УДАЛЕНИЕ НЕВИДИМЫХ ЛИНИЙ И ПОВЕРХНОСТЕЙ

Для правильного восприятия человеком формы объемных тел при их формировании с помощью средств КГр необходимо обеспечивать удаление невидимых линий, ребер, поверхностей и части тел, затеняемых другими графическими объектами. Сложность задачи удаления определяется многообразием обрабатываемых графических объектов и их взаимным расположением породило множество алгоритмов ее решения. Лучшего универсального алгоритма удаления линий и поверхностей не существует. Такая работа зависит от вида объектов, от детализации элементов изображения и требуемой скорости формирования результирующего изображения. Эти противоречивые требования приводят к появлению новых подходов при решении ставящихся задач. Однако можно выделить ряд особенностей, присущих многим из алгоритмов. Так многие используют сортировку поверхностей, часть же выявляют, что видно в окне или в определенной точке экрана и т.д. Теоретически трудоемкость алгоритмов определяется обычно либо квадратом участвующих в сцене объектов (в объектном пространстве) – n^2 или произведением числа объектов сцены n и размером пространства изображения (экрана) N, выраженном в пикселах, – nN.

Таким образом, все алгоритмы можно разделить на 2 группы:

1) решают задачу видимости в пространстве объектов, т.е. как расположены объекты друг относительно друга в пространстве;
2) рассматривают в плоскости визуализации (что видно, а что не видно на экране).

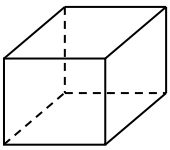
Как правило, 1-я труппа используется:

- когда объектов не очень много, т.к. это можно сделать быстро (сопоставить объекты друг с другом).

2-я группа – когда сцена сложная (много объектов), и тогда время обработки пропорционально количеству пикселов, участвующих в представлении объекта на экране.

В курсе рассмотрим некоторые основные алгоритмы, используемые в КГр. Однако надо понимать, что алгоритмы, используемые при решении реальных задач, могут представлять собой не всегда представленные алгоритмы в “чистом” виде, а являются совокупностью наиболее эффективных частей различных алгоритмов, используемых разработчиками при решении поставленной задачи. Т.е. базовые алгоритмы могут комбинироваться при решении конкретных задач..

Алгоритм удаления невидимых линий, используемый при изображении отдельных выпуклых тел.

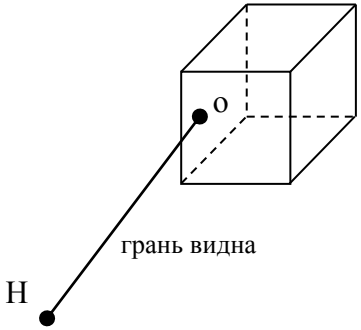


Рассмотрим алгоритм определения видимости простого выпуклого тела:

Он основывается на том, что:

- если грань видна – то и все ребра такой грани видны:

- если грань не видна – то и все ребра будут не видны



Если найти координаты внутренней точки выпуклого тела “О” и провести через нее и точку “Н”, в которой расположен наблюдатель, прямую линию и сформировать плоскость, которая совпадает с гранью выпуклого тела, то можно довольно просто определить, видна или не видна эта грань.

Так, если плоскость пересекает линию на участке от “О” (внутренней точки) до точки “Н”, то такая грань будет видна и будут видны все ребра тела, принадлежащие этой грани.

Иначе грань не видна.

Для этого в качестве тестовой функции можно использовать матричное уравнение плоскости, определяемое тремя принадлежащими грани точками:

$$f(x,y,z)=\begin{vmatrix} x-x_1 & y-y_1 & z-z_1 \\ x_2-x_1 & y_2-y_1 & z_2-z_1 \\ x_3-x_1 & y_3-y_1 & z_3-z_1 \end{vmatrix}$$

При этом если при подстановки в это уравнение координат точки $f(x,y,z)=0$ – то точка P(x,y,z) принадлежит плоскости, иначе (если > 0 или < 0) данная точка лежит в одном из полупространств, т.е. точка лежит вне плоскости.

Так как для каждой грани выпуклого тела можно определить три точки и составить тестовую функцию для соответствующей грани. В эту функцию подставляют координаты внутренней точки “О” и точки наблюдателя “Н” определяют значения тестовых функций:

$$f(x_0,y_0,z_0)=f(O)$$

$$f(x_H,y_H,z_H)=f(H)$$

Если эти функции разного знака:

$f(O)*f(H)<0$ – то плоскость пересекает прямую на участке от “О” до “Н” и анализируемая грань видна, и все ребра ее видны – их все можно отобразить;

иначе:

$f(H)*f(O)\geq 0$ – грань не видна. (Равенство 0 говорит о том, что плоскость проходит через точку “Н”, и плоскость рассматриваемой грани для наблюдателя вырождается в линию.) Ребра невидимых граней можно считать невидимыми и не изображать или изображать невидимыми, например, пунктирными линиями.

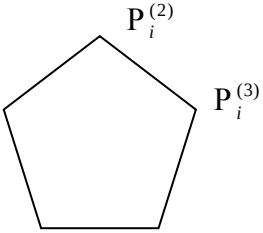
Координаты внутренней точки “О” выпуклого тела можно определить, сложив координаты 3-х точек, которые принадлежат одной грани тела и не лежат на одной прямой, с координатами 4-ой точки, принадлежащей любой другой грани, и разделив результат суммирования на 4.

Алгоритм, обеспечивающий сортировку граней по z – координате.

Исходными данными алгоритма являются

1) $A_i x + B_i y + C_i z + D_i = 0$ - уравнения плоскостей (i-ой грани),

2) P_i – вершины i- той грани тел, участвующих в выявлении видимости,



$$P_i(x_i^{(1)}, y_i^{(1)}, z_i^{(1)})$$

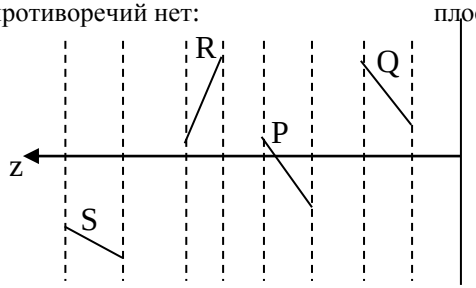
3) цвет каждой грани.

Первый этап.

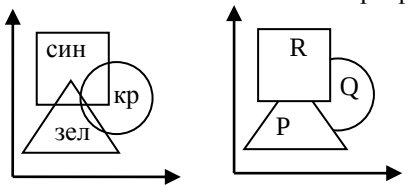
На первом этапе для каждой грани находится значения $Z_{\max i}$ и эти грани упорядочиваются по возрастанию или убыванию по этой координате:

$Q_i \quad P_{i+1} \quad R_{i+2} \quad S_{i+3}$ - упорядоченные грани по возрастанию $Z_{\max}(\uparrow)$

Если противоречий нет: плоскость визуализации XOУ:

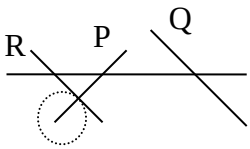


Если $z_{\min i+1} \geq z_{\max i}$ для каждого выполняется, т.е. каждая грань лежит в своем диапазоне, и эти диапазоны по координате Z не накладываются друг на друга, то формально можно выводить эти грани последовательно, начиная с грани Q. В таких случаях при выводе на экран дисплея последующей грани, если ее площадь в плоскости визуализации накладывается на плоскость ранее выведенных граней, то обеспечивается автоматическая перекраска пикселов ранее выведенных граней. В результате на экране дисплея формируется реально видимое изображение.



Иными словами, если противоречий нет, то можно изображение выводить на экран.

Если есть противоречия, т.е. например:

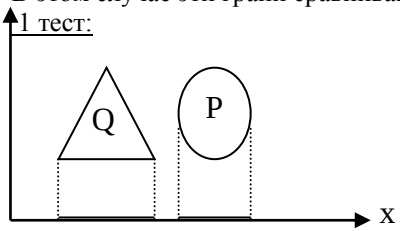


то эта часть потеряется

В подобных случаях следует вначале разрешить противоречия. Противоречия разрешаются путем использования следующих 5 тестов.

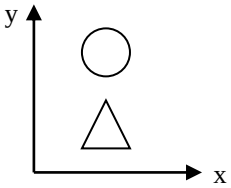
Например, возникли противоречия между гранями Q и P по координате Z. В этом случае эти грани сравниваются вначале по их оболочкам:

1 тест:



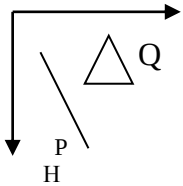
если по Z нарушается, а по координате X они разнесены (оболочки по X не перекрываются), то последовательность вывода граней неважна,

2 тест:



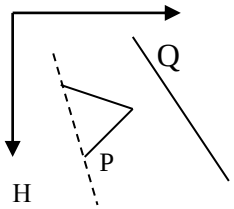
по X накладываются, а по Y разнесены – можно выводить

3 тест:



Проверяется с помощью тестовой функции не находится ли многоугольник Q дальше, чем плоскость P от наблюдателя, т.е. проверяются не находится ли точка H и все вершины многоугольника Q по разные стороны от плоскости, совпадающей с гранью P

4 тест:

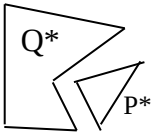


Проверяется с помощью тестовой функции не находятся ли все вершины многоугольника P по одну сторону с наблюдателем H по отношению к плоскости, совпадающей с плоскостью грани Q

Если хотя бы один тест 3 или 4 проходит – то противоречие считается разрешенным и соответствующие грани отображаются.

Если не проходят оба теста (3 и 4) грани P и Q в списке должны поменять их порядок, при этом грани в списке помечают, что свидетельствует об их перемещении, и возвращаются к проверке на 3 или 4 тест. Если после изменения порядка их описания в списке Z оба этих теста не прошли опять (второй раз их порядок не меняют), то следует перейти к проверки этих граней следующим видом теста (тестом 5).

Тест 5:



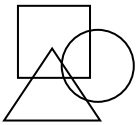
В этом тесте, проводимом над проекциями граней на плоскость визуализации (X0Y). В тесте выявляется, накладываются ли проекции граней друг на друга. Для этого в циклах по всем ребрам ищутся точки пересечения ребер одной грани с ребрами другой. Если пересечения ребер есть, то проверяется, только ли накладывается проекция одной грани на проекцию другой или плоскости граней взаимно проникают одна сквозь другую. Если они только накладываются, то порядок вывода должен определяться условиями наложения. Если взаимно проникают, то следует определить линию пересечения и в соответствии с этой линией разделить описание одной из граней на два многоугольника (например, Q_1^* и Q_2^*) и определить какой из них ближе к наблюдателю по отношению ко второй грани, а какой дальше. В соответствии с этим и следует определять порядок вывода части разрезанной грани и неразрезанной.

Таким образом, если грани взаимно проникающие, то из P^*, Q^* следует образовать P^*, Q_1^*, Q_2^* и один многоугольник необходимо переместить в списке впепел: Q_2^*, P^*, Q_1^*

Разрешив противоречия, следует последовательно вывести все грани всех объектов сцены.

Алгоритм выявления видимых частей сложной сцены, основанный на использовании Z-буфера.

Исходные данные для алгоритма: описание всех граней (количество граней, количество вершин каждой грани и значение их координат, а также цвет каждой грани). На основании этих данных определяются уравнения для плоскостей граней и границы, соответствующих им многоугольников.



Кроме буфера визуализации (экрана), который предназначен для хранения целочисленных значений цвета каждого пиксела экрана, алгоритм использует дополнительный буфер, в котором для каждого пикселя фиксируется вещественное максимальное значение z координаты, соответствующей X_iY_j рассматриваемого пиксела грани (многоугольника). Этот пиксел должен быть виден в текущий момент работы алгоритма.

При работе алгоритма с учетом уравнения плоскости каждого рассматриваемого текущего многоугольника грани:

$$A\,x_i + By_i + Cz_i + D_i = 0$$

для каждой его точки определяется: $z_{i,\,j}\Big|_{x_i, y_{j=const}}$

При этом предварительно в во все ячейки Z-буфера заносится значение «0» ($z_{б}|_{i,\,j} = 0$)

Далее берется многоугольник и для каждой точки многоугольника определяется:

$$z_{i,j} = -\left(\frac{A_i}{C_i}x_i + \frac{B_i}{C_i}y_i + \frac{D_i}{C_i}\right).$$

Затем рассчитанное $z_{i,\,j}$ сравнивается со значением в Z-буфере, т.е с $z_{б}|_{i,\,j}$

Если $z_{i,\,j} > z_{б}|_{i,\,j}$, то $z_{б}|_{i,\,j} = z_{i,\,j}$ и этому пикселю присваивается цвет соответствующий текущему многоугольнику.

Если $z_{i,\,j} < z_{б}|_{i,\,j}$, то значение $z_{б}|_{i,\,j}$ не корректируют, и цвет пиксела не меняют.

И так в циклах по X и Y просматриваются все точки всех многоугольников. Для ускорения работы при переходе вдоль оси X от одной точки к другой Y_j = const и тогда значение Z от точки к точке меняется на постоянную величину $\frac{A_i}{C_i}$:

$$z_i\Big|_{xi+1,y} = z_i\Big|_{xi+1,y} - \frac{A_i}{C_i}$$

Алгоритм с использованием Z-буфера в настоящее время находит широкое применение при большом количестве объектов, так как быстродействие ЭВМ существенно возросло, а память стала очень дешевой.