Министерство образования Российской федерации
Самарский государственный аэрокосмический
университет имени академика С.П. Королева
Кафедра технической кибернетики

ИНФОРМАТИКА.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

САМАРА
2004

Составитель Колчин Ю.В.

Информатика.

Метод. указания / Самар. гос. аэрокосм. ун-т. Сост. Колчин Ю.В.

Самара, 2004. – 21 с.

Кратко изложены теория и задания к выполнению лабораторных работ курса “Информатика” раздел Компьютерная графика.

Методические указания предназначены для студентов факультета информатики, обучающихся по специальности 010200 (направлению 510200) Прикладная математика и информатика.

Содержание

Растровое представление отрезков.

Экран дисплея с электронно-лучевой трубкой можно рассматривать как матрицу дискретных элементов - пикселей, каждый из которых может быть подсвечен, то есть он относится к растровым устройствам. Процесс определения пикселей, наилучшим образом аппроксимирующих заданный отрезок, называется разложением в растр. В сочетании с процессом построчной визуализации изображения он известен как преобразование растровой развертки

Общие требования к алгоритмам вычерчивания отрезков следующие:

- отрезки должны выглядеть прямыми
- концы отрезков должны находиться в заданных точках
- яркость вдоль отрезка должна быть постоянной и не зависеть от длины и наклона
- рисовать нужно быстро.

Эти требования возможно выполнить лишь частично. Концы отрезков располагаются на пикселях ближайших к истинным координатам отрезков. Расположение вдоль прямой и постоянная вдоль всего отрезка яркость достигаются лишь при проведении горизонтальных, вертикальных и наклоненных под углом 45° прямых. Даже в этом случае на единицу длины отрезка приходится различное количество пикселей. Для всех других углов наклона отрезков разложение в растр приведет к неравномерности яркости.

Обычно шаг вдоль большей из проекций на координатные оси равен расстоянию между соседними пикселями, а для изменения другой координаты

текущего пикселя проводятся дополнительные расчеты. Алгоритм расчета и определяет быстроту растровой развертки.

Один из методов разложения отрезка в растр состоит в решении дифференциального уравнения, описывающего этот процесс. Для прямой линии имеем

$$dy / dx = \text{const} \text{ или } \Delta y / \Delta x = \frac{y2 - y1}{x2 - x1}$$

Решение представляется в виде

$xi+1 = xi + \Delta x$

$yi+1 = yi + \Delta x\ (y2 - y1) / (x2 - x1)$ [1]

где x1, y1 и x2, y2 - концы разлагаемого отрезка и yi - начальное значение для очередного шага вдоль отрезка. Фактически уравнение [1] представляет собой рекуррентное соотношение для последовательных значений y вдоль нужного отрезка. Этот метод, используемый для разложения в растр отрезков, называется цифровым дифференциальным анализатором (ЦДА). В простом ЦДА либо Δx, либо Δy (большее из приращений) выбирается в качестве единицы растра.

Алгоритм Брезенхема.

Хотя алгоритм Брезенхема был первоначально разработан для цифровых графопостроителей, однако он в равной степени подходит для использования растровыми устройствами с ЭЛТ. Алгоритм выбирает оптимальные растровые координаты для представления отрезка. В процессе работы одна из координат - либо x, либо y (в зависимости от углового коэффициента) - изменяется на единицу. Изменение другой координаты (на 0 или 1) зависит от расстояния между действительным положением отрезка и ближайшими координатами сетки. Такое расстояние мы назовем ошибкой.

Алгоритм построен так, что требуется проверить лишь знак этой ошибки. На рис1 это иллюстрируется для отрезка в первом октанте, т.е. для отрезка с

угловым коэффициентом, лежащим в диапазоне от 0 до 1. Из рисунка можно заметить, что если угловой коэффициент отрезка из точки (0,0) больше, чем 1/2, то пересечение с прямой x = 1 будет расположено ближе к прямой y = 1, чем к прямой y = 0. Следовательно, точка растра (1,1) лучше аппроксимирует ход отрезка, чем точка (1,0). Если угловой коэффициент меньше 1/2, то верно обратное. для углового коэффициента, равного 1/2, нет какого либо предпочтительного выбора.

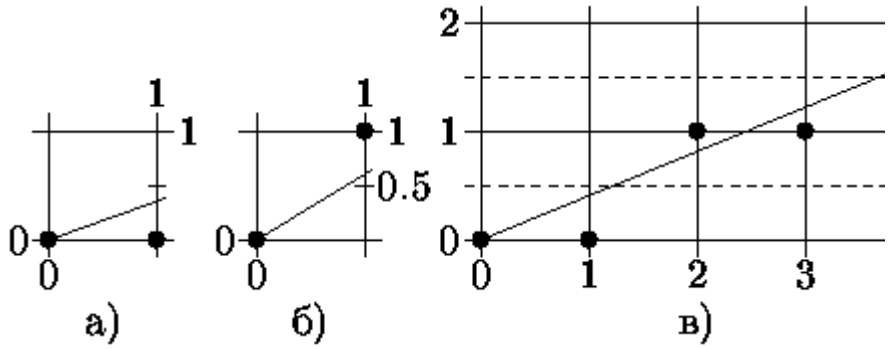

Рис.1. Основная идея алгоритма Брезенхема.

Желательно проверять только знак ошибки, то она первоначально устанавливается равной -1/2. Таким образом, если угловой коэффициент отрезка больше или равен 1/2, то величина ошибки в следующей точке растра с координатами (1,0) может быть вычислена как

$$e = e + m$$

где m - угловой коэффициент

Приведем алгоритм Брезенхема для первого октанта, т.е. для случая 0 =< dy =< dx.

Алгоритм Брезенхема разложения в растр отрезка для первого октанта

предполагается, что концы отрезка (x1,y1) и (x2,y2) не совпадают

Integer - функция преобразования в целое

x, y, dx, dy - целые

e - вещественное

инициализация переменных

```
x = x1
y = y1
dx = x2 - x1
dy = y2 - y1
Инициализация с поправкой на половину пиксела
e = dy/dx - 1/2
начало основного цикла
for i = 1 to dx
    plot (x,y)
    while ( e => 0 )
        y = y + 1
        e = e - 1
    end while
    x = x + 1
    e = e + dy/dx
next i
finish
```

Пример Алгоритм Брезенхема.

Рассмотрим отрезок проведенный из точки (0,0) в точку (5,5). Разложение отрезка в растр по алгоритму Брезенхема приводит к такому результату:

```
начальные установки
x = 0
y = 0
```

dx = 5

dy = 5

e = 1 - 1/2 = 1/2

результаты работы пошагового цикла

	Plot	e	x	y
		1/2	0	0
1	(0,0)			
		-1/2	0	1
		1/2	1	1
2	(1,1)			
		-1/2	1	2
		1/2	2	2
3	(2,2)			
		-1/2	2	3
		1/2	3	3
4	(3,3)			
		-1/2	3	4
		1/2	4	4
5	(4,4)			
		-1/2	5	5
		1/2	5	5

Результат совпадает с ожидаемым. Заметим, что точка растра с координатами (5,5) не активирована. Эту точку можно активировать путем изменения цикла for-next на 0 to dx. Активацию точки (0,0) можно устранить, если поставить оператор Plot непосредственно перед строкой next i.

После внесения в величину ошибки поправочного коэффициента 2*dx алгоритм становится целочисленным.

Задание:

1. Записать и реализовать целочисленный алгоритм Брезенхема для октанта Отличного от первого. При отображении результатов на экране пиксели должны изображаться окружностями или квадратами с габаритами не менее 5 пикселей.

2. Укажите в чем отличие Вашего алгоритма от алгоритма для первого октанта.

3. Как изменить алгоритм Брезенхема для рисования отрезков произвольной толщины?

4. Какие Вы можете предложить способы устранения ступенчатости при растровой развертке отрезков?

Варианты заданий:

№ варианта	№ октанта	знак dx	знак dy	Длины проекций
1	2	dx >= 0	dy >= 0	dy >= dx
2	3	dx =< 0	dy >= 0	dy >= -dx
3	4	dx =< 0	dy >= 0	dy =< -dx
4	5	dx =< 0	dy =< 0	- dy =< -dx
5	6	dx =< 0	dy =< 0	-dy >= -dx
6	7	dx >= 0	dy =< 0	-dy >= dx
7	8	dx >= 0	dy =< 0	- dy =< dx

Алгоритмы двумерного отсечения.

Применяемое в настоящее время программное обеспечение широко использует окна при отображении графической информации. Обычно эти окна – прямоугольники со сторонами параллельными осям координат. В процессе работы с такими окнами меняются их габариты, масштаб отображения внутри окна, одни окна перекрываются другими. Все это делает актуальным задачу определения графических примитивов или их частей попадающих в область окна. Алгоритмы решающие подобные задачи называются алгоритмами отсечения. Алгоритмы отсечения применяются также при удалении невидимых линий, построении теней и ряде других задач компьютерной графики.

Рассмотрим некоторые из наиболее распространенных алгоритмов двумерного отсечения.

Обычно в окно попадает лишь часть объекта. При этом необходимо определить те графические примитивы, которые не попадают в окно. Кроме того, часть примитивов целиком лежат внутри окна и потому изображаются без изменений. Будем рассматривать наиболее распространенные в графике примитивы – отрезки.

Прямоугольное окно множество выпуклое. Для проверки принадлежности отрезка окну достаточно определить лежат ли концы отрезка $P_1 – (X_1,Y_1)$ и $P_2 – (X_2,Y_2)$ внутри окна с границами $X_л$,$X_п$,$Y_н$,$Y_в$.

$$X_л \le X_1 \le X_п$$

$$X_л \le X_2 \le X_п$$

$$Y_н \le Y_1 \le Y_в$$

$$Y_н \le Y_2 \le Y_в$$

Отрезок тривиально невидим если оба его конца лежат с вне какой либо из сторон окна, например $X_1 \le X_л$ и $X_2 \le X_л$.

В остальных случаях отрезок пересекает стороны окна или их продолжения. При этом он может быть частично видим либо не имеет видимых частей. При анализе видимости находят точки пересечения отрезка с линиями ограничивающими окно. В алгоритме Сазерленда Коэна отрезок последовательно отсекается каждой из сторон окна.

В лабораторной работе предлагается реализовать иной подход: отсекать отрезок вертикальной или горизонтальной полосой. Совместное использование этих этапов приводит к верному результату.

Методы отсечения отличаются способом определения координат точки пересечения с границами окна. Рассмотрим отсечение вертикальной полосой.

1 метод использует уравнение прямой с угловым коэффициентом $y=kx+b$. В подобном виде невозможно представить вертикальные отрезки, но они параллельны границам области отсечения. В остальных случаях значение коэффициентов можно найти из условия прохождения прямой через концы отрезков.

Координаты точек пересечения с левой стороной $(X_л, Y_л)$ где $Y_л = kX_л + b$, с правой стороной $(X_п, Y_п)$ где $Y_п = kX_п + b$.

2 метод использует параметрическое представление отрезка

$P(t)=P_1-t(P_2-P_1)$ при $t \in [0,1]$ точка лежит внутри отрезка. Для координат x,y имеем $x(t)=x_1-t(x_2-x_1)$, $y(t)=y_1-t(y_2-y_1)$. По известным координатам например $x_л$ находим значение параметра $t_л$ и соответствующее значение $y_л$.

$$t_л = \frac{x_л - x_1}{x_2 - x_1}$$

$$y_л = y_1 - t_л(y_2 - y_1)$$

3 метод – разбиение средней точкой. Алгоритм итерационный. Пусть один из концов отрезка лежит внутри, а другой вне области отсечения. Вычисляются координаты середины отрезка. Если средняя точка лежит внутри области, одна из его половин видима. В противном случае одна из половин невидима. В обоих случаях вторая половина опять делится пополам. Алгоритм заканчивает работу при попадании точки на границу области отсечения. При таком подходе видимая часть отрезка состоит из кусочков. Для устранения этого эффекта в процессе работы алгоритма запоминаются наиболее удаленные от концов отрезка видимые точки, они и определяют его видимую часть.

4 метод применяется в алгоритме Кируса – Бека. Этот метод позволяет проводить отсечение произвольным выпуклым многоугольником. Отрезок P_1P_2 задается параметрически. Для определения точки пересечения необходимо значение вектора внутренней нормали N к стороне многоугольника и координаты точки, лежащей на этой стороне T. Для точек на прямой содержащей отрезок P_1P_2 имеем:

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

Рассмотрим вектор W, соединяющий точку N с точкой P(t).

$$W=P(t)-T=(P_1-T)-t(P_2-P_1)$$

В точке пересечения со стороной вектора W и N ортогональны. Их скалярное произведение (W,N)=0. Окончательно имеем:

$$t=\frac{(N,(P_1-T))}{(N,(P_2-P_1))}$$

Для произвольного выпуклого многоугольника точки пересечения упорядочиваются по значению параметра t и подразделяются на точки входа и выхода из области. При отсечении полосой анализ упрощается до проверки условия $t\in[0,1]$.

Задание:

1. Реализовать отсечение полосой, для своего варианта, один из алгоритмов отсечения.
2. Протестировать программу для случаев полной видимости, тривиальной невидимости, пересечения одной и обеих границ полосы.
3. Чему соответствует в алгоритме Кируса Бека случай равенства нулю знаменателя дроби при вычислении t?

Варианты заданий:

№ варианта	№ реализуемого метода	полоса отсечения
1	1	вертикальная
2	2	вертикальная
3	3	вертикальная
4	4	вертикальная
5	1	горизонтальная
6	2	горизонтальная
7	3	горизонтальная
8	4	горизонтальная

Представление и преобразование точек и отрезков. Отображение, вращение, масштабирование.Проекции.

Представление и преобразование точек.

Представление точек традиционно осуществляется в декартовых координатах следующем виде:

На плоскости [x, y]

В пространстве [x, y, z]

Преобразование точек.

Операция переноса точек на величину вектора (dx,dy) дает результат

$$x'=x+dx,\ y'=y+dy$$

В векторной форме это преобразование записывается в виде:

$$[x',y'] = [x,y] + [dx,dy]$$

Масштабирование:

$$x' = x_* Sx,\ y' = y_* Sy$$

можно записать в матричном виде:

$$[x',y'] = [x,y] * \begin{bmatrix} Sx & 0 \\ 0 & Sy \end{bmatrix}$$

Операция поворота на угол α задает преобразование координат:

$$x' = x_*\cos\alpha - y_*\sin\alpha$$

$$y' = x_*\sin\alpha + y_*\cos\alpha$$

или в матричном виде:

$$[x',y'] = [x,y] * \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix}$$

Переход к однородным координатам путем добавления к двум координатам точки третьей координаты со значением равным единице, позволяет все операции осуществлять умножением координат точек на соответствующие матрицы:

Преобразование переноса:

$$[x',y',1] = [x,y,1] * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ dx & dy & 1 \end{bmatrix}$$

Масштабирование:

$$[x',y',1] = [x,y,1] * \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Поворот:

$$[x',y',1] = [x,y,1] * \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Представляют интерес еще несколько матриц преобразования координат точек двумерного пространства:
тождественному преобразованию соответствует умножение на единичную матрицу:

$$[x',y',1] = [x,y,1] * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

отображение симметрично относительно оси Oy:

$$[x',y',1] = [x,y,1] * \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

относительно оси Ox:

$$[x',y',1] = [x,y,1] * \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

относительно начала координат:

$$[x',y',1] = [x,y,1] * \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Таким образом, все рассмотренные нами преобразования двумерного пространства реализуются умножением однородных координат точек справа на соответствующую матрицу размерности 3x3.

Каждое последующее преобразование – умножение справа на очередную матрицу. Результат умножения двух матриц размерности 3x3 – матрица такой же размерности. Это обстоятельство позволяет накапливать последовательность преобразований в одной результирующей матрице и получать образ точек объектов умножением справа на эту матрицу.

Рассмотрим это на примере:

Исходный объект – ромб ABCD. A=(1,2), B=(4,4), C=(7,2), D=(4,0)

Преобразование P_1 – масштабирование вдоль оси Ox в два раза, вдоль оси Oy – в три.

Матрица этого преобразования:

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Преобразование P_2 – поворот вокруг центра координат на угол 30º. Матрица:

$$\begin{bmatrix} 0{,}866 & 0{,}5 & 0 \\ -0{,}5 & 0{,}866 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Сформируем матрицу строки которой – однородные координаты вершин ромба и умножим ее на матрицу результирующего преобразования $P_1 * P_2$

$$\begin{vmatrix} 1 & 2 & 1 \\ 4 & 4 & 1 \\ 7 & 2 & 1 \\ 4 & 0 & 1 \end{vmatrix} * \begin{vmatrix} 1{,}732 & 1 & 0 \\ -1{,}5 & 2{,}598 & 0 \\ 0 & 0 & 1 \end{vmatrix} = \begin{vmatrix} -1{,}268 & 6{,}196 & 1 \\ 0{,}928 & 14{,}392 & 1 \\ 9{,}124 & 12{,}196 & 1 \\ 6{,}928 & 4 & 1 \end{vmatrix}$$

Результат – матрица новых координат точек A′, B′, C′, D′.

Операция умножения матриц не коммутативна. Если поменять порядок преобразований будем иметь матрицу $P_2 * P_1$ и результат: A^*, B^*, C^*, D^*.

$$\begin{vmatrix} 1 & 2 & 1 \\ 4 & 4 & 1 \\ 7 & 2 & 1 \\ 4 & 0 & 1 \end{vmatrix} * \begin{vmatrix} 1{,}732 & 1{,}5 & 0 \\ -1 & 2{,}598 & 0 \\ 0 & 0 & 1 \end{vmatrix} = \begin{vmatrix} -0{,}268 & 6{,}696 & 1 \\ 2{,}928 & 16{,}392 & 1 \\ 10{,}124 & 15{,}696 & 1 \\ 6{,}928 & 6 & 1 \end{vmatrix}$$

Матрицы преобразований в трехмерном пространстве.

Использование однородных координат точек трехмерного пространства: [x,y,z,1] приводит к аналогичному результату. При этом матрицы преобразований имеют следующий вид:

Преобразование переноса:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ dx & dy & dz & 1 \end{bmatrix}$$

Масштабирование:

$$\begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Поворот вокруг оси Oz:

$$\begin{bmatrix} \cos\alpha & \sin\alpha & 0 & 0 \\ -\sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Поворот вокруг оси Ox:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Поворот вокруг оси Oy:

$$\begin{bmatrix} \cos\alpha & 0 & -\sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ \sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Математическое описание плоских геометрических проекций.

Для простоты будем считать, что при центральном проецировании картинная плоскость перпендикулярна оси z и совпадает с плоскостью z = d, а при параллельном совпадает с плоскостью z = 0. Проекции рассматриваются в системе координат наблюдателя, которая является левосторонней. Система координат, в которой ось x направлена вправо, ось y - вверх, а ось z - внутрь экрана, естественно согласуется с экраном дисплея.

Каждую из проекций можно описать матрицей размером 4x4. Этот способ оказывается удобным, поскольку появляется возможность объединить матрицу проецирования с матрицей преобразования, представив в результате две операции (преобразование и проецирование) в виде одной матрицы.

Матрица для центрального проецирования в таком случае будет такой:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1/d \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Матрица для косоугольной проекции:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ l\cos\alpha & l\sin\alpha & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Для проекции Кавалье l = 1, угол α составляет 45°. Для проекции Кабине l =1/2, а α=arctg(2) = 63,4°.

В случае ортогональной проекции l = 0 и α = 90°, поэтому $M_{орт}$ есть частный случай $M_{кос}$.

Матрица для ортогональной проекции:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Аксонометрические проекции можно получить с помощью двух последовательных поворотов объекта (сначала вокруг оси Oy на угол α, а затем вокруг оси Ox на угол β) и последующего ортогонального проецирования на плоскость xOy. Наиболее распространены два типа аксонометрических проекций: изометрия ($\alpha = \pm 45°$, $\beta = \pm 35{,}26°'$) и диметрия ($\alpha = \pm 29{,}52°$, $\beta = \pm 26{,}23°'$).

Для центральной проекции иногда удобнее считать, что плоскость проекции имеет уравнение z = 0, а центр проекции находится в точке на оси Oz с координатой z = -d. В этом случае матрица проецирования имеет вид:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Задание.

1. Выбрать объект для отображения. В качестве объекта можно использовать кривую заданную параметрически в трехмерном пространстве или каркасную модель.
2. Реализовать несколько видов преобразований объекта, включая вращения. Список преобразований согласовать с преподавателем. Преобразования должны фиксироваться с помощью матрицы результирующего преобразования 4x4.

3. Последним этапом должно быть одно из преобразований проецирования.
4. Что заносится в матрицу результирующего преобразования при инициализации?

Удаление невидимых линий и поверхностей.

Задача удаления невидимых линий и поверхностей является одной из наиболее сложных в компьютерной графике. Алгоритмы удаления служат для определения ребер и поверхностей, которые невидимы для наблюдателя, находящегося в заданной точке пространства.

В ряде алгоритмов при определении видимости используют вектор нормали к грани многогранника. Нормаль вычисляется как векторное произведение пары неколлинеарных векторов принадлежащих грани.

После проецирования грани отображаются в многоугольники на плоскости проекции. Для определения видимости грани вычисляется скалярное произведение вектора внутренней нормали и вектором параллельным направлению проецирования. Видимость определяется знаком этого произведения. Отображению подлежат лишь ребра видимых граней.

Можно также оперировать не ребрами, а гранями закрашивая соответствующие им многоугольники пикселями яркость которых рассчитывается по закону Ламберта.

$$I=I_0 k\cos\alpha$$

где I_0-интенсивность источника света, k – коэффициент отражения материала, α - угол между вектором внешней нормали и направлением на источник света.

Для удаления невидимых граней применяют также алгоритмы упорядочивающие грани по глубине и алгоритмы, использующие Z-буфер.

Задание.

1. Для объекта из предыдущего задания подобрать алгоритм удаления невидимых ребер или граней и реализовать его.
2. Как определить является нормаль к грани внешней или внутренней?

Литература.

1. Роджерс Д. Алгоритмические основы компьютерной графики. М. Мир. 1986.

2. Роджерс Д., Адамс Дж. Математические основы машинной графики. Пер. с англ. М.: Мир, 2001. 604 с. ил.

3. Дж.Фоли, А.вэн Дэм Основы интерактивной машинной графики. М. Мир, 1985 в 2-х книгах.