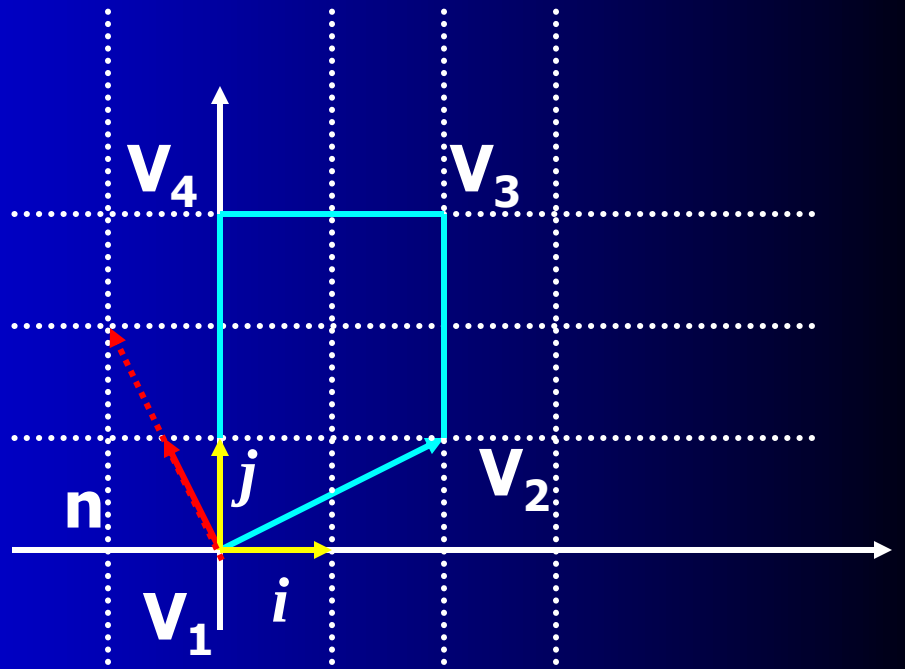
$\mathbf{V}_x=2$, $\mathbf{V}_y=1$

Вектор нормали:

$\mathbf{n} = -(\mathbf{V}_y/\mathbf{V}_x)\boldsymbol{i} + \boldsymbol{j} =$

$= -1/2\boldsymbol{i} + \boldsymbol{j}$

Так как длина нормали не критична, то можно удлинить вектор вдвое.



Определение факта внутренней ориентации нормали

Если вектор стороны многоугольника образован как разность векторов пары его смежных вершин $\mathbf{V}_{i-1}$ и $\mathbf{V}_i$, то следует проверить скалярное произведение нормали данной стороны и вектора от $\mathbf{V}_{i-1}$ до $\mathbf{V}_{i+1}$. Если знак произведения положительный, то **n**- внутренняя нормаль, если отрицательный – то внешняя.

В последнем случае внутреннюю нормаль можно получить умножением **n** на –1.

Пример:

n умножим на $\mathbf{V_1V_3}$

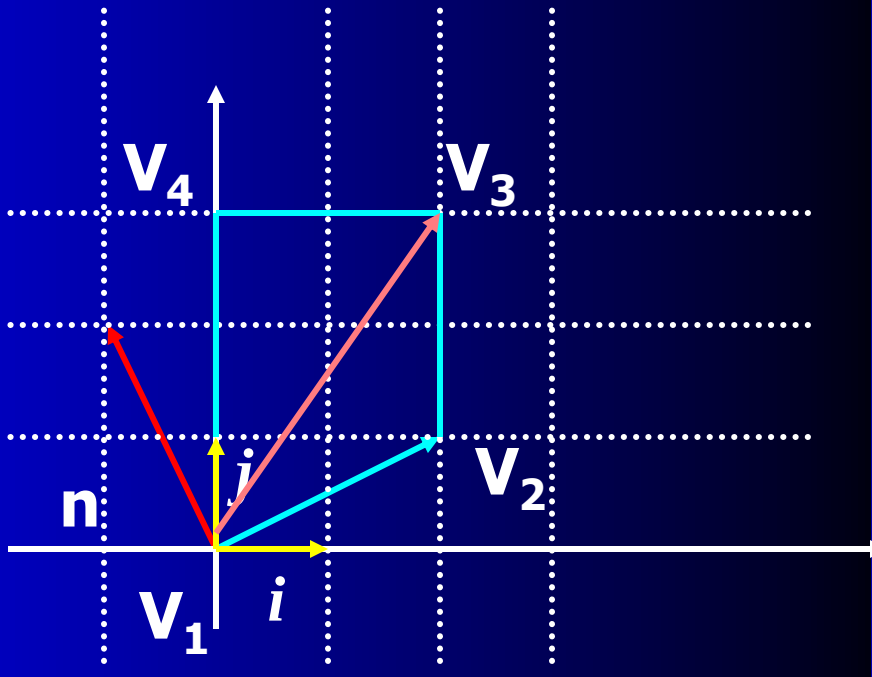
$\mathbf{V_1V_3} = 2\boldsymbol{i} + 3\boldsymbol{j}$

$\mathbf{n} = -\boldsymbol{i} + 2\boldsymbol{j}$

$\mathbf{n}\,\mathbf{V_1V_3} = (-\boldsymbol{i} + 2\boldsymbol{j})(2\boldsymbol{i} + 3\boldsymbol{j}) =$

$= -2+6 = 4$

4> 0 – нормаль внутренняя!



Разбиение невыпуклых многоугольников

Во многих алгоритмах имеется требование о выпуклости многоугольника. Один из простейших способов одновременной проверки и разбиения многоугольника следующий:

Обходим вершины многоугольника против часовой стрелки.

Для каждой i-й вершины производится перенос многоугольника (или системы координат), с тем что бы вершина была в т. (0,0).

Поворот многоугольника по часовой стрелке относительно (0,0), так что бы вершина (i+1) оказалась на положительной полуоси x.

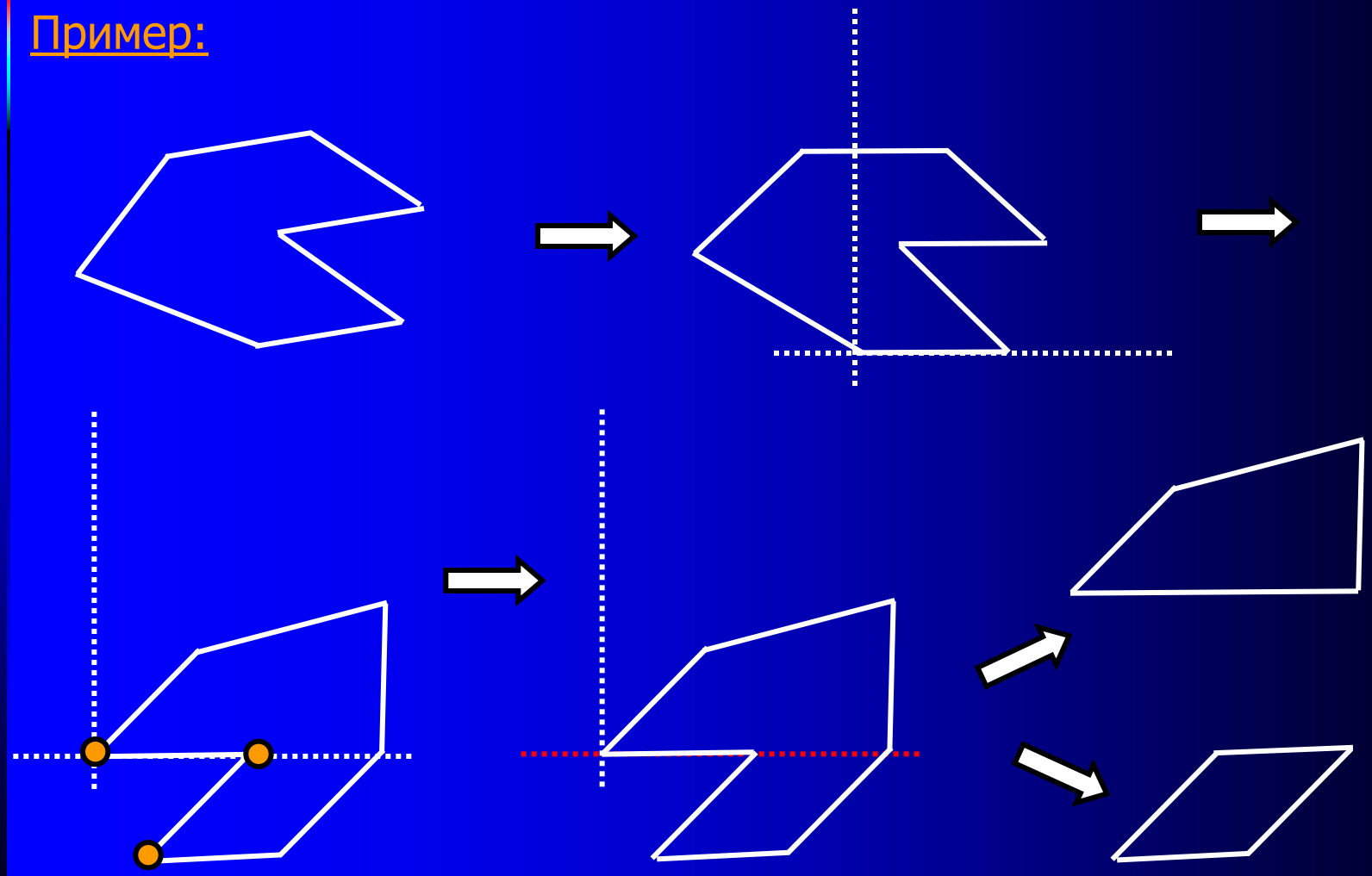
Проанализировать знак ординаты вершины (i+2). Если он не отрицательный, то многоугольник в (i+1)-й вершине выпуклый.

Если отрицательный – то многоугольник не выпуклый – требует разбивки. Разрезаем его по положительной полуоси OX. Получаем два новых многоугольника. Рекурсивно анализируем каждый из них.

Переход к новой вершине.

Разбиение невыпуклых многоугольников

Пример:

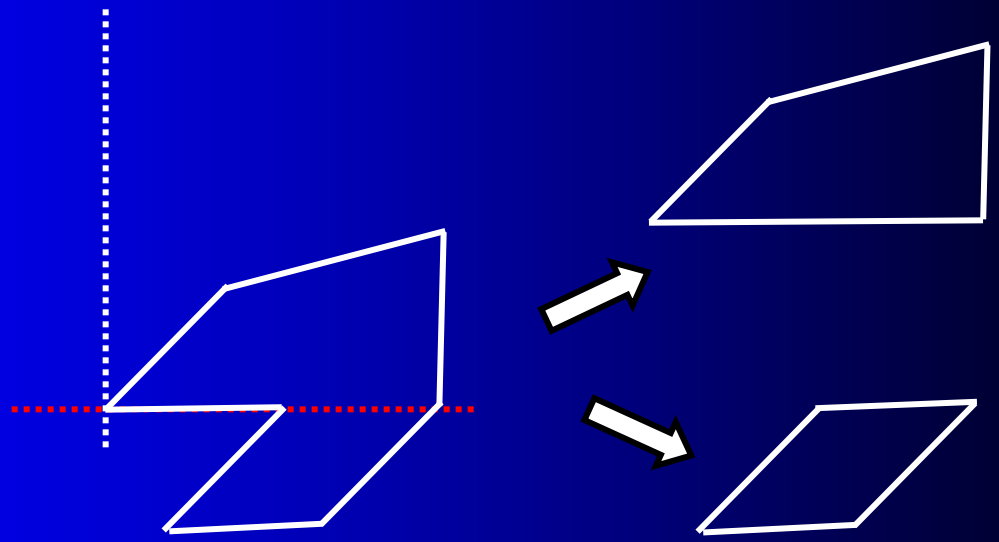


Разбиение невыпуклых многоугольников

Недостатки алгоритма:

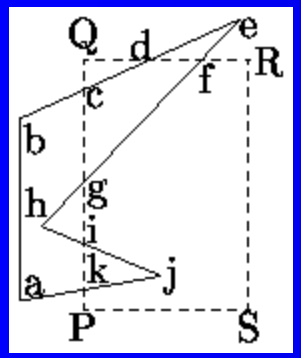
-не обеспечивает минимальность по количеству получаемых многоугольников,

- не корректно работает, если имеется самопересечение сторон многоугольника.

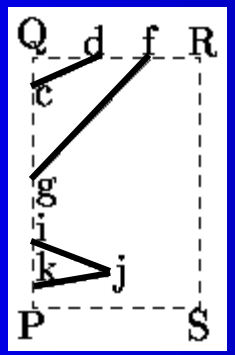


Отсечение многоугольников

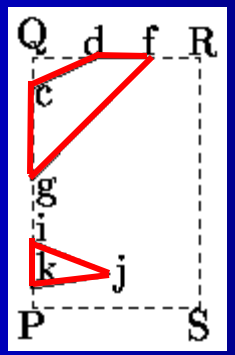
Проблема. На рис. а) показаны окно отсечения PQRS и исходный многоугольник abehj. В результате применения алгоритмов отсечения линий мы получим набор несвязанных между собой отрезков cd, fg, ij, jk. - рис. б). Желаемый же результат отсечения, необходимый при закраске отсеченного многоугольника, представлен на рис. в). Видно, что в состав отсеченного многоугольника должны войти фрагменты границ окна отсечения - ребра cg, df, ki.



а



б



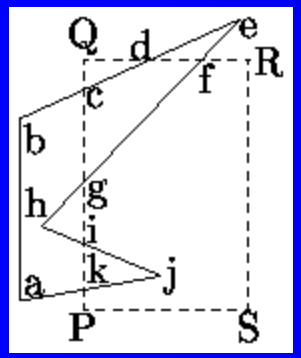
в

Отсечение многоугольников

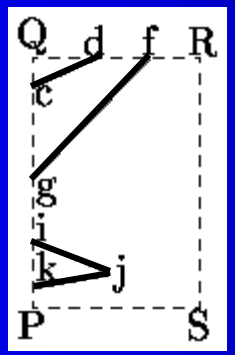
Одним из выходов могло бы быть автоматическое соединение предыдущих и последующих точек пересечения рёбер многоугольника с окном отсечения.

Однако в данном случае возможно возникновение паразитных рёбер в результирующем многоугольнике.

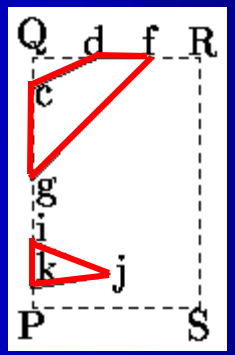
В примере - ребро ig на рис. г).



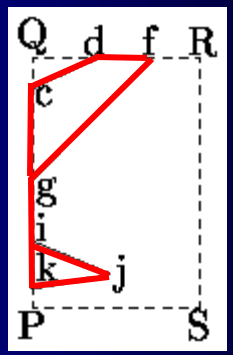
а



б



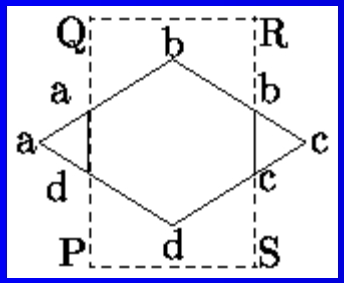
в



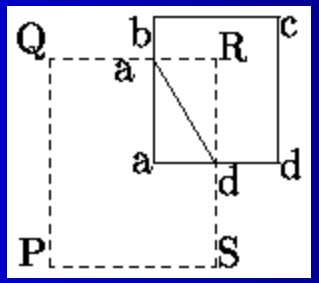
г

Отсечение многоугольников

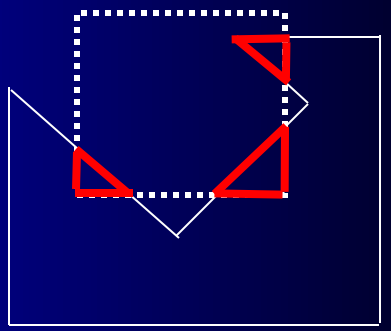
В ряде случаев, например, при отсечении по границе экрана эти паразитные ребра несущественны. Рассмотренный подход позволяет правильно отсечь для простых случаев рис. а) и дает ошибку, если отсекаемый многоугольник охватывает вершину окна см. рис. б). В последнем случае результат отсечения - треугольник a, as, ds, вместо прямоугольника a, as, R, ds.



а



б



Отсечение многоугольников

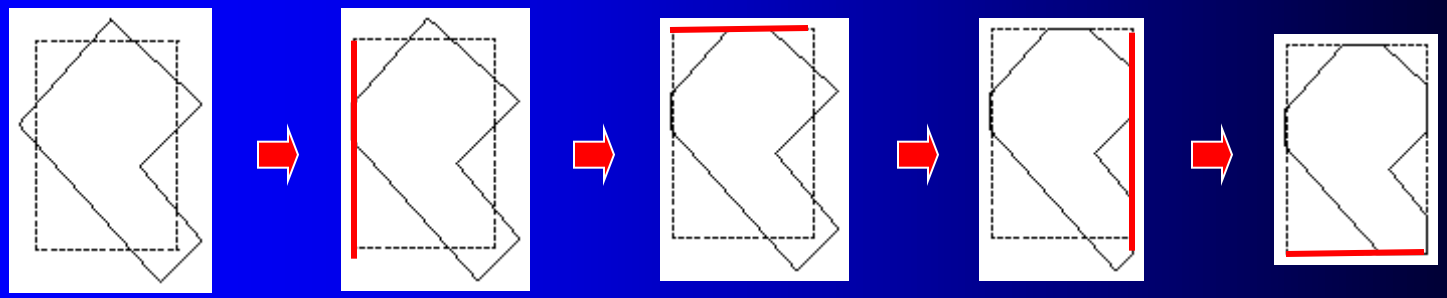
Таким образом, алгоритм отсечения многоугольника должен в результате отсечения давать один или несколько замкнутых многоугольников. При этом могут быть добавлены новые ребра, а имеющиеся или сохранены или разделены или даже отброшены. Существенно, чтобы границы окна, которые не ограничивают видимую часть отсекаемого многоугольника, не входили в состав результата отсечения. Если это не выполняется, то возможна излишняя закраска границ окна.

В общем случае, при отсечении многоугольников требуется решать два типа задач – внутренне (отображение части изображения попавшего в окно) и внешнее (отображение изображения, находящегося вне окна).

Алгоритм Сазерленда-Ходгмана (Ходжмена) (Sutherland & Hodgman)

Основная идея алгоритма:

весь многоугольник последовательно отсекается каждой границей окна, как это показано на рис.



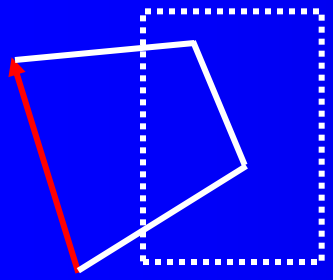
Алгоритм способен отсекать любой многоугольник (выпуклый или невыпуклый) относительно окна отсечения, являющегося выпуклым многоугольником.

Наличие отверстий в окне отсечения либо в отсекаемом многоугольнике не допускается.

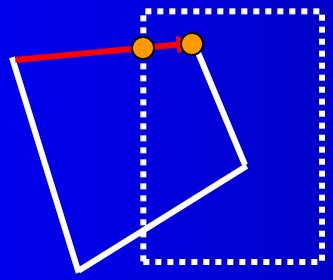
Алгоритм Сазерленда-Ходгмана (Ходжмена) (Sutherland & Hodgman)

Порядок обхода многоугольника по часовой стрелке (но не принципиален). Координаты первой вершины (начала обхода) запоминаются в переменной F (от first).

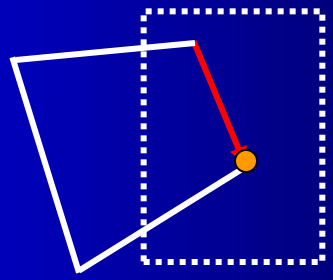
Возможны четыре случая положения ребра по отношению к отсекающей прямой.



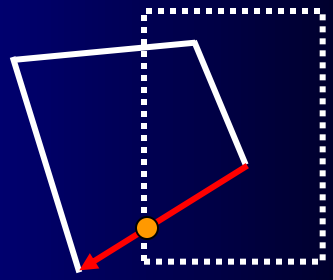
В список новых вершин не заносится



В список заносится 2 новые вершины



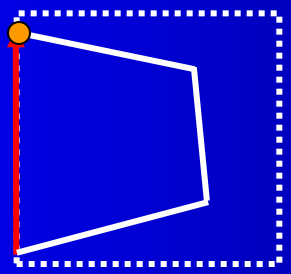
В список заносится только одна вершина



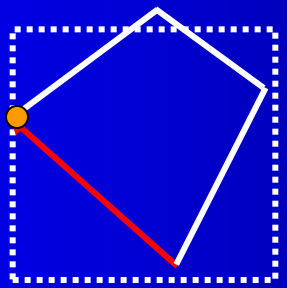
В список заносится одна граничная точка

Алгоритм Сазерленда-Ходгмана (Ходжмена) (Sutherland & Hodgman)

Частные случаи:



Ребро на границе окна считается отображаемым в кадре. В список заносится одна вершина (как в 3-м случае)



Касания не считаются пересечениями. Точка касания - считается отображаемой в кадре. В список заносится одна вершина (как в 3-м случае)

Алгоритм Сазерленда-Ходгмана (Ходжмена) (Sutherland & Hodgman)

Алгоритм:

Для удобства на входе - P1…PN-1PN (=P1…PN-1P1)–замкнутый цикл.

```
{   Цикл по сторонам окна – прямую L поочерёдно переопределяют
        по каждой стороне окна.
    //  Отсечение P1…PN-1PN прямой L:
        for (i=2;i<=N;i++)
        {       if (Pi-1Pi пересекает L)
                {   I=intersection(Pi-1Pi,L);
                    output (I);
                }
                if (Pi видима) output (Pi);
        }
}
```

Плюсы алгоритма:

Простота;
Надо мало памяти - только 2 списка точек;
Работает для любых замкнутых цепочек;

Минусы алгоритма:

Годен только для выпуклых отсекателей;