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 качестве стандартного отсекателя рассматривается прямоугольник со сторонами, параллельными осям координат. Произвольный отсекатель - это обычно многоугольник, который может быть как выпуклым, так и невыпуклым, а также может иметь отверстия. Помимо отсекателя должны быть также известны геометрические 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 составляющих - отрезков и многоугольников - может рассматриваться как довольно интересная самостоятельная математическая задача. Однако в рамках машинной графики она приобретает и важное прикладное значение.

Отсечение используется в алгоритмах устранения ступенчатости, при создании трехмерных и реалистических изображений для удаления невидимых 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 выделения тех фрагментов, которые попадают в поле видимости наблюдателя. Здесь рассматриваются алгоритмы отсечения отрезков и многоугольников на плоскости. Данные геометрические объекты являются наиболее распространенными при построении реальных сцен. Отсечение в пространстве либо
сводится к отсечению на плоскости, либо сами плоские алгоритмы без труда распространяются на трехмерный случай. Об этом будет сказано по ходу изложения материала.

3.6.2 ОТСЕЧЕНИЕ ОТРЕЗКОВ РЕГУЛЯРНЫМ ОТСЕКАТЕЛЕМ

Регулярным (стандартным) отсекателем на плоскости является прямоугольник со сторонами, параллельными координатным осям объектного пространства или экрана. Такое окно задается левым, правым, верхним и нижним двумерными ребрами. Для выполнения отсечения необходимо задать абсциссы $X_л$, $X_п$ левого и правого ребер и ординаты $Y_н$,$Y_в$ нижнего и верхнего ребер. Цель отсечения 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.

Поэтому желательно быстро определять такие точки и отрезки. Точки, лежащие целиком внутри окна, удовлетворяют условию $(X_л \le X \le X_п) \wedge (Y_н \le Y \le Y_п)$, где (X,Y) - координаты точки. Считается, что точки, лежащие на границе окна, принадлежат внутренней области окна.

Отрезок целиком лежит внутри окна, если обе его концевые точки лежат внутри окна. Однако обратное утверждение, к сожалению, верно не всегда. Отрезок, концевые точки которого лежат вне окна, может быть как полностью невидимым, 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 концевые точки отрезка невидимы, то он будет заведомо невидимым, если они (вершины отрезка) одновременно лежат левее или правее или ниже или выше окна (рис.3.6.1).

С помощью следующего простого теста можно определить полностью видимые, часть полностью невидимых отрезков, остающиеся отрезки требуют дальнейшего 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 $(X_a < X_л) \vee (X_b > X_л)$ то переход к п.2;
   если $(X_b < X_л) \vee (X_b > X_л)$ то переход к п.2;
   если $(Y_a < Y_н) \vee (Y_a > Y_в)$ то переход к п.2;
   если $(Y_b < Y_н) \vee (Y_b > Y_в)$ то переход к п.2;
   иначе отрезок видим, переход к п.3.
2. Определение полностью невидимого отрезка:
   если $(X_a < X_л) \wedge (X_b < X_л)$ то переход к п.3 (отрезок невидим);
   если $(X_a > X_п) \wedge (X_b > X_п)$ то переход к п.3(отрезок невидим);
   если $(Y_a < Y_н) \wedge (Y_b < Y_в)$ то переход к п.3(отрезок невидим);
   если $(Y_a > Y_в) \wedge (Y_b > Y_в)$ то переход к п.3(отрезок невидим);
   иначе отрезок частично видим или является невидимым (требуется дополнительный анализ.
3. 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 четырехразрядный (битовый) код (рис.3.6.2). Биты этого кода заполняются по следующему правилу

T_1 =1, если точка лежит левее окна, и 0 в противном случае;

T_2 =1, если точка лежит правее окна, и 0 в противном случае;

T_3 =1, если точка лежит ниже окна, и 0 в противном случае;

T_4 =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 невидимых отрезков. Если побитное логическое произведение кодов концов отрезка не равно нулю, то отрезок является полностью невидимым и его можно отбросить. Однако, если логическое произведение равно нулю, то ничего определенного 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 тривиально невидимые отрезки. Отрезки, для которых логическое произведение кодов концов равно нулю, а побитная сумма этих кодов для концевых точек (одной или обеих) отлична от нуля, требуется выполнение собственно алгоритма отсечения с целью нахождения точек пересечения отрезка с границами отсекателя 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 отрезков 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

$Y=m(X-X_1)+Y_1$ или $Y=m(X-X_2)+Y_2$, где (X_1,Y_1) и (X_2,Y_2) - координаты двух точек, через которые проходит отрезок, а $m=(Y_2 - Y_1)/(X_2- X_1)$ - тангенс угла наклона отрезка.

Поскольку для отсекателя заданы абсциссы левой и правой границ, ординаты нижней и верхней границ, то для нахождения координат точек пересечения отрезка с границами отсекателя достаточно подставить в уравнение отрезка абсциссы 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 пересечения с боковыми ребрами необходимо найти только ординату точки пересечения, а абсцисса совпадает с абсциссой ребра отсекателя. При нахождении точки пересечения с нижним или верхним ребрами отсекателя известна ордината ребра отсекателя, ее значение совпадает с ординатой точки пересечения и остается 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 отсекателя находятся из следующих выражений:

$$\begin{aligned} &Y= m(X_л-X_1)+Y_1 \text{ , } m\neq\infty \quad \text{(с левой границей)} \\ &Y= m(X_п-X_1)+Y_1 \text{ , } m\neq\infty \quad \text{(с правой границей)} \\ &X=(1/m)(Y_н-Y_1)+X_1 \text{ , } m\neq0 \quad \text{(с нижней границей)} \\ &X=(1/m)(Y_в-Y_1)+X_1 \text{ , } m\neq0 \quad \text{(с верхней границей).} \end{aligned} \qquad (3.6.1)$$

Указанные ограничения на значение тангенса угла наклона отрезка выполняются, т.к. в алгоритмах предварительно анализируется характер отрезка (вертикальный, горизонтальный или общего положения) и в дальнейшем для вертикальных отрезков не ищутся точки пересечения с боковыми ребрами, а для

горизонтальных отрезков не ищутся точки пересечения с нижним и верхним ребрами отсекателя.

3.6.2.1. ПРОСТОЙ АЛГОРИТМ ОТСЕЧЕНИЯ ОТРЕЗКА

Изложенные соображения положены в основу простого алгоритма отсечения отрезка регулярным отсекателем. На основе сформированных кодов концов отрезка проверяется полная видимость, невидимость отрезка или видимость одной 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 отсекателя. 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 нахождения двух точек пересечения визуализируется полученный результат. Если же пересечение не является корректным, то выполненные вычисления оказываются фактически напрасными и в этом случае ищется пересечение со следующей стороной отсекателя (рис.3.6.3). Если не было найдено ни одного корректного пересечения, то отрезок является невидимым.

1. Ввод координат отсекателя $X_л$, $X_п$, $Y_н$, $Y_в$.
2. Ввод координат концов отрезка $P_1(X_1,Y_1)$, $P_2(X_2,Y_2)$.
3. Вычисление кодов концов отрезка T1, T2.

Вычисление сумм кодов концов отрезка $S_1=\sum_{i=1}^{4} T1_i$, $S_2=\sum_{i=1}^{4} T2_i$.

4. Установка признака видимости отрезка pr =1 (pr =1 - отрезок видимый; pr = -1 - отрезок невидимый).
5. Задание начального значения тангенса угла наклона отрезка $m=10^{30}$ (большое число, вначале предполагается, что отрезок вертикальный).
6. Проверка полной видимости отрезка:

если $(S_1=0)\&(S_2=0)$=истина, то отрезок видимый; выполнение в этом случае следующих действий: занесение в результат координат концов отрезка $R_1=P_1$, $R_2=P_2$ и переход к п. 31.

7. Вычисление логического произведения кодов концов отрезка $PL=\sum_{i=1}^{4} T1_i T2_i$.
8. Проверка тривиальной невидимости отрезка: если PL≠0, то отрезок невидим. В этом случае установка признака pr= -1 и переход к п. 31.
9. Проверка видимости первого конца отрезка:

если $S_1=0$ (первый конец видим), то выполнение следующих действий: $R_1=P_1$ (занесение этой вершины в результат), $Q=P_2$ (занесение координат другой вершины в рабочую переменную Q), i=2 (номер шага отсечения), переход к п.15.

10. Проверка видимости второго конца отрезка:

если $S_2=0$ (второй конец видим), то выполнение следующих действий: $R_1= P_2$ (занесение этой вершины в результат), $Q=P_1$ (занесение координат другой вершины в рабочую переменную Q), i=2 (номер шага отсечения), переход к п.15.

11. Установка начального значения шага отсечения i=0.

12. Вычисление текущего номера шага отсечения i=i+1.

13. Проверка завершения процедуры отсечения: если i>2, то переход к п.31.

14. Занесение в рабочую переменную Q координат i-ой вершины Q=Pi.

15. Определение расположения отрезка: если $X_2=X_1$ (отрезок вертикальный), то переход к п.23 (не может быть пересечения с левой и правой границами отсекателя).

16. Вычисление тангенса угла наклона отрезка $m=(Y_2-Y_1)/(X_2-X_1)$.

17. Проверка возможности пересечения с левой границей отсекателя: если $Q_x>X_л$ (пересечения нет), то переход к п.20.

18. Вычисление ординаты точки пересечения отрезка с левой границей отсекателя: $Y_p=m(X_л - Q_x)+Q_y$.

19. Проверка корректности найденного пересечения: если $(Y_p\geq Y_н)\&(Y_p\leq Y_в)$= истина (пересечение корректное), то выполнение следующих действий: $R_{i.x}=X_л$, $R_{i..y}=Y_p$ (занесение полученных координат в результат), переход к п.12.

20. Проверка возможности пересечения отрезка с правой границей отсекателя: если $Q_x<X_п$ (пересечения нет), то переход к п.23.

21. Вычисление ординаты точки пересечения с правой границей: $Y_p=m(X_п - Q_x)+Q_y$.

22. Проверка корректности найденного пересечения: если $(Y_p\geq Y_н)\&(Y_p\leq Y_в)$= истина (пересечение корректно), то выполнение следующих действий $R_{i.x}=X_п$, (занесение полученных координат в результат), переход к п.12.

23. Проверка горизонтальности отрезка: если m=0, то переход к п.12.

24. Проверка возможности пересечения с верхней границей отсекателя: если $Q_y<Y_в$ (пересечения нет), то переход к п.27.

25. Вычисление абсциссы точки пересечения с верхней границей: $X_p=(Y_в - Q_y)/m+Q_x$.

26. Проверка корректности найденного пересечения: если $(X_p\geq X_л)\&(X_p\leq X_п)$= =истина (пересечение корректно), то выполнение следующих действий: $R_{i.x}=X_p$; $R_{i.y}=Y_в$ (занесение полученных координат в результат); переход к п. 12.

27. Проверка возможности пересечения с нижней границей отсекателя: если $Q_x>Y_н$ (пересечения нет), то переход к п. 30 (вершина невидима и ни одно пересечение не является корректным, следовательно, отрезок невидим).

28. Вычисление абсциссы точки пересечения с нижней границей: $X_p=(Y_н - Q_y)/m+Q_x$.

29. Проверка корректности найденного пересечения: $(X_p\geq X_л)\&(X_p\leq X_п)$= =истина (пересечение корректно), то выполнение следующих действий: $R_{i.x}=X_p$; $R_{i.y}=Y_в$ (занесение полученных координат в результат); переход к п. 12.

30. Установка признака видимости pr= -1 (отрезок невидим полностью, так как ни одно пересечение не оказалось корректным).

31. Проверка признака видимости: если pr=1, то вычерчивание отрезка R_1R_2.

32. Конец.

3.6.2.2.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 вершины 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 попарное логическое произведение кодов концов будет равно нулю.

В алгоритме предусматривается предварительный анализ сумм кодов концов отрезка и попарных логических произведений этих кодов с целью определения тривиально видимых и тривиально невидимых отрезков.

Если отрезок не является ни тривиально видимым, ни тривиально невидимым, 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 меняются 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 тривиально невидимый.

Для удобства работы данные, задающие отсекатель, заносятся в массив "окно" O(4), в котором $O_1=X_л$, $O_2=X_п$, $O_3=Y_н$, $O_4=Y_в$. 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 расположение 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 $X_л$ (O_1), $X_п$ (O_2), $Y_н$ (O_3), $Y_в$ (O_4).
2. Ввод координат концов отрезка $P_1(X_1,Y_1)$, $P_2(X_2,Y_2)$.
3. Установка начального значения флага Fl=0.
4. Проверка вертикальности отрезка: если $P_{2.x}-P_{1.x}=0$ (вертикальный), то Fl= -1, иначе вычислить тангенс угла наклона отрезка $m=(P_{2.y}-P_{1.y})/(P_{2.x}-P_{1.x})$.
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 P_1P_2 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.
9. Проверка видимости обеих вершин отрезка относительно текущей i-ой стороны окна: если $T1i=T2_i$, то переход к п.18.
10. Проверка видимости первой вершины: если $T1_i=0$ (вершина видима), то обмен местами вершин: $R=P_1$; $P_1=P_2$; $P_2=R$.
11. Проверка вертикальности отрезка: если Fl= -1, то переход к п. 14.

12. Анализ номера шага отсечения: если i≥3, то переход к п. 14.

13. Вычисление координат точки пересечения с i-ым ребром отсекателя (левым или правым): $P_{1.y}=m(O_i-P_{1.x})+P_{1.y}$; $P_{1.x}=O_i$. Переход к п. 18.

14. Проверка горизонтальности отрезка: если Fl=1, то переход к п. 18.

15. Проверка вертикальности отрезка: если Fl=-1, то переход к п.17.

16. Вычисление абсциссы точки пересечения отрезка общего положения со стороной отсекателя (верхней или нижней): $P_{1.x}=(O_i-P_{1.y})/m+P_{1.x}$.

17. Присвоение ординате вершины отрезка ординаты стороны отсекателя: $P_{1.y}=O_i$.

18. 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).

19. Вычерчивание отрезка P_1P_2.

20. Конец.

При работе с вертикальными отрезками нет необходимости определять номер шага отсечения. Невидимый вертикальный отрезок относительно бокового ребра отсекателя будет полностью невидимым, это обнаружится в самом 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 представить следующим образом.

1. Вычисление кодов первой вершины отрезка T1.

2. Вычисление кодов второй вершины отрезка T2.

3. Вычисление суммы кодов первой вершины $S_1=\sum_{i=1}^{4} T1_i$,

4. Вычисление суммы кодов второй вершины $S_2=\sum_{i=1}^{4} T2_i$

5. Определение полной видимости отрезка: если $(S_1=0)\&(S_2=0)$=истина, то pr =1 и переход к п. 8.

6. Вычисление попарного логического произведения кодов концов отрезка: $p=\sum_{i=1}^{4} T1_i\, T2_i$.

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

8. Конец.

Логическую сумму можно вычислить или как арифметическую или как логическую сумму.

3.6.2.3. 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 отсутствует, точка пересечения определяется с использованием двоичного поиска.

Рассматриваемый алгоритм является частным случаем алгоритма отсечения Сазерленда-Коэна. Он был разработан Спруллом и Сазерлендом для аппаратной 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 объектов синтезируемой сцены и проверок, определяющих полную видимость отрезков (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 равные 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 точки. Таким образом, процесс определения пересечения сводится к двоичному поиску, 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 точкой S. Однако это разбиение приводит к одинаковым результатам для обеих половин. Далее разобьём отрезок SK средней точкой R. Теперь отрезок SR полностью видим, а отрезок RK видим частично. Запомним точку R как текущую видимую 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 окна 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 пересечение с заранее заданной точностью.

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

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

-проверка на полную невидимость отрезка, в положительном случае отрезок 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 исследуемого отрезка средней точкой. Применение вышеизложенных проверок и оценок к двум полученным частям отрезка. Если часть отрезка является полностью невидимой, то средняя точка даёт верхнюю оценку для наиболее удаленной 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 проверяется на видимость и анализ завершается.

Более подробно алгоритм разбиения средней точкой можно представить следующим образом:

1. Ввод координат отсекателя $X_л$, $X_п$, $Y_н$, $Y_в$.
2. Ввод координат концов отрезка $P_1(X_1,Y_1)$, $P_2(X_2,Y_2)$.
3. Ввод точности ε вычисления точки пересечения отрезка с границей отсекателя.
4. Установка номера шага отсечения i=1.
5. Вычисление кодов концевых точек и запись их в соответствующие массивы T1 и T2 размерностью 1x 4, вычисление сумм кодов концов $S_1=\sum_{i=1}^{4} T1_i$, $S_2=\sum_{i=1}^{4} T2_i$
6. Проверка полной видимости отрезка. Если коды обоих концов отрезка равны нулю (полная видимость отрезка), то переход к п. 9.
7. Проверка полной невидимости отрезка. Вычисление побитного логического произведения кодов концевых точек отрезка. Если произведение отлично от нуля (отрезок невидим), то переход к п. 10.
8. Анализ частично видимого отрезка в том случае, если побитовое логическое произведение кодов его концов равно нулю:

8.1. Поиск наиболее удаленной от P_1 видимой точки S исследуемого отрезка. Запоминание исходной точки P_1 в промежуточной переменной R.

8.2. Проверка на окончание процесса решения: если i>2, то определение логического произведения pr кодов концов отрезка. Если pr≠0, то переход к п.10, иначе переход к п.9.

8.3. Проверка точки P_2 на наиболее удаленную от P_1 видимую точку отрезка. Если сумма всех элементов массива T2 равна нулю (S_2), то переход к пункту 8.12.

8.4. Проверка нахождения точки пересечения отрезка с границами отсекателя. Если $|P_1 - P_2| \le \varepsilon$ (расстояние между концевыми точками исследуемого отрезка меньше допустимой погрешности), то переход к пункту 8.12.

8.5. Вычисление средней точки $P_{ср.}$ отрезка: $P_{ср.} = (P_1 + P_2)/2$ ($P_{ср.x} = (P_{1.x} + P_{2..x})/2$; $P_{ср.y} = (P_{1.y} + P_{2.y})/2$).

8.6. Запоминание текущей точки P_1: Pm=P1.

8.7. Замена точки P_1 на среднюю точку: $P_1 = P_{ср}$.

8.8. Вычисление нового кода T1 точки P_1.

8.9. Вычисление произведения pr кодов концов нового отрезка P_1P_2.

8.10.Проверка полной невидимости отрезка P_1P_2. Если побитовое логическое произведение pr кодов концевых точек равно нулю, то переход к пункту 8.4. В противном случае отрезок P_1P_2 невидим.

8.11. Возврат к предыдущему отрезку P_1P_2 : $P_1 = Pm$, $= P_{ср}$, переход к пункту 8.4. (Вычислена наиболее удаленная от точки P_1 видимая точка отрезка).

8.12..Поиск наиболее удаленной от P_2 видимой точки отрезка. Замена мест точек P_1 и P_2 : $P_1 = P_2$, P_2=R. Увеличение шага выполнения отсечения i=i+1. Переход к п.5.

9. Визуализация отрезка.
10.Конец.

При программной реализации алгоритма целесообразно для вычисления кодов концевых точек отрезков и их логических произведений использовать отдельные функции.

3.6.3. 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 сетки или не является прямоугольным. Поэтому Кирус и Бек предложили алгоритм отсечения окном произвольной выпуклой формы.

В этом алгоритме для определения местоположения точки относительно окна отсечения используется вектор нормали и параметрическая форма задания отрезка. Параметрическое уравнение отрезка P_1P_2 имеет вид:

$P(t)= P_1 + (P_2- P_1)t; \quad 0\le t \le 1$, где t - параметр.

Фактически это уравнение является векторным, оно сводится к двум одномерным параметрическим уравнениям следующего вида:

$$P_x(t)= P_{1.x} + (P_{2.x}- P_{1.x})t; \quad 0\le t \le 1$$
$$P_y(t)= P_{1.y} + (P_{2.y}- P_{1.y})t; \quad 0\le t \le 1 \qquad (3.6.2)$$

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):

$$t = (P(t)-P_1)/(P_2 - P_1)$$

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

$t = \frac{X_л - P_{1.x}}{P_{2.x} - P_{1.x}}$ $0\le t \le 1$ (для левой границы)

$t = \frac{X_п - P_{1.x}}{P_{2.x} - P_{1.x}}$ $0\le t \le 1$ (для правой границы)

$t = \frac{Y_н - P_{1.y}}{P_{2.y} - P_{1.y}}$ $0\le t \le 1$ (для нижней границы)

$t = \frac{Y_в - P_{1.y}}{P_{2.y} - P_{1.y}}$ $0\le t \le 1$ (для верхней границы)

Если в результате вычислений получаются значения параметра t, выходящие за пределы интервала $0\le t \le 1$, то такие решения отвергаются, так как они соответствуют точкам, лежащим вне рассматриваемого отрезка. Естественной является 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 невидимого 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 видимые 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 $n_в$, в произвольной точке A, лежащей на любой границе C, удовлетворяет следующему условию: $n_в (B - A) \geq 0$, где B - любая другая точка на границе отсекающей области. Правильность условия иллюстрируется на рис. 3.6.8. Угол α между внутренней нормалью $n_в$ к точке A и вектором, проведенным из данной точки к любой другой, расположенной на границе отсекающей области, всегда находится внутри интервала: $-\pi/2 \leq \alpha \leq \pi/2$. Таким образом, косинус угла α всегда положителен, а, следовательно, положительно и скалярное произведение этих векторов. В свою очередь, угол между внешней нормалью $n_{вш}$ и любым из подобных векторов равен $(\pi-\alpha)$, косинус которого всегда отрицателен.

Выше уже говорилось, что для анализа частично невидимых отрезков необходимо определять точки пересечения исследуемых отрезков с границами окна отсечения. Для этого представим исследуемый отрезок в параметрическом виде:

$$P(t) = P_1 + (P_2 - P_1)\, t , \qquad (3.6.3)$$

где t - параметр; при наложении ограничений на t: $0 \leq t \leq 1$, получим именно отрезок, а не бесконечную прямую. Далее предположим, что f - граничная точка выпуклой области C, а $n_в$ - внутренняя нормаль к одному из ограничивающих эту область ребер. Тогда для любой точки отрезка AB, то есть для любого значения параметра t:

* из условия $n_в[P(t) - f] < 0$ следует, что вектор[P(t) - f] направлен из области C;
* из условия $n_в[P(t) - f] = 0$ следует, что вектор [P(t) - f] принадлежит плоскости, которая проходит через граничную точку k и перпендикулярна нормали $n_в$;
* из условия $n_в [P(t) - f] > 0$ следует, что вектор [P(t) - f] направлен внутрь C (рис.3.6.9).

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

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.

$$n_{вi} [P(t) - f_i] , \qquad (3.6.4)$$

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

будет положительно, равно нулю или отрицательно, в зависимости от того, будет 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 пересечения отрезка с ребром полигонального выпуклого окна отсечения:

$$n_{вi}\ [\ P_1 + (P_2-P_1)\ t - f_i] = 0, \qquad (3.6.5)$$

которое приведем к виду:

$$n_{вi}\ [P_1 - f_i] + n_{вi}\ [P_2-P_1]\ t\ = 0. \qquad (3.6.6)$$

В уравнении (3.6.6) вектор $[P_2-P_1]$ определяет ориентацию отрезка, а вектор $[P_1 - f_i]$ пропорционален расстоянию от начала отрезка до исследуемой граничной точки. Обозначим вектор ориентации как $D = P_2-P_1$, а для второго вектора введем некоторый коэффициент $Wi= P_1- f_i$. С учетом последних обозначений уравнение (3.6.6) можно записать в следующем виде:

$$t\ (\ n_{вi}\ D\ ) + Wi\ \ n_{вi}\ = 0 \qquad (3.6.7)$$

Решая уравнение (2.6.7) относительно t, получим:

$$t = -\ (\ Wi\ \ n_{вi})\ /(\ D\ n_{вi})\ ,\quad D \neq 0,\quad i=1,2..m\ . \qquad (3.6.8)$$

Анализируя уравнение (3.6.8), замечаем, что знаменатель $D\ n_{вi}$ 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 числитель $Wi\ n_{вi} < 0$, на границе, когда он равен нулю и внутри, когда числитель $Wi\ n_{вi} > 0$. Однако этот случай не представляет большого интереса, т.к. нас интересуют отрезки, а не точки. Во-вторых, знаменатель равен нулю, когда равно нулю скалярное произведение $D\ n_{вi,}$ 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 отсечения. Если значения t не принадлежат интервалу $0 \le t \le 1$, то их не рассматривают, 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 $0 \le t \le 1$. Полученные решения следует разбить на две группы: верхнюю и нижнюю, в зависимости от близости найденной точки пересечения к началу или концу отрезка. Чтобы отрезок был видимым относительно всего отсекателя, он должен быть видим относительно всех ребер отсекателя одновременно. Концам видимой части отрезка будут соответствовать два значения параметра t, одно из которых является максимальным значением из нижней группы t_{maxmin}, а второе - минимальным из верхней группы t_{minmax} (рис. 3.6.10). Найденное значение параметра t для очередной точки пересечения рассматривают в качестве возможного верхнего предела $t_в$, если знаменатель $Dn_{вi} < 0$; в случае же, когда знаменатель положителен, значение параметра t определяет точку, которую относят к нижней границе видимости $t_н$.

Алгоритм Кируса - Бека может быть представлен следующим образом:

1. Ввод координат концевых точек отрезка $P_1(P_{1.x}, P_{1.y})$, $P_2(P_{2.x}, P_{2.y})$
2. Ввод числа сторон m выпуклого многоугольника - окна отсечения и координат 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 $t_н = 0$, $t_в = 1$.
5. Начало цикла по всем сторонам отсекающего окна (i = 1..m). Выполнение для каждой i-ой стороны окна следующих действий:

5.1. Вычисление вектора внутренней нормали к очередной i-ой стороне окна отсечения - $n_{вi}$

5.2. Определение граничной точки f_i каждой стороны отсекающего окна (точки, лежащей на i-ой стороне окна).

5.3. Вычисление вектора $Wi = P_1 - f_i$.

5.4 Вычисление скалярного произведения векторов $W_{ск\,i} = Wi\ n_{вi}$.

5.5 .Вычисление скалярного произведения векторов $D_{ск\,i} = Dn_{вi}$.

5.6. Проверка на равенство нулю скалярного произведения $D_{ск}$ (вырождение отрезка в точку или его параллельность стороне отсекателя). Если $Dn_{вi} = 0$, тогда переход к пункту 5.9 .

5.7. Вычисление параметра t: $t = -(W_{скi} / D_{скi})$ (отрезок не вырожден в точку и не параллелен стороне отсекателя).

5.8. Определение верхнего и нижнего пределов параметра t :

5.8.1. Поиск нижней границы параметра t, если $D_{скi} > 0$: 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 P_2.; если $t \le 1$, то $t_н = \max (t, t_н)$ (выбор максимального значения из текущего значения параметра t и ранее вычисленного значения нижней границы параметра).

5.8.2. Поиск верхней границы параметра t, если $D_{скi} < 0$: 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 P_1.; если $t \ge 0$, то $t_в = \min (t, t_в)$ (выбор минимального значения из текущего значения параметра t и ранее вычисленного значения верхней границы параметра).

5.9. Проверка видимости точки, в которую выродился отрезок, или проверка видимости произвольной точки отрезка в случае его параллельности стороне отсекателя: если $W_{скi} < 0$, то отрезок (точка) невидим и переход к п. 7; если $W_{скi} > 0$, то отрезок (точка) видим относительно текущей стороне отсекателя и переход к п. 5.10.

5.10. Конец цикла по сторонам отсекателя.

6. Проверка фактической видимости отсеченного отрезка. Если $t_н > t_в$, то переход к пункту 7 , иначе визуализация отрезка в интервале от $P(t_н)$ до $P(t_в)$.

7. Конец алгоритма.

При программной реализации целесообразно скалярные произведения векторов вычислять с помощью отдельной функции.

В заключение, рассмотрим пример отсечения отрезка выпуклым шестиугольником. На рис. 3.6.11 показан шестиугольник P1-P6 , служащий окном отсечения, 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, соответствующую ребру P5P6. Вектор ориентации D = B - A = [3 3] - [-1 1] = [4 2]. Внутренняя нормаль $\mathbf{n_в}$ к ребру P5P6 равна $\mathbf{n_в}$ = [-1 0], тогда D $\mathbf{n_в}$ = -4. Для граничной точки k (2,3) коэффициент W = A - k = [-1 1] - [2 3] = [-3 -2], а W $\mathbf{n_в}$ = 3. Так как D $\mathbf{n_в}$ = -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 интервале: $1/4 \le t \le 3/4$.

Таблица 3.1

Ребро	Нормаль $\mathbf{n_в}$	Граничная точка k	Коэффициент W	W $\mathbf{n_в}$	D $\mathbf{n_в}$	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 областью окна отсечения. Аналогично отсекается отрезок и внешней областью отсекающего окна. Для этого определяются, а затем вычерчиваются части отрезка, лежащие вне окна отсечения.

3.6.4.ОПРЕДЕЛЕНИЕ ФАКТА ВЫПУКЛОСТИ МНОГОУГОЛЬНИКА. ВЫЧИСЛЕНИЕ ВНУТРЕННИХ НОРМАЛЕЙ

Алгоритм Кируса - Бека позволяет производить отсечение отрезков произвольным выпуклым окном, поэтому необходимо уметь устанавливать факт выпуклости 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 рассматриваемый многоугольник. Будем считать, что это произведение имеет положительный 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 отрицательные знаки, то многоугольник невыпуклый. Если все знаки векторных произведений смежных сторон неотрицательные, то отсекающий многоугольник выпуклый, а внутренние нормали ориентированы влево от его контура. Если 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 произведений. 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 вершинах многоугольника. Если среди знаков этих произведений будут как положительные, так и отрицательные, то многоугольник - невыпуклый. Если все знаки произведений одинаковые, то сразу сделать вывод о характере многоугольника нельзя, а следует в качестве базовой выбрать следующую вершину многоугольника и повторить вычисления. Если для всех вершин многоугольника 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 дальнейшие 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 A_x ,A_y -компоненты вектора A стороны многоугольника (считаются известными), n_x, n_y - искомые компоненты вектора нормали. Тогда можно записать следующее равенство:

$$nA=(n_x i +n_y j)(A_x i +A_y j)=(n_x A_x + n_y A_y)=0$$

$$n_x A_x = - n_y A_y$$

Положив n_y равным 1, получаем $n= -A_y/ A_x\ i + j$ ($A_x \neq 0$, т.е. сторона многоугольника не параллельна оси ординат). Если сторона многоугольника параллельна оси ординат, т.е. $A_x=0$, $A_y \neq 0$, то вектор нормали параллелен оси абсцисс и следует положить $n_x=1$, $n_y=0$.

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 смежных его вершин A_{i-1} и A_i и если скалярное произведение нормали n и вектора от A_{i-1} до A_{i+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 выпуклости многоугольника с одновременным определением вектора внутренней нормали и разбиения исследуемого многоугольника на выпуклые многоугольники в случае его невыпуклости. Этот способ основан на использовании переносов и поворотов исследуемого многоугольника. Предполагается, что вершины многоугольника пронумерованы в направлении против часовой стрелки. Рассматриваемый 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 таким образом, чтобы следующая (i+1)- ая вершина оказалась на положительной полуоси абсцисс x.
3. Определить знак ординаты y всех (i+2)-ых вершин. Если знаки ординат y всех (i+2)-ых вершин неотрицательные, то многоугольник выпуклый относительно очередной стороны, соединяющей i-ую и (i+1)-ую вершины.

   В повернутой системе координат внутренняя нормаль к ребру, соединяющему i-ую и (i+1)-ую вершины, имеет нулевую абсциссу и ординату, знак которой совпадает со знаком ординаты (i+2)-ой вершины. Для определения ориентации внутренней нормали исходного многоугольника следует 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 вырожден, 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 многоугольника, пересекающие ось абсцисс, и для них находится ближайшая к началу координат точка пересечения с этой осью, абсцисса которой больше значения x_{i+1} ($x > x_{i+1}$). Далее формируются два многоугольника: первый многоугольник образован вершинами, начиная с (i+1)-ой и кончая найденной точкой пересечения (многоугольник лежит целиком ниже оси абсцисс), второй многоугольник образован всеми вершинами исходного многоугольника, 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, получаемые в процессе разбиения, не станут выпуклыми.

3.6.5.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 отсечением. 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, расположенное в окнах с меньшим приоритетом, отсекается (внутренним образом) самим окном и отсекается внешним образом окнами, имеющими более высокий приоритет.

При внутреннем отсечении получаются значения параметра t ($t_н$, $t_в$), определяющие видимую часть отрезка (лежащую в пределах окна). Если же взять отрезки, для которых значения этого параметра лежат в интервалах (0, $t_н$) и ($t_в$, 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 достигается путем соединения новым ребром вершин, соседних с той вершиной многоугольника, в которой нарушается выпуклость многоугольника). Затем выполняется внутреннее отсечение новым (дополненным) многоугольником. На следующем шаге отсеченный отрезок подвергается внешнему отсечению по границам дополняющего многоугольника. В итоге получаем отрезок, отсеченный невыпуклым окном.

3.6.6.ОТСЕЧЕНИЕ МНОГОУГОЛЬНИКОВ

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

3.6.6.1. АЛГОРИТМ ОТСЕЧЕНИЯ МНОГОУГОЛЬНИКА ВЫПУКЛЫМ ОКНОМ (алгоритм Сазерленда-Ходжмена)

Алгоритм Сазерленда-Ходжмена позволяет провести отсечение произвольного (выпуклого или невыпуклого) многоугольника по границам выпуклого отсекателя. Идея алгоритма достаточно проста. На каждом шаге отсечения исходный 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 P_1, P_2, . . ., P_N(две последовательные вершины являются концами одного ребра, например, P_1P_2, P_2P_3, ... PnP1.). Отсекатель также задается количеством вершин M и их координатами C_1, C_2, . .., C_m..

Исходный многоугольник отсекается сначала, например, левой стороной отсекателя (выберем направление по часовой стрелке), в результате чего получается промежуточный многоугольник (рис.3.6.19). Этот многоугольник отсекается далее верхней границей отсекателя, получается второй промежуточный многоугольник. Далее процесс продолжается аналогичным образом пока не будет выполнено отсечение последней границей отсекателя.

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

Будем рассматривать каждую точку P_i (кроме первой P_1) из списка вершин отсекаемого многоугольника как конечную точку ребра, а начальной точкой S этого ребра является вершина P_{i-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 является полностью видимым;
- ребро целиком лежит по невидимую сторону ребра отсекателя и оно является полностью невидимым;
- ребро видно частично, причем точка S невидима, а точка P_i видима, т.е. ребро входит в область видимости (внутрь отсекателя);
- ребро видно частично, причем точка S видима, а точка P_i 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 точек. Если исследуемое ребро полностью невидимо (второй случай), то в результирующий список заносить ничего не надо. Если ребро видимо полностью (первый 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 присутствовать в результирующем списке. Если ребро многоугольника входит внутрь отсекателя (третий случай), то необходимо найти точку пересечения ребер многоугольника и отсекателя и занести ее в результирующий список, конечная точка P_i в этом случае также видима и она тоже подлежит занесению в результирующий список.

Особым образом обрабатывается первая точка многоугольника: для нее требуется определить только видимость. Если точка видима, то она заносится в результирующий список и становится начальной точкой первого ребра. Если же она невидима, то она просто становится начальной точкой ребра и в результирующий 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 слева) границы отсекателя, по которую лежит эта точка. Если граница отсекателя обходится по часовой стрелке, то его внутренняя область лежит по правую сторону от границы. При противоположном направлении обхода она лежит по левую сторону.

Ранее были уже рассмотрены способы решения стоящей задачи. Так в алгоритме Кируса-Бека использовалось скалярное произведения вектора внутренней нормали на вектор, начало которого лежит в произвольной точке прямой (на границе отсекателя), а конец - в исследуемой точке. Второй способ заключается в подстановке координат исследуемой точки в пробную функцию, роль которой играет уравнение прямой. Авторы рассматриваемого алгоритма предложили третий способ (рис.3.6.21), основанный на вычислении векторного произведения векторов. Если две точки A_1A_2 лежат на ребре отсекателя, а A_3 является пробной точкой, то следует вычислить векторное произведение векторов A_1A_3*A_1A_2. 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 $Z=(X_3-X_1)(Y_2-Y_1)-(X_2-X_1)(Y_3-Y_1)$. Анализируя знак этой компоненты, можно сделать вывод о взаимном положении точки A_3 и отрезка A_1A_2: 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 полностью невидимым, если оба конца невидимы. Если же один конец видим, а другой 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 использования стандартного отсекателя прямоугольной формы со сторонами, параллельными координатным осям можно использовать и другие алгоритмы - простой, Сазерленда-Коэна.

Воспользуемся здесь, как и в алгоритме Кируса-Бека, параметрическим заданием отрезков. Пусть P_1P_2 - концевые точки ребра отсекаемого

многоугольника, а C_1, C_2 - концевые точки ребра отсекателя. В параметрической форме эти отрезки будут описаны следующим образом:

$$P(t)= P_1 + (P_2 - P_1)t, \quad 0\le t \le 1$$

$$C(s)=C_1+ (C_2 - C_1)s, \quad 0\le s \le 1$$

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 пересекаются не сами отрезки, а прямые, проходящие через них (продолжения отрезков). В рассматриваемом алгоритме необходимо определять точки пересечения ребра отсекаемого многоугольника с прямой, проходящей через ребро отсекателя. Поэтому ограничение, накладываемое на параметр s, учитывать не следует. Чтобы не производить лишних вычислений, в алгоритме предварительно устанавливается факт пересечения ребер отсекаемого многоугольника и отсекателя. Для этого используются результаты проверки концов ребра отсекаемого многоугольника на видимость. Если концевые точки отрезка имеют разную видимость (одна вершина видима, а другая невидима), то пересечение есть, в противном случае пересечения не будет.

Алгоритм Сазерленда-Ходжмена может быть представлен следующим образом.

1. Ввод исходных данных: N_p - количества вершин отсекаемого многоугольника, P - массива координат вершин отсекаемого многоугольника, N_c - количества 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 N_c.
   2.1. Обнуление количества вершин результирующего многоугольника N_q.
   2.2. Цикл по всем ребрам отсекаемого многоугольника (переменная цикла j изменяется от 1 до N_c).
   2.2.1. Анализ номера обрабатываемой вершины многоугольника: если j=1 (первая вершина), то ее координаты запоминаются в переменной F ($F=P_1$). Переход к п. 2.2.5.
      2.2.2. Определение факта пересечения ребра многоугольника SP_j и ребра отсекателя C_jC_{j+1}.
      2.2.3. Если пересечение ребер многоугольников установлено, то определение координат точки T пересечения этих ребер, иначе переход к п. 2.2.5.
      2.2.4. Увеличение на единицу количества вершин результирующего многоугольника Nq= Nq+1. Занесение в массив координат результирующего многоугольника координат найденной точки Q(Nq)=T.
      2.2.5. Изменение начальной точки ребра многоугольника: присвоение переменной S значения переменной P_j : $S = P_j$.

2.2.6. Проверка видимости вершины S относительно ребра C_jC_{j+1}. Если вершина видима, то занесение ее координат в массив Q: Nq= Nq+1; Q(Nq)=S.

2.2.7. Конец цикла по переменной j (цикл отсечения ребер многоугольника по текущей границе отсекателя).

2.3. 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).

2.4. Проверка факта пересечения ребра многоугольника SF с ребром отсекателя C_jC_{j+1}.

2.5.Если пересечение ребер многоугольников установлено, то определение координат точки T пересечения этих ребер, иначе переход к п. 2.

2.6.Увеличение на единицу количества вершин результирующего много угольника Nq= Nq+1. Занесение в массив координат результирующего многоугольника координат найденной точки Q(Nq)=T.

2.7. Присвоение полученных значений количества вершин и их координат результирующего многоугольника значениям количества вершин и их координат исходного многоугольника: $N_p = N_q$, P=Q (полученный многоугольник отсекается далее следующей стороной отсекателя).

2.8.Конец цикла по переменной i (цикл отсечения по всем границам отсекателя).

2.9.Визуализация полученного многоугольника P.

2.10. Конец алгоритма.

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

При реализации алгоритма Сазерленда - Ходжмена одна из возникающих проблем - рисование ложных ребер (рис.3.6.22). Ложные ребра - это ребра, которые являются дополнительными, они не являются границами результирующих многоугольников. Ложные ребра появляются в том случае, когда в результате отсечения получается несколько не связанных друг с другом многоугольников. Ложные ребра (они совпадают с ребрами отсекателя или их частями) как раз и соединяют между собой получаемые многоугольники. Ложные ребра легко 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 алгоритмах удаления невидимых поверхностей, устранение таких ребер обязательно.

3.6.6.2. 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 невидимых поверхностей, необходимо выполнить отсечение невыпуклыми многоугольниками. Вейлер и Азертон предложили весьма мощный алгоритм, позволяющий отсекать невыпуклые многоугольники, содержащие внутренние отверстия, многоугольником невыпуклой формы, также содержащим внутренние отверстия.

Границы многоугольников, получаемых в результате отсечения, совпадают 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 многоугольников. Координаты точек пересечения можно найти, решив простейшую систему двух уравнений с двумя неизвестными следующего вида:

$X_1 +(X_2-X_1)t=X_3+(X_4-X_3)s$

$Y_1+(Y_2-Y_1)t=Y_3+(Y_4-Y_3)s$

где (X_1,Y_1), (X_2,Y_2) - координаты концов ребра одного многоугольника;

(X_3,Y_3), (X_4,Y_4) - координаты концов ребра другого многоугольника;

t, s - параметры, причем $0\leq t \leq 1$, $0\leq s \leq 1$.

Найденные значения параметров t, s должны удовлетворять указанным условиям, 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 количество 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 произведением векторов, построенных на ребрах многоугольников. Если произведение 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 циклические списки вершин отсекаемого и отсекающего многоугольников.

Далее можно приступить к собственно отсечению, которое будет сводиться к просмотру созданных списков границ. При этом можно отметить универсализм рассматриваемого алгоритма, позволяющего производить как

внутреннее, так и внешнее отсечение. В первом случае находятся многоугольники (являющиеся частями исходного многоугольника), лежащие внутри отсекателя, во втором случае - многоугольники, лежащие за пределами отсекателя (рис.3.6.23)

Для выполнения внутреннего отсечения выбирается очередная точка из списка входных точек. Далее осуществляется поиск этой точки в списке границ отсекаемого многоугольника, после чего просматривается список вершин этого многоугольника до встречи с точкой пересечения. В этом случае осуществляется переход к списку границ отсекателя, содержащему одноименную точку пересечения. Список вершин отсекателя также просматривается от начала к концу до встречи с точкой пересечения, после чего осуществляется возврат к списку вершин отсекаемого многоугольника. Процесс завершается в момент нахождения начальной точки пересечения.

Внешнее отсечение выполняется аналогично, но начальная точка выбирается из списка выходных точек пересечения, а просмотр списков вершин отсекателя производится в обратном порядке - от конца к началу. Следует заметить, что если очередная точка входа (выхода) ранее уже попала в список вершин результирующего многоугольника, то просмотр списков вершин отсекаемого многоугольника и отсекателя, начиная с этой точки, новых многоугольников не даст (получится один из уже рассмотренных многоугольников).

Реализуя вышеописанную последовательность действий, получим правильный результат только в том случае, если все границы отсекаемого и отсекающего многоугольников пересекаются. Если же нет пересечения границ, то необходимо установить соответствие между полученной границей (она будет внешней или внутренней) и границей самого многоугольника или отсекателя, которая будет являться результирующей внутренней или внешней границей.

Границы многоугольника, лежащие внутри отсекателя, будут являться границами результирующего внутреннего многоугольника. Границы, лежащие вне отсекателя, будут являться границами получаемых внешних многоугольников.

Границы отсекателя, попавшие внутрь многоугольника, образуют в нем отверстия. В итоге границы отсекающего многоугольника должны быть помещены в оба списка принадлежности: внутренний и внешний.

Формально алгоритм Вейлера-Азертона можно записать следующим образом:

1. Ввод количества вершин внешней границы отсекаемого многоугольника и их координат.
2. Ввод количества отверстий в отсекаемом многоугольнике, по каждому отверстию - количества вершин и их координат.
3. Ввод количества вершин внешней границы отсекателя и их координат.
4. Ввод количества отверстий в отсекателе, по каждому отверстию - количества вершин и их координат.
5. Формирование кольцевых двунаправленных списков вершин по всем границам отсекаемого многоугольника.
6. Формирование кольцевых двунаправленных списков вершин по всем границам отсекателя.

7. Вычисление координат всех точек пересечения отсекаемого многоугольника и отсекателя.

8. Добавление всех найденных точек пересечения на соответствующие места в списки вершин многоугольников.

9. Определение типа каждой точки пересечения и формирование двух списков - точек входа и точек выхода.

10. 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, попавшие внутрь отсекаемого многоугольника, в оба списка принадлежности. (Границы отсекателя, лежащие за пределами отсекаемого многоугольника, проигнорировать).

11. Проведение собственно отсечения. Организация для этого цикла по всем точкам входа.

11.1.Нахождение в списке вершин отсекаемого многоугольника очередной точки входа.

11.2. Просмотр списка вершин отсекаемого многоугольника до нахождения точки пересечения. Копирование просмотренных вершин в список внутренней принадлежности.

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

11.4. Просмотр списка вершин отсекателя до нахождения точки пересечения. Копирование просмотренных вершин в список внутренней принадлежности.

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

11.6. Определение необходимости формирования второй границы отсеченного многоугольника: если не все границы отсекаемого многоугольника имеют пересечение с границами отсекателя, то необходим поиск другой границы, иначе переход к п.19.

11.7. Получение второй границы результирующего многоугольника. В этом случае можно использовать правило: если пересечение имела только внешняя граница отсекаемого многоугольника, то полученный результат является внешней границей. Внутренняя граница будет совпадать с внутренними границами самого многоугольника и отсекателя (если она лежит внутри отсекаемого 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 результирующего многоугольника совпадает с внешней границей исходного многоугольника (если эта граница лежит внутри отсекателя) или с внешней границей отсекателя (отсекатель лежит внутри отсекаемого многоугольника).

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

12. Конец.

Для проведения внешнего отсечения цикл, начинающийся в п.11, следует проводить по всем точкам из списка выходов, а просмотр списков границ отсекателя 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 пересечения. Следует проверить принадлежность точки, ближайшей к точке пересечения и расположенной по направлению вектора стороны отсекаемого многоугольника. Если эта точка принадлежит ребру (а не его продолжению), то ее следует считать точкой пересечения, в противном случае она является точкой касания.