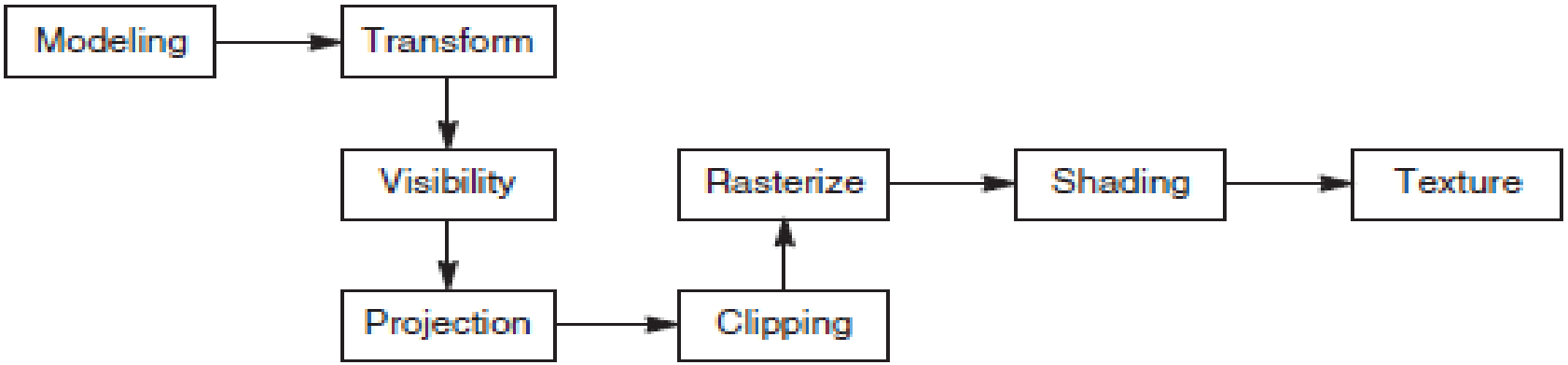
Компьютерная графика

Курс лекций

Тема №7. Алгоритмы отсечения.

Графический конвейер

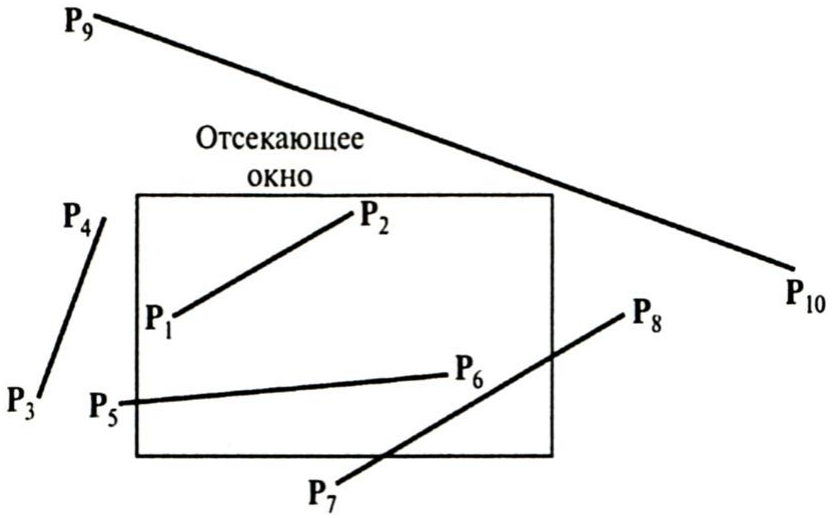


- моделирование (modeling) – математическое описание объектов, всей сцены, источников света, с учётом расположения
- отображение (rendering):
  - преобразования (transformation) – задание местоположения;
  - определение видимости (visibility) – область видимости (field of view) + нелицевые поверхности → отсечение (clipping);
  - проекция на картинную плоскость (projection);
  - растеризация (rasterization);
  - закраска (shading);
  - текстурирование (texturing).

Отсечение

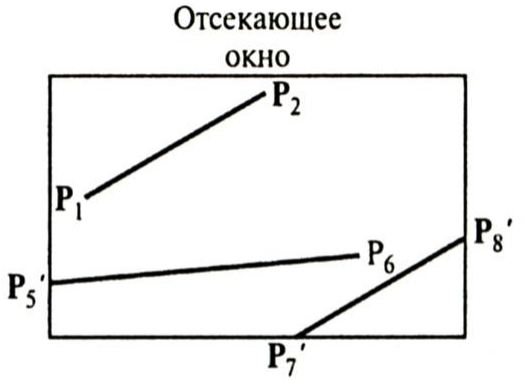
- *алгоритмом отсечения (отсечением)* называется любая процедура, которая удаляет те точки изображения, которые находятся внутри (или снаружи) заданной области пространства;

- отсекающее окно или объем называются *отсекателем:*
  - регулярной формы (прямоугольное окно или параллелепипед, стороны которого параллельны осям координат);
  - нерегулярной формы (выпуклый или невыпуклый многоугольник или многогранник).



До отсечения

а)



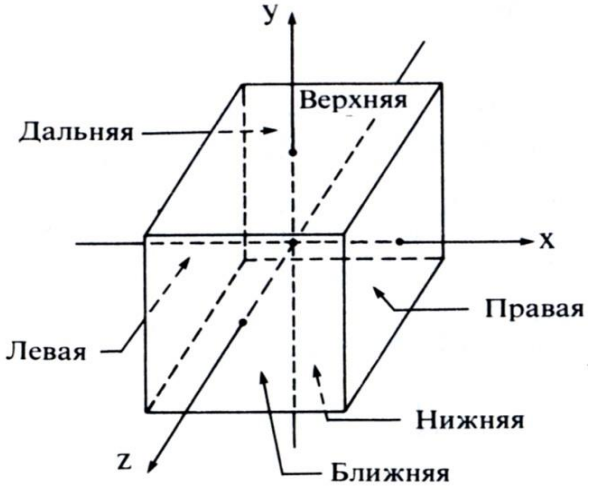
После отсечения

б)

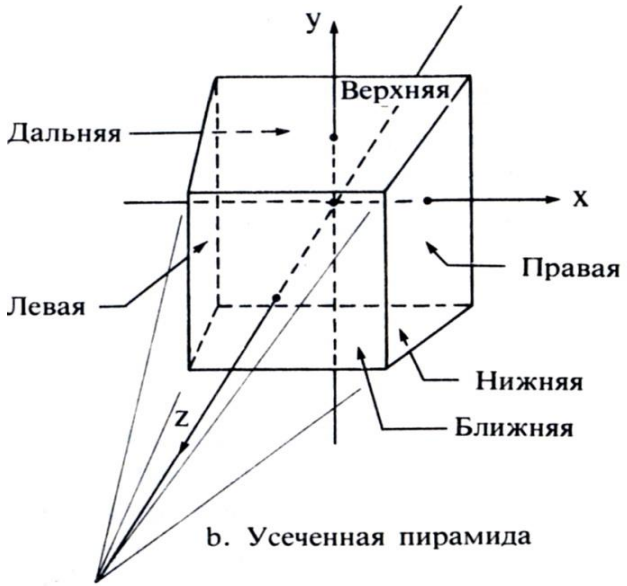
Классификация алгоритмов отсечения

- **по типу обрабатываемых объектов:**
  - отсечение точки;
  - отсечение линии (отрезка);
  - отсечение области (многоугольника);
  - отсечение кривой;
  - отсечение текста (литер);
- **по размерности:**
  - двумерное отсечение;
  - трехмерное отсечение;
- **по расположению результата отсечения относительно отсекателя:**
  - внутреннее отсечение;
  - внешнее отсечение.

$$xw_{\min} \le x \le xw_{\max},$$
$$yw_{\min} \le y \le yw_{\max}.$$



a. Параллелепипед



b. Усеченная пирамида

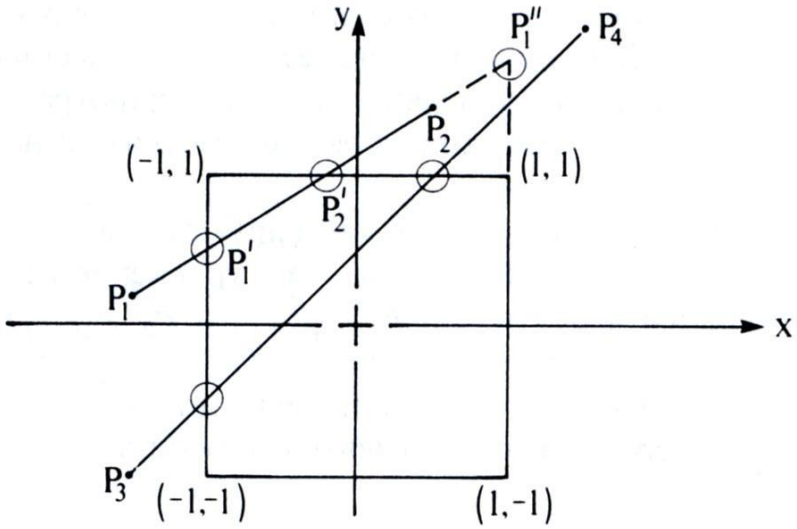
Отсечение отрезков

- простой алгоритм отсечения;
- алгоритм Коэна-Сазерленда (двумерного отсечения регулярной прямоугольной областью);
- алгоритм разбиения средней точкой;
- алгоритм Кируса-Бека (двумерного параметрического отсечения выпуклым многоугольником);
- отсечение отрезка невыпуклым окном;
- обобщение алгоритмов Коэна-Сазерленда и Кируса-Бека для трехмерного случая.

Простой алгоритм отсечения отрезка

- основан на отсечении точки;
- реализует двумерное отсечение отрезка регулярным окном;
- для каждой из сторон отсекателя:
  - рассчитывается точка пересечения прямой, содержащей сторону отсекателя, с прямой, содержащей отрезок;
  - анализируется принадлежность точки пересечения области отсекателя;
- недостаток: определяются 4 точки пересечения;
- вычисление точки пересечения является ресурсоемкой операцией, количество которых необходимо минимизировать.

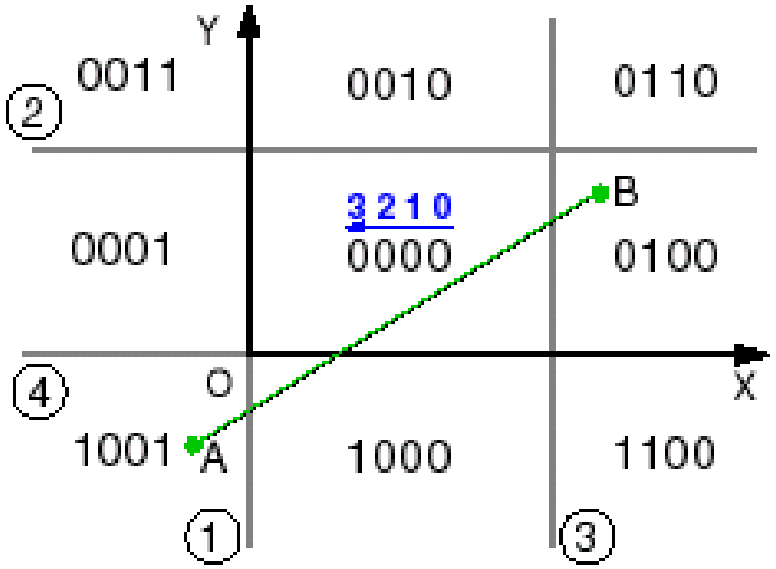
$$xw_{\min} \le x \le xw_{\max},$$
$$yw_{\min} \le y \le yw_{\max}.$$



с левой:	$x_{Л},\ y = m(x_{Л} - x_1) + y_1$	$m \neq \infty$
с правой:	$x_{П},\ y = m(x_{П} - x_1) + y_1$	$m \neq \infty$
с верхней:	$y_{В},\ x = x_1 + (1/m)(y_{В} - y_1)$	$m \neq 0$
с нижней:	$y_{Н},\ x = x_1 + (1/m)(y_{Н} - y_1)$	$m \neq 0$

Вычисление кодов концов отрезка (двумерный случай)

- каждому концу отрезка сопоставляется 4х-битный код:
  - 0 бит равен 1, если точка лежит левее окна;
  - 1 бит - если точка лежит выше окна;
  - 2 бит - если точка лежит правее окна;
  - 3 бит - если точка лежит ниже окна;
- случай тривиальной видимости отрезка: A = B = 0;
- случай тривиальной невидимости: A&B != 0 (побитовое И);
- отрезок может пересекать те стороны окна, для которых установлен соответствующий бит в A^B (побитовое ИЛИ).

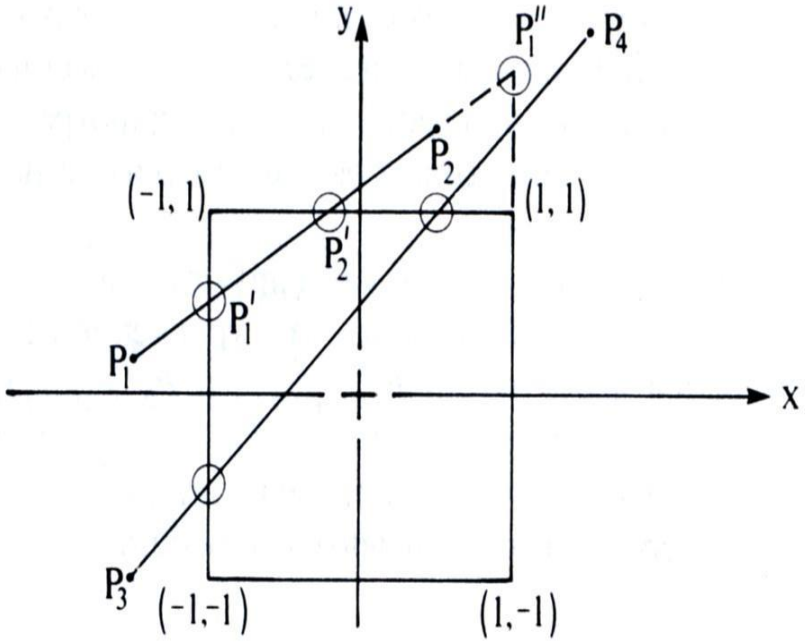


код **A** = **1001**

код **B** = **0100**

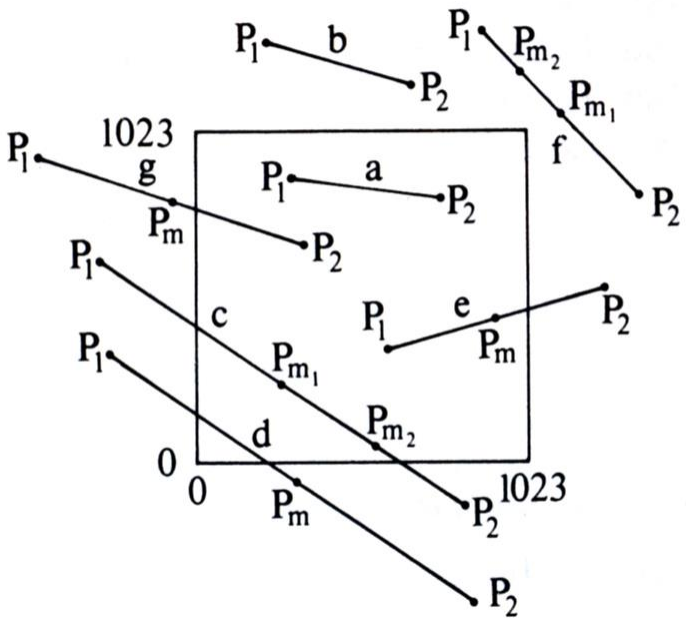
Алгоритм Коэна-Сазерленда (Cohan-Sutherland)

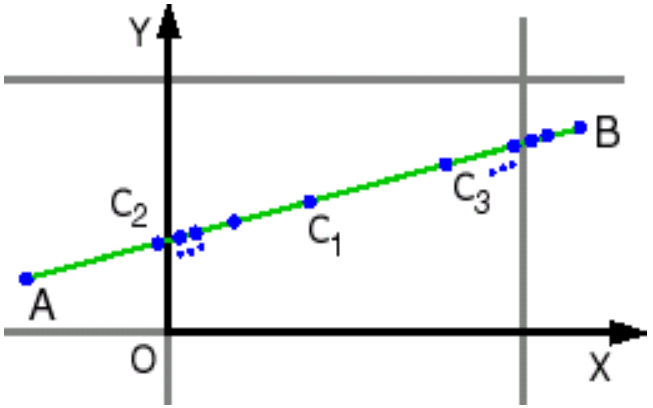
- вычислить коды концов отрезка;
- проанализировать условие тривиальной видимости отрезка;
- проанализировать условие тривиальной невидимости отрезка;
- пока результирующий отрезок не удовлетворяет условиям тривиальной видимости / невидимости:
  - в качестве начальной точки выбрать конец отрезка, лежащий вне окна;
  - определить точку пересечения отрезка с той стороной отсекателя, для которой установлен соответствующий бит в коде начальной точки;
  - отбросить часть отрезка от начальной точки до точки пересечения;
  - вычислить код для точки пересечения;
  - проанализировать условия тривиальной видимости / невидимости.



Алгоритм разбиения средней точкой (Спрулл-Сазерленд)

- точка пересечения со стороной отсекателя ищется путем рекурсивного разбиения отрезка средней точкой (алгоритм эффективен для аппаратной реализации);
- алгоритм сводится к нахождению для каждого из концов отрезка наиболее удаленной видимой точки (в общем случае – 2 двоичных поиска):
  - если отрезок тривиально невидим, то он игнорируется и процедура завершается;
  - если концевая точка видима, она будет наиболее удаленной видимой точкой, и процедура поиска завершается;
  - определяется P_m – средняя точка отрезка P_1P_2:
    - если отрезок вырождается (средняя точка совпадает с концами отрезка с заданной точностью, например, в 1 пиксель), то процедура завершается (точка найдена, необходимо определить ее видимость);
    - если P_mP_2 тривиально невидим, то продолжить процедуру с отрезком P_1P_m;
    - иначе продолжить процедуру с отрезком P_mP_2





Определение положения точки относительно прямой (плоскости)

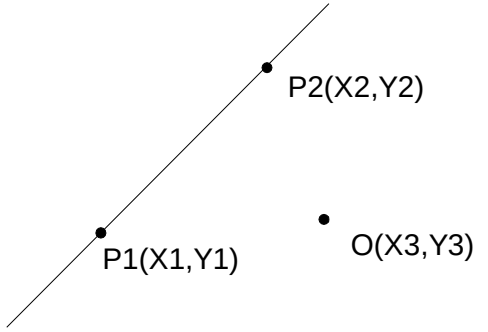
- через уравнение (неявное задание) ориентированной прямой:

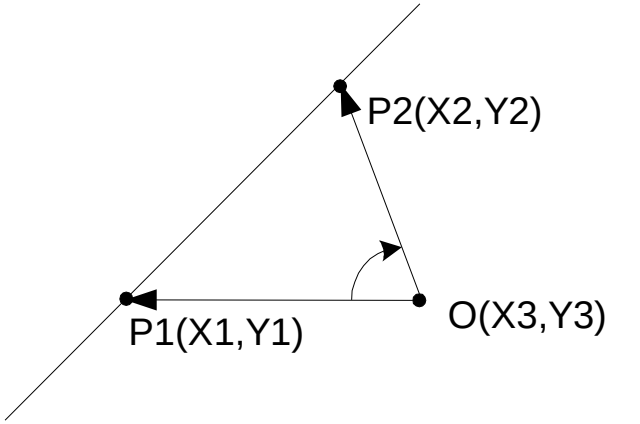
  $F(x,y)=(y2-y1)*(x-x1)-(x2-x1)*(y-y1)=0$

  – если F(x3,y3)=0, то точка O(x3,y3) лежит на прямой;
  – если F(x3,y3)>0, то точка O(x3,y3) расположена справа от прямой;
  – если F(x3,y3)<0, то точка O(x3,y3) расположена слева от прямой;
- через скалярное произведение нормали к отрезку и вектора, направленного из произвольной точки на отрезке в тестируемую точку (анализируется знак скалярного произведения):

  $\mathbf{N} = (y2-y1,-(x2-x1))$

  $F(x3,y3)=(\mathbf{N}, \mathbf{P_1O})$
- через векторное произведение векторов, направленных из тестируемой точки в вершины отрезка (анализируется знак координаты *z* векторного произведения).





$$OP_1 \times OP_2 = \begin{vmatrix} X1-X3 & Y1-Y3 & 0 \\ X2-X3 & Y2-Y3 & 0 \\ i & j & k \end{vmatrix}$$

Пересечение прямых и плоскостей

- пересечение двух отрезков прямых:

$$A + \mathbf{b}t = C + \mathbf{d}u.$$

$$\mathbf{b}t = \mathbf{c} + \mathbf{d}u.$$

$$t = \frac{\mathbf{d}^{\perp} \cdot \mathbf{c}}{\mathbf{d}^{\perp} \cdot \mathbf{b}}. \qquad u = \frac{\mathbf{b}^{\perp} \cdot \mathbf{c}}{\mathbf{d}^{\perp} \cdot \mathbf{b}}$$

$$I = A + \left(\frac{\mathbf{d}^{\perp} \cdot \mathbf{c}}{\mathbf{d}^{\perp} \cdot \mathbf{b}} \right) \mathbf{b} \text{ (точка пересечения).}$$

  - для отрезков производится проверка $t,u \in [0,1]$

- пересечение прямой и плоскости:

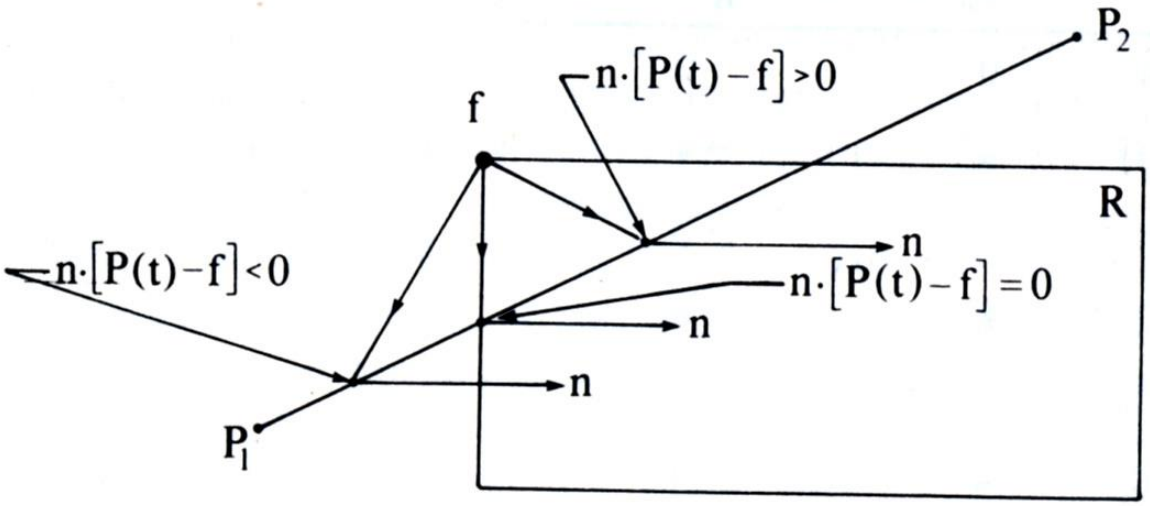
$$\mathbf{n}(A + \mathbf{c}t_{\text{hit}} - B) = 0.$$

$$\mathbf{n}(A - B) + \mathbf{n} \cdot \mathbf{c}t_{\text{hit}} = 0,$$

$$t_{hit} = \frac{\mathbf{n}(B - A)}{\mathbf{n} \cdot \mathbf{c}}$$

  - для отрезка производится проверка $t \in [0,1]$

Параметрическое отсечение



$$P(t) = P_1 + (P_2 - P_1)\,t,\ 0 \leqslant t \leqslant 1.$$

$$\mathbf{n}_i \cdot [\mathbf{P}(\mathbf{t}) - \mathbf{f}_i] \qquad i = 1,\ 2,\ 3, \ldots$$

$$\mathbf{n}_i \cdot [\mathbf{P}_1 + (\mathbf{P}_2 - \mathbf{P}_1)\mathbf{t} - \mathbf{f}_i] = 0$$

$$\mathbf{n}_i \cdot [\mathbf{P}_1 - \mathbf{f}_i] + \mathbf{n}_i \cdot [\mathbf{P}_2 - \mathbf{P}_1]\mathbf{t} = 0$$

$$\mathbf{D} = \mathbf{P}_2 - \mathbf{P}_1$$
$$\mathbf{w}_i = \mathbf{P}_1 - \mathbf{f}_i$$

$$\mathbf{t}(\mathbf{n}_i \cdot \mathbf{D}) + \mathbf{w}_i \cdot \mathbf{n}_i = 0$$

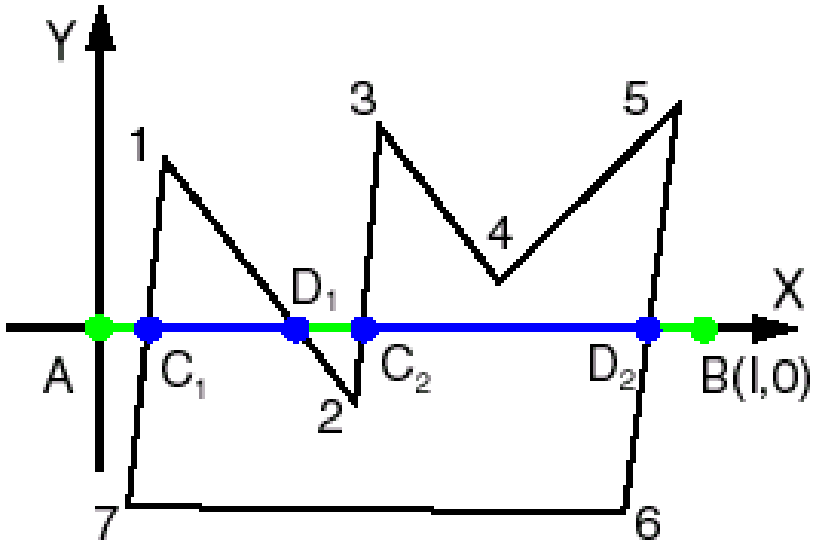
$$t = -\frac{\mathbf{w}_i \cdot \mathbf{n}_i}{\mathbf{D} \cdot \mathbf{n}_i} \qquad \mathbf{D} \neq 0 \qquad i = 1, 2, 3, \ldots$$

Алгоритм Кируса-Бека (Cyrus-Beck)

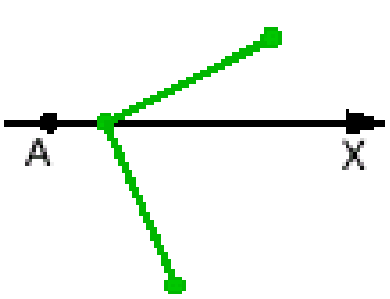
- реализует параметрическое отсечение выпуклым окном;
- инициализация: вычислить **D**, $t_н=0$, $t_в=1$;
- в цикле по ребрам отсекателя:
  - если $\mathbf{D}*\mathbf{n}_i = 0$ (отрезок параллелен ребру отсекателя):
    - если $\mathbf{w}_i*\mathbf{n}_i < 0$ – отрезок полностью лежит вне окна;
    - иначе – перейти к следующему ребру;
  - если $\mathbf{D}*\mathbf{n}_i > 0$ - поиск нижней границы параметра t (точки «входа» отрезка внутрь области): $t_н = \max(t, t_н)$, если $t \leq 1$;
  - если $\mathbf{D}*\mathbf{n}_i < 0$ - поиск верхней границы параметра t (точки «выхода» отрезка из области): $t_в = \min(t, t_в)$, если $t \geq 0$;
- видимая область отрезка: $[t_н;t_в]$ (если $t_н = \max(t_{in}) \leq t_в = \min(t_{out})$)

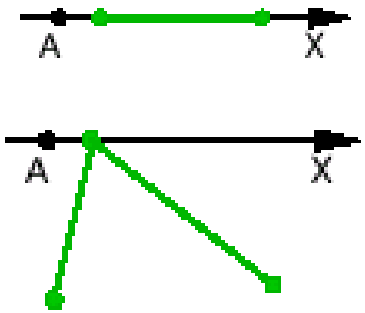
Отсечение невыпуклым многоугольником без самопересечений

- создать список точек пересечения отрезка с многоугольником – для каждого ребра отсекателя возможны 4 случая:
  - обе вершины ребра лежат по одну сторону от отрезка → пересечений нет;
  - вершины ребра лежат по разные стороны от отрезка → есть пересечение;
  - ребро лежит на прямой, содержащей отрезок → пересечений нет;
  - при прохождении отрезка через вершину отсекателя пересечение учитывается, только если смежные с вершиной ребра лежат по разные стороны от отрезка;
- результирующий набор полученных точек пересечения (дополненный конечными точками отрезка) упорядочивается, и пары точек пересечения формируют отсеченные сегменты.

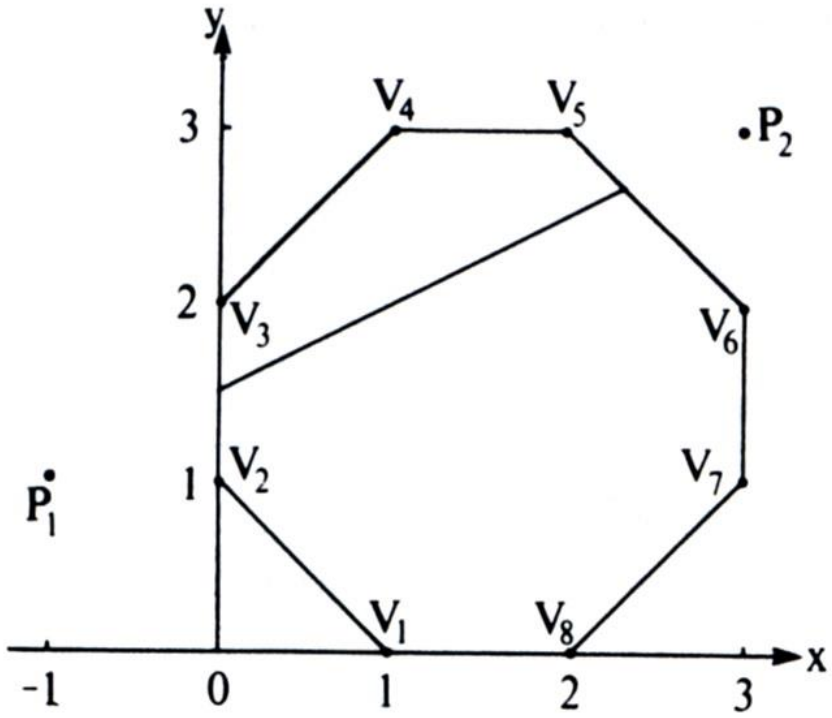


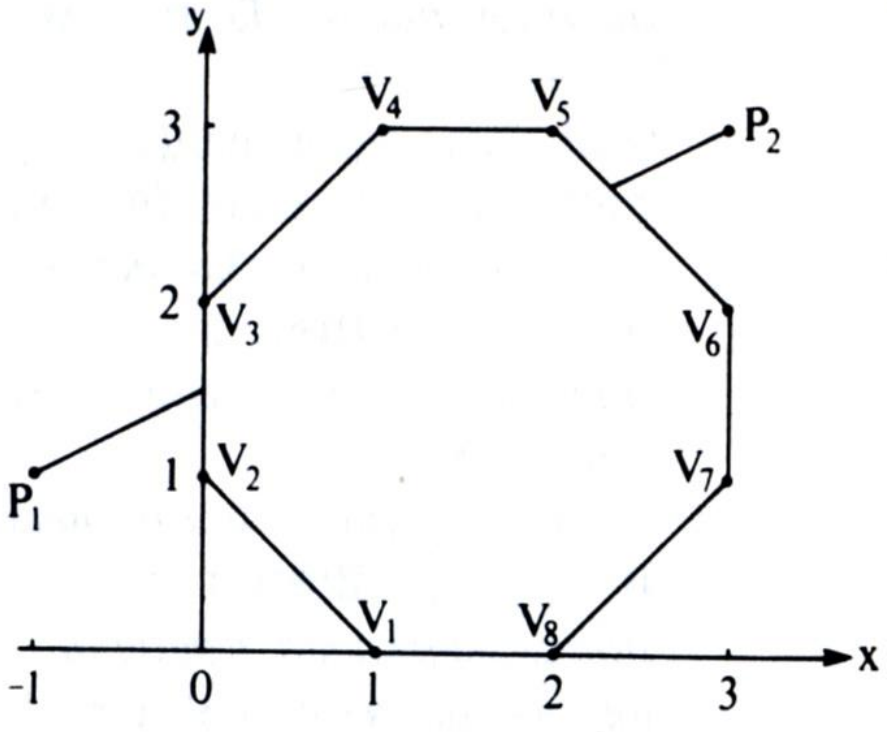
[0,1] ∩ ([t1,t2] ∪ [t3,t4] ∪ …)





Внутреннее и внешнее отсечение

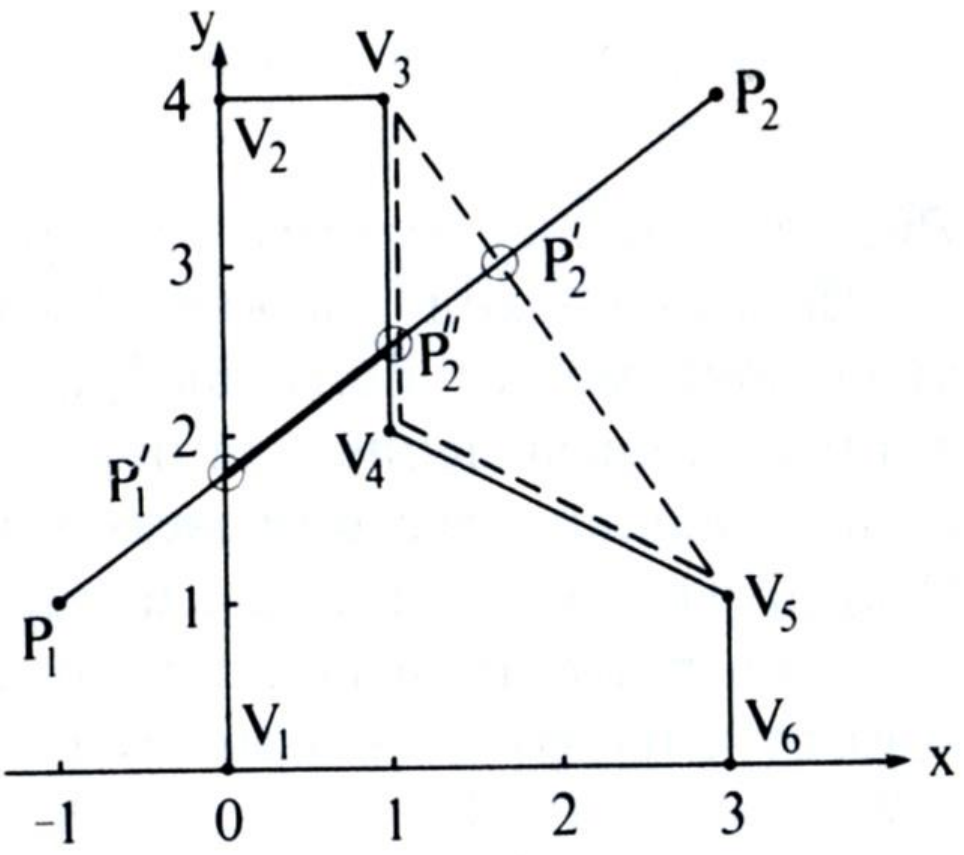




- внутреннее отсечение: $t \in [t_н;t_в]$;
- внешнее отсечение: $t \in [0;t_н] \cup [t_в;1]$.

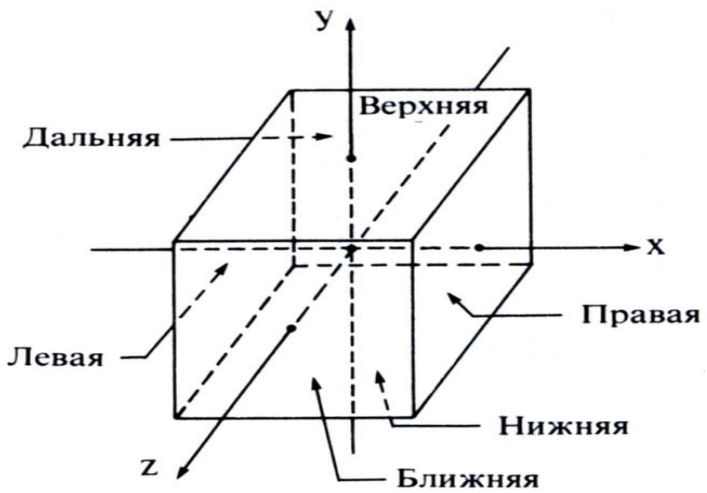
Отсечение невыпуклым отсекателем

- комбинация двух процедур отсечения:
  - внутреннее отсечение отсекателем, дополненным до выпуклого многоугольника;
  - внешнее отсечение дополнением вогнутого отсекателя до выпуклого.

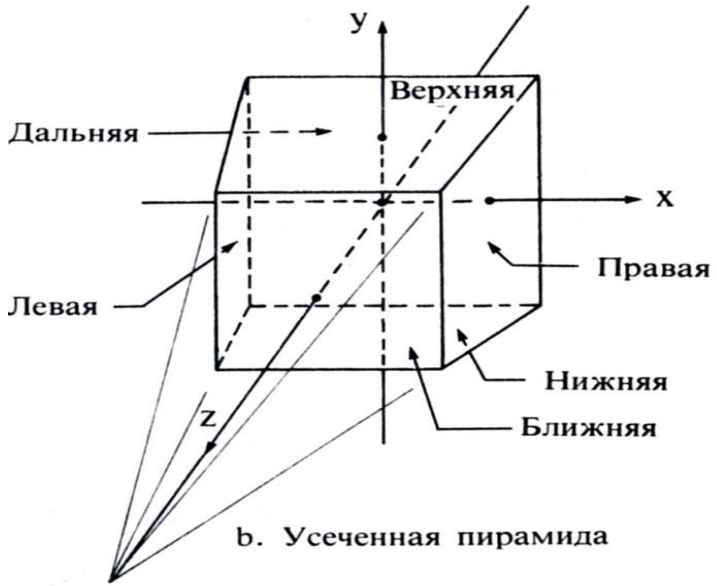


Трехмерное отсечение

- чтобы учесть преобразование проецирования, отсечение должно производиться в однородных координатах, а любой объем наблюдения может быть приведен к нормированному кубу;
- алгоритмы трехмерного отсечения:
  - трехмерный алгоритм Коэна-Сазерленда;
  - трехмерный алгоритм разбиения средней точкой;
  - трехмерный алгоритм Кируса-Бека.

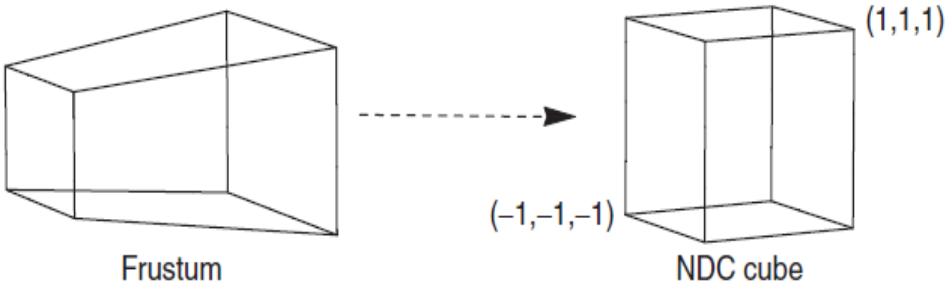


a. Параллелепипед



b. Усеченная пирамида

Отсечение и объём видимости



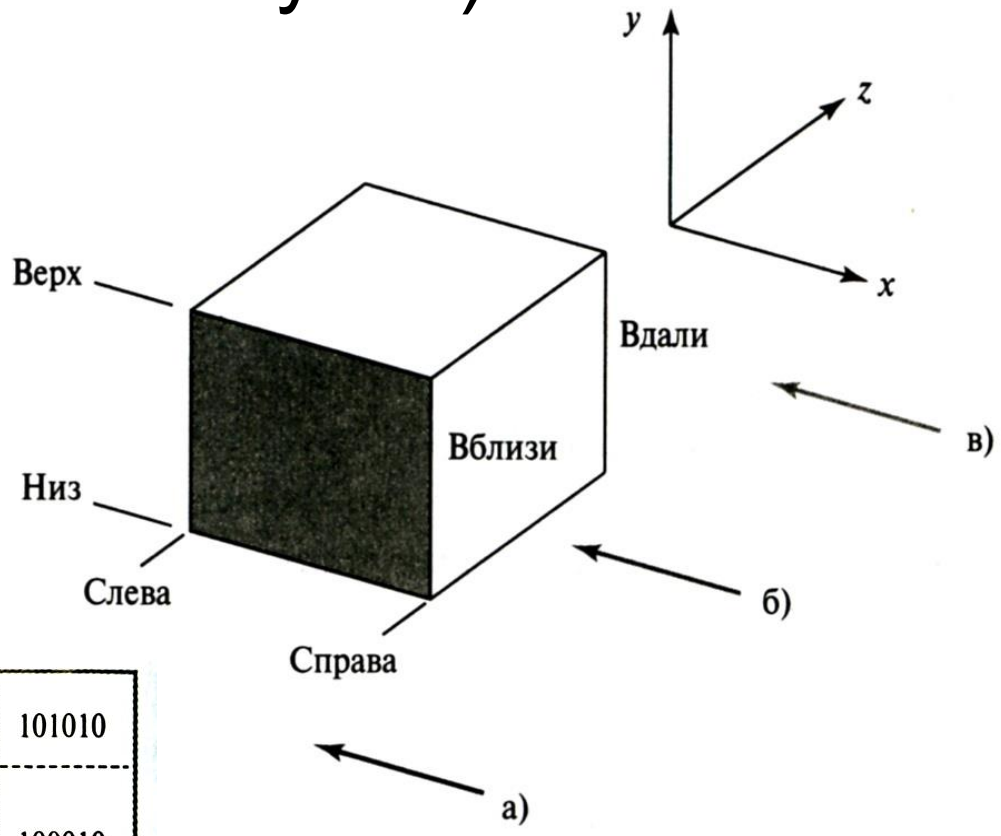
	Normalized	Camera
Near	$(0, 0, 1, 1)$	$M_2 + M_3$
Far	$(0, 0, -1, 1)$	$-M_2 + M_3$
Top	$(0, -1, 0, 1)$	$-M_1 + M_3$
Bottom	$(0, 1, 0, 1)$	$M_1 + M_3$
Left	$(1, 0, 0, 1)$	$M_0 + M_3$
Right	$(-1, 0, 0, 1)$	$-M_0 + M_3$

- процедура отсечения является одним из этапов удаления тех граней объектов, которые являются невидимыми, а именно – тех, которые не попадают в объём видимости;
- ключевая операция – определение положения точки относительно плоскости;
- отсечение может производиться после проективного преобразования, но более эффективно – в видовых координатах (если M – матрица преобразования, то $(M^{-1})^T$ – матрица преобразования нормалей, соответственно, M_{per}^T задаёт преобразование в видовые координаты нормалей плоскостей, задающих объём видимости, в таблице M_i – *i*-я строка M_{per});
- возможна оптимизация за счёт использования охватывающих оболочек (bounding volumes) – параллелепипедов, сфер и т.п.

Вычисление кодов концов отрезка (трехмерный случай)

011001	011000	011010
010001	010000	010010
010101	010100	010110

а) коды области перед ближней плоскостью



001001	001000	001010
000001	000000	000010
000101	000100	000110

б) коды области между ближней и дальней плоскостью

101001	101000	101010
100001	100000	100010
100101	100100	100110

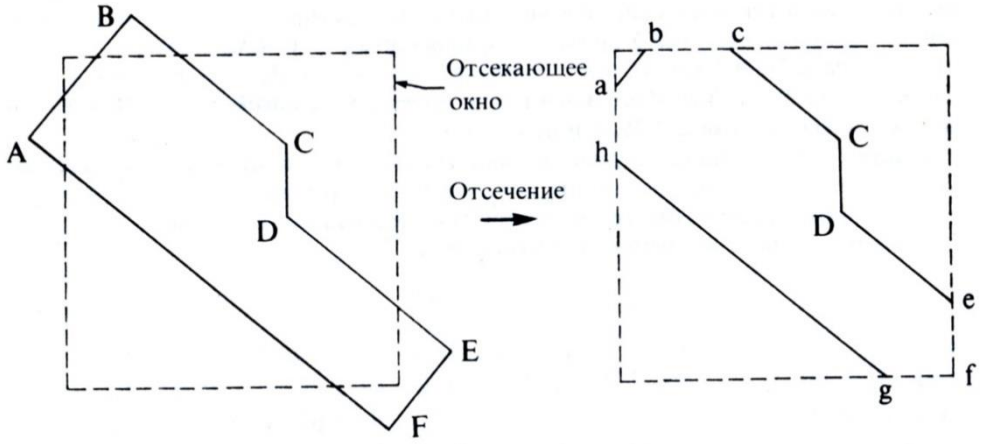
в) коды области за дальней плоскостью

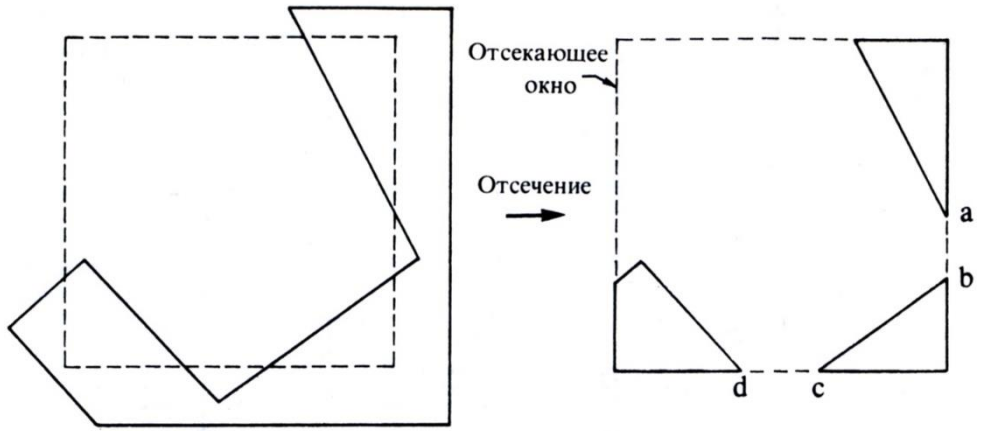
Отсечение многоугольников

- алгоритм Сазерленда-Ходжмена (последовательного отсечения произвольного многоугольника выпуклым отсекателем);
- алгоритм Вейлера-Азертона (отсечения многоугольника произвольным отсекателем);
- модификации алгоритма Вейлера-Азертона для реализации булевских операций над многоугольниками.

Отсечение многоугольников: процедура отсечения

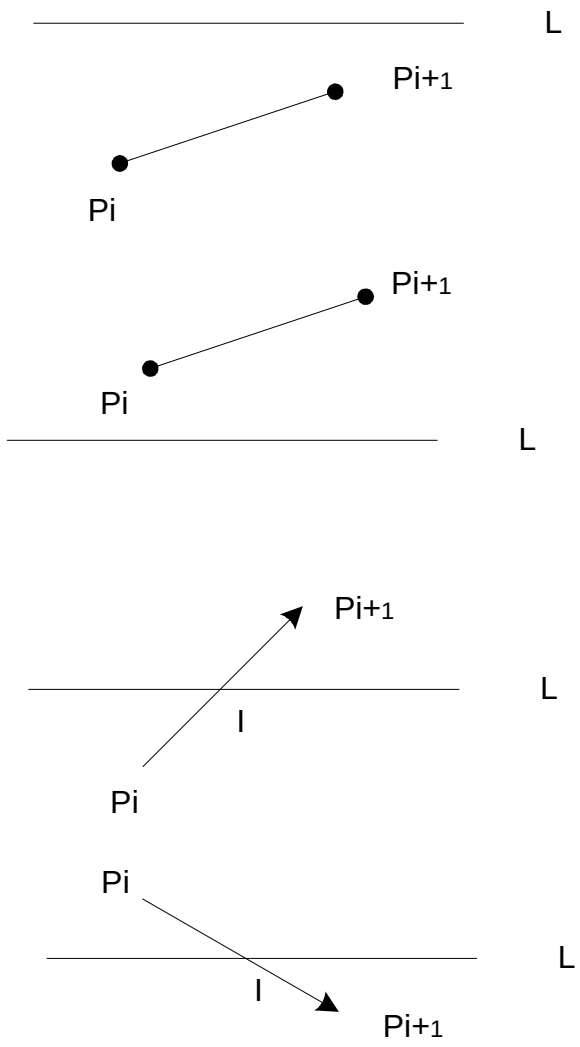
- входные данные:
  - отсекаемый многоугольник;
  - отсекатель;
- результат: набор из одного или нескольких отсеченных многоугольников;
- проблемы:
  - генерация дополнительных ребер для замыкания контуров многоугольников;
  - удаление «ложных» ребер, возникших в процессе отсечения.





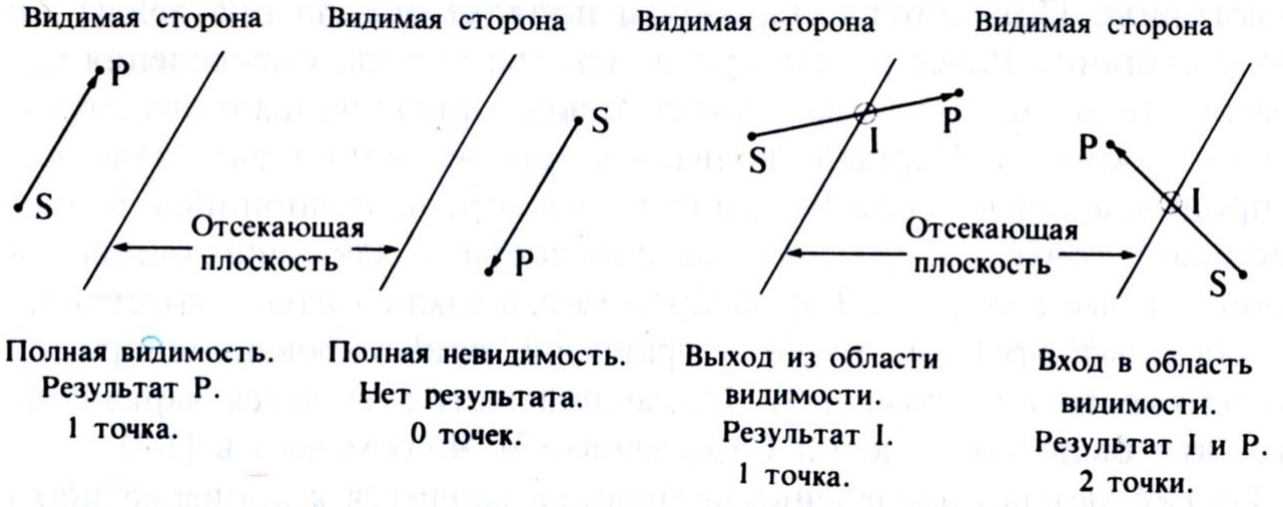
Отсечение многоугольника выпуклым отсекателем

- для любого ребра выпуклого отсекателя его внутренняя область лежит по одну сторону от прямой, содержащей данное ребро;
- следовательно, возможно последовательное (поочередное) отсечение многоугольника вдоль прямых, содержащих ребра отсекателя;
- для ориентированного ребра отсекаемого многоугольника возможны четыре варианта его расположения относительно ребра отсекателя и полуплоскости, соответствующей его внутренней области:
  - отрезок целиком лежит во внутренней полуплоскости;
  - отрезок целиком лежит во внешней полуплоскости;
  - отрезок выходит за границу внутренней полуплоскости;
  - отрезок входит во внутреннюю область.



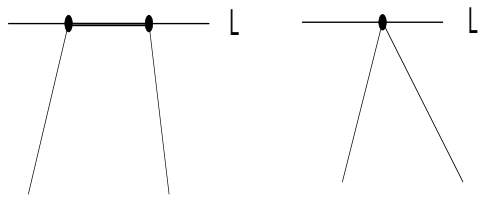
Алгоритм Сазерленда-Ходжмана (Sutherland-Hodgman, 1974)

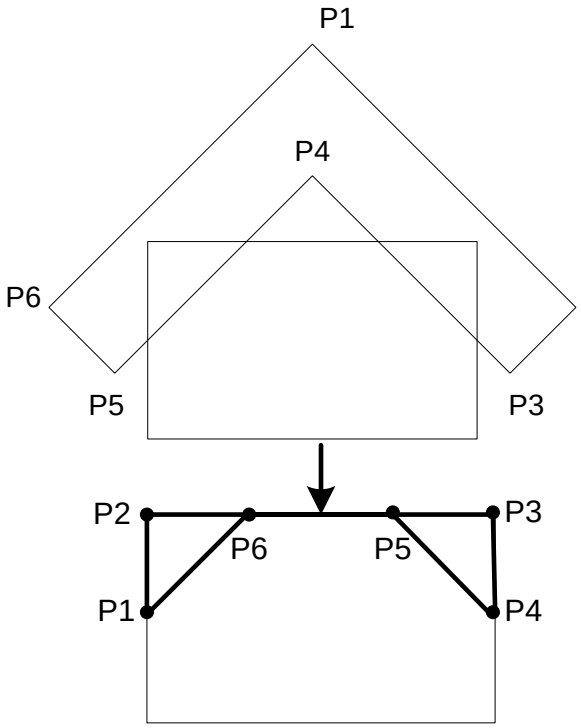
- реализует последовательное отсечение произвольного многоугольника выпуклым отсекателем;
- цикл по ребрам отсекателя:
  - формирование нового списка вершин путем последовательного рассмотрения текущего списка ребер отсекаемого многоугольника, для каждого ребра SP:
    - полная видимость: результат ← P;
    - полная невидимость: результат ← ∅;
    - выход ребра из области видимости: результат ← I;
    - вход ребра в область видимости: результат ← {I, P};
  - переход к следующему ребру отсекателя.



Алгоритм Сазерленда-Ходжмана: особые случаи и модификации

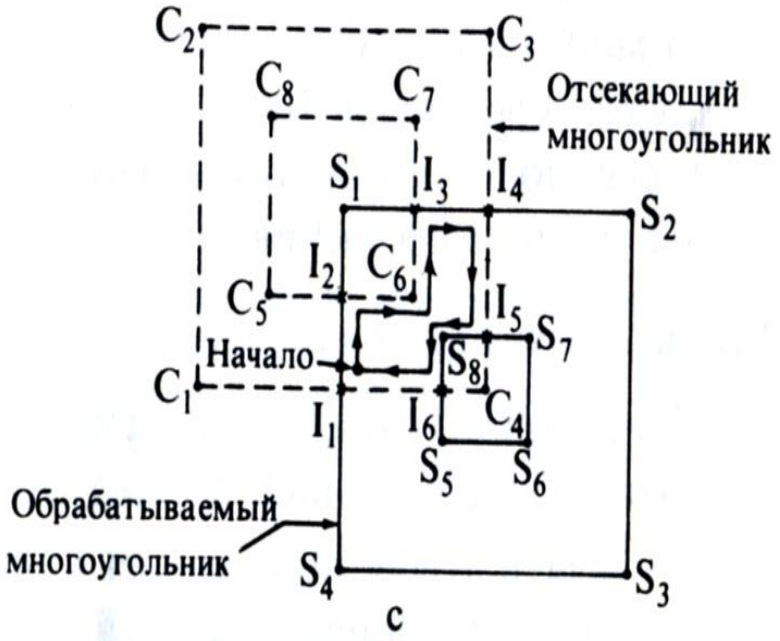
- особые случаи:
  - определение пересечений отсекаемого многоугольника и отсекателя: касания не считаются пересечениями;
  - возникновение ложных ребер:
    - исключить данную возможность путем разбиения исходного вогнутого многоугольника;
    - анализ точек пересечения вдоль каждого из ребер отсекателя (по завершении отсечения по данному ребру), разбиение их на пары с учетом точек касания и модификация списка ребер;
- модификация алгоритма: последовательное отсечение ребер (вершин) многоугольника всеми границами отсекателя.





Алгоритм Вейлера-Азертона (Weiler-Atherton, 1977)

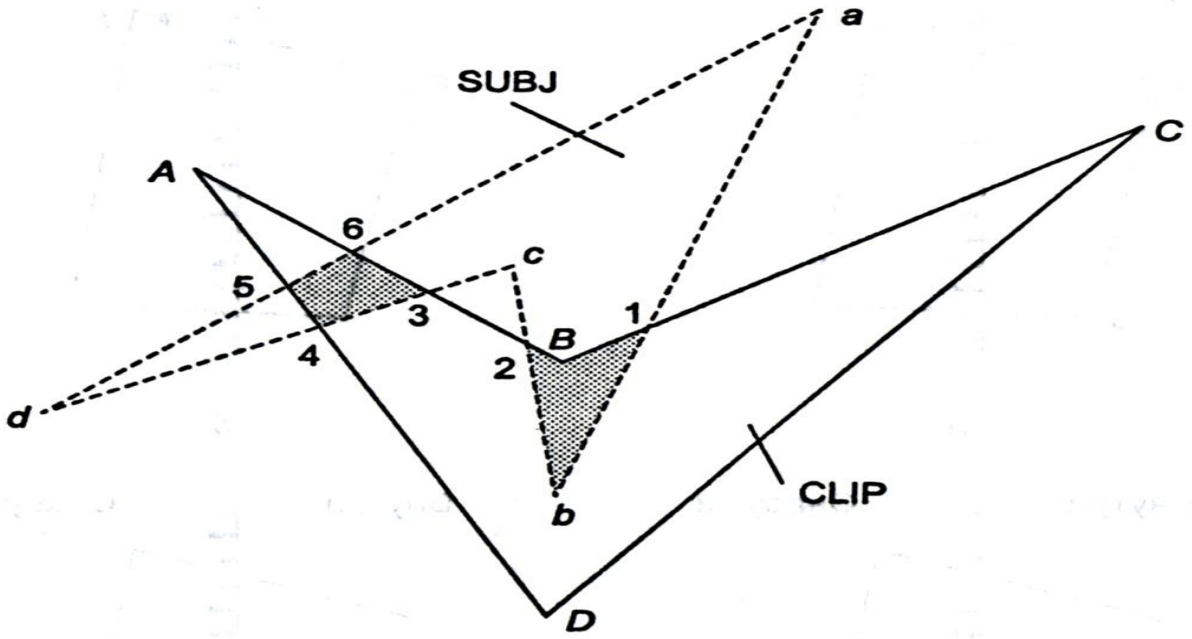
- отсечение произвольного многоугольника произвольным отсекателем, возможно наличие отверстий;
- внешние границы и границы отверстий задаются контурами с противоположными направлениями обхода (прямое направление обхода внешнего контура – по часовой стрелке, внутреннего контура (отверстия) – против часовой стрелки);
- возможно внешнее/внутреннее отсечение (interior/exterior clipping), а также реализация булевых операций над многоугольниками.

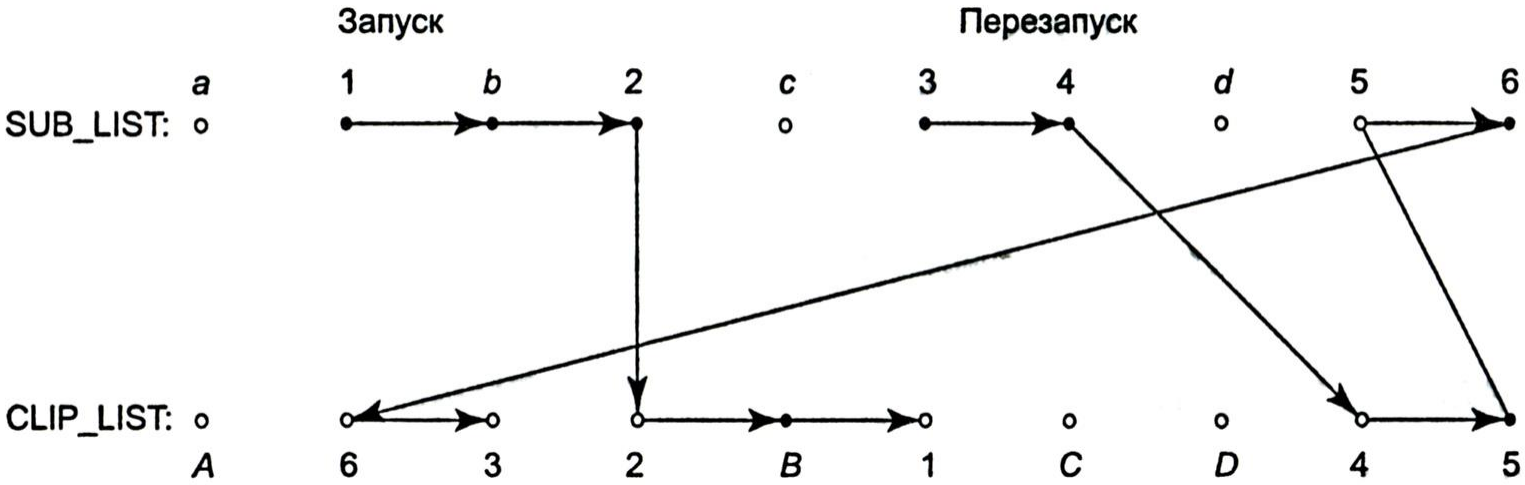


Алгоритм Вейлера-Азертона (внутреннее отсечение)

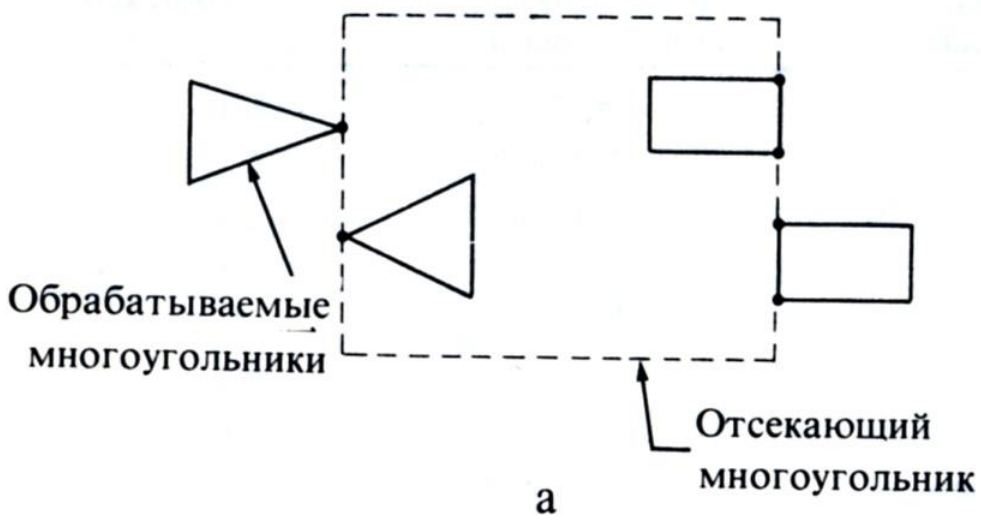
1. определение точек пересечения ребер отсекателя и отсекаемого многоугольника, их классификация на точки входа в отсекатель и точки выхода из отсекателя;
2. дополнения списков вершин отсекаемого многоугольника и отсекателя точками пересечения;
3. для каждой необработанной вершины из списка входов:
   a. обход отсекаемого многоугольника в прямом направлении до точки выхода;
   b. обход отсекателя в прямом направлении до точки входа;
   c. если достигнута исходная точка входа – 4, иначе – 3a;
4. если все точки входа исчерпаны – завершение, иначе – 3;
5. обработка непересекающихся контуров:
   a. игнорируются границы отсекателя, лежащие вне отсекаемого многоугольника, а также границы отсекаемого многоугольника вне отсекателя;
   b. границы отсекателя, лежащие внутри отсекаемого многоугольника, а также границы отсекаемого многоугольника внутри отсекателя, переносятся в результирующий набор контуров.
6. все отверстия (внутренние границы) сопоставляются с соответствующими внешними границами.

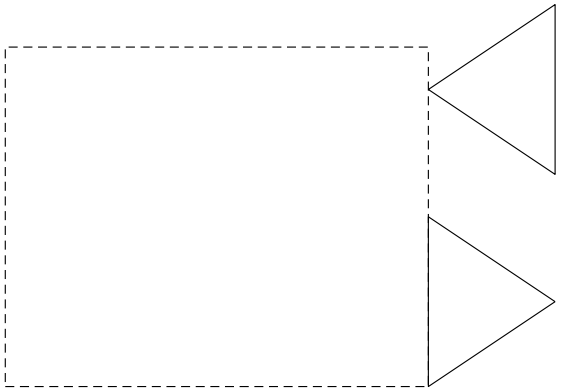
Алгоритм Вейлера-Азертона: пример



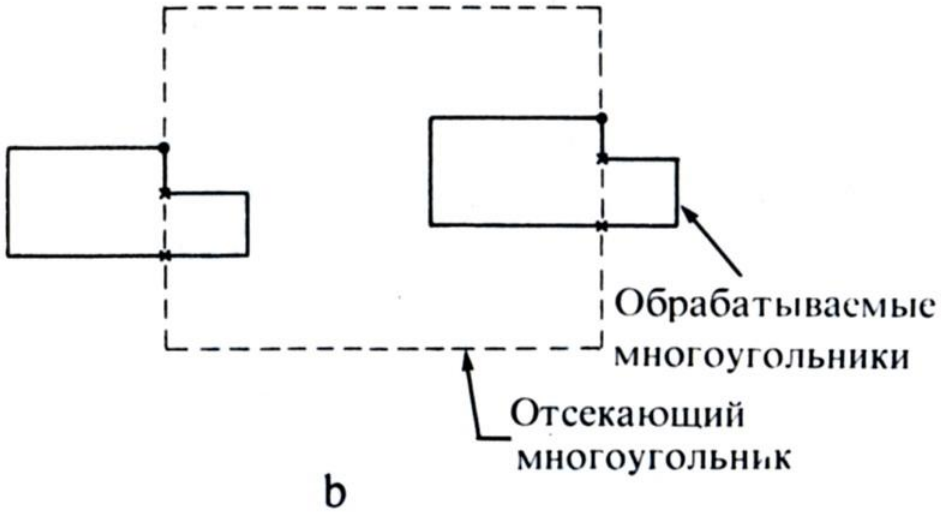


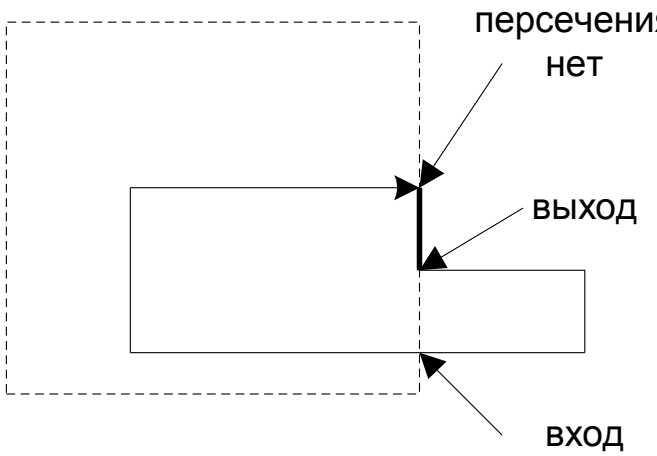
Алгоритм Вейлера-Азертона: особые случаи при классификации и вычислении пересечений





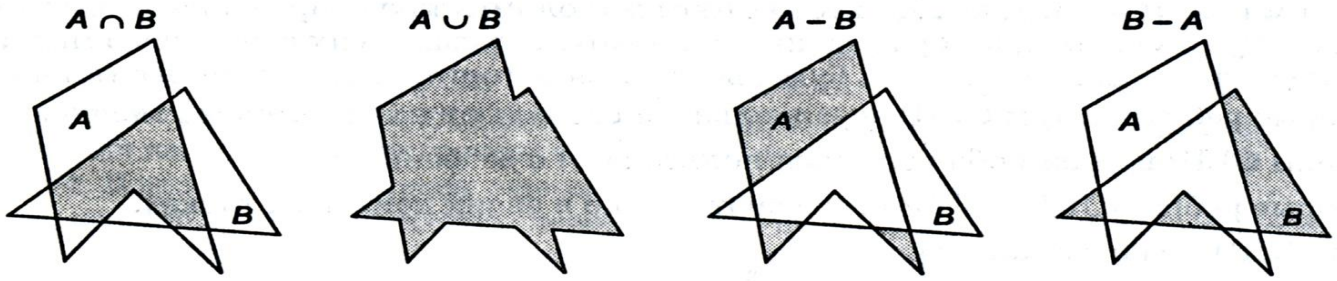
Если касание, то считаем,
что пересечения нет





Алгоритм Вейлера-Азертона и реализация булевых операций над многоугольниками

- пересечение: стандартный алгоритм внутреннего отсечения:
  - стартовая точка – точка входа;
  - прямой порядок обхода отсекаемого многоугольника;
  - прямой порядок обхода отсекателя;
- объединение:
  - стартовая точка – точка выхода;
  - прямой порядок обхода отсекаемого многоугольника;
  - прямой порядок обхода отсекателя;
- разность отсекаемого многоугольника и отсекателя: внешнее отсечение:
  - стартовая точка – точка выхода;
  - прямой порядок обхода отсекаемого многоугольника;
  - обратный порядок обхода отсекателя.



Дополнительные плоскости отсечения в OpenGL

- произвольно ориентированные плоскости, в дополнение к шести плоскостям отсечения объема видимости;
- определение параметров дополнительной плоскости отсечения (плоскость задается значениями коэффициентов A,B,C,D уравнения плоскости):

  void glClipPlane(GLenum *plane*, **const GLdouble ***_equation_ **)**
- активизация / отключение дополнительной плоскости отсечения:

  void glEnable(GL_CLIP_PLANE*i*)

  void glDisable(GL_CLIP_PLANE*i*)
- получение информации о дополнительных плоскостях отсечения

  GLboolean glIsEnabled(GL_CLIP_PLANE*i*)

  void glGetClipPlane(GLenum *plane*, **GLdouble ***_equation_ **)**
- определение максимального поддерживаемого количества дополнительных плоскостей отсечения:

  void glGetIntegerv(GL_MAX_CLIP_PLANES, *num*)

Лабораторная работа № 5

- Реализовать один из алгоритмов отсечения определенного типа в пространстве заданной размерности (в соответствии с вариантом).
- Ввод исходных данных каждого из алгоритмов производится интерактивно с помощью клавиатуры и/или мыши (за исключением особо оговоренных случаев).

Вопросы к экзамену

- Двумерное отсечение. Простой алгоритм отсечения отрезка. Алгоритм отсечения средней точкой.
- Алгоритм Коэна-Сазерленда двумерного отсечения отрезка. Обобщение алгоритма для трехмерного случая.
- Параметрическое отсечение. Алгоритм Кируса-Бека двумерного отсечения отрезка. Обобщение алгоритма для трехмерного случая.
- Внутреннее и внешнее отсечение. Отсечение отрезка невыпуклым многоугольником (без самопересечений).
- Алгоритм Сазерленда-Ходжмена отсечения многоугольников.
- Алгоритм Вейлера-Азертона отсечения многоугольников.