



```
Head = new Cell;
Head->LeftTag = true;
Head->RightTag = true;
Head->Left = Head;
Head->Right = Head;
current = Head;
}

template <class T> void Bina(T*parent, T*el)
{
    Cell *temp,*node;
    node = ToCell(el);
    temp = parent->Left;
    parent->Left = node;
    node->Left = temp;
    node->Right = parent;
    node->LeftTag = parent->LeftTag;
    parent->LeftTag = false;
    node->RightTag = true;

    if(!node->LeftTag){
        temp = node->Left;
        temp = GoToDownRight(temp);
        temp->Right = node;
    }

    temp = parent->Right;
    parent->Right = node;
    node->Right = temp;
    node->RightTag = parent->RightTag;
    parent->RightTag = false;
    node->LeftTag = true;
}
```

Алгоритмы отсечения

Лекция № 6

Введение

ОТСЕЧЕНИЕ ОТСЕЖКОВ

- Двумерный алгоритм Козна-Сазерленда
- Двумерный FC-алгоритм
- Двумерный алгоритм Лианга-Барски
- Двумерный алгоритм Кируса-Бека
- Алгоритм с использованием векторных произведений
- Разбиение невыпуклых многоугольников
- Сравнение алгоритмов двумерного отсечения

ОТСЕЧЕНИЕ МНОГОУГОЛЬНИКА

- Алгоритм Сазерленда-Ходгмана
- Простой алгоритм отсечения многоугольника
- Алгоритм отсечения многоугольника Вейлера-Азертонна

Отсечение отрезков

Если изображение выходит за пределы экрана, то либо увеличивается время построения, либо это приводит к искажению картины, либо просто не допускают выхода за пределы экрана.

Первое сообщение об аппаратуре отсечения, использующей алгоритм отсечения делением отрезка пополам и реализованной в устройстве Clipping Divider, появилось в 1968 г.

Robert F. Sproull and Ivan E. Sutherland. A Clipping Divider // AFIP Fall Joint Computer Conference. San Francisco, 1968.

Отсекаемые отрезки могут быть трех классов - **целиком видимые, целиком невидимые и пересекающие окно.**

Два основных типа алгоритмов отсечения:

- использующие кодирование концов отрезка или всего отрезка: алгоритм Коэна-Сазерленда (Cohen-Sutherland, CS-алгоритм) и FC-алгоритм (Fast Clipping - алгоритм),
- использующие параметрическое представление отсекаемых отрезков и окна отсечения: алгоритм Кируса-Бека (Cyrus-Beck, CB - алгоритм) и более поздний алгоритм Лианга-Барски (Liang-Barsky, LB-алгоритм).

Двумерный алгоритм Козна-Сазерленда

Этот алгоритм особенно эффективен в двух крайних случаях:

- большинство примитивов содержится целиком в большом окне,
- большинство примитивов лежит целиком вне относительно маленького окна.

Идея алгоритма:

Окно отсечения и прилегающие к нему части плоскости вместе образуют 9 областей.

Каждой из областей присвоен 4-х разрядный код. Две конечные точки отрезка получают 4-х разрядные коды, соответствующие областям, в которые они попали. Смысл разрядов кода:

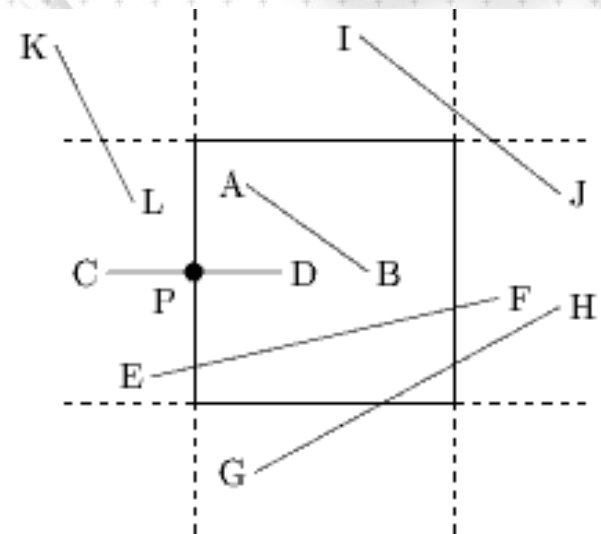
1 pp = 1 - точка над верхним краем окна;

2 pp = 1 - точка под нижним краем окна;

3 pp = 1 - точка справа от правого края окна;

4 pp = 1 - точка слева от левого края окна.

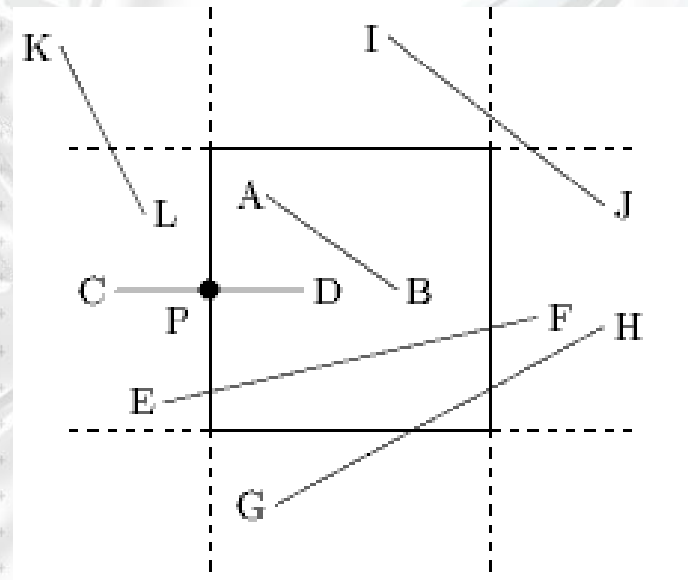
1001	1000	1010
-----	-----	-----
0001	Окно 0000	0010
-----	-----	-----
0101	0100	0110



Двумерный алгоритм Козна-Сазерленда

Определение того лежит ли отрезок целиком внутри окна или целиком вне окна выполняется следующим образом:

- если коды обоих концов отрезка равны 0, то отрезок целиком внутри окна, отсечение не нужно, отрезок принимается как **тривиально видимый** (отрезок AB);
- если логическое & кодов обоих концов отрезка не равно нулю, то отрезок **целиком вне окна**, отсечение не нужно, отрезок отбрасывается как тривиально невидимый (отрезок KL);
- если логическое & кодов обоих концов отрезка равно нулю, то отрезок **подозрительный**, он может быть частично видимым (отрезки CD, EF, GH) или целиком невидимым (отрезок IJ); для него нужно определить координаты пересечений со сторонами окна и для каждой полученной части определить тривиальную видимость или невидимость. При этом для отрезков CD и IJ потребуется вычисление одного пересечения, для остальных (EF и GH) - двух.



Двумерный алгоритм Козна-Сазерленда

Схема алгоритма Козна-Сазерленда:

1. Рассчитать коды конечных точек отсекаемого отрезка. В цикле повторять пункты 2-6.
2. Если логическое И кодов конечных точек не равно 0, то отрезок целиком вне окна. Он отбрасывается и отсечение закончено.
3. Если оба кода равны 0, то отрезок целиком видим. Он принимается и отсечение закончено.
4. Если начальная точка внутри окна, то она меняется местами с конечной точкой.
5. Анализируется код начальной точки для определения стороны окна, с которой есть пересечение и выполняется расчет пересечения. При этом вычисленная точка пересечения заменяет начальную точку.
6. Определение нового кода начальной точки.

Двумерный FC-алгоритм

В 1987 г. Собков, Поспишил и Янг предложили FC-алгоритм (Fast Clipping).

Пространство разбивается на 9 неперекрывающихся областей. Коды, назначаемые концам отрезков, попавших в ту или иную область.

Отрезок видим только в области 5, т.е. отрезок, координаты которого удовлетворяют условиям: $X_{\text{лев}} \leq X \leq X_{\text{прав}}$ и $Y_{\text{низ}} \leq Y \leq Y_{\text{верх}}$.

Каждая конечная точка отрезка V0V1 окажется с одной из этих областей. Комбинация кодов концов отрезка, называемая кодом линии, используется для определения возможных вариантов расположения отрезка и, следовательно, отсечения.

Код линии формируется из кодов концов отрезка следующим образом:

$\text{LineCode}(V0, V1) = (\text{Code}(V0) \times 16) + \text{Code}(V1)$,
здесь $\text{Code}(V1)$ обозначает код конечной точки V1,
 $\text{Code}(V0) \times 16$ означает сдвиг кода начальной точки V0 влево на 4 разряда.

Так как каждый код может принимать одно из 9 значений, то всего имеется 81 возможный вариант расположения отрезка. Но, если $\text{Code}(V0)$ равен $\text{Code}(V1)$, то $\text{LineCode}(V0, V1)$ равен $\text{LineCode}(V1, V0)$.
Имеется всего 9 таких случаев: 1-1, 2-2, ... 9-9.

Следовательно, число различных случаев уменьшается до 72.

1001 0x9 1	1000 0x8 2	1010 0xA 3	Y _{верх}
0001 0x1 4	0000 0x0 5	0010 0x2 6	
0101 0x5 7	0100 0x4 8	0110 0x6 9	Y _{низ}
X _{лев}		X _{прав}	

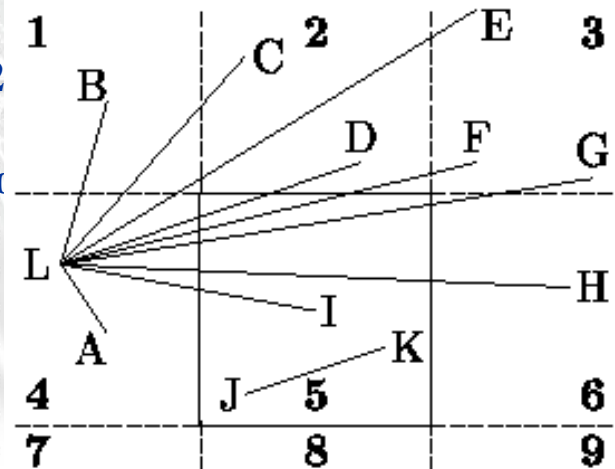
Двумерный FC-алгоритм

Каждый LineCode требует своего набора вычислений для определения отсеечения отрезка за минимальное время. Обозначения:

- начальная точка отрезка считается точкой номер 0 (V_0),
- конечная точка отрезка считается точкой номер 1 (V_1),
- ClipA_B обозначает алгоритм расчета перемещения конечной точки номер A на сторону окна B.

Всего имеется 8 основных случаев отсеечения:

1. Начальная и конечная точки отрезка обе в области 5 (отрезок JK).
2. Начальная и конечная точки отрезка обе в области 4 (отрезок LA).
3. Начальная точка в области 4, конечная - в области 1 (отрезок LB).
4. Начальная точка в области 4, конечная - в области 2 (отрезки LC и LD). Отрезки явно пересекает Хлев, так что вначале надо определить соответствующую координату, используя алгоритм Clip0_Xleft. Для отрезка LC это дает $V_0y > Y_{\text{верх}}$, так что отрезок должен быть отброшен. Отрезок LD входит в окно с левой стороны и может выходить через верх. Следовательно, следующее отсеечение должно быть Clip1_Top, после которого отрезок принимается.

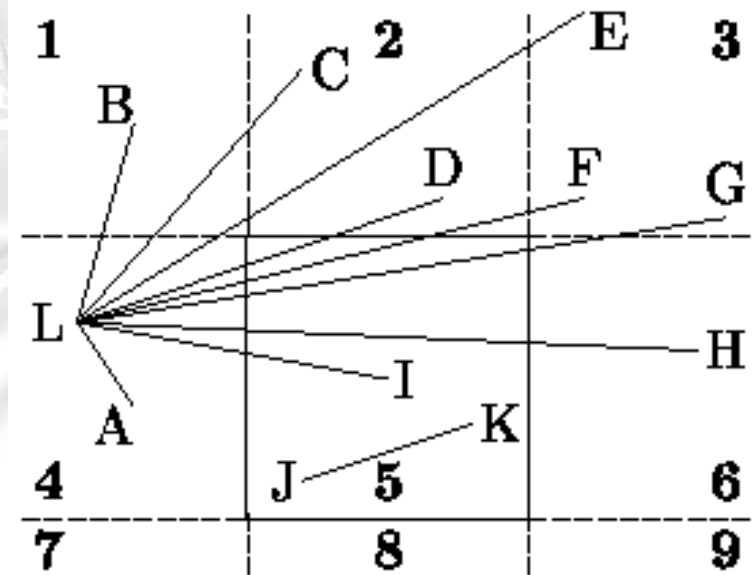


Двумерный FC-алгоритм

5. Начальная точка в области 4, конечная - в 3 (отрезки LE, LF и LG). Отрезки явно пересекают Хлев. Так же как и для случая 4 вначале применяется Clip0_Xleft и отрезок LE отбрасывается, если $V0y > Y_{\text{верх}}$. Если же получаем $V0y \leq Y_{\text{верх}}$, то отрезок должен выйти из области видимости через верхнее или правое ребро. Применяем отсечение Clip1_Top и сравниваем новое значение X-координаты конечной точки - $V1x$ с $X_{\text{прав}}$. Если $V1x \leq X_{\text{прав}}$, то отрезок (LF) проходит через верхнюю сторону, отрезок принимается. Иначе отрезок (LG) проходит через правую сторону и требуется отсечение Clip1_Right.

6. Начальная точка в области 4, конечная - в 6 (отрезок LH). Данный отрезок видим. Вначале используем Clip0_Xleft, затем Clip1_Right и принимаем отрезок.

7. Начальная точка в области 4, конечная - в 5 (отрезок LI). Данный отрезок видим. Просто используем Clip0_Xleft и принимаем отрезок.



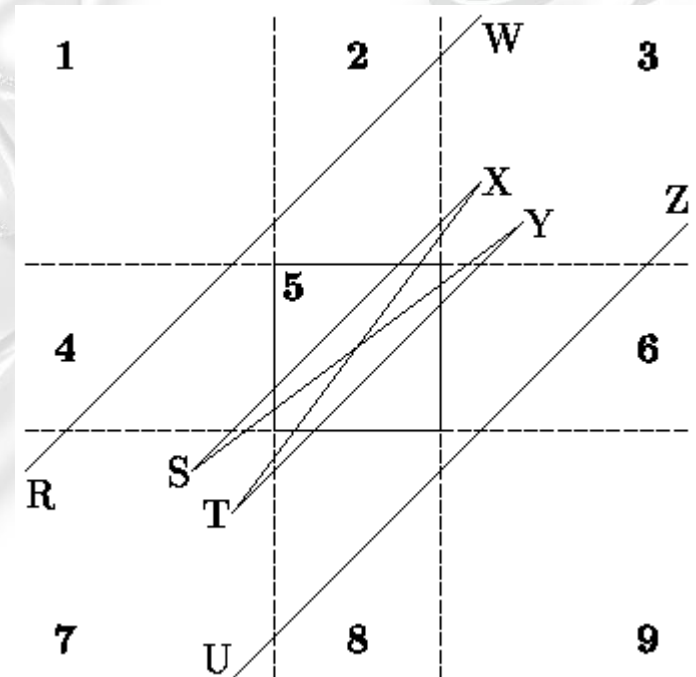
Двумерный FC-алгоритм

8. Начальная точка V_0 (R, S, T или U) в области 7, конечная точка V_1 (W, X, Y или Z) - в области 3. В этом случае могут быть отброшены только два типа отрезков. Для минимизации вычислений используем Clip0_Xleft. Если $V_0y > Y_{\text{верх}}$, то это первый случай отбрасывания (отрезок RW). Clip1_Xright и проверка $V_1y < Y_{\text{низ}}$ задают второй случай отбрасывания (отрезок UZ). Все другие отрезки должны быть видимы.

Если $V_0y < Y_{\text{низ}}$, тогда $V_0 = T$, иначе $V_0 = S$. Если $V_0y < Y_{\text{низ}}$, то Clip1_Ybottom даст точку V_0 на ребре окна. Аналогично, если $V_1y > Y_{\text{верх}}$, то $V_1 = X$ и здесь требуется Clip1_Ytop перед приемом отрезка. Если $V_1y < Y_{\text{верх}}$, тогда $V_1 = Y$.

Из этих восьми случаев легко симметрично сгенерировать все остальные.

Главное отличие FC-алгоритма от алгоритма Козна-Сазерленда состоит в упорядочивании действий по отсечению. Эффективность алгоритма Козна-Сазерленда ограничивается последовательным характером и фиксированным порядком действий по отсечению. Кроме этого, повышению эффективности FC-алгоритма по сравнению с CS-алгоритмом способствует отсутствие ненужных циклов и, следовательно, перевычислений кодов конечных точек.



Двумерный алгоритм Лианга-Барски

В 1982 г. Лианг и Барски предложили алгоритмы отсечения прямоугольным окном с использованием параметрического представления для двух, трех и четырехмерного отсечения.

Продолжим каждую из четырех границ окна до бесконечных прямых. Каждая из таких прямых делит плоскость на 2 области: видимую и невидимую части плоскости. Таким образом, окно отсечения может быть определено как область, которая находится на внутренней стороне всех линий границ.

Пусть конечные точки отрезка есть V_0 и V_1 с координатами (x_0, y_0) и (x_1, y_1) .

Тогда параметрическое представление линии может быть задано уравнениями:

$$x = x_0 + dx \cdot t, \quad y = y_0 + dy \cdot t,$$

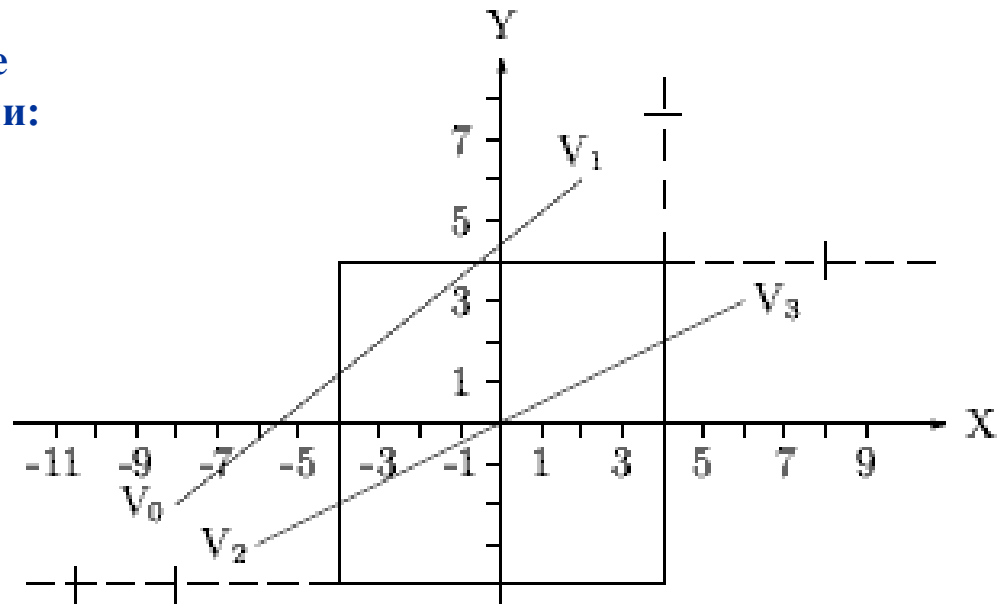
где $dx = x_1 - x_0$, $dy = y_1 - y_0$.

Или в общем виде:

$$V(t) = V_0 + (V_1 - V_0) \cdot t.$$

Подставляя параметрическое представление в неравенства окна, получим:

$$\begin{aligned} -dx \cdot t &\leq x_0 - X_{\text{лев}} \quad \text{и} \quad dx \cdot t \leq X_{\text{прав}} - x_0, \\ -dy \cdot t &\leq y_0 - Y_{\text{низ}} \quad \text{и} \quad dy \cdot t \leq Y_{\text{верх}} - y_0. \end{aligned}$$



Двумерный алгоритм Лианга-Барски

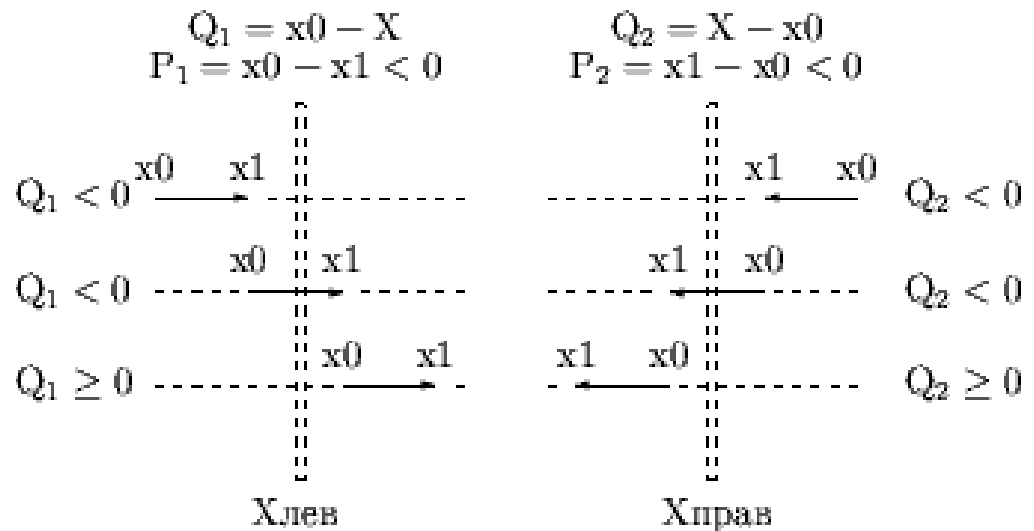
Неравенства имеют одинаковую форму $P_i t \leq Q_i$ для $i = 1, 2, 3, 4$, где $P_1 = -dx$; $Q_1 = x_0 - X_{\text{лев}}$; $P_2 = dx$; $Q_2 = X_{\text{прав}} - x_0$; $P_3 = -dy$; $Q_3 = y_0 - Y_{\text{низ}}$; $P_4 = dy$; $Q_4 = Y_{\text{верх}} - y_0$.

Каждое из неравенств соответствует одной из граничных линий (левой, правой, нижней и верхней) и описывает ее видимую сторону.

Удлиним V_0V_1 в бесконечную прямую. Тогда каждое неравенство задает диапазон значений параметра t , для которых эта удлиненная линия находится на видимой стороне соответствующей линии границы. Более того, конкретное значение параметра t для точки пересечения есть $t = Q_i/P_i$. Причем, если $Q_i \geq 0$, тогда V_0 находится на видимой стороне линии границы, включая и ее. Если же $Q_i < 0$, тогда V_0 находится на невидимой стороне.

Если $P_i < 0$, тогда соответствующее неравенство становится $t \geq Q_i / P_i$.

Диапазон значений t , для которых удлиненная линия находится на видимой стороне соответствующей граничной линии, имеет минимум в точке пересечения направленной удлиненной линии, заданной вектором V_0V_1 и идущей с невидимой на видимую сторону граничной линии (так как только на границе t равно Q_i / P_i , а в остальной части видимой стороны больше).



Двумерный алгоритм Лианга-Барски

Аналогично, если $P_i > 0$, тогда соответствующее неравенство становится $t \leq Q_i / P_i$.

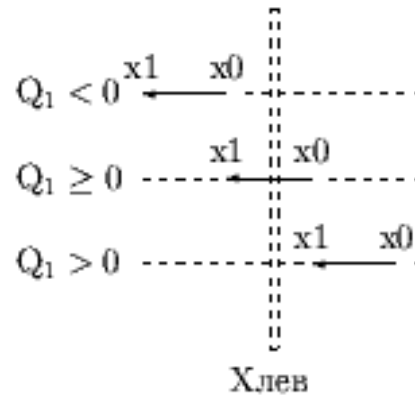
Так как значения параметра t только на границе равны Q_i/P_i , а в остальной видимой части меньше Q_i/P_i , то значение параметра t имеет максимум на границе.

Наконец, если $P_i = 0$, тогда соответствующее неравенство превращается в $0 \leq Q_i$.

Заметим, что здесь нет зависимости от t , т.е. неравенство выполняется для всех t , если $Q_i \geq 0$ и не имеет решения при $Q_i < 0$.

$$Q_1 = x_0 - X$$

$$P_1 = x_0 - x_1 > 0$$



$$Q_2 = X - x_0$$

$$P_2 = x_1 - x_0 > 0$$

