3.6. ОТСЕЧЕНИЕ НА ПЛОСКОСТИ

3.6.1. ОСНОВНЫЕ ПОЛОЖЕНИЯ. ИСПОЛЬЗОВАНИЕ ОТСЕЧЕНИЯ В МАШИННОЙ ГРАФИКЕ

Отсечение - это операция удаления изображения за пределами выделенной области, называемой окном. Отсечение может проводиться как на плоскости, так и в пространстве. Чтобы выполнить данную операцию, необходимо прежде всего задать тип отсекателя. Отсекатели могут иметь некоторую стандартную форму, а могут быть и произвольного вида. При отсечении на плоскости в каче- стве стандартного отсекателя рассматривается прямоугольник со сторонами, параллельными осям координат. Произвольный отсекатель - это обычно много- угольник, который может быть как выпуклым, так и невыпуклым, а также мо- жет иметь отверстия. Помимо отсекателя должны быть также известны геомет- рические характеристики изображенных объектов. В результате отсечения должны получиться геометрические характеристики объектов, остающихся в пределах окна отсечения в результате выполнения рассматриваемой операции.

Задача отсечения изображения, в частности его простейших состав- ляющих - отрезков и многоугольников - может рассматриваться как довольно интересная самостоятельная математическая задача. Однако в рамках машин- ной графики она приобретает и важное прикладное значение.

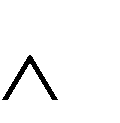
Отсечение используется в алгоритмах устранения ступенчатости, при создании трехмерных и реалистических изображений для удаления невиди- мых линий и поверхностей, для учета теней, формирования фактуры. Например, практически целиком на отсечении основываются такие алгоритмы удаления невидимых линий и поверхностей, как алгоритм Варнока, алгоритм Вейлера- Азертона.

Отсечение используется при построении сцен реальных объектов для вы- деления тех фрагментов, которые попадают в поле видимости наблюдателя.

Здесь рассматриваются алгоритмы отсечения отрезков и многоугольников на плоскости. Данные геометрические объекты являются наиболее распростра- ненными при построении реальных сцен. Отсечение в пространстве либо сводится к отсечению на плоскости, либо сами плоские алгоритмы без тру- да распространяются на трехмерный случай. Об этом будет сказано по ходу изложения материала.

* + 1. ОТСЕЧЕНИЕ ОТРЕЗКОВ РЕГУЛЯРНЫМ ОТСЕКАТЕЛЕМ Регулярным (стандартным) отсекателем на плоскости является прямоуголь-

ник со сторонами, параллельными координатным осям объектного пространства или экрана. Такое окно задается левым, правым, верхним и нижним двумерны- ми ребрами. Для выполнения отсечения необходимо задать абсциссы Xл, Xп ле- вого и правого ребер и ординаты Yн,Yв нижнего и верхнего ребер. Цель отсече- ния будет состоять в определении точек, отрезков или их частей, которые лежат внутри отсекателя.

Поскольку отсечение выполняется достаточно часто в машинной графике, то актуальным является вопрос обеспечения высокой эффективности работы этих алгоритмов. Часто большое количество исследуемых точек и отрезков реальной сцены лежит полностью в пределах окна или полностью за пределами окна. По- этому желательно быстро определять такие точки и отрезки. Точки, лежащие целиком внутри окна, удовлетворяют условию (XлX Xп) (Yн Y Yп), где



(X,Y) - координаты точки. Считается, что точки, лежащие на границе окна, при- надлежат внутренней области окна.

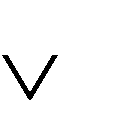
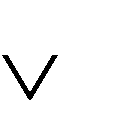
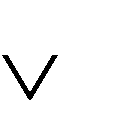
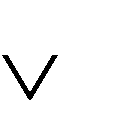
Отрезок целиком лежит внутри окна, если обе его концевые точки лежат внутри окна. Однако обратное утверждение, к сожалению, верно не всегда. От- резок, концевые точки которого лежат вне окна, может быть как полностью не- видимым, так и частично видимым. Полностью невидимым называется отрезок, целиком лежащий вне отсекателя. Частично видимым называется отрезок, одна часть которого лежит в пределах отсекателя, а другая - вне его. Если обе конце- вые точки отрезка невидимы, то он будет заведомо невидимым, если они (вер- шины отрезка) одновременно лежат левее или правее или ниже или выше окна (рис.3.6.1).

С помощью следующего простого теста можно определить полностью види- мые, часть полностью невидимых отрезков, остающиеся отрезки требуют даль- нейшего исследования для установления факта их частичной видимости или полной невидимости. Собственно для идентификации именно таких отрезков и требуются специальные алгоритмы отсечения.

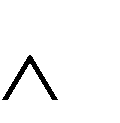
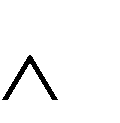
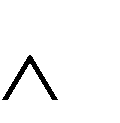
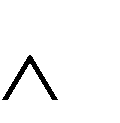
Простой алгоритм определения видимости

Обозначим концевые точки отрезка как (a,b). Тест определяет полностью видимый отрезок как такой, у которого ни одна из координат концов отрезка не находится вне отсекателя. Если же оба конца отрезка лежат целиком слева, справа, снизу или сверху от окна, то он является невидимым.

1. Определение полностью видимого отрезка: если (Xa<Xл) (Xb>Xл) то переход к п.2; если (Xb<Xл) (Xb>Xл) то переход к п.2; если (Ya<Yн) (Ya>Yв) то переход к п.2; если (Yb<Yн) (Yb>Yв) то переход к п.2; иначе отрезок видим, переход к п.3.



1. Определение полностью невидимого отрезка:



если (Xa<Xл) (Xb<Xл) то переход к п.3 (отрезок невидим); если (Xa>Xп) (Xb>Xп) то переход к п.3(отрезок невидим); если (Ya<Yн) (Yb<Yв) то переход к п.3(отрезок невидим); если (Ya>Yв) (Yb>Yв) то переход к п.3(отрезок невидим);

иначе отрезок частично видим или является невидимым (требуется до-

полнительный анализ.

1. Конец.

Д.Коэн и А.Сазерленд предложили формализовать описанную процедуру. Для определения принадлежности точки одной из девяти областей, на которые разбивается плоскость продолжениями ребер отсекателя, вводится четырехраз- рядный (битовый) код (рис.3.6.2). Биты этого кода заполняются по следующему правилу

T1 =1, если точка лежит левее окна, и 0 в противном случае; T2 =1, если точка лежит правее окна, и 0 в противном случае; T3 =1, если точка лежит ниже окна, и 0 в противном случае; T4 =1, если точка лежит выше окна, и 0 в противном случае. Первым считается крайний правый бит.

Точка лежит внутри отсекателя, если все биты ее кода - нулевые. Отрезок будет полностью видимым, если коды обоих его концов нулевые. Введенные коды концов отрезка можно использовать и для определения полностью неви-

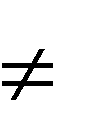
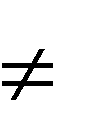
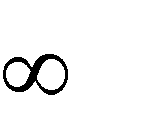
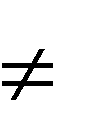
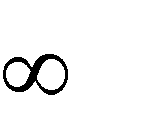
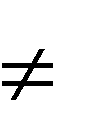
димых отрезков. Если побитное логическое произведение кодов концов отрезка не равно нулю, то отрезок является полностью невидимым и его можно отбро- сить. Однако, если логическое произведение равно нулю, то ничего определен- ного об отрезке сказать нельзя. В этом случае отрезок может быть целиком или частично видимым или даже целиком невидимым. Таким образом, на основе анализа кодов концов отрезка легко определяются полностью видимые и триви- ально невидимые отрезки. Отрезки, для которых логическое произведение ко- дов концов равно нулю, а побитная сумма этих кодов для концевых точек (од- ной или обеих) отлична от нуля, требуется выполнение собственно алгоритма отсечения с целью нахождения точек пересечения отрезка с границами отсека- теля или установления факта полной невидимости отрезка.

Пересечение отрезка с ребром отсекателя ищется как решение системы двух уравнений с двумя неизвестными. Уравнениями системы являются уравнения, задающие положение отрезка и ребра отсекателя на плоскости. Решение такой системы - координаты точки пересечения этих двух отрезков. Уравнения отрез- ков можно задавать как в параметрической, так и в непараметрической формах. В алгоритмах отсечения в силу большего удобства решения конкретной задачи используется параметрический способ задания отрезка в виде

Y=m(X-X1)+Y1 или Y=m(X-X2)+Y2, где (X1,Y1) и (X2,Y2) - координаты двух точек, через которые проходит отрезок, а m=(Y2 - Y1)/(X2- X1) - тангенс угла на- клона отрезка.

Поскольку для отсекателя заданы абсциссы левой и правой границ, ординаты нижней и верхней границ, то для нахождения координат точек пересечения от- резка с границами отсекателя достаточно подставить в уравнение отрезка абс- циссы боковых ребер отсекателя для нахождения ординат точек пересечения с этими ребрами . Поскольку точка пересечения принадлежит одновременно и отрезку и соответствующей границе отсекателя, то для нахождения точки пере- сечения с боковыми ребрами необходимо найти только ординату точки пересе- чения, а абсцисса совпадает с абсциссой ребра отсекателя. При нахождении точки пересечения с нижним или верхним ребрами отсекателя известна ордина- та ребра отсекателя, ее значение совпадает с ординатой точки пересечения и ос- тается найти только абсциссу точки пересечения. Для этого известная ордината подставляется в уравнение отрезка, откуда и находится искомая абсцисса.

В итоге координаты искомых точек пересечения отрезка с границами отсека- теля находятся из следующих выражеий:



Y= m(Xл-X1)+Y1 , m (с левой границей)

Y= m(Xп-X1)+Y1 , m (с правой границей) (3.6.1) X=(1/m)(Yн-Y1)+X1 , m 0 (с нижней границей)

X=(1/m)(Yв-Y1)+X1 , m 0 (с верхней границей).

Указанные ограничения на значение тангенса угла наклона отрезка выпол- няются, т.к. в алгоритмах предварительно анализируется характер отрезка (вер- тикальный, горизонтальный или общего положения) и в дальнейшем для верти- кальных отрезков не ищутся точки пересечения с боковыми ребрами, а для го- ризонтальных отрезков не ищутся точки пересечения с нижним и верхним реб- рами отсекателя.

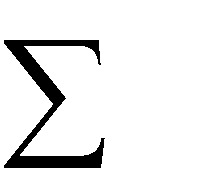
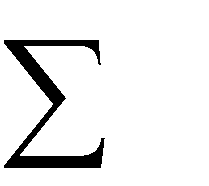
* + - 1. ПРОСТОЙ АЛГОРИТМ ОТСЕЧЕНИЯ ОТРЕЗКА Изложенные соображения положены в основу простого алгоритма отсечения

отрезка регулярным отсекателем. На основе сформированных кодов концов от- резка проверяется полная видимость, невидимость отрезка или видимость од-

ной из вершин отрезка. Если отрезок не является полностью видимым, то далее ищутся точки пересечения отрезка с границами отсекателя. При этом сначала анализируется возможность пересечения отрезка с очередной границей отсека- теля. Если пересечение возможно, то находятся координаты точки пересечения, после чего проводится анализ ее корректности. Под корректностью понимается принадлежность найденной точки ребру отсекателя, а не его продолжению. В случае корректности найденного пересечения точка заносится в результат и ищется вторая точка пересечения отрезка с границами отсекателя. После нахо- ждения двух точек пересечения визуализируется полученный результат. Если же пересечение не является корректным, то выполненные вычисления оказы- ваются фактически напрасными и в этом случае ищется пересечение со сле- дующей стороной отсекателя (рис.3.6.3). Если не было найдено ни одного кор- ректного пересечения, то отрезок является невидимым.

1. Ввод координат отсекателя Xл, Xп, Yн, Yв.
2. Ввод координат концов отрезка P1(X1,Y1), P2(X 2,Y 2).
3. Вычисление кодов концов отрезка T1, T2.

4 4



Вычисление сумм кодов концов отрезка S1= *T*1*i* , S2= *T* 2*i* .

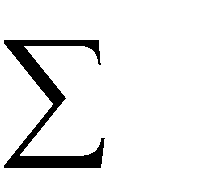
*i* 1 *i* 1

1. Установка признака видимости отрезка pr =1 (pr =1 - отрезок видимый; pr = -1 - отрезок невидимый).
2. Задание начального значения тангенса угла наклона отрезка m=1030 (большое число, вначале предполагается, что отрезок вертикальный).
3. Проверка полной видимости отрезка:

если (S1=0)&(S 2=0)=истина, то отрезок видимый; выполнение в этом случае следующих действий: занесение в результат координат концов отрезка R1=P1 , R 2=P2 и переход к п. 31.

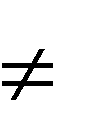
1. Вычисление логического произведения кодов концов отрезка

4



PL= *T*1*i T*2*i* .

*i* 1

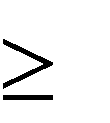
1. Проверка тривиальной невидимости отрезка: если PL 0, то отрезок неви- дим. В этом случае установка признака pr= -1 и переход к п. 31.
2. Проверка видимости первого конца отрезка:

если S1=0 (первый конец видим), то выполнение следующих

действий: R1=P1 (занесение этой вершины в результат), Q=P2 (занесение коор- динат другой вершины в рабочую переменную Q), i=2 (номер шага отсечения), переход к п.15.

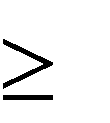
1. Проверка видимости второго конца отрезка:

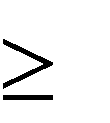
если S2=0 (второй конец видим), то выполнение следующих действий: R1= P2 (занесение этой вершины в результат), Q=P1 (занесение координат дру- гой вершины в рабочую переменную Q), i=2 (номер шага отсечения), переход к п.15.

1. Установка начального значения шага отсечения i=0.
2. Вычисление текущего номера шага отсечения i=i+1.
3. Проверка завершения процедуры отсечения: если i>2, то переход к п.31.
4. Занесение в рабочую переменную Q координат i-ой вершины Q=Pi.
5. Определение расположения отрезка: если X2=X1 (отрезок вертикаль- ный), то переход к п.23 (не может быть пересечения с левой и правой грани- цами отсекателя).
6. Вычисление тангенса угла наклона отрезка m=(Y2-Y1)/(X 2-X1).
7. Проверка возможности пересечения с левой границей отсекателя: если Qx>Xл (пересечения нет), то переход к п.20.
8. Вычисление ординаты точки пересечения отрезка с левой границей от- секателя: Yр=m(Xл - Qx)+Qy .
9. Проверка корректности найденного пересечения: если (Yр Yн)&(Yp Yв)= истина (пересечение корректное), то выполнение следующих действий: Ri.x

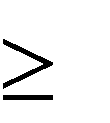
=Xл, Ri..y =Yр (занесение полученных координат в результат), переход к п.12.

1. Проверка возможности пересечения отрезка с правой границей отсека- теля: если Qx<Xп (пересечения нет), то переход к п.23.
2. Вычисление ординаты точки пересечения с правой границей: Yр=m(Xп

- Qx)+Qy .

1. Проверка корректности найденного пересечения: если (Yр Yн)& (Yр Yв)= истина (пересечение корректно), то выполнение следующих действий Ri.x =Xп, ( занесение полученных координат в результат ), переход к п.12.
2. Проверка горизонтальности отрезка: если m=0, то переход к п.12.
3. Проверка возможности пересечения с верхней границей отсекателя: если Qy <Yв (пересечения нет), то переход к п.27.
4. Вычисление абсциссы точки пересечения с верхней границей: Xр=(Yв -- Qy)/m+Qx.
5. Проверка корректности найденного пересечения: если (Xр Xл)&(Xр Xп)=

=истина (пересечение корректно), то выполнение следующих действий: Ri.x=Xр; Ri.y=Yв (занесение полученных координат в результат); переход к п. 12.

1. Проверка возможности пересечения с нижней границей отсекателя: если Qx>Yн (пересечения нет), то переход к п. 30 (вершина невидима и ни одно пересечение не является корректным, следовательно, отрезок невидим).
2. Вычисление абсциссы точки пересечения с нижней границей: Xр=(Yн -- Qy)/m+Qx.
3. Проверка корректности найденного пересечения: (Xр Xл)&(Xр Xп)=

=истина (пересечение корректно), то выполнение следующих действий: Ri.x=Xр; Ri.y=Yв (занесение полученных координат в результат); переход к п. 12.

1. Установка признака видимости pr= -1 (отрезок невидим полностью, так как ни одно пересечение не оказалось корректным).
2. Проверка признака видимости: если pr=1, то вычерчивание отрезка R1R2.
3. Конец.
   * + 1. АЛГОРИТМ ОТСЕЧЕНИЯ САЗЕРЛЕНДА-КОЭНА

Алгоритм Сазерленда-Коэна, как и в предыдущем случае, предусматривает нахождение точек пересечения отрезка со сторонами окна прямоугольной формы. Однако здесь не производится проверка корректности найденных точек пересечения. Найденная точка пересечения разбивает отрезок на две части: полностью невидимую относительно очередной стороны отсекателя и видимую часть. Невидимой будет та часть отрезка, которая заключена от вер- шины отрезка, невидимой относительно текущей стороны окна, до точки пересечения. Этот факт используется в алгоритме для определения части отрезка, подлежащей отсечению. Это связано с тем, что точка, попадающая на ребро отсекателя, имеет нулевой код (она считается видимой), поэтому по- парное логическое произведение кодов концов будет равно нулю.

В алгоритме предусматривается предварительный анализ сумм кодов кон- цов отрезка и попарных логических произведений этих кодов с целью опре- деления тривиально видимых и тривиально невидимых отрезков.

Если отрезок не является ни тривиально видимым, ни тривиально неви- димым, то производится собственно его отсечение (рис.3.6.4). При этом предполагается, что невидимой относительно каждого ребра должна быть первая вершина отрезка. Поэтому в случае необходимости вершины меня- ются местами. Следует отметить, что одновременно обе вершины не могут быть невидимыми относительно одного текущего ребра отсекателя, так как этот факт позволил бы на предварительном этапе отбросить отрезок как триви- ально невидимый.

Для удобства работы данные, задающие отсекатель, заносятся в массив "окно" O(4), в котором O1=Xл, O2=Xп, O3=Yн, O4=Yв. Отсечение производится в определенном порядке: левой, правой, нижней, верхней границами отсекателя. Поэтому для отыскания точек пересечения в выражения (3.6.1) следует на i-ом шаге подставлять i-ые элементы массива O. Такое задание исходных данных позволяет для горизонтального отрезка не проводить третий и четвертый этапы, а для вертикального отрезка - первый и второй этапы.

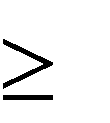
В алгоритме используется признак (флаг) Fl, определяющий располо- жение отрезка: Fl= -1 - отрезок вертикальный, Fl=0 - общего положения, Fl=1

- горизонтальный.

Сам алгоритм Сазерленда-Коэна можно представить в следующем виде:

1. Ввод координат отсекателя Xл (O1), Xп (O2), Yн (O3), Yв (O4).
2. Ввод координат концов отрезка P1(X1,Y1), P2(X 2,Y2).
3. Установка начального значения флага Fl=0.
4. Проверка вертикальности отрезка: если P2.x-P1.x=0 (вертикальный), то Fl= -1, иначе вычислить тангенс угла наклона отрезка m=(P2.y -P1.y)/(P2.x-P1.x).
5. Проверка горизонтальности отрезка: если m=0, то Fl=1.
6. Начало цикла по i от 1 до 4 отсечения отрезка по всем четырем сторонам отсекателя.
7. Обращение к алгоритму (подпрограмме) определения видимости отрезка P1P2 относительно заданного окна. Подпрограмма возвращает признак pr, принимающий следующие значения: pr=1 - отрезок видимый; pr= -1 - отрезок полностью невидимый; pr=0 - отрезок может быть частично видимым.
8. Анализ полученного признака видимости:

если pr= -1, то переход к п. 20; если pr=1, то переход к п. 19.

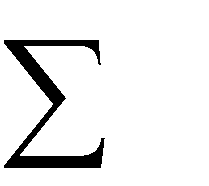
1. Проверка видимости обеих вершин отрезка относительно текущей i-ой стороны окна: если T1i=T2i, то переход к п.18.
2. Проверка видимости первой вершины: если T1i=0 (вершина видима), то обмен местами вершин: R=P1; P1=P2; P2 =R.
3. Проверка вертикальности отрезка: если Fl= -1, то переход к п. 14.
4. Анализ номера шага отсечения: если i 3, то переход к п. 14.
5. Вычисление координат точки пересечения с i-ым ребром отсекателя (левым или правым): P1.y=m(Oi-P1.x)+P1.y; P1.x=Oi. Переход к п. 18.
6. Проверка горизонтальности отрезка: если Fl=1, то переход к п. 18.
7. Проверка вертикальности отрезка: если Fl=-1, то переход к п.17.
8. Вычисление абсциссы точки пересечения отрезка общего положения со стороной отсекателя (верхней или нижней): P1.x=(Oi-P1.y)/m+P1.x .
9. Присвоение ординате вершины отрезка ординаты стороны отсекате- ля: P1.y=Oi .
10. Конец цикла по i (вычисление нового значения параметра цикла i=i+1, анализ его значения и переход на повторное выполнение цикла или выход из цикла).
11. Вычерчивание отрезка P1P2.
12. Конец.

При работе с вертикальными отрезками нет необходимости определять но- мер шага отсечения. Невидимый вертикальный отрезок относительно боко- вого ребра отсекателя будет полностью невидимым, это обнаружится в са- мом начале. Если же он видим относительно бокового ребра, то на первых двух шагах автоматически произойдет переход к следующему шагу цикла.

Алгоритм определения видимости отрезка относительно окна можно пред- ставить следующим образом.

1. Вычисление кодов первой вершины отрезка T1.
2. Вычисление кодов второй вершины отрезка T2.

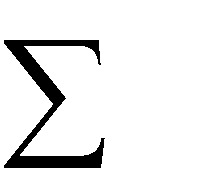
4



1. Вычисление суммы кодов первой вершины S1= *T*1*i* ,

*i* 1

4



1. Вычисление суммы кодов второй вершины S2= *T* 2*i*

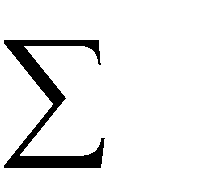
*i* 1

1. Определение полной видимости отрезка: если (S1=0)&(S2=0)=истина, то pr

=1 и переход к п. 8.

1. Вычисление попарного логического произведения кодов концов отрезка:

4



p= *T*1*i T*2*i* .

*i* 1

1. Определение полной невидимости отрезка: если p=1, то pr= -1, иначе pr=0.
2. Конец.

Логическую сумму можно вычислить или как арифметическую или как логи- ческую сумму.

* + - 1. ОТСЕЧЕНИЕ ОТРЕЗКА С ПОМОЩЬЮ АЛГОРИТМА РАЗБИЕНИЯ

СРЕДНЕЙ ТОЧКОЙ

В предыдущих алгоритмах координаты точки пересечения отрезка с ребром отсекателя определялись непосредственно путем решения соответствующих уравнений. В данном алгоритме непосредственное вычисление координат от- сутствует, точка пересечения определяется с использованием двоичного поиска. Рассматриваемый алгоритм является частным случаем алгоритма отсечения Сазерленда-Коэна. Он был разработан Спруллом и Сазерлендом для аппарат- ной реализации, так как программная реализация данного алгоритма медленнее, чем алгоритма Сазерленда-Коэна; аппаратная же, за счет того, что сложение и

деление на 2 очень быстры, намного эффективнее.

Алгоритм предусматривает вычисление кодов концевых точек отрезков объ- ектов синтезируемой сцены и проверок, определяющих полную видимость от- резков (коды обоих концов отрезка равны 0 ), и полную невидимость отрезков ( побитное логическое произведение кодов концевых точек не равно нулю). Те же отрезки, о которых нельзя судить столь однозначно, разбивают на две рав- ные части. К каждой из половин отрезков применяют те же проверки до тех пор, пока не будет обнаружено пересечение со стороной окна отсечения, либо

пока он не выродится в точку. Затем определяется видимость вычисленной точ- ки. Таким образом, процесс определения пересечения сводится к двоичному по- иску, столь эффективно реализуемому аппаратно.

Рассмотрим более подробно работу алгоритма на примере отрезка GK (рис.3.6.5). После анализа кодов точек G и K нельзя однозначно судить о его полной видимости или полной невидимости, поэтому разобьём его средней точ- кой S. Однако это разбиение приводит к одинаковым результатам для обеих по- ловин. Далее разобьём отрезок SK средней точкой R. Теперь отрезок SR полно- стью видим, а отрезок RK видим частично. Запомним точку R как текущую ви- димую точку, наиболее удаленную от G и продолжим разбиение отрезка RK. При обнаружении следующей видимой средней точки, она запоминается как текущая, наиболее удаленная от G. Разбиения продолжаются до тех пор, пока не будет обнаружено пересечение с нижней стороной окна с заданной точностью. Это пересечение и будет являться самой удаленной от G видимой точкой. Далее аналогичным образом обрабатывается отрезок GS. Для отрезка GK наиболее удаленной от K видимой точкой будет точка пересечения с левой стороной ок- на отсечения.

При отсечении отрезков объектов синтезируемой сцены возможны также случаи расположения отрезков подобно отрезкам EF и NM рис.3.6.5. У отрезка NM полностью виден конец N, поэтому производится поиск точки пересечения только с одной (в данном случае левой) стороной окна отсечения. Отрезок же EF невидим, однако в отличие от отрезка CD, он пересекает диагональ окна и требует дополнительных проверок

Каждое разбиение средней точкой дает грубую оценку искомых пересечений, поэтому разбиения производятся до тех пор, пока не будет обнаружено пересе- чение с заранее заданной точностью.

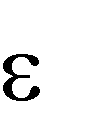
Исходя из возможных вышеизложенных проверок частей отрезков на види- мость и невидимость, полный анализ каждой концевой точки отрезка состоит из следующих трех этапов:

-проверка на видимость, в положительном случае она принимается за наи- более удаленную видимую точку и анализ завершен, в отрицательном - переход к следующему этапу;

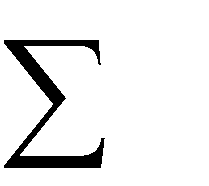
-проверка на полную невидимость отрезка, в положительном случае отре- зок не рисуется, следовательно, никакая информация о нем не запоминается и анализ завершен, в отрицательном - переход к следующему этапу;

-грубая оценка наиболее удаленной видимой точки путем деления иссле- дуемого отрезка средней точкой. Применение вышеизложенных проверок и оценок к двум полученным частям отрезка. Если часть отрезка является полно- стью невидимой, то средняя точка даёт верхнюю оценку для наиболее удален- ной видимой точки. Если же средняя точка видима, то она дает нижнюю оценку для наиболее удаленной видимой точки. Аналогично исследуется другая часть исходного отрезка. Процедуры с частями отрезков повторяются до тех пор, пока часть отрезка не становится так мала, что его средняя точка совпадает с его концами с заранее заданной точностью. Тогда полученная средняя точка прове- ряется на видимость и анализ завершается.

Более подробно алгоритм разбиения средней точкой можно представить сле- дующим образом:

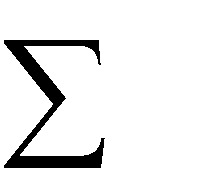
1. Ввод координат отсекателя Xл, Xп, Yн, Yв.
2. Ввод координат концов отрезка P1(X1,Y1), P2(X 2,Y 2).
3. Ввод точности вычисления точки пересечения отрезка с границей отсе- кателя.
4. Установка номера шага отсечения i=1.
5. Вычисление кодов концевых точек и запись их в соответствующие масси-

4



вы T1 и T2 размерностью 1х 4, вычисление сумм кодов концов S1= *T*1*i* , S2=

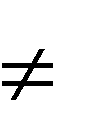
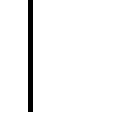
*i* 1

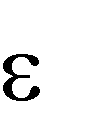
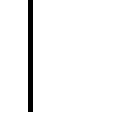


4

= *T* 2*i*

*i* 1

1. Проверка полной видимости отрезка. Если коды обоих концов отрезка рав- ны нулю (полная видимость отрезка), то переход к п. 9.
2. Проверка полной невидимости отрезка. Вычисление побитного логического произведения кодов концевых точек отрезка. Если произведение отлично от ну- ля (отрезок невидим), то переход к п. 10.
3. Анализ частично видимого отрезка в том случае, если побитовое логическое произведение кодов его концов равно нулю:
   1. Поиск наиболее удаленной от P1 видимой точки S исследуемого отрезка. Запоминание исходной точки P1 в промежуточной переменной R.
   2. Проверка на окончание процесса решения: если i>2, то определение логиче- ского произведения pr кодов концов отрезка. Если pr 0, то переход к п.10, иначе переход к п.9.
   3. Проверка точки P2 на наиболее удаленную от P1 видимую точку отрезка. Если сумма всех элементов массива T2 равна нулю (S2), то переход к пункту 8.12.
   4. Проверка нахождения точки пересечения отрезка с границами отсекателя. Если P1- P2 (расстояние между концевыми точками исследуемого отрезка меньше допустимой погрешности), то переход к пункту 8.12.



* 1. Вычисление средней точки Pср. отрезка: Pср. = (P1 + P2 )/2 (Pср.x = (P1.x + P2..x )/2 ; Pср.y = (P1.y + P2.y )/2).
  2. Запоминание текущей точки P1: Pm=P1.
  3. Замена точки P1 на среднюю точку: P1= Pср .
  4. Вычисление нового кода T1 точки P1.
  5. Вычисление произведения pr кодов концов нового отрезка P1P2. 8.10.Проверка полной невидимости отрезка P1P2. Если побитовое логическое произведение pr кодов концевых точек равно нулю, то переход к пункту 8.4. В противном случае отрезок P1P2 невидим.

8.11. Возврат к предыдущему отрезку P1P2 : P1 = Pm, = Pср , переход к пункту

8.4. ( Вычислена наиболее удаленная от точки P1 видимая точка отрезка). 8.12..Поиск наиболее удаленной от P2 видимой точки отрезка. Замена мест то- чек P1 и P2 : P1 = P2 , P2=R. Увеличение шага выполнения отсечения i=i+1. Пере- ход к п.5.

1. Визуализация отрезка. 10.Конец.

При программной реализации алгоритма целесообразно для вычисления ко- дов концевых точек отрезков и их логических произведений использовать от- дельные функции.

* + 1. ОТСЕЧЕНИЕ ПРОИЗВОЛЬНОГО ОТРЕЗКА ПРОИЗВОЛЬНЫМ ВЫПУКЛЫМ ОКНОМ (Алгоритм Кируса - Бека)

В трех уже рассмотренных алгоритмах отсечения отрезков предполагалось, что отсекатель является прямоугольником со сторонами, параллельными осям координат. Однако, очень часто оно повернуто относительно координатной сет- ки или не является прямоугольным. Поэтому Кирус и Бек предложили алго- ритм отсечения окном произвольной выпуклой формы.

В этом алгоритме для определения местоположения точки относительно окна отсечения используется вектор нормали и параметрическая форма задания от- резка. Параметрическое уравнение отрезка P1P2 имеет вид:



P(t)= P1 + (P2- P1)t; 0 t 1, где t - параметр.

Фактически это уравнение является векторным, оно сводится к двум одно- мерным параметрическим уравнениям следующего вида:



t 1

t 1

Px(t)= P1.x + (P2.x- P1.x)t; 0

Py(t)= P1.y + (P2.y- P1.y)t; 0 ( 3.6.2)

Для прямоугольного окна со сторонами, параллельными осям координат, точки пересечения отрезков с его границами определяются достаточно просто, поскольку одна из координат точки пересечения заранее известна и остается вычислить только вторую координату из (2.6.2 ):

t = (P(t)-P1)/(P2 - P1)

Значения параметра t для точек пересечения отрезка с границами отсекателя определяются из следующих соотношений:

t = *Xл P*1.*x* 0



t 1

(для левой границы)

*P*2.*x P*1.*x*

t = *Xп P*1.*x* 0



t 1

(для правой границы)

*P*2 .*x P*1.*x*

t = *Yн P*1.*y* 0



t 1

(для нижней границы)

*P*2 .*y P*1.*y*

t = *Yв P*1. *y* 0



t 1

(для верхней границы)

*P*2 . *y P*1. *y*

Если в результате вычислений получаются значения параметра t, выходящие

за пределы интервала 0 1, то такие решения отвергаются, так как они соот-

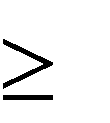


t

ветствуют точкам, лежащим вне рассматриваемого отрезка. Естественной явля- ется попытка определения видимости отрезка на основе анализа получаемых значений параметра t (рис.3.6.6).

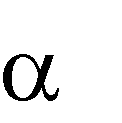
Так, для полностью видимого отрезка точки пересечения лежат за пределами отрезка, поэтому значения параметра t будут лежать за пределами допустимого интервала (рис.3.6.7). Данный факт можно было бы рассматривать в качестве критерия определения видимых отрезков. Однако рассмотрение полностью не- видимого отрезка показывает, что и для такого отрезка значения параметра t, соответствующие точкам пересечения со сторонами отсекателя, также лежат за пределами допустимого интервала. Таким образом, не удается получить простой критерий, используя который легко было бы идентифицировать полностью ви- димые и полностью невидимые отрезки. Поэтому требуется специальный метод отсечения параметрически заданных отрезков выпуклым отсекателем.

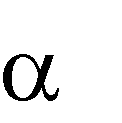
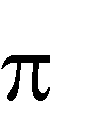
Пусть в качестве окна отсечения используется выпуклая область C, которая может быть любым плоским выпуклым многоугольником. Тогда, внутренняя нормаль nв , в произвольной точке A, лежащей на любой границе C, удовлетво-

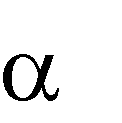
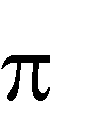
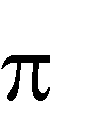
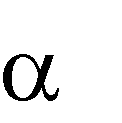


0

ряет следующему условию: nв ( B - A ) , где B - любая другая точка на грани-

це отсекающей области. Правильность условия иллюстрируется на рис. 3.6.8. Угол между внутренней нормалью nв к точке A и вектором, проведенным из данной точки к любой другой, расположенной на границе отсекающей области,

всегда находится внутри интервала: - /2 /2. Таким образом, косинус угла всегда положителен, а, следовательно, положительно и скалярное произведе- ние этих векторов. В свою очередь, угол между внешней нормалью nвш и любым



из подобных векторов равен ( - ), косинус которого всегда отрицателен.

Выше уже говорилось, что для анализа частично невидимых отрезков необ- ходимо определять точки пересечения исследуемых отрезков с границами окна отсечения. Для этого представим исследуемый отрезок в параметрическом ви- де:

P(t) = P1 + (P2 - P1) t , ( 3.6.3)



где t - параметр; при наложении ограничений на t: 0 t 1, получим именно отрезок, а не бесконечную прямую. Далее предположим, что f - граничная точ- ка выпуклой области С, а nв **-** внутренняя нормаль к одному из ограничиваю- щих эту область ребер. Тогда для любой точки отрезка AB, то есть для любого значения параметра t:

* из условия nв[P(t) - f] <0 следует, что вектор[P(t) - f] направлен из облас- ти C;
* из условия nв[P(t) - f] =0 следует, что вектор [P(t) - f] принадлежит плос- кости, которая проходит через граничную точку k и перпендикулярна нормали nв ;
* из условия nв [P(t) - f] >0 следует, что вектор [P(t) - f] направлен внутрь

C (рис.3.6.9).

Из всех этих условий следует, что прямая пересекает замкнутую выпуклую об- ласть C ( выпуклый многоугольник в нашем двумерном случае) равно в двух точках. Если эти две точки не принадлежат одной граничной плоскости или ребру, тогда уравнение nв [ C(t) - f] = 0 имеет только одно решение. Если точка f лежит на той граничной плоскости или на том ребре, для которых nв является внутренней нормалью , то точка на отрезке P(t) , которая удовлетворяет уравне- нию nв [ P(t) - f] = 0 , будет точкой пересечения этого отрезка с ребром окна от- сечения.

Таким образом, скалярное произведение внутренней нормали к i-му ребру окна отсечения на вектор, начинающийся в точке, лежащей на i - ом ребре окна отсечения, и заканчивающийся в любой точке исследуемого отрезка, т.е.

nвi [P(t) - fi] , (3.6.4)

где i=1,2,3, ..m; m - число сторон окна отсечения

будет положительно, равно нулю или отрицательно, в зависимости от того, бу- дет ли исследуемая точка лежать внутри окна отсечения, на границе окна или вне его.

Подставив параметрическое представление отрезка (3.6.3) в соотношение скалярного произведения (3.6.4), получим уравнение для определения точки пе- ресечения отрезка с ребром полигонального выпуклого окна отсечения:

nвi [ P1 + (P2-P1) t - fi] =0, (3.6.5)

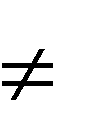
которое приведем к виду:

nвi [P1 - fi] + nвi [P2-P1] t = 0. (3.6.6)

В уравнении (3.6.6) вектор [P2-P1] определяет ориентацию отрезка, а вектор [P1

* fi] пропорционален расстоянию от начала отрезка до исследуемой граничной точки. Обозначим вектор ориентации как D = P2-P1, а для второго вектора вве- дем некоторый коэффициент Wi= P1- fi. С учетом последних обозначений урав- нение (3.6.6) можно записать в следующем виде:

t ( nвi D ) + Wi nвi **=** 0 (3.6.7)

Решая уравнение (2.6.7) относительно t, получим:

t = - ( Wi nвi) **/**( D nвi) , D 0, i=1,2..m . (3.6.8)

Анализируя уравнение (3.6.8), замечаем, что знаменатель D nвi принимает нулевое значение в двух случаях. Во-первых, в случае вырождения отрезка в точку, то есть когда P2=P1. Эта точка лежит вне окна отсечения, когда числи- тель Wi nвi < 0 , на границе, когда он равен нулю и внутри, когда числитель Wi nвi >0. Однако этот случай не представляет большого интереса, т.к. нас интере- суют отрезки, а не точки. Во-вторых, знаменатель равен нулю, когда равно нулю скалярное произведение D nвi, т.е. когда отрезок параллелен ребру отсекателя.

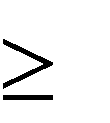
Используя уравнение (3.6.8), можно определить значения параметра t, при которых происходят пересечения исследуемого отрезка с ребрами окна отсече- ния. Если значения t не принадлежат интервалу 0 t 1, то их не рассматри- вают, поскольку они соответствуют точкам, лежащим вне исходного отрезка. Несмотря на то, что отрезок может пересечь замкнутую выпуклую область не более, чем в двух точках, при решении уравнения (3.6.8) можно получить более двух решений в интервале 0 t 1. Полученные решения следует разбить на две группы: верхнюю и нижнюю, в зависимости от близости найденной точки пе- ресечения к началу или концу отрезка. Чтобы отрезок был видимым относи- тельно всего отсекателя, он должен быть видим относительно всех ребер отсе- кателя одновременно. Концам видимой части отрезка будут соответствовать два значения параметра t, одно из которых является максимальным значением из нижней группы tmaxmin, а второе - минимальным из верхней группы tminmax (рис. 3.6.10). Найденное значение параметра t для очередной точки пересечения рас- сматривают в качестве возможного верхнего предела tв, если знаменатель Dnвi < 0; в случае же, когда знаменатель положителен, значение параметра t опреде- ляет точку, которую относят к нижней границе видимости tн.



Алгоритм Кируса - Бека может быть представлен следующим образом:

1. Ввод координат концевых точек отрезка P1(P1.x, P1.y) , P2(P2.x, P2.y)
2. Ввод числа сторон m выпуклого многоугольника - окна отсечения и коор- динат его вершин (массив C).
3. Вычисление вектора ориентации отрезка D = S2 - S1.
4. Инициализация пределов значений параметра t при условии, что отрезок полностью видимый, то есть tн = 0, tв = 1.
5. Начало цикла по всем сторонам отсекающего окна (i = 1..m). Выполнение для каждой i-ой стороны окна следующих действий:
   1. Вычисление вектора внутренней нормали к очередной i-ой стороне окна отсечения - nвi
   2. Определение граничной точки fi каждой стороны отсекающего окна (точки, лежащей на i-ой стороне окна).
   3. Вычисление вектора Wi = P1 - fi .
   4. Вычисление скалярного произведения векторов Wск i= Wi nвi .
   5. .Вычисление скалярного произведения векторов Dск i= Dnвi .
   6. Проверка на равенство нулю скалярного произведения Dск (вырождение отрезка в точку или его параллельность стороне отсекателя). Если Dnвi =

=0 , тогда переход к пункту 5.9 .

* 1. Вычисление параметра t: t = - ( Wскi **/** Dскi) (отрезок не вырожден в точ- ку и не параллелен стороне отсекателя).
  2. Определение верхнего и нижнего пределов параметра t :
     1. Поиск нижней границы параметра t, если Dскi > 0 : если t>1, то переход к п.7 (отрезок невидим, т.к. нижний предел параметра t превышает единицу и пересечение с отсекателем имеет место не для самого отрезка, а для его продолжения за вершину P2.; если t 1, то tн = max ( t, tн ) (выбор максимального значения из теку- щего значения параметра t и ранее вычисленного значения ниж- ней границы параметра).
     2.  Поиск верхней границы параметра t, если Dскi < 0 : если t<0, топереход к п.7 (отрезок невидим, т.к. верхний предел параметра t отрицателен и пересечение с отсекателем имеет место не для самого отрезка, а для его продолжения за вершину P1.; если t 0, то tв = min ( t, tв ) (выбор минимального значения из текущего значения параметра t и ранее вычисленного значения верхней границы параметра).
  3. Проверка видимости точки, в которую выродился отрезок, или проверка видимости произвольной точки отрезка в случае его параллельности стороне отсекателя: если Wскi < 0, то отрезок (точка) невидим и переход к п. 7; если Wскi

>0, то отрезок (точка) видим относительно текущей стороне отсекателя и пере- ход к п. 5.10.

* 1. Конец цикла по сторонам отсекателя.

1. Проверка фактической видимости отсеченного отрезка. Если tн > tв, то переход к пункту 7 , иначе визуализация отрезка в интервале от P(tн) до P(tв).
2. Конец алгоритма.

При программной реализации целесообразно скалярные произведения векто- ров вычислять с помощью отдельной функции.

В заключение, рассмотрим пример отсечения отрезка выпуклым шестиуголь- ником. На рис. 3.6.11 показан шестиугольник P1-P6 , служащий окном отсече- ния, и отрезок AB, который отсекается данным окном. В таблице 3.1 приведены необходимые вычисления, полученные после работы алгоритма Кируса - Бека. В качестве примера заполнения таблицы, рассмотрим ее строку, соответствую- щую ребру P5P6. Вектор ориентации D = B - A = [ 3 3 ] - [-1 1 ] = [ 4 2 ]. Внут- ренняя нормаль **nв** к ребру P5P6 равна **nв** = [ -1 0 ], тогда D **nв** = -4. Для гра- ничной точки k ( 2,3 ) коэффициент W = A - k = [-1 1 ] - [2 3 ] = [ -3 -2 ], a W **nв** = 3. Так как D **nв** = -4 < 0, то найден верхний предел и tв = -3/ (-4) = 3/4. Из анализа таблицы 1 видно, что максимальное среди нижних значений t - tн

=1/4, а минимальное среди верхних t - tв = 3/4 , то есть отрезок AB видим в ин- тервале: 1/4 t 3/4 .



Таблица 3.1

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ребро | Нормаль  **nв** | | Гранич- ная точка  k | Коэффици- ент W | | W **nв** | D **nв** | Нижний предел  tн | Верхний предел  tв |
| P1P2 | [ 1 | 1 ] | ( 1 , 0 ) | [ -2 | 1 ] | -1 | 6 | 1/6 |  |
| P2P3 | [ 1 | 0 ] | ( 0 , 2 ) | [ -1 | -1 ] | -1 | 4 | 1/4 |  |
| P3P4 | [ 1 | - 1 ] | ( 0 , 2 ) | [ -1 | -1 ] | 0 | 2 | 0 |  |
| P4P5 | [ 0 | -1 ] | ( 2 , 3 ) | [ -3 | -2 ] | 2 | -2 |  | 1 |
| P5P6 | [ -1 | 0 ] | ( 2 , 3 ) | [ -3 | - 2 ] | 3 | -4 |  | 3/4 |
| P6P1 | [ 0 | 1 ] | ( 1 , 0 ) | [ -2 | 1 ] | 1 | 2 | -1/2 |  |

В данном примере отсечение отрезка производится внутренней обла- стью окна отсечения. Аналогично отсекается отрезок и внешней областью отсе- кающего окна. Для этого определяются, а затем вычерчиваются части отрезка, лежащие вне окна отсечения.

* + 1. ОПРЕДЕЛЕНИЕ ФАКТА ВЫПУКЛОСТИ МНОГОУГОЛЬНИКА. ВЫЧИСЛЕНИЕ ВНУТРЕННИХ НОРМАЛЕЙ

Алгоритм Кируса - Бека позволяет производить отсечение отрезков произ- вольным выпуклым окном, поэтому необходимо уметь устанавливать факт вы- пуклости отсекателя. Существует ряд способов проверки многоугольника на выпуклость. Одним из них является вычисление векторных произведений его смежных сторон (рис.3.6.12). Результирующий вектор, получаемый в результате умножения векторов, перпендикулярен плоскости, в которой лежит рассматри- ваемый многоугольник. Будем считать, что это произведение имеет положи- тельный знак, если кратчайший поворот для совмещения первого вектора со вторым производится против часовой стрелки. Произведение будет иметь знак минус, если кратчайший поворот первого вектора для совмещения со вторым должен производиться по часовой стрелке.

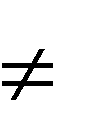
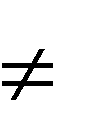
Правило определения выпуклости многоугольника следующее. Если знаки всех векторных произведений смежных сторон многоугольника равны нулю, то многоугольник вырождается в отрезок. Если есть, как положительные так и от- рицательные знаки, то многоугольник невыпуклый. Если все знаки векторных произведений смежных сторон неотрицательные, то отсекающий многоуголь- ник выпуклый, а внутренние нормали ориентированы влево от его контура. Ес- ли же все знаки неположительны, то многоугольник также является выпуклым, а внутренние нормали ориентированы вправо от его контура.

Второй способ (рис.3.6.13) также основан на вычислении векторных произ- ведений. В этом случае одна из вершин выбирается в качестве базовой и из нее проводятся вектора в остальные вершины многоугольника. Далее вычисляются векторные произведения векторов, заканчивающихся в последовательных вер- шинах многоугольника. Если среди знаков этих произведений будут как поло- жительные, так и отрицательные, то многоугольник - невыпуклый. Если все знаки произведений одинаковые, то сразу сделать вывод о характере много- угольника нельзя, а следует в качестве базовой выбрать следующую вершину многоугольника и повторить вычисления. Если для всех вершин многоугольни- ка знаки произведений будут одинаковые, то многоугольник будет выпуклым. Если же для какой-то вершины знаки произведений получатся разными, то это будет свидетельствовать о невыпуклости многоугольника, в этом случае даль- нейшие проверки проводить не следует.

Помимо установления факта выпуклости многоугольник алгоритм требует вычисления внутренних нормалей к сторонам отсекателя. Причем достаточно определить направление нормали. Для этого используется свойство скалярного произведения двух перпендикулярных векторов - оно равно нулю. Пусть Ax ,A y

-компоненты вектора A стороны многоугольника (считаются известными), nx, ny - искомые компоненты вектора нормали. Тогда можно записать следующее равенство:

nA=( nxi +nyj)( Axi +Ayj )=( nxAx + nyAy)=0 nxAx = - nyAy

Положив ny равным 1, получаем n= -Ay/ Ax i + j (Ax 0, т.е. сторона много- угольника не параллельна оси ординат). Если сторона многоугольника парал- лельна оси ординат, т.е. Ax=0, Ay 0, то вектор нормали параллелен оси абсцисс и следует положить nx=1, ny=0.

Найденную нормаль надо проверить, является ли она внутренней или внешней. Для этой проверки можно воспользоваться следующим фактом. Если вектор стороны многоугольника образован как разность векторов пары смеж- ных его вершин Ai-1 и Ai и если скалярное произведение нормали n и вектора от Ai-1 до Ai+1 положительно, то n - внутренняя нормаль. В противном случае n - внешняя нормаль. Для получения внутренней нормали полученное значение n надо умножить на -1.

Можно предложить и третий способ (рис.3.6.14) определения факта выпук- лости многоугольника с одновременным определением вектора внутренней нормали и разбиения исследуемого многоугольника на выпуклые многоуголь- ники в случае его невыпуклости. Этот способ основан на использовании пере- носов и поворотов исследуемого многоугольника. Предполагается, что вершины многоугольника пронумерованы в направлении против часовой стрелки. Рас- сматриваемый алгоритм может быть представлен в следующем виде:

1. Осуществить перенос многоугольника таким образом, чтобы очередная i- ая вершина совпала с началом координат.
2. Осуществить поворот многоугольника относительно начала координат та- ким образом, чтобы следующая (i+1)- ая вершина оказалась на положи- тельной полуоси абсцисс x.
3. Определить знак ординаты у всех (i+2)-ых вершин. Если знаки ординат у всех (i+2)-ых вершин неотрицательные, то многоугольник выпуклый от- носительно очередной стороны, соединяющей i-ую и (i+1)-ую вершины.

В повернутой системе координат внутренняя нормаль к ребру, соеди- няющему i-ую и (i+1)-ую вершины, имеет нулевую абсциссу и ординату, знак которой совпадает со знаком ординаты (i+2)-ой вершины. Для опре- деления ориентации внутренней нормали исходного многоугольника сле- дует выполнить обратные повороты.

Если ордината (i+2)-ой вершины равна нулю, то i-ая, (i+1)-ая и (i+2)-ая вершины лежат на одной прямой, т.е. ребра, соединяющие i-ую и (i+1)-ую вершины, а также (i+1)-ую и (i+2)-ую вершины коллинеарны. Если ординаты всех (i+2)-ых вершин равны нулю, то многоугольник вы- рожден, т.е. является отрезком.

Если знак ординаты (i+2)-ой вершины отрицателен, то многоугольник невыпуклый, его следует разбить на части. Многоугольник разрезается вдоль положительной полуоси x. Для этого ищутся стороны многоуголь- ника, пересекающие ось абсцисс, и для них находится ближайшая к началу координат точка пересечения с этой осью, абсцисса которой больше зна- чения xi+1 (x> xi+1). Далее формируются два многоугольника: первый мно- гоугольник образован вершинами, начиная с (i+1)-ой и кончая найденной точкой пересечения (многоугольник лежит целиком ниже оси абсцисс), второй многоугольник образован всеми вершинами исходного много- угольника, не вошедшими в первый многоугольник, и точкой пересечения. Полученные многоугольники могут быть невыпуклым, поэтому описанная процедура применяется к ним до тех пор, пока все многоугольники, полу- чаемые в процессе разбиения, не станут выпуклыми.

* + 1. ОТСЕЧЕНИЕ НЕВЫПУКЛЫМ ОТСЕКАТЕЛЕМ. СТИРАНИЕ (ВНЕШНЕЕ ОТСЕЧЕНИЕ)

Стирание - это операция удаления изображения в пределах выделенного окна (отсекателя). В результате стирания должно остаться изображение, находящееся вне отсекателя. Часто операцию стирания называют также внешним отсечени- ем. При отсечении окном невыпуклой формы приходится использовать внешнее отсечение. Кроме того, внешнее отсечение может использоваться при работе на экране дисплея с несколькими окнами (рис.3.6.15). Изображение, расположен- ное в окнах с меньшим приоритетом, отсекается (внутренним образом) самим окном и отсекается внешним образом окнами, имеющими более высокий при- оритет.

При внутреннем отсечении получаются значения параметра t (tн, tв), опреде- ляющие видимую часть отрезка (лежащую в пределах окна). Если же взять от- резки, для которых значения этого параметра лежат в интервалах (0, tн) и (tв, 1), то мы выполним фактически внешнее отсечение (рис.3.6.17).

Для выполнения отсечения окном невыпуклой формы (рис.3.6.16) следует преобразовать исходный невыпуклый многоугольник в выпуклый (это достига- ется путем соединения новым ребром вершин, соседних с той вершиной много- угольника, в которой нарушается выпуклость многоугольника). Затем выполня- ется внутреннее отсечение новым (дополненным) многоугольником. На сле- дующем шаге отсеченный отрезок подвергается внешнему отсечению по грани- цам дополняющего многоугольника. В итоге получаем отрезок, отсеченный не- выпуклым окном.

* + 1. ОТСЕЧЕНИЕ МНОГОУГОЛЬНИКОВ

Многоугольник можно рассматривать как совокупность отрезков и попы- таться применить ранее рассмотренные алгоритмы отсечения отрезков для от- сечения ребер многоугольника. Однако это может привести к тому, что перво- начально замкнутая фигура (многоугольник) превратится в незамкнутую. Более того, можно получить несколько незамкнутых многоугольников или просто со- вокупность отрезков. Однако, если рассматривать многоугольник как часть плоскости, ограниченную замкнутой ломаной линией, то такой подход будет явно неверным. В результате проведения операции отсечения необходимо полу- чить один или несколько замкнутых многоугольников (рис.3.6.18).

* + - 1. АЛГОРИТМ ОТСЕЧЕНИЯ МНОГОУГОЛЬНИКА ВЫПУКЛЫМ ОКНОМ (алгоритм Сазерленда-Ходжмена)

Алгоритм Сазерленда-Ходжмена позволяет провести отсечение произволь- ного (выпуклого или невыпуклого) многоугольника по границам выпуклого от- секателя. Идея алгоритма достаточно проста. На каждом шаге отсечения исход- ный и промежуточные многоугольники отсекаются последовательно очередной границей отсекателя. Отсечение многоугольника относительно одной прямой не представляет больших затруднений.

Обычно исходный многоугольник задается количеством своих вершин N(в порядке их обхода ) и их координатами P1, P2, . . ., PN(две последовательные вершины являются концами одного ребра, например, P1P2, P2P.3, ... PnP1.). От- секатель также задается количеством вершин M и их координатами C1, C 2, . .., Cm..

Исходный многоугольник отсекается сначала, например, левой стороной от- секателя (выберем направление по часовой стрелке), в результате чего получа- ется промежуточный многоугольник (рис.3.6.19). Этот многоугольник отсекает-

ся далее верхней границей отсекателя, получается второй промежуточный мно- гоугольник. Далее процесс продолжается аналогичным образом пока не будет выполнено отсечение последней границей отсекателя.

На каждом шаге отсечения алгоритм работает со списком вершин много- угольника и в результате также получается список вершин нового многоуголь- ника. Причем все вершины нового многоугольника лежат по видимую сторону очередной границы отсекателя. Каждое ребро многоугольника отсекается неза- висимо от других, поэтому для подробного рассмотрения сути алгорита доста- точно рассмотреть все возможные комбинации взаимного расположения ребра отсекаемого многоугольника и ребра отсекателя.

Будем рассматривать каждую точку Pi (кроме первой P1) из списка вершин отсекаемого многоугольника как конечную точку ребра, а начальной точкой S этого ребра является вершина Pi-1, предшествующая рассматриваемой в этом списке. Возможны следующие четыре случая взаимного расположения ребра отсекаемого многоугольника и ребра отсекателя (рис.3.6.20):

* + ребро целиком лежит по видимую сторону ребра отсекателя и оно являет- ся полностью видимым;
  + ребро целиком лежит по невидимую сторону ребра отсекателя и оно явля- ется полностью невидимым;
  + ребро видно частично, причем точка S невидима, а точка Pi видима, т.е. ребро входит в область видимости (внутрь отсекателя);
  + ребро видно частично, причем точка S видима, а точка Pi невидима, т.е. ребро выходит из области видимости (выходит из отсекателя).

Рассмотрев эти варианты, можно сделать вывод о необходимости занесения в список вершин отсеченного многоугольника нуля, одной или двух видимых то- чек. Если исследуемое ребро полностью невидимо (второй случай), то в резуль- тирующий список заносить ничего не надо. Если ребро видимо полностью (пер- вый случай), то в результирующий список заносится одна вершина (конечная), т.к. вершина S на предыдущем шаге являлась конечной точкой предыдущего ребра и была занесена в результирующий список. Если ребро многоугольника видимо частично и выходит из отсекателя (четвертый случай), то необходимо также в результирующий список занести одну точку. Этой единственной точкой будет точка пересечения ребра многоугольника с ребром отсекателя. Поэтому потребуется предварительно определить координаты этой точки пересечения. Видимая начальная точка, как и в предыдущем случае, будет уже присутство- вать в результирующем списке. Если ребро многоугольника входит внутрь отсе- кателя (третий случай), то необходимо найти точку пересечения ребер много- угольника и отсекателя и занести ее в результирующий список, конечная точка Pi в этом случае также видима и она тоже подлежит занесению в результирую- щий список.

Особым образом обрабатывается первая точка многоугольника: для нее тре-

буется определить только видимость. Если точка видима, то она заносится в ре- зультирующий список и становится начальной точкой первого ребра. Если же она невидима, то она просто становится начальной точкой ребра и в результи- рующий список не заносится.

Специального рассмотрения требует также и последнее ребро. Первая точка запоминается в промежуточной переменной, например, F. Полученное ребро PnF обрабатывается затем обычным образом.

Рассмотрим теперь способы определения видимости вершин многоугольника и нахождения координат точек пересечения ребер многоугольника и отсекателя. Определение видимости точки означает определение стороны (справа или сле- ва) границы отсекателя, по которую лежит эта точка. Если граница отсекателя обходится по часовой стрелке, то его внутренняя область лежит по правую сто- рону от границы. При противоположном направлении обхода она лежит по ле- вую сторону.

Ранее были уже рассмотрены способы решения стоящей задачи. Так в алго- ритме Кируса-Бека использовалось скалярное произведения вектора внутрен- ней нормали на вектор, начало которого лежит в произвольной точке прямой (на границе отсекателя), а конец - в исследуемой точке. Второй способ заключа- ется в подстановке координат исследуемой точки в пробную функцию, роль ко- торой играет уравнение прямой. Авторы рассматриваемого алгоритма предло- жили третий способ (рис.3.6.21), основанный на вычислении векторного произ- ведения векторов. Если две точки A1A2 лежат на ребре отсекателя, а A3 является пробной точкой, то следует вычислить векторное произведение векторов A1A3\*A1A2. Поскольку эти вектора лежат в одной плоскости (три точки задают плоскость), то результирующий вектор будет перпендикулярен этой плоскости и отличной от нуля у него будет только компонента Z=(X3-X1)(Y2-Y1)-(X2- X1)(Y3-Y1). Анализируя знак этой компоненты, можно сделать вывод о взаим- ном положении точки A3 и отрезка A1A2: если знак положительный, то точка лежит справа от отрезка, если знак - отрицательный, то слева, если компонента нулевая, то точка лежит на прямой, проходящей через отрезок.

Отрезок будет видимым, если оба его конца видимы, отрезок будет полно-

стью невидимым, если оба конца невидимы. Если же один конец видим, а дру- гой невидим, то ребро многоугольника пересекает прямую, проходящую через ребро отсекателя. В этом случае вычисляются координаты точки пересечения. Для решения этой задачи можно использовать ранее рассмотренные алгоритмы отсечения отрезка: Кируса-Бека, разбиения средней точкой. В случае использо- вания стандартного отсекателя прямоугольной формы со сторонами, параллель- ными координатным осям можно использовать и другие алгоритмы - простой, Сазерленда-Коэна.

Воспользуемся здесь, как и в алгоритме Кируса-Бека, параметрическим зада- нием отрезков. Пусть P1P2 - концевые точки ребра отсекаемого многоугольника, а C1, C 2 - концевые точки ребра отсекателя. В параметрической форме эти от- резки будут описаны следующим образом:



P(t)= P1 + ( P2 - P1)t, 0 t 1 C(s)=C1+ (C2 - C1)s, 0 s 1

В точке пересечения выполняется равенство P(t)=C(s). Данное равенство фактически является векторным, на его основе можно записать систему двух уравнений с двумя неизвестными P.x(t)=C.x(s), P.y(t)=C.y(s). Если система не имеет решений, то отрезки параллельны, если решение есть, но оно достигается при значениях параметров t или s, выходящих за требуемый диапазон, то пере- секаются не сами отрезки, а прямые, проходящие через них (продолжения от- резков). В рассматриваемом алгоритме необходимо определять точки пересече- ния ребра отсекаемого многоугольника с прямой, проходящей через ребро отсе- кателя. Поэтому ограничение, накладываемое на параметр s, учитывать не сле- дует. Чтобы не производить лишних вычислений, в алгоритме предварительно устанавливается факт пересечения ребер отсекаемого многоугольника и отсека-

теля. Для этого используются результаты проверки концов ребра отсекаемого многоугольника на видимость. Если концевые точки отрезка имеют разную ви- димость (одна вершина видима, а другая невидима), то пересечение есть, в про- тивном случае пересечения не будет.

Алгоритм Сазерленда-Ходжмена может быть представлен следующим обра- зом.

1. Ввод исходных данных: Np - количества вершин отсекаемого многоуголь- ника, P - массива координат вершин отсекаемого многоугольника, Nc - ко- личества вершин отсекателя, C - массива координат вершин отсекателя. Элементами массивов являются записи, каждая из которых содержит два поля - координаты x и y вершины. Для удобства работы алгоритма первая вершина отсекателя заносится в массив C дважды: на первое место и еще раз в конец массива (это сделано потому, что последнее ребро отсекателя образуется последней и первой вершинами многоугольника).
2. Цикл по всем ребрам отсекателя (переменная цикла i изменяется от 1 до Nc.
   1. Обнуление количества вершин результирующего многоугольника Nq.
   2. Цикл по всем ребрам отсекаемого многоугольника (переменная цикла j изменяется от 1 до Nc ).
      1. Анализ номера обрабатываемой вершины многоугольника: если j=1 (первая вершина), то ее координаты запоминаются в переменной F (F=P1). Переход к п. 2.2.5.
      2. Определение факта пересечения ребра многоугольника SPj и ребра отсекателя CjCj+1.
      3. Если пересечение ребер многоугольников установлено, то опре- деление координат точки T пересечения этих ребер, иначе переход к п. 2.2.5.
      4. Увеличение на единицу количества вершин результирующего многоугольника Nq= Nq+1. Занесение в массив координат резуль- тирующего многоугольника координат найденной точки Q(Nq)=T.
      5. Изменение начальной точки ребра многоугольника: присвоение переменной S значения переменной Pj : S = Pj .
      6. Проверка видимости вершины S относительно ребра CjCj+1. Если вершина видима, то занесение ее координат в массив Q: Nq= Nq+1; Q(Nq)=S.
      7. Конец цикла по переменной j (цикл отсечения ребер многоуголь- ника по текущей границе отсекателя).
   3. Проверка ненулевого количества вершин в результирующем массиве: если Nq=0, то переход к п. (многоугольник невидим относительно текущей границы отсекателя, следовательно, он невидим относительно всего отсекателя).
   4. Проверка факта пересечения ребра многоугольника SF с ребром отсе- кателя CjCj+1.
   5. Если пересечение ребер многоугольников установлено, то определение координат точки T пересечения этих ребер, иначе переход к п. 2.
   6. Увеличение на единицу количества вершин результирующего много уголь- ника Nq= Nq+1. Занесение в массив координат результирующего многоуголь- ника координат найденной точки Q(Nq)=T.
   7. Присвоение полученных значений количества вершин и их координат результирующего многоугольника значениям количества вершин и их коорди- нат исходного многоугольника: Np =Nq , P=Q (полученный многоугольник отсе- кается далее следующей стороной отсекателя).
   8. Конец цикла по переменной i (цикл отсечения по всем границам отсека- теля).
   9. Визуализация полученного многоугольника P.
   10. Конец алгоритма.

Вышеописанный алгоритм Сазерленда - Ходжмена требует вычисления и запоминания большого количества вершин промежуточных многоугольников. Для сокращения количества вычислений вместо отсечения каждого ребра мно- гоугольника одной границей, ограничивающей окно отсечения, можно отсекать каждое из ребер многоугольника последовательно всеми границами окна отсе- чения. Тогда, при практической реализации алгоритма, после отсечения очеред- ного ребра многоугольника по одной из границ окна отсечения, программа ре- курсивно обращается сама к себе, чтобы отсечь результат предыдущего обраще- ния по следующей границе окна.

При реализации алгоритма Сазерленда - Ходжмена одна из возникающих проблем - рисование ложных ребер (рис.3.6.22). Ложные ребра - это ребра, ко- торые являются дополнительными, они не являются границами результирую- щих многоугольников. Ложные ребра появляются в том случае, когда в резуль- тате отсечения получается несколько не связанных друг с другом многоуголь- ников. Ложные ребра (они совпадают с ребрами отсекателя или их частями) как раз и соединяют между собой получаемые многоугольники. Ложные ребра лег- ко обнаружить в процессе визуализации: часть ребра отсекателя (или все ребро), рисуемая дважды, является ложной. Во многих приложениях появление таких ложных ребер несущественно; например, при растровой развертке сплошных областей. Однако, в ряде других приложений; например, в некоторых алгорит- мах удаления невидимых поверхностей, устранение таких ребер обязательно.

* + - 1. ОТСЕЧЕНИЕ ПРОИЗВОЛЬНОГО МНОГОУГОЛЬНИКА ОКНОМ ПРОИЗВОЛЬНОЙ ФОРМЫ (Алгоритм Вейлера-Азертона)

Рассмотренный алгоритм Сазерленда - Ходжмена обладает существенным недостатком: он позволяет использовать в качестве отсекателя только выпуклые многоугольники. Однако в ряде приложений, например, при удалении невиди- мых поверхностей, необходимо выполнить отсечение невыпуклыми много- угольниками. Вейлер и Азертон предложили весьма мощный алгоритм, позво- ляющий отсекать невыпуклые многоугольники, содержащие внутренние отвер- стия, многоугольником невыпуклой формы, также содержащим внутренние от- верстия.

Границы многоугольников, получаемых в результате отсечения, совпа- дают либо с границами исходного многоугольника, либо с участками границ отсекателя. Никаких других новых границ не возникает.

Для реализации отсечения следует описать каждую границу отсекаемого многоугольника и отсекателя в виде циклического списка их вершин. При этом внешние границы многоугольников должны обходиться по часовой стрелке, а внутренние границы - против часовой стрелки. Это приводит к тому, что при обходе вершин многоугольника в порядке их следования внутренняя

область будет расположена справа от границы. Таким образом, всегда имеется минимум два списка вершин.

Границы отсекаемого многоугольника и отсекателя могут пересекаться. Алгоритм требует нахождения всех точек пересечения границ много- угольников. Координаты точек пересечения можно найти, решив про- стейшую систему двух уравнений с двумя неизвестными следующего вида:

X1 +(X2-X1)t=X3+(X4-X3)s Y1+(Y2-Y1)t=Y3+(Y4-Y3)s

где (X1,Y1), (X2,Y2) - координаты концов ребра одного многоуголь-

ника;

(X3,Y3), (X4,Y4) - координаты концов ребра другого многоугольника; t, s - параметры, причем 0 t 1, 0 s 1.



Найденные значения параметров t, s должны удовлетворять указанным ус- ловиям, так как только в этом случае пересекаются действительно ребра многоугольников, а не их продолжения.

Найденные точки пересечения необходимо отсортировать на две группы: в одну войдут точки входа, в другую - точки выхода.

Точка пересечения является точкой входа, если в ней ребро отсекаемого многоугольника входит внутрь отсекателя; точка пересечения является точкой выхода, если в ней ребро многоугольника выходит из отсекателя. Общее коли- чество точек пересечения должно быть четным, так как эти точки образуют пары: каждой входной точке соответствует точка выхода. Для распознавания характера точки пересечения можно воспользоваться векторным произве- дением векторов, построенных на ребрах многоугольников. Если произве- дение вектора ребра отсекаемого многоугольника на вектор ребра отсекателя положительно, то их точка пересечения является точкой входа, в противном случае - точкой выхода.

Найденные точки пересечения необходимо добавить в ранее созданные цик- лические списки вершин отсекаемого и отсекающего многоугольников.

Далее можно приступить к собственно отсечению, которое будет сводиться к просмотру созданных списков границ. При этом можно отметить универса- лизм рассматриваемого алгоритма, позволяющего производить как внутрен- нее, так и внешнее отсечение. В первом случае находятся многоугольни- ки (являющиеся частями исходного многоугольника), лежащие внутри отсе- кателя, во втором случае - многоугольники, лежащие за пределами отсека- теля (рис.3.6.23)

Для выполнения внутреннего отсечения выбирается очередная точка из списка входных точек. Далее осуществляется поиск этой точки в списке гра- ниц отсекаемого многоугольника, после чего просматривается список вер- шин этого многоугольника до встречи с точкой пересечения. В этом случае осуществляется переход к списку границ отсекателя, содержащему одноимен- ную точку пересечения. Список вершин отсекателя также просматривает- ся от начала к концу до встречи с точкой пересечения, после чего осущест- вляется возврат к списку вершин отсекаемого многоугольника. Процесс за- вершается в момент нахождения начальной точки пересечения.

Внешнее отсечение выполняется аналогично, но начальная точка выбирает- ся из списка выходных точек пересечения, а просмотр списков вершин отсе- кателя производится в обратном порядке - от конца к началу. Следует заме- тить, что если очередная точка входа (выхода) ранее уже попала в список вершин результирующего многоугольника, то просмотр списков вершин отсе-

каемого многоугольника и отсекателя, начиная с этой точки, новых много- угольников не даст (получится один из уже рассмотренных многоугольников).

Реализуя вышеописанную последовательность действий, получим пра- вильный результат только в том случае, если все границы отсекаемого и отсекающего многоугольников пересекаются. Если же нет пересечения гра- ниц, то необходимо установить соответствие между полученной границей (она будет внешней или внутренней) и границей самого многоугольника или отсека- теля, которая будет являться результирующей внутренней или внешней грани- цей.

Границы многоугольника, лежащие внутри отсекателя, будут являться границами результирующего внутреннего многоугольника. Границы, лежа- щие вне отсекателя, будут являться границами получаемых внешних мно- гоугольников.

Границы отсекателя, попавшие внутрь многоугольника, образуют в нем от- верстия. В итоге границы отсекающего многоугольника должны быть помеще- ны в оба списка принадлежности: внутренний и внешний.

Формально алгоритм Вейлера-Азертона можно записать следующим обра- зом:

* + - * 1. Ввод количества вершин внешней границы отсекаемого многоуголь-

ника и их координат.

* + - * 1. Ввод количества отверстий в отсекаемом многоугольнике, по каждому отверстию - количества вершин и их координат.
        2. Ввод количества вершин внешней границы отсекателя и их координат.
        3. Ввод количества отверстий в отсекателе, по каждому отверстию - ко- личества вершин и их координат.
        4. Формирование кольцевых двунаправленных списков вершин по всем границам отсекаемого многоугольника.
        5. Формирование кольцевых двунаправленных списков вершин по всем границам отсекателя.
        6. Вычисление координат всех точек пересечения отсекаемого мно- гоугольника и отсекателя.
        7. Добавление всех найденных точек пересечения на соответствующие места в списки вершин многоугольников.
        8. Определение типа каждой точки пересечения и формирование двух списков - точек входа и точек выхода.
        9. Определение границ многоугольников, не имеющих пересечений. Границы отсекаемого многоугольника, лежащие вне отсекателя, поместить в список внешней принадлежности, границы, попавшие внутрь отсекателя - в список внутренней принадлежности. Поместить границы отсекателя, по- павшие внутрь отсекаемого многоугольника, в оба списка принадлежно- сти. (Границы отсекателя, лежащие за пределами отсекаемого многоугольни- ка, проигнорировать).
        10. Проведение собственно отсечения. Организация для этого цикла по всем точкам входа.

Нахождение в списке вершин отсекаемого многоугольника очередной точки входа.

Просмотр списка вершин отсекаемого многоугольника до нахожде- ния точки пересечения. Копирование просмотренных вершин в список внут- ренней принадлежности.

Переход к списку вершин отсекателя и поиск в нем одноименной точки пересечения.

Просмотр списка вершин отсекателя до нахождения точки пересе- чения. Копирование просмотренных вершин в список внутренней принадлеж- ности.

Сравнение найденной точки пересечения с первой точкой: если эти точки не совпадают, то переход к п.11.2., иначе к п. 11.6.

Определение необходимости формирования второй границы от- сеченного многоугольника: если не все границы отсекаемого многоугольника имеют пересечение с границами отсекателя, то необходим поиск другой гра- ницы, иначе переход к п.19.

Получение второй границы результирующего многоугольника. В этом случае можно использовать правило: если пересечение имела только внешняя граница отсекаемого многоугольника, то полученный результат явля- ется внешней границей. Внутренняя граница будет совпадать с внутренними границами самого многоугольника и отсекателя (если она лежит внутри отсе- каемого многоугольника), если отверстия отсекаемого многоугольника не лежат внутри отверстий отсекателя. В противном случае внутренняя граница будет совпадать с границей отверстия отсекателя.

Если пересечение имела только внутренняя граница с внешней границей отсекателя, то будет получен нужный результат. Если же пересечение имели внутренние границы многоугольников, то внешняя граница ре- зультирующего многоугольника совпадает с внешней границей исходного многоугольника (если эта граница лежит внутри отсекателя) или с внешней границей отсекателя (отсекатель лежит внутри отсекаемого многоугольника).

Конец цикла (выбор следующей точки входа из списка, если он не пуст и переход к п.11.1., иначе выход из цикла).

* + - * 1. Конец.

Для проведения внешнего отсечения цикл, начинающийся в п.11, следует проводить по всем точкам из списка выходов, а просмотр списков границ отсе- кателя производить в обратном порядке, а в правиле п.11.7. внутренние и внешние границы меняются местами.

Следует отметить, что определение точек пересечения ребер отсекаемого многоугольника и отсекателя требует определенной аккуратности. В частности, надо четко различать собственно пересечение ребер от их касания (рис.3.6.24). Случай, когда вершина или ребро отсекаемого многоугольника касаются ребра отсекателя (вершина отсекаемого многоугольника инцидентна ребру отсекателя или ребро отсекаемого многоугольника совпадает со стороной отсекателя), не является пересечением. При реализации алгоритма следует проверять только такие точки пересечения, которые совпадают с вершинами ребер отсекаемого многоугольника.

Можно предложить следующий подход к определению характера точки пе- ресечения. Следует проверить принадлежность точки, ближайшей к точке пере- сечения и расположенной по направлению вектора стороны отсекаемого много- угольника. Если эта точка принадлежит ребру (а не его продолжению), то ее следует считать точкой пересечения, в противном случае она является точкой касания.