5.2. Алгоритмы отсечения многоугольников

Многоугольники особенно важны в растровой графике как средство задания поверхностей. Будем называть многоугольник, используемый в качестве окна отсечения, отсекателем, а многоугольник, который отсекается, — отсекаемым.

Алгоритм отсечения многоугольника должен в результате отсечения давать один или несколько замкнутых многоугольников (рис. 5.19). При этом могут быть добав-

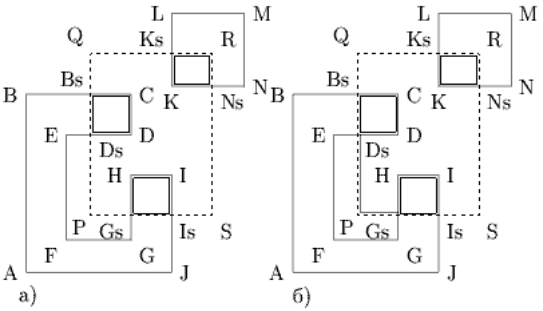

Рис. 5.19. Получение результата отсечения многоугольников.

лены 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 принципе задачу отсечения многоугольника можно решить с использованием рассмотренных выше алгоритмов отсечения линий, если рассматривать многоугольник просто как набор векторов, а не как сплошные закрашиваемые области. При этом отрезки, составляющие многоугольник, просто последовательно отсекаются сторонами окна (рис. 5.20).

Если же в результате отсечения должен быть получен замкнутый многоугольник, то формируется вектор, соединяющий последнюю видимую точку с точкой пересечения с окном (рис. 5.21, а). Проблема возникает при окружении отсекаемым многоугольником угла окна (см. 5.21, б).

Здесь мы рассмотрим пять алгоритмов корректно решающие задачу отсечения

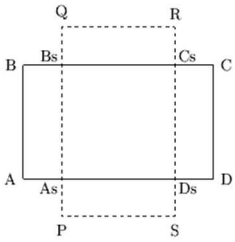

Рис. 5.20. Отсечение набора отрезков.

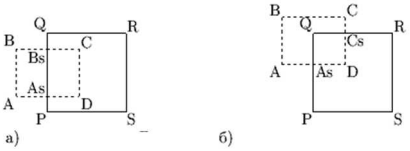

Рис. 5.21. Отсечение набора отрезков.

сплошного многоугольника. Первые четыре алгоритма быстро работают, но генерируют лишние ребра, как это продемонстрировано на рис. 5.19, б. Последний алгоритм свободен от указанного недостатка.

В общем, при отсечении многоугольников возникают два типа задач — отображение части изображения попавшей в окно и наоборот, отображение изображения, находящегося вне окна. Все здесь рассматриваемые алгоритмы могут использоваться в обоих случаях.

5.2.1. Алгоритмы поэтапного отсечения многоугольника ребрами области видимости

Следующие два алгоритма имеют в своей основе один и тот же метод. Область видимости в этих алгоритмах — выпуклый многоугольник.

Как отсекаемый многоугольник, так и многоугольник области видимости, представляются списком вершин в порядке некоторого обхода. В дальнейшем будем считать, что обход осуществляется по часовой стрелке.

Многоугольник последовательно отсекается каждой границей окна, как это показано на рис. 5.22

На каждом этапе отсечения многоугольник отсекается одним ребром выпуклого

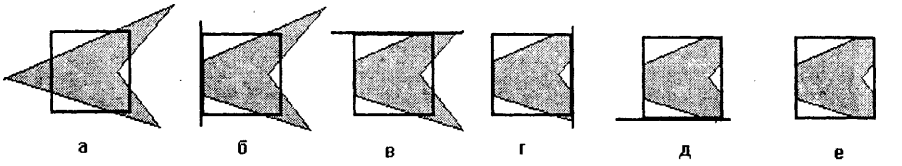

Рис. 5.22. Поэтапное отсечение многоугольника.

окна отсечения. В результате такого отсечения формируется новый многоугольник, который затем отсекается следующим ребром и т. д., пока не будет выполнено отсечение последним ребром окна.

Основная процедура здесь — процедура отсечения отдельным ребром. При выполнении этой процедуры последовательно просматривается каждое ребро многоугольника и решается вопрос о том, какая часть этого ребра будет записана в список многоугольника—результата отсечения.

Возможны 9 различных случаев расположения отсекающего ребра окна и отсекаемой стороны, показанных на рис. 5.23–5.25.

Рис. 5.23. Начальная точка на стороне внешности i-го ребра OB

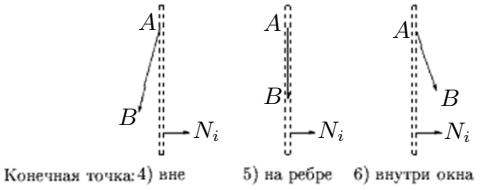

Рис. 5.24. Начальная точка на несущей прямой i-го ребра OB

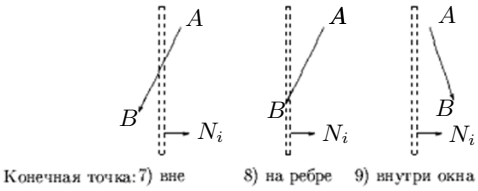

Рис. 5.25. Начальная точка на стороне внутренности i-го ребра OB

Здесь, A и B — начальная и конечная точки отсекаемой стороны многоугольника, соответственно; N_i — нормаль к i-му ребру окна отсечения, направленная внутрь окна.

Из этих рисунков очевиден результат, формирующий список нового многоугольника, зависящий от случая взаимного расположения:

1) Нет выходных данных.
2) В выходные данные заносится конечная точка.
3) Рассчитывается пересечение и в выходные данные заносятся точка пересечения и конечная точка.
4) Нет выходных данных.
5) В выходные данные заносится конечная точка.
6) В выходные данные заносится конечная точка.
7) Рассчитывается пересечение и в выходные данные заносится только точка пересечения.
8) В выходные данные заносится конечная точка.
9) В выходные данные заносится конечная точка.

Общая схема рассматриваемых методов — алгоритма Сазерленда—Ходгмана и алгоритма Кируса—Бека, представлена алгоритмом 8. Данные алгоритмы различаются только способами определения того, какой случай из девяти вышеперечисленных случаев имеет место. В алгоритме 8 эти различия будут иметь место в шагах PosA и PosB. Кроме этого, в алгоритмах разные подходы к поиску точки пересечения на шаге FindC.

Алгоритм 8: Алгоритм Сазерленда—Ходгмана / Кируса—Бека. Общая схема

Вход: список S многоугольников.
Выход: S_1 — список видимых частей многоугольников из S.
начало алгоритма
  Пусть F_i — все вершины области видимости, $1 \leqslant i \leqslant m$;
  $S_1 = \varnothing$;
  цикл пока S *не пуст* **выполнять**
    Взять из S список многоугольника P; n — количество вершин в P;
    Присвоить $i = 1$;
    цикл пока $i \leqslant m$ **выполнять**
      $k = 1$; $A = P_n$;
PosA Определить положение точки A относительно $i-$ границы;
      $P' = \varnothing$; $n_1 = 0$;
      цикл пока $k \leqslant n$ **выполнять**
        $B = P_k$;
PosB   Определить положение точки B относительно $i-$ границы;
        если *AB пересекается с несущей прямой i-ой границы* **то**
FindC   Найти точку пересечения C;
          Занести точку пересечения C в P'; $n_1 = n_1 + 1$;
        если *точка B на i-ой границе или на стороне внутренности* **то**
          Занести точку B в P'; $n_1 = n_1 + 1$;
        Присвоить $A = B$;
      Присвоить $P = P'$; $n = n_1$;
    если $P \neq \varnothing$ **то** положить P в список S_1;
  Выдать S_1;
конец алгоритма

Алгоритм Сазерленда—Ходгмана

Для определения случая расположения ребра многоугольника и прямой, ограничивающей область видимости, используется псевдоскалярное произведение вектора F_iF_{i+1}, проведенного из начальной в конечную точку текущего ребра области видимости, на вектор F_iA из начальной точки текущего ребра окна в очередную вершину A многоугольника (рис. 5.26). Обозначим через $Q_i(A)$ такое произведение.

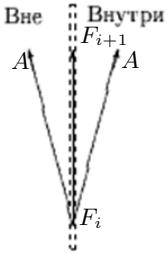

Рис. 5.26. Расположение точки относительно границы.

Если ребра многоугольников обходятся по часовой стрелке (в левой системе координат), то если $Q_i(A) < 0$, — точка A находится на стороне внутренности от i-го ребра области видимости. Если $Q_i(A) = 0$, то точка A находится на несущей прямой i-го ребра.

Для определения, какой из 9-ти случаев имеет место для отрезка AB, нужно найти $Q_i(A)$, $Q_i(B)$ и оценить их знаки.

Подход к нахождению точки пересечения в этом алгоритме такой же, как в алгоритме Скала: находим параметр t для точки пересечения:

$$t = \frac{AF_i \times F_iF_{i+1}}{(B-A) \times F_iF_{i+1}} =$$

$$\frac{AF_i \times F_iF_{i+1}}{(B-A+F_i-F_i) \times (F_iF_{i+1})} = \frac{AF_i \times F_iF_{i+1}}{((F_i-A) \times F_iF_{i+1}) - ((F_i-B) \times F_iF_{i+1})} =$$

$$\frac{AF_i \times F_iF_{i+1}}{(AF_i \times F_iF_{i+1}) - (BF_i \times F_iF_{i+1})} = \frac{Q_i(A)}{Q_i(A) - Q_i(B)},$$

после чего, из параметрического уравнения отрезка AB находим соответствующую точку.

Простой алгоритм отсечения многоугольника (Алгоритм Кируса—Бека)

Данный алгоритм использует те же подпрограммы обработки многоугольного окна отсечения, что и алгоритм Кируса — Бека для отсечения отрезков.

Для определения взаимного расположения, подобно алгоритму отсечения Кируса — Бека, используется скалярное произведение вектора нормали N_i на вектор, проведенный из начала ребра в анализируемую точку.

Таким образом, для определения взаимного расположения начальной A и конечной B точек отсекаемой стороны и ребра отсечения с вектором его начала F_i, надо вычислить:

$$Q_i(0) = (A - F_i)N_i$$

$$Q_i(1) = (B - F_i)N_i.$$

Расчет пересечения, если он требуется, производится аналогично алгоритму Кируса — Бека с использованием параметрического представления линии.

Из равенства 5.10 следует

$$(B - A - F_i + F_i)N_i t - (F_i - A)N_i = 0,$$

откуда

$$((B - F_i)N_i - (A - F_i)N_i)t + (A - F_i)N_i = 0,$$

или, в вышеприведенных обозначениях,

$$(Q_i(1) - Q_i(0))t + Q_i(0) = 0.$$

Таким образом, точке пересечения соответствует значение параметра t, равное:

$$t = \frac{Q_i(0)}{Q_i(0) - Q_i(1)}.$$

Значения координат пересечения находятся из параметрического уравнения отрезка.

5.2.2. Однопроходные алгоритмы отсечения многоугольников

В следующих двух алгоритмах производится отсечение многоугольников относительно прямоугольного окна. В таком случае результат отсечения должен представлять из себя список многоугольника, состоящий из некоторого набора вершин исходного многоугольника, некоторого количества точек пересечения ребер многоугольника

со сторонами области видимости и, возможно, точек—углов области видимости, — так называемых, поворотных точек.

Весь процесс отсечения заключается в одном проходе исходного списка многоугольника, в котором проводится анализ каждого ребра, и в список результата заносятся соответствующие точки. В ходе анализа ребро отсекается. Если после отсечения остаются видимые части, то в список результата заносятся соответствующие вершины и точки пересечения. В любом случае делается заключение о необходимости добавления в результат одной или двух поворотных точек.

Анализ на добавление поворотных точек может осуществляться разными способами. На рисунке 5.27 изображены примеры возможного расположения ребер многоугольника

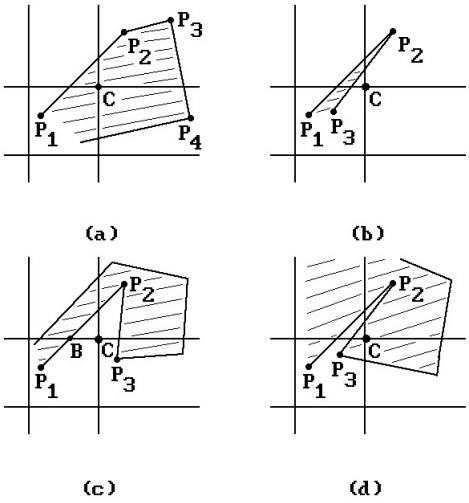

Рис. 5.27. Анализ отрезка на предмет добавления поворотной точки

и точки угла окна. В тех алгоритмах, в которых проводится только один проход многоугольника, в течении этого прохода невозможно предсказать, будет ли многоугольник охватывать угол или нет (например, на рисунке 5.27, анализируя только лишь отрезок P_1P2 невозможно однозначно решить вопрос о добавлении точки C в результат). Поэтому в этих алгоритмах, в целях ускорения работы, вопрос о добавлении поворотной точки решается следующим образом: поворотная точка добавляется если рассматриваемое ребро многоугольника проходит через соответствующую угловую область за пределами области видимости (т.е., для всех примеров на рисунке 5.27 поворотная

точка будет добавлена). Таким образом, в результат могут быть добавлены лишние точки углов окна видимости, что может привести к появлению в многоугольнике—результате областей, вырожденных в отрезок — побочный эффект, который можно легко устранить с помощью дополнительной обработки.

Алгоритм Лианга—Барски

Основная идея алгоритма заключается в следующем. Считаем, что область видимости ограничивают четыре прямые. Две прямые ограничивают область видимости по x и две прямые — по y. Рассматриваемому отрезку—ребру отсекаемого многоугольника придается направление (от начальной точки до конечной). Если ребро не параллельно никакой из прямых, ограничивающих область видимости, то найдутся четыре точки пересечения несущей прямой этого ребра с продолженными границами окна. Пусть для этих точек пересечения найдены параметры t параметрического уравнения отрезка. Причем относительно двух границ отрезок направлен в сторону внутренности области видимости. Обозначим через $t_{1\,in}$ и $t_{2\,in}$ (см. рис. 5.28) параметры для точек пересечения с этими границами (будем называть эти точки «входящие точки пересечения»). Относительно двух других границ отрезок направлен в сторону внешности. Обозначим параметры для этих точек пересечения (для «выходящих точек пересечения») через $t_{1\,out}$ и $t_{2\,out}$ (см. рис. 5.28). Анализируя величины этих парамет-

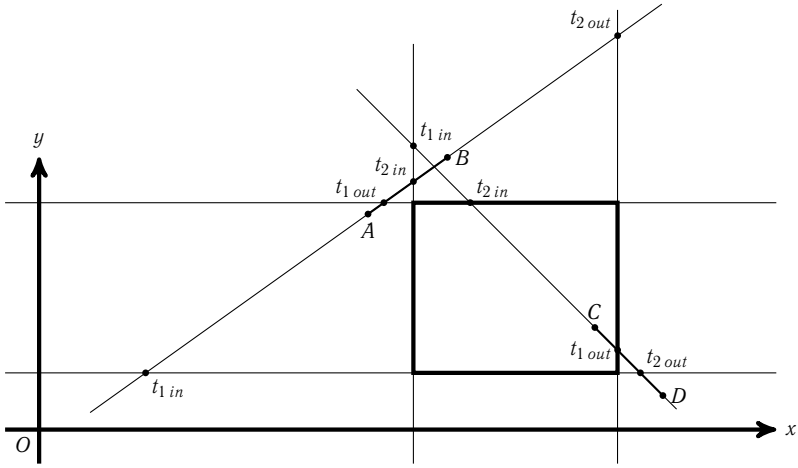

Рис. 5.28. Параметры для точек пересечения несущих прямых с границами окна

ров делается вывод о расположении соответствующих точек пересечения и самого отрезка в целом относительно области видимости.

Если упорядочить по возрастанию параметры для этих четырех точек пересечения, то минимальным параметром будет параметр для одной из входящих точек пересечения. Пусть это будет $t_{1\,in}$. Последний параметр в упорядоченной последовательности — параметр для одной из выходящих точек пересечения. Будем считать, что это $t_{2\,out}$. Параметр $t_{2\,in}$ может быть меньше $t_{1\,out}$ (в этом случае несущая прямая отрезка проходит через область видимости. См. рис. 5.28, отрезок CD), а может быть больше, чем $t_{1\,out}$ (несущая прямая не проходит через область видимости, а следовательно, ребро отсекаемого многоугольника полностью невидимо. См. рис. 5.28, отрезок AB). В первом случае вопрос о видимости или частичной видимости ребра решается сравнением параметров $t_{2\,in}$ и $t_{1\,out}$ с нулем и единицей (параметрами для видимой части отрезка). Если отрезок пересекает одну из границ области видимости или обе границы, то точки пересечения для соответствующих параметров заносятся в список вершин результата, а если отрезок заканчивается в области видимости ($t_{2\,in} \leqslant 1 \leqslant t_{1\,out}$), то в результат заносится и конечная точка отрезка. Во втором случае точек пересечения нет, но отрезок может проходить насквозь через угловую область за пределами окна наблюдения (не начинаться и не заканчиваться в этой угловой области, а проходить насквозь), что выявляется сравнением тех же $t_{2\,in}$ и $t_{1\,out}$ с нулем и единицей. Если это так, то в результат заносится поворотная точка — соответствующая угловая точка области видимости. Наконец, и в случае видимости отрезка, и в случае его нахождения за пределами окна, проводится проверка на окончание отрезка в угловой области за пределами окна. Если конечная точка отрезка располагается в такой области, то в результат заносится соответствующая угловая точка окна.

Общая схема метода представлена в алгоритме 9.

Алгоритм 9: Отсечение многоугольника по Лиангу—Барски. Общая схема

Вход: список S многоугольников.
Выход: S_1 — список видимых частей многоугольников из S.
начало алгоритма
  $S_1 = \varnothing$;
  цикл пока S *не пуст* **выполнять**
    Взять из S список многоугольника P; n — количество вершин в P;
    $k = 1$; $A = P_n$;
    $P' = \varnothing$;
    цикл пока $k \leqslant n$ **выполнять**
      $B = P_k$;
      Определить направление отрезка AB;
      Вычислить $t_{1\,out}$, $t_{2\,out}$;
      если $0 < t_{2\,out}$ **то**
        Вычислить $t_{2\,in}$;
        если $t_{1\,out} < t_{2\,in}$ **то** *// отрезок полностью невидим*
          если $0 < t_{1\,out} \leqslant 1$ **то** *// отрезок заходит в угловую область*
            Добавить в P' точку соответствующего угла окна;
        иначе
          если $0 < t_{1\,out}$ *и* $t_{2\,in} \leqslant 1$ **то** *// отрезок видим*
            если $0 < t_{2\,in}$ **то**
              $A' = A + (B - A)t_{2\,in}$;
              Добавить A' в P';
            если $t_{1\,out} \leqslant 1$ **то**
              $B' = A + (B - A)t_{1\,out}$;
              Добавить B' в P';
            иначе Добавить B в P';
          если $t_{2\,out} \leqslant 1$ **то** *// отрезок заканчивается в угловой области*
            Добавить в P' точку соответствующего угла окна;
      $A = B$; $k = k + 1$;
    если $P \neq \varnothing$ **то** положить P в список S_1;
  Выдать S_1;
конец алгоритма

Алгоритм Мэл'от

Алгоритм Мэл'от (Maillot) является своеобразным продолжением алгоритма Коэна—Сазерленда для отсечения многоугольников. Автор метода предлагает дополнить код области при кодировании, введенном в алгоритме Коэна—Сазерленда, дополнительным разрядом (см. рис. 5.29). Дополнительный разряд будет получать значение «один» если область угловая. В противном случае — значение «ноль».

Для каждого ребра многоугольника проводится отсечение по алгоритму Коэна—Сазерленда обычным образом, без использования дополнительного разряда. Во время отсечения инициализируются два флага: `SEGM` — говорит о том, что после отсечения остались видимые части отрезка, `CLIP` — говорит о том, что начальная точка исход-

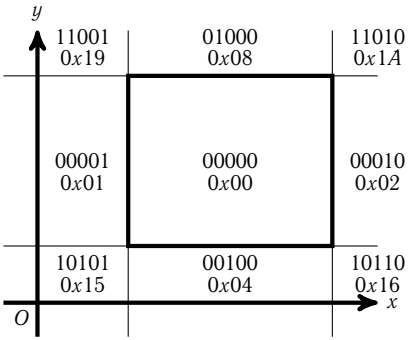

Рис. 5.29. Кодирование областей в алгоритме Мэл'от

ного отрезка была отсечена.

Если после отсечения ребра остались видимые части, то алгоритм Мэл'от проверяет значение флага `CLIP`. Если начальная точка была отсечена, то в результат заносится начальная точка видимой части. В любом случае в результат заносится конечная точка видимой части отрезка.

Если отрезок невидим, производится его анализ чтобы выявить, не пересекает ли отрезок угловую область. Это имеет место, если

1. начало и конец отрезка находятся в разных неугловых областях (рисунок 5.30, отрезок DF);
2. начало отрезка находится в неугловой, а конец отрезка в угловой области по разные стороны от окна видимости (рисунок 5.30, отрезок DE);
3. начало отрезка находится в угловой, а конец отрезка в неугловой области по разные стороны от окна видимости (рисунок 5.30, отрезок AC);
4. и начало и конец отрезка в угловых областях, расположенных по диагонали от окна видимости (рисунок 5.30, отрезки AB и GH).

Все эти случаи выявляются при рассмотрении расширенных кодов концов исходного отрезка. Более того, в первых трех случаях из кодов соответствующих однозначно определяется поворотная точка, которую нужно добавить в результат. В четвертом случае для выявления поворотной точки отрезок делится пополам. Если точка середины отрезка попала не в ту же угловую область, в которой находится один из концов отрезка, то поворотная точка определяется однозначно. В противном случае тест повторяется, но уже для половины отрезка.

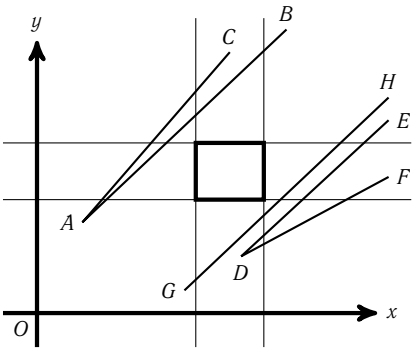

Рис. 5.30. Варианты пересечения отрезком угловой области

Когда поворотная точка возникает в результате наличия отрезка типа DF на рис. 5.30, код угловой области, соответствующий этой поворотной точке вычисляется как сумма кодов областей начала и конца отрезка с добавлением к нему 16 (заполнение единицей добавочного разряда).

Код угловой области, соответствующий поворотной точке, возникшей в результате наличия отрезков типа AC, DE на рис. 5.30, предлагается вычислять как сумму кода угловой области, в которой начинается/заканчивается отрезок, со значением элемента массива значений Tcc, в котором для кода боковой области хранится недостающая величина. Так для кодировки областей, представленной на рис. 5.29, массив Tcc будет состоять из следующих величин:

$$Tcc[0x01] = -1;\ Tcc[0x02] = 1;\ Tcc[0x04] = -4;\ Tcc[0x08] = 4.$$

Так, например, для вычисления кода области, соответствующей поворотной точке для отрезка AC нужно сложить код точки A со значением элемента Tcc для кода точки C:

$$0x15 + Tcc[0x08] = 0x15 + 4 = 0x19$$

После выполнения вышеперечисленных действий (в случае видимости отрезка и его невидимости) выполняется простой тест конечной точки исходного отрезка, чтобы выявить необходимость добавления поворотной точки в случае ее расположения в угловой области. Общая схема метода представлена в алгоритме 10. Здесь $ExtCode(A)$ обозначает функцию, возвращающую расширенный код области, в которую попала точка A, в соответствии с рисунком 5.29.

Алгоритм 10: Отсечение многоугольника по Мэл'от. Общая схема

Вход: список S многоугольников.
Выход: S_1 — список видимых частей многоугольников из S.
начало алгоритма
  $S_1 = \varnothing$;
  цикл пока S *не пуст* **выполнять**
    Взять из S список многоугольника P; n — количество вершин в P;
    $k = 1$; $A = P_n$; вычислить $C_{start} = ExtCode(A)$;
    $P' = \varnothing$; ;
    цикл пока $k \leqslant n$ **выполнять**
      $B = P_k$; вычислить $C_{end} = ExtCode(B)$;
      Выполнить алгоритм Коэна—Сазерленда. Получить значения `CLIP`, `SEGM`, и, возможно, точки A' и B';
      $C_2 = C_{end}$;
      если `SEGM` **то**
        если `CLIP` **то** добавить A' в P';
        Добавить B' в P';
      иначе
        если $C_{end} \& 0x10 \neq 0$ **то** *// конечная точка в угловой области*
          если $C_{start} \& C_{end} \& 0x0F = 0$ **то** *// начальная и конечная*
            // точки по разные стороны области видимости
            если $C_{start} \& 0x10 = 0$ **то** $C_1 = C_{end} + Tcc(C_{start})$;
            иначе
              $A' = A$; $B' = B$; $C_{11} = C_{start}$; $C_{12} = C_{end}$;
              $D = A$; $C_1 = C_{start}$;
              цикл пока $C_1 = C_{11}$ *или* $C_1 = C_{12}$ **выполнять**
                если $C_1 = C_{11}$ **то** $A' = D$ **иначе** $B' = D$;
                Вычислить D — точку середины отрезка $A'B'$;
                Вычислить $C_1 = ExtCode(D)$;
              если $C_1 \& C_{12} \neq 0$ **то** $C_1 = C_{11} + Tcc(C_1)$;
              иначе $C_1 = C_{12} + Tcc(C_1)$;
            Определить угол, соответствующий C_1. Добавить его в P';
          иначе
            если $C_{start} \& C_{end} \& 0x0F = 0$ **то**
              если $C_{start} \& 0x10 \neq 0$ **то** $C_2 = C_{start} + Tcc(C_{end})$;
              иначе $C_2 = C_{start} + C_{end} + 16$;
        если $C_2 \& 0x10 \neq 0$ **то**
          Определить угол, соответствующий C_2. Добавить его в P';
        $A = B$; $k = k + 1$; $C_{start} = C_{end}$;
      если $P \neq \varnothing$ **то** положить P в список S_1;
    Выдать S_1;
конец алгоритма