Машинная графика
Computer Graphics

Лекция 6.

«Отсечение на плоскости»

План лекции

- Постановка проблемы
- Отсечение точки
- Отсечение отрезка
- Алгоритм Коэна-Сазерленда
- Параметрическое задание отрезка
- Простое двухмерное отсечение

- Алгоритм разбиения средней точкой

Определение. Области применения

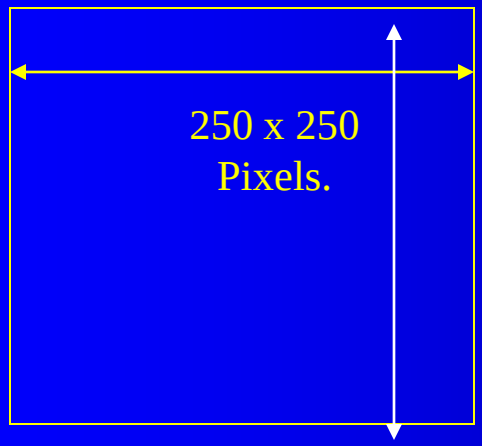
Отсечение - это процесс выделения некоторой части базы данных, описывающей изображение.

Основное применение алгоритмов отсечения - это отбор той информации, которая необходима для визуализации конкретной сцены или вида, как части более обширной обстановки.

Но кроме этого отсечение применяется в алгоритмах удаления невидимых линий и поверхностей, при построении теней, а также при формировании фактуры.

Отсечение растра

Окно в мировых координатах.

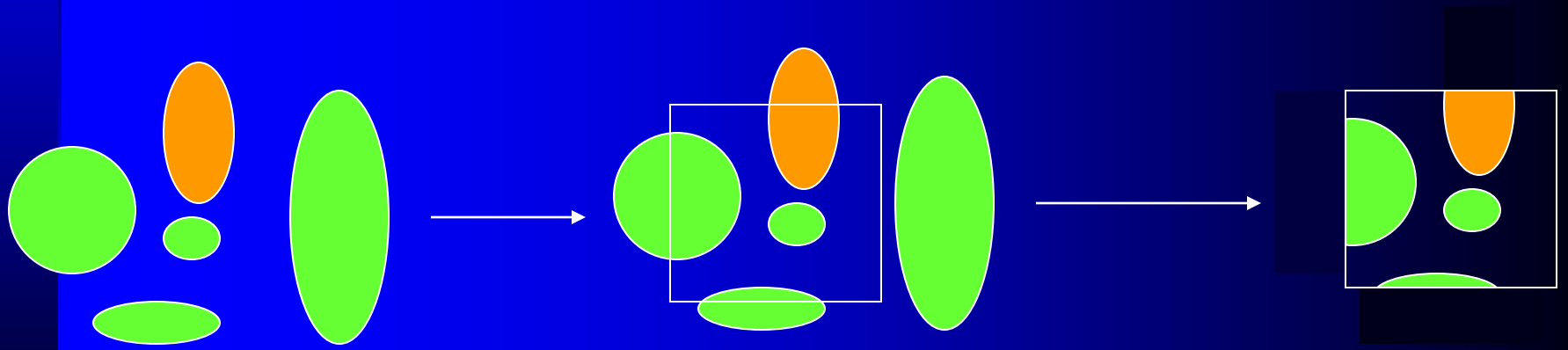


Вид в коорд. устройства вывода

Постановка проблемы

Отсечение растрового изображения требует лишь пересчёта координат для отображаемого фрагмента изображения.

С векторной моделью ситуация несколько иная. Каждый элемент сцены должен быть проверен – попадает он в кадр и если да, то полностью или частично.



Классификация алгоритмов

Алгоритмы отсечения бывают : двумерными и трехмерными.

Могут применяться как к регулярным, так и к нерегулярным областям и объемам. Под регулярными понимаются канонические области и объемы. В частности, к ним относятся прямоугольники и параллелепипеды со сторонами, параллельными осям координат.

Алгоритмы могут быть реализованы: аппаратно и программно.

Реализованные программно, зачастую оказываются недостаточно быстродействующими для приложений, ориентированных на процессы, протекающие в реальном времени. Поэтому трех- и двумерные алгоритмы отсечения реализуются аппаратными или микропрограммными средствами. В подобных реализациях обычно ограничиваются дву- или трехмерными отсекателями типовых форм.

Однако, с появлением сверхбольших интегральных схем (СБИС) открылись возможности для более общих реализации, позволяющих работать в реальном времени как с регулярными, так и с нерегулярными областями и телами.

Двухмерное отсечение точки

На рис. 1 показана плоская сцена и отсекающее окно регулярной формы. Окно задается левым (Л), правым (П), верхним (В) и нижним (Н) двумерными ребрами.

Регулярным отсекающим окном является прямоугольник, стороны которого параллельны осям координат объектного пространства или осям координат экрана. Целью алгоритма отсечения является определение тех точек, отрезков или их частей, которые лежат внутри отсекающего окна. Эти точки, отрезки или их части остаются для визуализации. А все остальное отбрасывается.

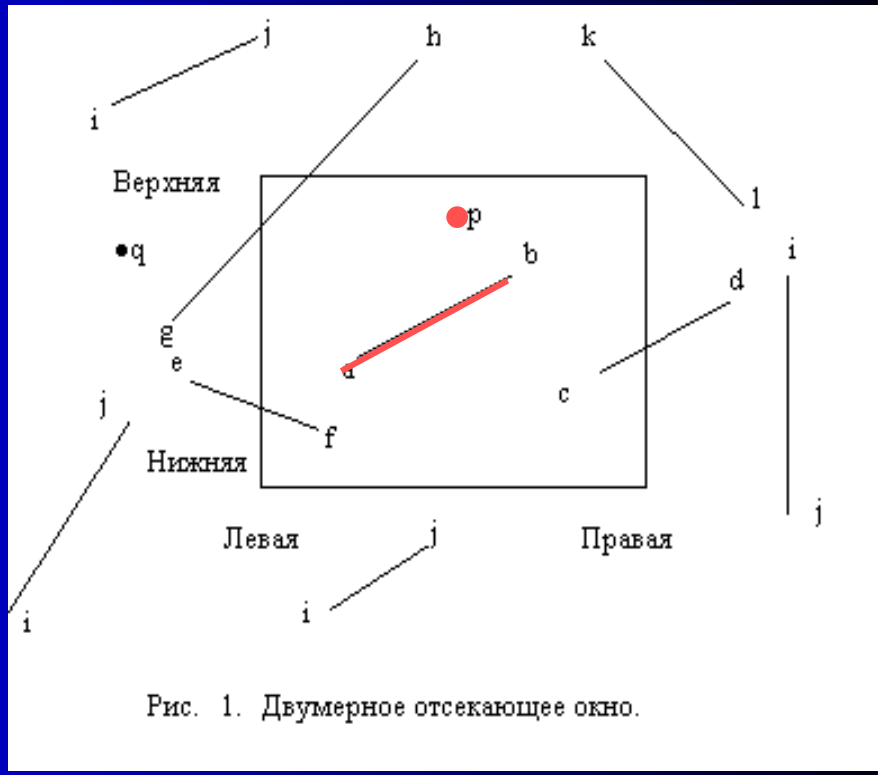


Рис. 1. Двумерное отсекающее окно.

Двухмерное отсечение точки

Точки, лежащие внутри отсекающего окна, удовлетворяют условию:

$x_л <= x <= x_п$ и $y_н <= y <= y_в$.

Если любое из неравенств не удовлетворяется, точка отсекается.

Отсечение отдельных точек применяется довольно редко, но в некоторых сценах, например, моделируемых «частицами» (облака, морская пена, дым, взрывы и т.п.) оно имеет место быть.

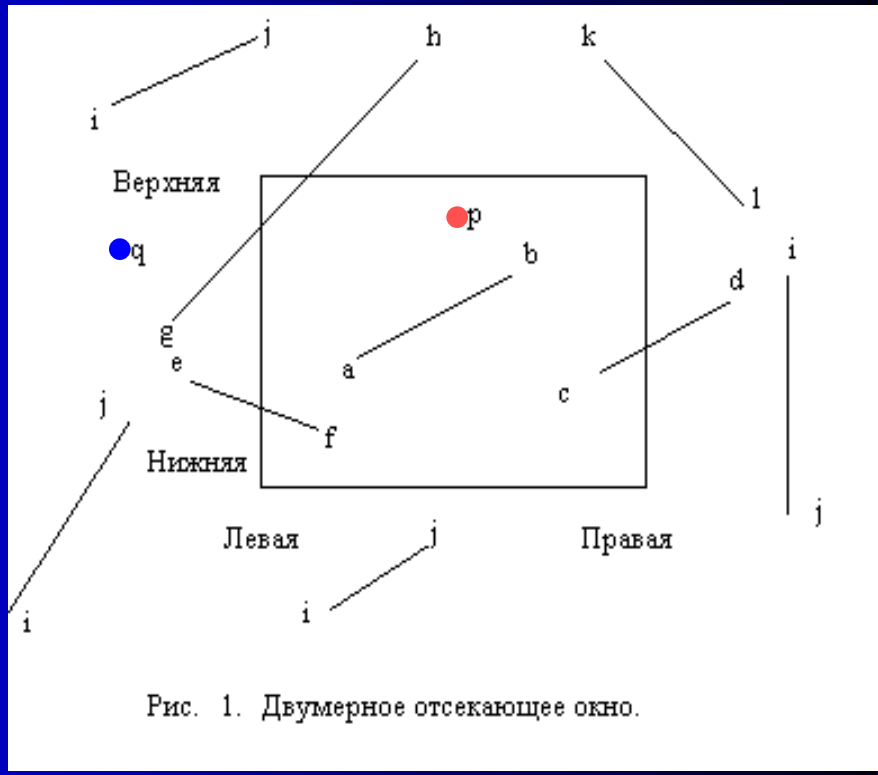


Рис. 1. Двумерное отсекающее окно.

Знак равенства в уравнениях показывает, что точки, лежащие на границе окна, считаются находящимися внутри него.

Двухмерное отсечение отрезка

Поскольку в обычных сценах или картинках необходимо отсекать большое число отрезков или точек, то эффективность алгоритмов отсечения представляет особый интерес. Во многих случаях подавляющее большинство точек или отрезков лежит целиком внутри или вне отсекающего окна.

Поэтому важно уметь быстро отбирать отрезки, подобные *ab*, или точки, подобные *p*, и отбрасывать отрезки, подобные *ij*, или точки, подобные *q* на рис.1.

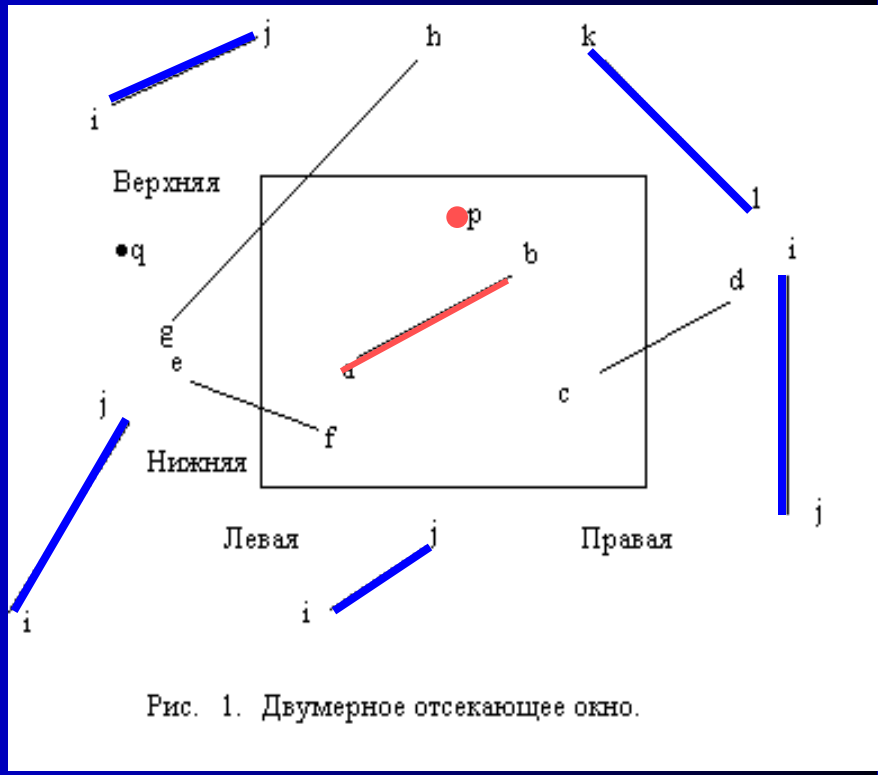


Рис. 1. Двумерное отсекающее окно.

Двухмерное отсечение отрезка

Отрезок лежит внутри окна и, следовательно, является видимым, если обе его концевые точки лежат внутри окна, например отрезок *ab* на рис. 1. Однако если оба конца отрезка лежат вне окна, то этот отрезок не обязательно лежит целиком вне окна, например отрезок *gh* на рис.1. Если же оба конца отрезка лежат справа, слева, выше или ниже окна, то этот отрезок целиком лежит вне окна, а значит, невидим. Проверка последнего условия устранит все отрезки, помеченные *ij* на рис.1. Но она не устранит ни отрезка *gh*, который видим частично, ни отрезка *kl*, который целиком невидим.

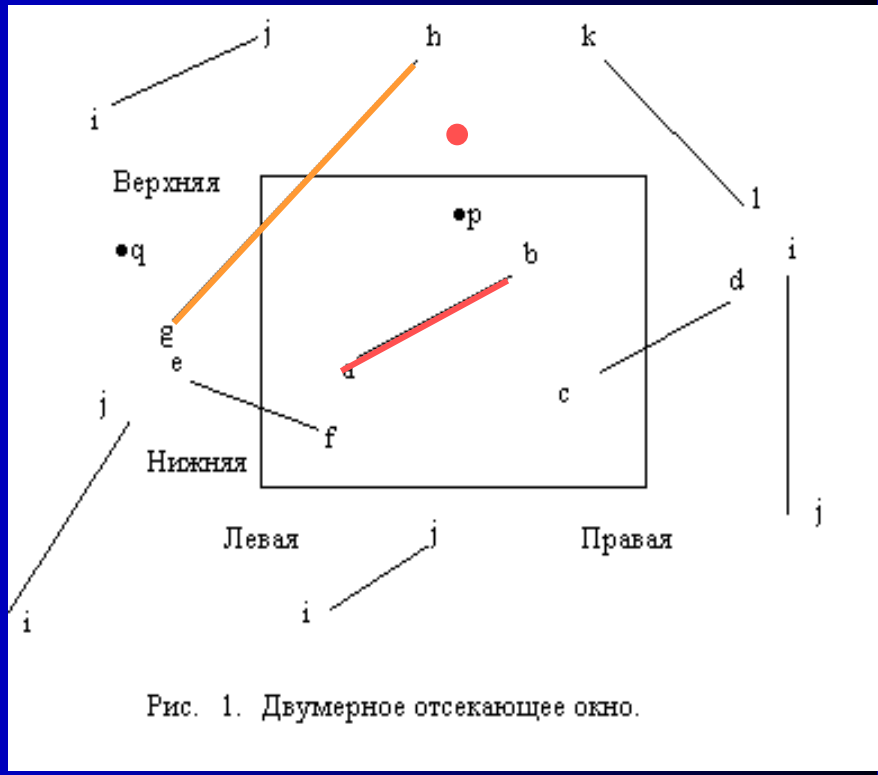


Рис. 1. Двумерное отсекающее окно.

Простой алгоритм определения полной видимости и тривиальной невидимости отрезка

Пусть *a* и *b* - концы отрезка, тогда можно предложить алгоритм, определяющий все полностью видимые и большинство невидимых отрезков.

a, b - концевые точки отрезка в координатном пространстве (x, y). Для каждого отрезка проверить полную видимость отрезка если одна из координат какого-нибудь конца отрезка находится вне окна, то отрезок не является полностью видимым.

if $x_a < x_л$ **or** $x_a > x_п$ **then** 1

if $x_b < x_л$ **or** $x_b > x_п$ **then** 1

if $y_a < y_н$ **or** $y_a > y_b$ **then** 1

if $y_b < y_н$ **or** $y_b > y_b$ **then** 1

Иначе: отрезок полностью видимый. Визуализировать отрезок

go to п.3.

Простой алгоритм определения полной видимости и тривиальной невидимости отрезка

Проверить полную невидимость отрезка если оба конца отрезка лежат слева, справа, сверху или снизу от окна, то факт невидимости отрезка тривиален.

if $x_a < x_л$ **and** $x_b < x_л$ **then 2**

if $x_a > x_п$ **and** $x_b > x_п$ **then 2**

п.1. **if** $x_a > x_в$ **and** $x_b > x_в$ **then 2**

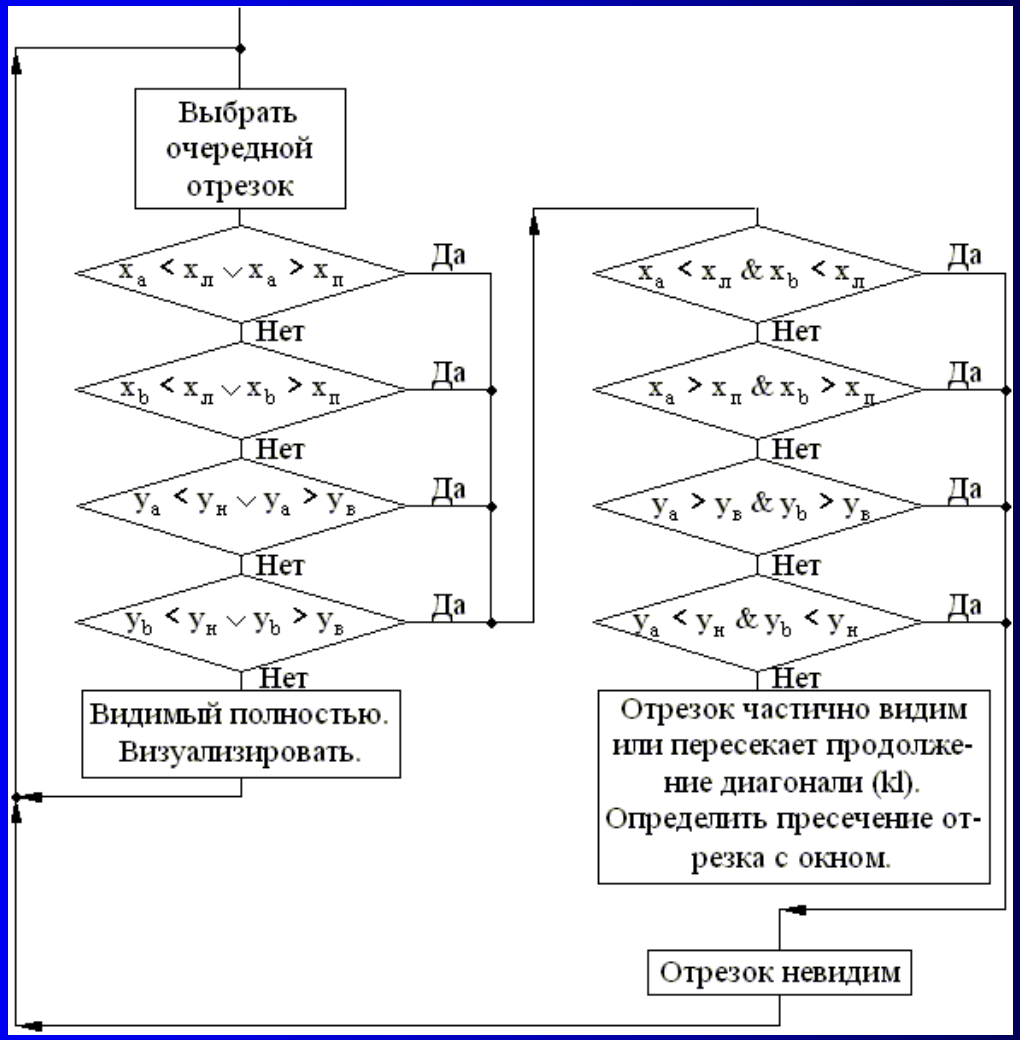
if $x_a < x_н$ **and** $x_b < x_н$ **then 2**

Иначе: отрезок частично видим или пересекает продолжение диагонали, оставаясь невидимым (kl). Определить пересечения отрезка с окном

п.2. Отрезок невидим

п.3 Переходит к следующему отрезку.

Блок-схема простого алгоритма определения полной видимости и тривиальной невидимости отрезка



Алгоритм Коэна-Сазерленда

Приведенные выше тесты полной видимости или невидимости и отрезков можно формализовать, используя метод Д. Коэна и А. Сазерленда. В этом методе для определения той из девяти областей, которой принадлежит конец ребра, вводится четырехразрядный (битовый) код. Коды этих областей показаны на рис. 3.

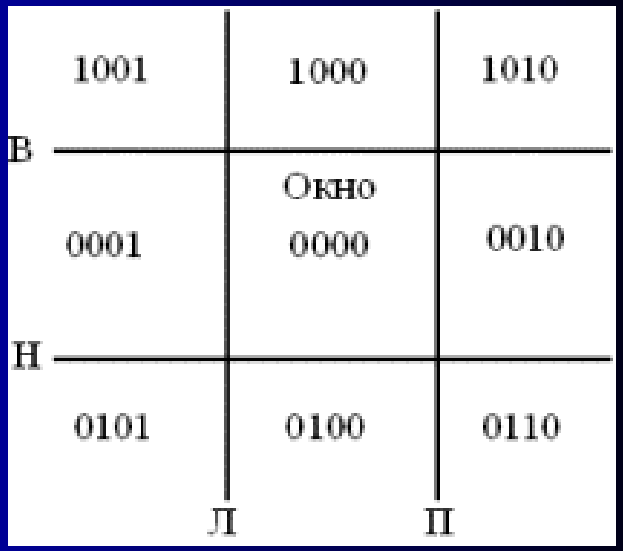
Крайний правый бит кода считается первым. В соответствующий бит заносится 1 при выполнении следующих условий:

для первого бита - если точка левее окна;

для второго бита - если точка правее окна;

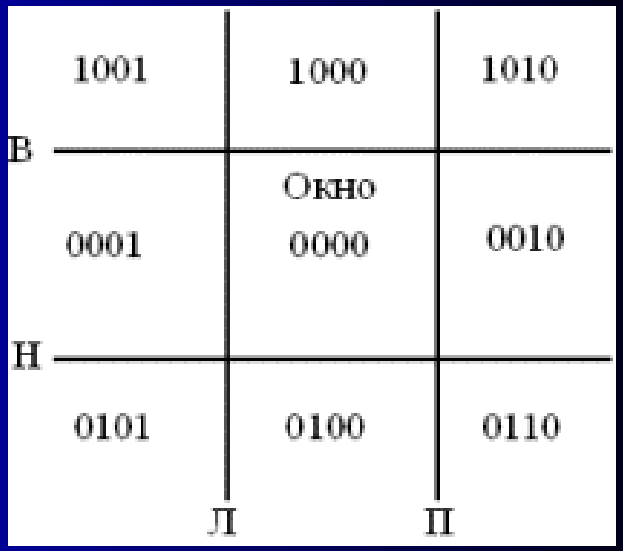
для третьего бита - если точка ниже окна;

для четвертого - если точка выше окна;

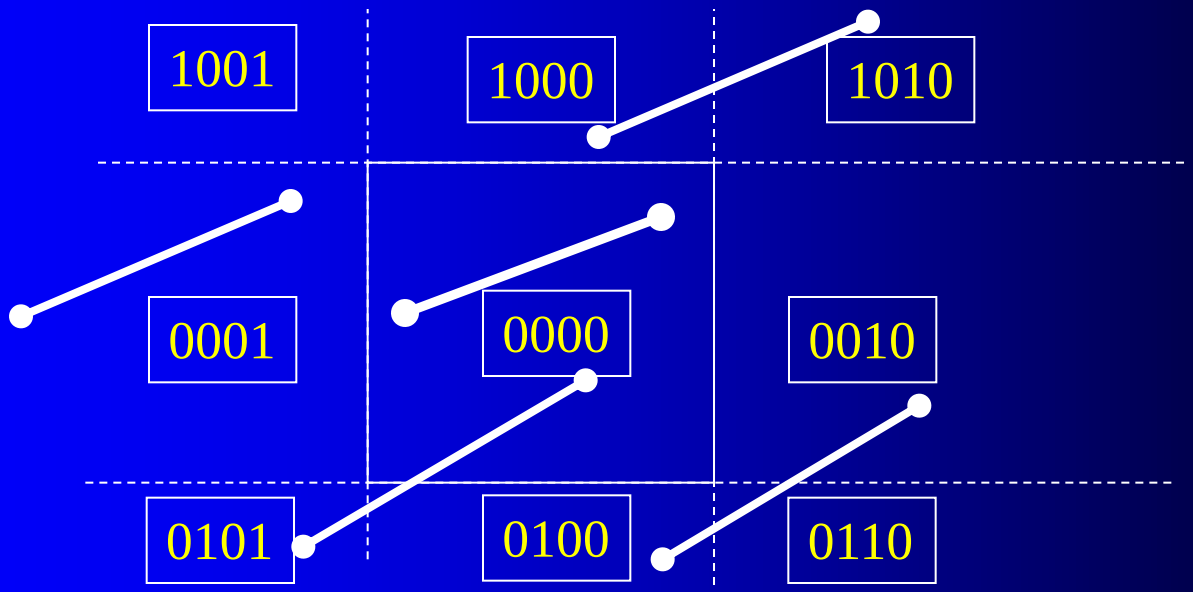


Алгоритм Коэна-Сазерленда

В противном случае в бит заносится нуль. Отсюда, очевидно, следует, что если коды обоих концов ребра равны нулю, то обе эти точки лежат внутри окна, и отрезок видимый. Коды концевых точек можно использовать также и для тривиального отбрасывания полностью невидимых отрезков. Если побитовое логическое произведение кодов концевых отрезка не равно нулю, то отрезок полностью невидим и его можно отбросить тривиально. Однако если логическое произведение равно нулю, то отрезок может оказаться целиком или частично видимым или даже целиком невидимым. Поэтому для определения полной видимости необходимо проверять значения кодов обоих концов отрезка по отдельности.



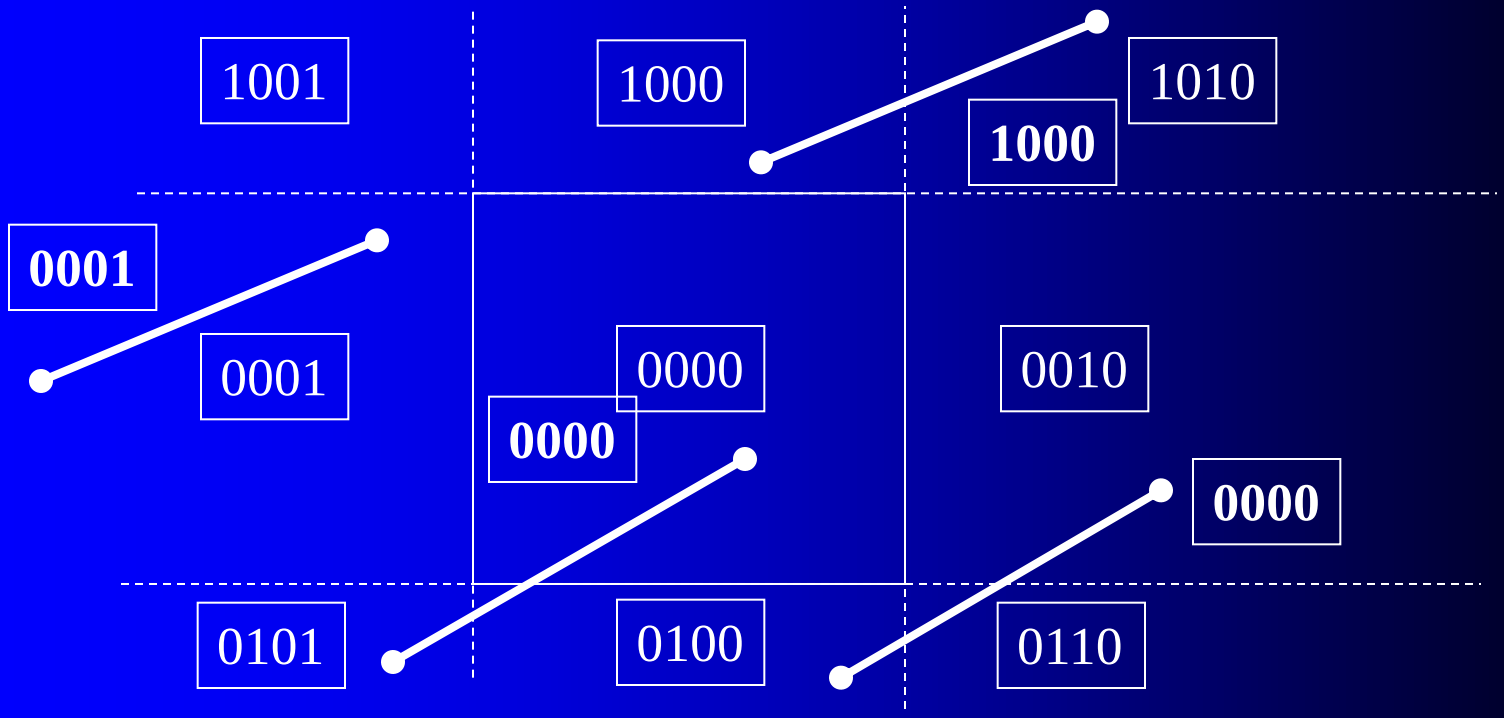
Алгоритм Коэна-Сазерленда



Если коды концов 0000, то отрезок полностью видим, иначе:

Произвести операцию логического И кодов концов отрезков

Алгоритм Коэна-Сазерленда



Если логическое И кодов концов не ноль, то
Отрезок полностью невидим - отбрасываем его.

Двухмерное отсечение отрезка

Результаты проверки отрезков.

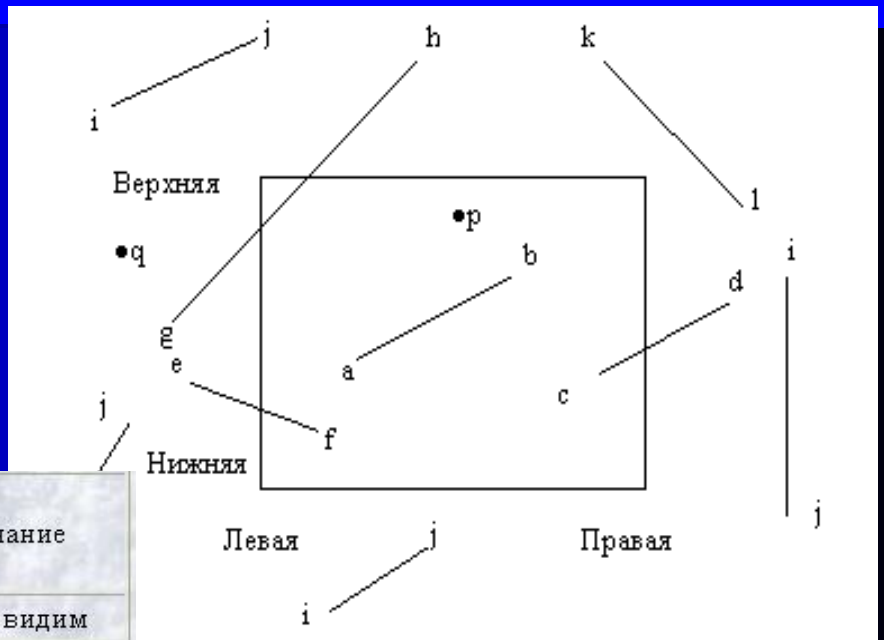


Рис. 1. Двумерное отсекающее окно.

Отрезок (рис. 1)	Коды концов (рис.9.2)	Результаты логического умножения	Примечание
ab	0000 0000	0000	Целиком видим
ij	0010 0110	0010	Целиком невидим
ij	1001 1000	1000	-"-
ij	0101 0001	0001	-"-
ij	0100 0100	0100	-"-
cd	0000 0010	0000	Частично видим
ef	0001 0000	0000	-"-
gh	0001 1000	0000	-"-
kl	1000 0010	0000	Целиком невидим

Простое двухмерное отсечение

Если определены целиком видимые и тривиально невидимые отрезки, то подпрограмме, вычисляющей пересечение отрезков, передаются только отрезки, которые, возможно, частично видимы, то есть те, для которых результат логического умножения кодов их концевых точек равен нулю.

Уравнение бесконечной прямой, проходящей через точки $P_1(x_1,y_1)$ и $P_2(x_2,y_2)$, имеет вид $y=t(x-x_1)+y_1$ или $y=t(x-x_2)+y_2$, где $m=(y_2-y_1)/(x_2-x_1)$ - это наклон данной прямой.

Точки пересечения этой прямой со сторонами окна имеют следующие координаты:

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:	$y_в$, $x=x_1+(1/m)(y_в-y_1)$	$m\neq0$
с нижней:	$y_н$, $x=x_1+(1/m)(y_н-y_1)$	$m\neq0$

Простое двухмерное отсечение

Для эффективной реализации данного алгоритма необходимо иметь в виду следующее:

1) Если наклон бесконечен, то отрезок параллелен левой и правой сторонам окна и надо искать его пересечение только с верхней и нижней сторонами.

2) Если наклон равен нулю, то отрезок параллелен верхней и нижней сторонам окна, а искать его пересечения надо только с левой и правой сторонами.

3) Если код одного из концов отрезка равен нулю, то этот конец лежит внутри окна, и поэтому отрезок может пересечь только одну сторону окна.

Простой алгоритм двумерного отсечения.

В общем простой алгоритм двумерного отсечения можно представить следующими основными шагами:

1. Вычисление кодов концевых точек отрезка. Занесение этих кодов в два массива M_1 и M_2 размерностью 1 4 каждый.

2. Инициализация флажка видимости.

3. Инициализация наклона.

4. Проверка полной видимости отрезка (коды обоих концов равны нулю): суммирование элементов M_1 и M_2. Если сумма элементов M_1 и сумма элементов M_2 равны нулю, то вычерчивание отрезка.

Простой алгоритм двумерного отсечения.

В общем простой алгоритм двумерного отсечения можно представить следующими основными шагами:

5. Проверка случая тривиальной невидимости. Если логическое произведение соответствующих элементов матриц не равно нулю, то флажок видимости устанавливается в "невидимость" и осуществляется переход к следующему отрезку.
6. Процедура отсечения отрезка каждой из сторон окна и получения точек пересечения с помощью проверки их принадлежности внутренней области окна. Эта последняя процедура реализуется по следующей схеме: сначала она применяется к отрезку P_1P_2 (рис. 4) и получается отрезок $P_1'P_2$, а затем - к отрезку $P_1'P_2$ и получается результирующий отрезок $P_1'P_2'$.

Пример 1. Вычисление пересечений отрезка со сторонами окна

Отрезок: от $P_1(-3/2, 1/6)$ до $P_2(1/2, 3/2)$
$m=(y_2-y_1)/(x_2-x_1)=(3/2-1/6)/[1/2-(-3/2)]=2/3$.
Его пересечения со сторонами окна таковы:
с левой: $x=-1$ $y=(2/3)[-1-(-3/2)]+1/6=1/2$
с правой: $x=1$ $y=(2/3)[1-(-3/2)]+1/6=11/6$
(последнее число больше, чем $y_в$, и поэтому отвергается) - проверка корректности пересечения
с верхней: $y=1$ $x=-3/2+(3/2)[1-1/6]=-1/4$
с нижней: $y=-1$ $x=-3/2+(3/2)[-1-(1/6)]=-13/4$
(последнее число меньше, чем $x_л$,
и поэтому отвергается)

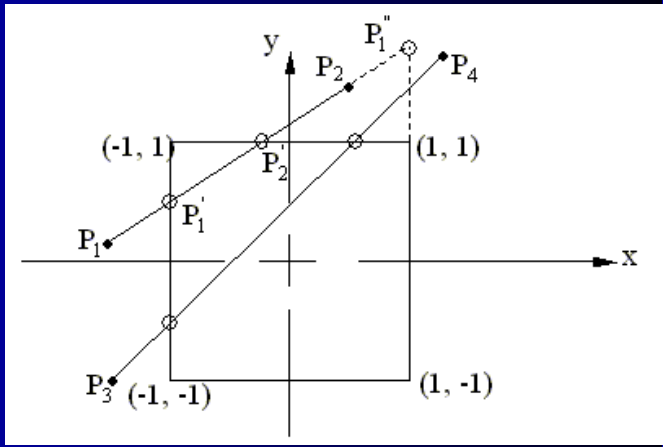


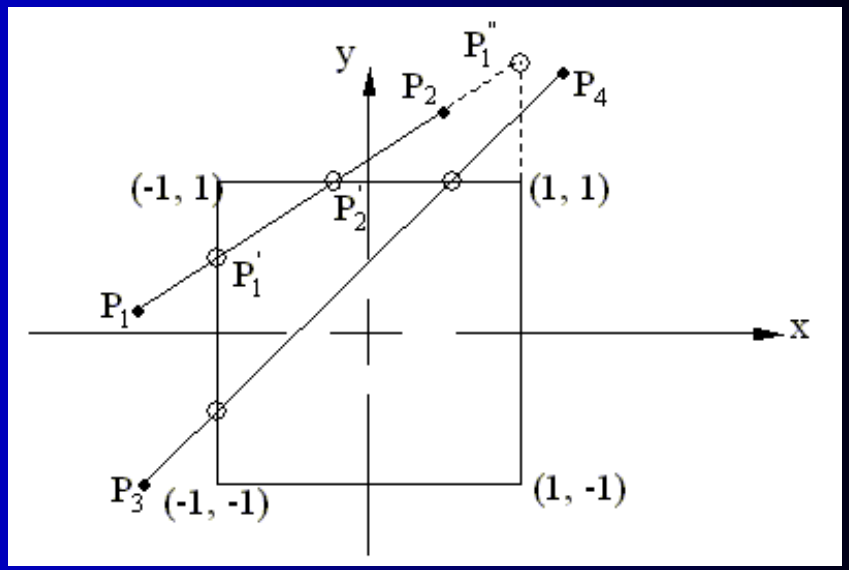
рис. 4

Аналогично, для отрезка от $P_3(-3/2, -1)$ до $P_4(3/2,2)$:

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{2-(-1)}{3/2-(-3/2)} = 1$$

Пример 1. Вычисление пересечений отрезка со сторонами окна

Пересечения отрезка P_3P_4:
с левой: $x=-1$ $y=(1)[-1-(-3/2)]+(-1)=-1/2$
с правой: $x=1$ $y=(1)[1-(-3/2)]+(-1)=3/2$
(последнее число больше, чем $y_в$, и поэтому отвергается).
с верхней: $y=1$, $x=-3/2+(1)[1-(-1)]=1/2$
с нижней: $y=-1$, $x=-3/2+(1)[-1-(-1)]=-3/2$
(последнее число меньше, чем $x_л$, и поэтому отвергается).



Алгоритм Сазерленда-Коэна

В данном отрезок тоже разбивается сторонами окна.

Отличие состоит в том, что здесь не производится проверки попадания точки пересечения внутрь окна, вместо этого каждая из пары получающихся частей отрезка сохраняется или отбрасывается в результате анализа кодов ее концевых точек.

Отрезок P_1P_2 на рисунке 4, разбитый левой стороной окна на P_1P_1' и $P_1'P_2$. Для алгоритма Сазерленда-Коэна это особый случай, поскольку коды концевых точек каждого из отрезков таковы, что оба они могут быть частично видимы. Ни один из них нельзя отвергнуть.

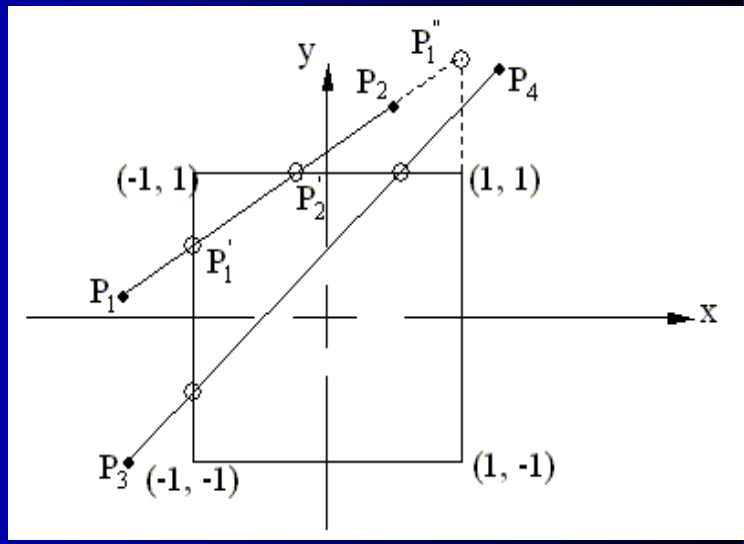
Ключом к алгоритму Сазерленда-Коэна является информация о том, что одна из концевых точек лежит вне окна. Поэтому тот отрезок, который заключен между этой точкой и точкой пересечения, можно отвергнуть как невидимый. Фактически это означает замену исходной концевой точки на точку пересечения.

Алгоритм Сазерленда-Коэна

Алгоритм Сазерленда-Коэна можно сформулировать следующим образом:

Для каждой стороны окна выполнить:

- Для каждого отрезка P_1P_2 определить, не является ли он полностью видимым или может быть тривиально отвергнут как невидимый.
- Если P_1 вне окна, то продолжить выполнение, иначе поменять местами P_1 и P_2.
- Заменить P_1 на точку пересечения P_1P_2 со стороной окна.



Пример 2: Алгоритм отсечения Сазерленда-Коэна

Рассмотрим вновь P_1P_2, показанный на рис. 4. Коды концевых точек $P_1(-3/2,1/6)$ и $P_2(1/2,3/2)$ равны (0001) и (1000) соответственно. Отрезок не является ни полностью видимым, ни тривиально невидимым. P_1P_2 пересекает левую сторону окна. P_1 - вне окна. Пересечение с левой стороной (x=-1) окна происходит в т.$P_1'(-1,1/2)$. Замена P_1 на P_1' дает новый отрезок – $P_1(-1,1/2)$ $P_2(1/2,3/2)$.

Коды: P_1 - (0000), P_2 - (1000). Отрезок не является ни полностью видимым, ни тривиально невидимым. Отрезок не пересекается с правой стороной окна.

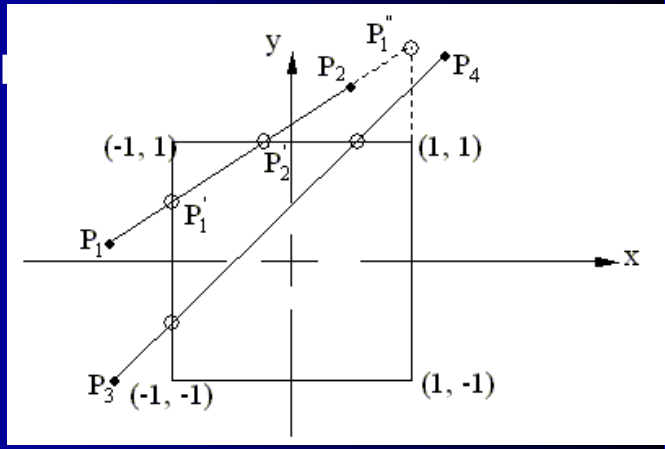


рис. 4

Пример 2: Алгоритм отсечения Сазерленда-Коэна

Перейти к нижней стороне. Коды по-прежнему: (0000), (1000). Отрезок не пересекается с нижней стороной окна.

Перейти к верхней стороне. Коды по-прежнему: (0000), (1000). Отрезок пересекается с верхней стороной окна. P_1 - не снаружи окна. Поменяв P_1 и P_2 местами, получаем: $P_1(1/2,3/2)$ $P_2(-1,1/2)$. Точка пересечения $P_1'(-1/4,1)$. Заменив P_1 на P_1', получим $P_1(-1/4,1)$ $P_2(-1,1/2)$.

Теперь P_1 - (0000), P_2 - (0000).

Отрезок полностью видим.

Начертить отрезок

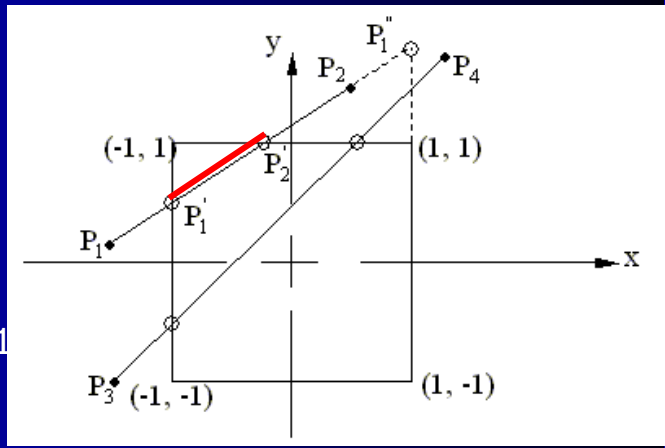
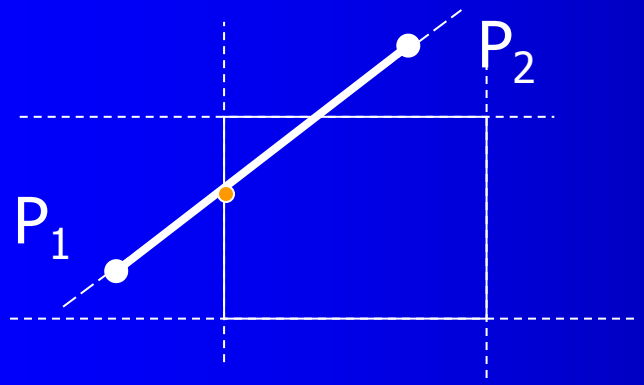
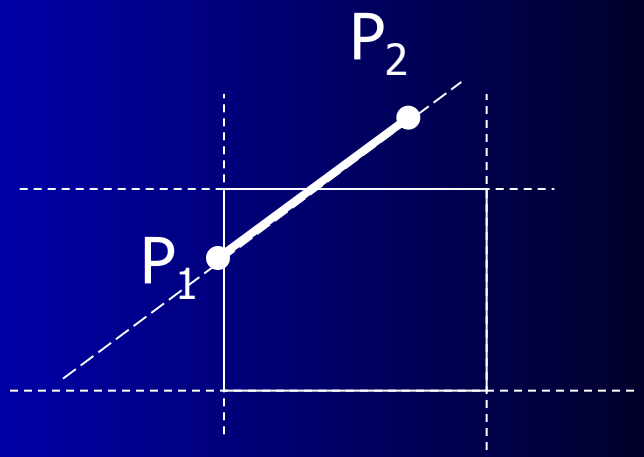


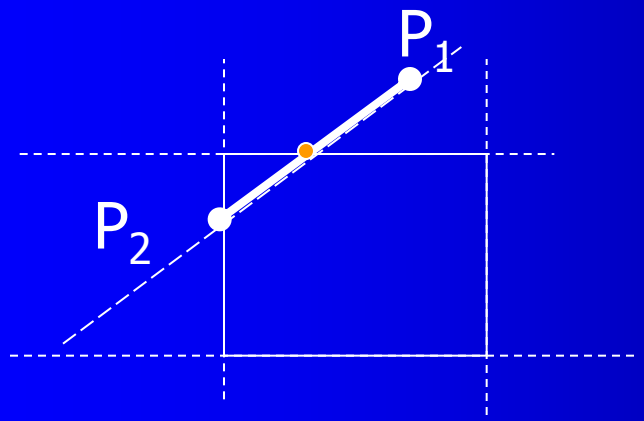
рис. 4

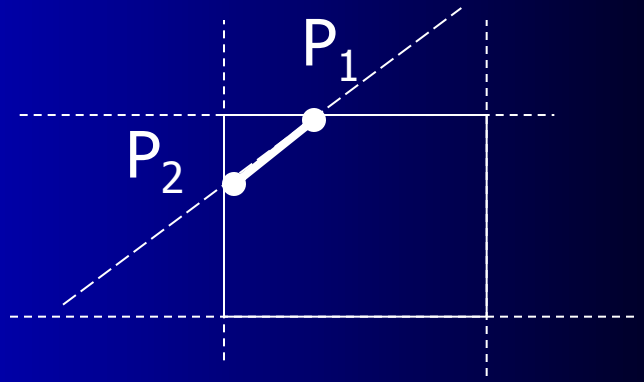
Алгоритм отсечения Сазерленда-Коэна

Итого:









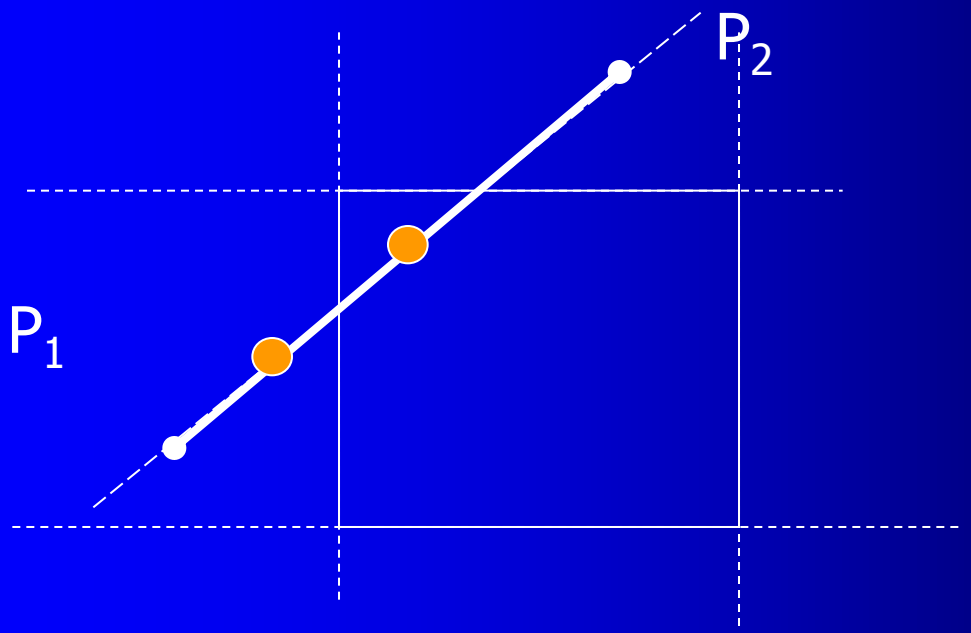
Алгоритм разбиения средней точкой

Точка пересечения отрезка может быть найдена путём деления отрезка точкой, находящейся на его середине.

В алгоритме используется метод Коэна-Сазерленда для проверки концов концевых точек отрезков, получающихся в результате деления исходного отрезка пополам. Те отрезки, которые не являются ни полностью видимыми, ни полностью невидимыми запоминаются и в свою очередь делятся на двое. Проверка выполняется для каждой из половин – невидимая часть отбрасывается, оставшаяся делится до тех пор, пока не будет обнаружено пересечение со стороной окна, либо пока отрезок не выродится в точку.

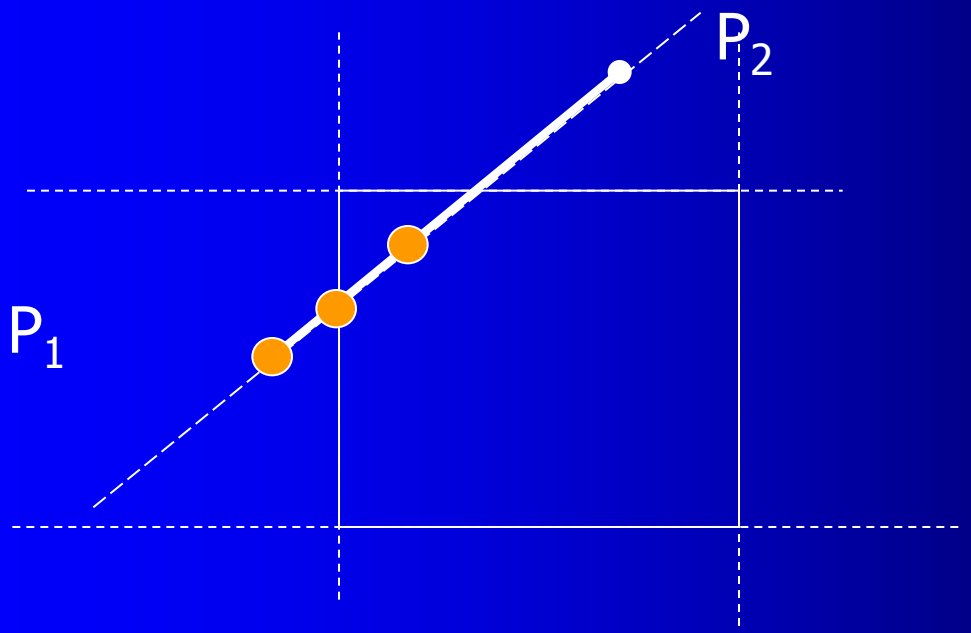
Алгоритм разбиения средней точкой

Пример:



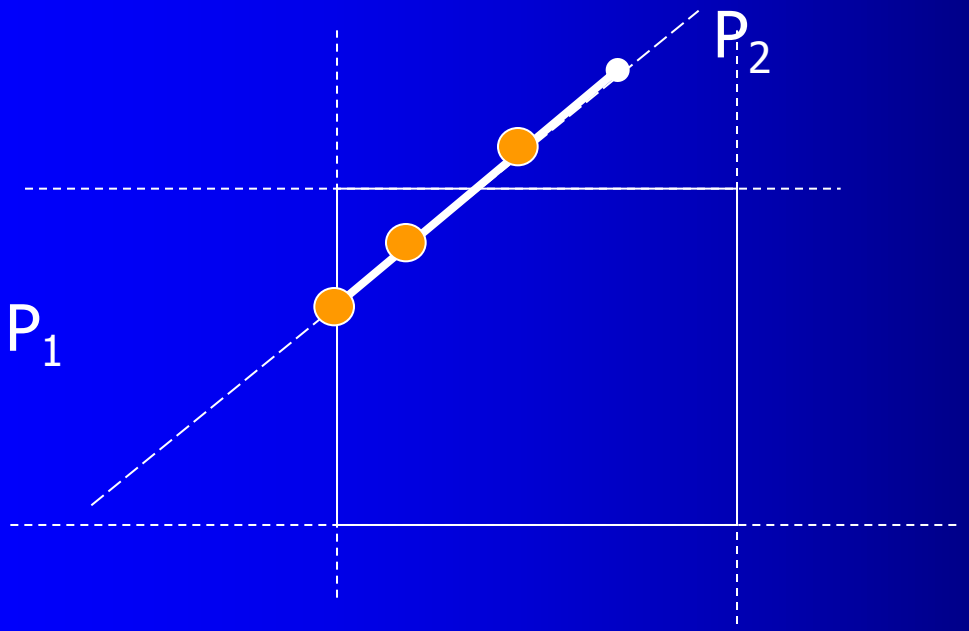
Алгоритм разбиения средней точкой

Пример:



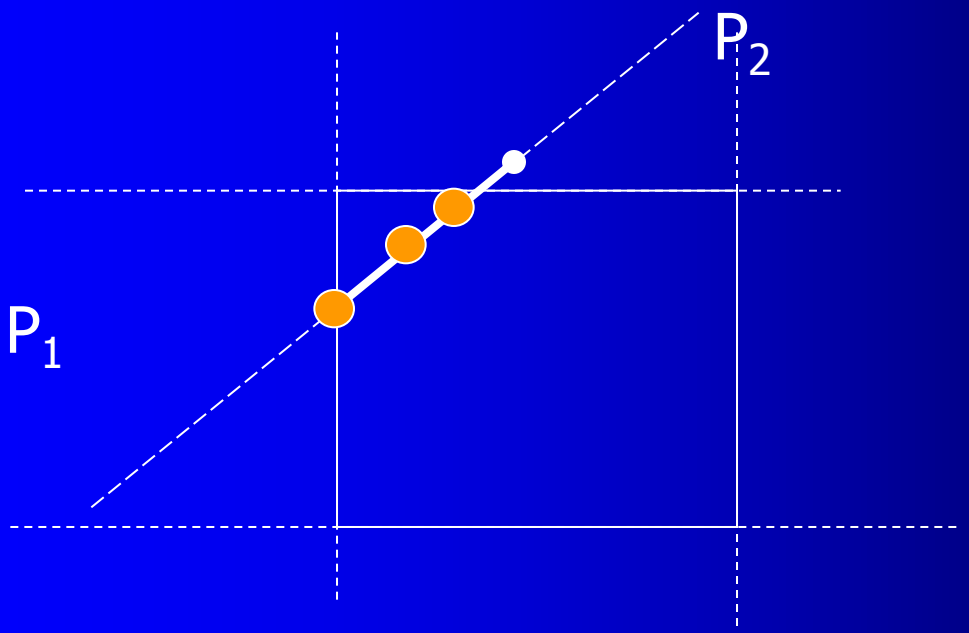
Алгоритм разбиения средней точкой

Пример:



Алгоритм разбиения средней точкой

Пример:



Отсечение окружностей и эллипсов