

Компьютерная графика

Задача синтеза сложного динамического изображения. Этапы синтеза изображения.	3
Последовательность и основное содержание.....	3
Преобразования на плоскости. Вывод расчетных соотношений. Матрицы преобразований.....	4
Построение плоских кривых. Выбор шага изменения аргумента. Алгоритм построения эллипса и окружности по методу средней точки.	4
Требования, предъявляемые к алгоритмам вычерчивания отрезков. Пошаговый алгоритм разложения отрезка в растр. Разложение в растр по методу цифрового дифференциального анализатора.....	5
Алгоритмы Брезенхема разложения отрезков в растр. Простой алгоритм Брезенхема.	
Целочисленный алгоритм Брезенхема. Общий алгоритм Брезенхема.	6
Основные расчетные соотношения и алгоритм Брезенхема для генерации окружности.	6
Растровая развертка сплошных областей. Алгоритм с упорядоченным списком ребер.	7
Заполнение многоугольников. Алгоритмы заполнения по ребрам, с перегородкой, со списком ребер и флагом.	7
Алгоритм заполнения с затравкой, простой алгоритм заполнения с затравкой. Построчный алгоритм заполнения с затравкой.....	8
Основы методов устранения ступенчатости. Алгоритм Брезенхема с устранением ступенчатости.	9
Двумерное отсечение. Простой алгоритм отсечения отрезка.....	10
Отсечение. Алгоритм Сазерленда-Коэна отсечения отрезка.	11
Отсечение Алгоритм разбиения средней точкой при отсечении отрезка.	12
Отсечение. Алгоритм Кируса-Бека отсечения отрезка.....	12
Внутреннее и внешнее отсечение. Определение выпуклости многоугольника; определение нормали; разбиение невыпуклых многоугольников.	13
Отсечение многоугольников. Алгоритм Сазерленда-Ходжмена.	13
Отсечение многоугольников невыпуклыми областями. Алгоритм Вейлера-Азерттона.	14
Модели трехмерных объектов. Требования, предъявляемые к моделям.	15
Операции преобразования в трехмерном пространстве. Матрицы преобразований.	16
Трехмерное отсечение. Виды отсекателей. Вычисление кодов концов отрезка для каждого типа отсекателей. Алгоритм отсечения отрезков средней точкой.	16
Отсечение отрезков в трехмерном пространстве. Трехмерный алгоритм Кируса-Бека.	17
Определение факта выпуклости трехмерных тел. Разбиение тела на выпуклые многогранники..	18
Алгоритм плавающего горизонта.	18
Задача удаления невидимых линий и поверхностей. Ее значение в машинной графике.	
Классификация алгоритмов по способу выбора системы координат (объектное пространство, пространство изображений). Задача удаления невидимых линий в объектном пространстве.....	19
Алгоритм Робертса.	19
Удаление невидимых линий и поверхностей в пространстве изображений. Алгоритм Варнока (разбиение окнами): последовательность действий и основные принципы. Типы многоугольников, анализируемых в алгоритме Варнока. Методы их идентификации.....	21
Алгоритм Вейлера-Азерттона удаления невидимых линий и поверхностей.....	24
Алгоритм, использующий Z-буфер.....	25

Алгоритм, использующий список приоритетов (алгоритм Художника).....	26
Алгоритм построчного сканирования, использующий Z-буфер. Интервальные методы построчного сканирования (основные предпосылки).....	26
Алгоритм определения видимых поверхностей путем трассировки лучей.	29
Построение реалистических изображений. Физические и психологические факторы, учитываемые при создании реалистичных изображений. Простая модель освещения.....	31
Построение реалистических изображений. Метод Гуро закраски поверхностей (получение сглаженного изображения).....	33
Построение реалистических изображений. Закраска Фонга (улучшение аппроксимации кривизны поверхности).....	35
Определение нормали к поверхности и вектора отражения в алгоритмах построения реалистических изображений. Определение направления отраженного и преломленного лучей.	35
Построение теней при создании реалистических изображений. Учет теней в алгоритмах удаления невидимых поверхностей.	36
Учет прозрачности в модели освещения. Учет прозрачности в алгоритмах удаления невидимых поверхностей.....	37
Учет фактуры при создании реалистических изображений.	38
Глобальная модель освещения. Алгоритм трассировки лучей с использованием глобальной модели освещения.....	39

Задача синтеза сложного динамического изображения. Этапы синтеза изображения. Последовательность и основное содержание.

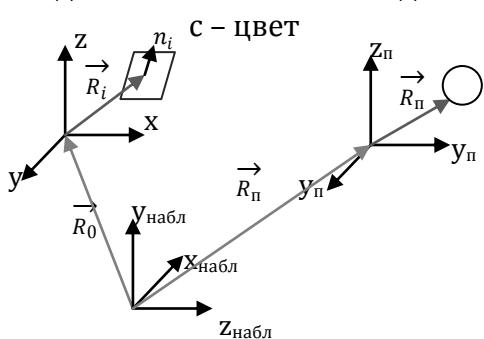
Компьютерная графика – совокупность методов и средств преобразования информации в и из графической формы с помощью ЭВМ. Деловая графика (графики, диаграммы); научная графика (чертежи, технологии моделирования); иллюстративная графика (карты, генерации символов и результатов операций); инженерная графика (иллюстрации к текстам).

Методы – математические, алгоритмические. Средства – технические, программные.

Уровни алгоритмов: Нижнего (базовые – точка, отрезок, эллипс); Среднего (плоские изображения с использованием готовых примитивов); Верхнего (трёхмерная графика, удаление невидимых линий, построение реалистических изображений).

Синтез сложного динамического изображения

неподвижная система подвижная система



I – интенсивность источника

β – коэффициент пропускания света через атмосферу

модели представления: каркасная, поверхностная, объёмная

положение поверхности может быть задано радиус-вектором, ориентация – вектором нормали

положение наблюдателя (ось Z – направление взгляда)

- 1) уравнение поверхности
- 2) цвет
- 3) оптические свойства (коэффициенты зеркального отражения, диффузионного отражения, пропускания, преломления)

Изображение строим в картинной плоскости (расположена перпендикулярно взгляду), на ней задаётся окно обзора (ось z через центр окна обзора)

f = 30-50Гц – частота генерации изображения

- 4) Источник света (положение в пространстве, цвет, интенсивность)
- 5) Для динамических объектов – уравнения воздействия, по которым можно рассчитать положение в интересующий момент времени
- 6) Характеристики окружающей среды (цвет фона)
- 7) Системы координат
- 8) Положение картинной плоскости, размер окна обзора

Этапы синтеза изображения

- 1) Разработка трёхмерной математической модели синтезируемой визуальной обстановки
- 2) Задание: положения наблюдателя, картинной плоскости, размеров окна вывода, значений управляющих сигналов
- 3) Определение операторов, определяющих пространственное перемещение объектов визуализации
- 4) Преобразования координат объектов в координаты наблюдателя
- 5) Отсечение объектов по границам пирамиды отсечения
- 6) Вычисление двумерных перспективных проекций объектов сцены на картинную плоскость
- 7) Удаление невидимых линий и поверхностей при заданном положении наблюдателя. Закрашивание и затенение видимых объектов.
- 8) Вывод полученного полутонового изображения на экран растрового дисплея.

Преобразования на плоскости. Вывод расчетных соотношений. Матрицы преобразований.

Преобразования на плоскости : $(X, Y) \rightarrow (X_1, Y_1)$

$$X_1 = Ax+By+C \quad Y_1 = Dx+Ey+F \quad (X_1 \ Y_1 \ 1) = (X \ Y \ 1)M \quad M = \begin{pmatrix} A & D & 0 \\ B & E & 0 \\ C & F & 1 \end{pmatrix}$$

Афинные преобразования – поверхность не вырождается в прямую, а прямая не вырождается в точку, сохраняется параллельность плоскостей и существуют обратные преобразования ($\det M \neq 0$).

Любое преобразование может быть представлено в виде трёх:

Перенос

dx, dy – 2 параметра

$$X_1 = X + dx \quad M = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ dx & dy & 1 \end{pmatrix}$$

$$Y_1 = Y = dy$$

Масштабирование

x_m, y_m, k_x, k_y – 4 параметра. $k_x \neq k_y$ – неоднородное масштабирование

$$X_1 = k_x X + (1-k_x)x_m \quad M = \begin{pmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{В матричной форме можно описать только отн. центра}$$

$$Y_1 = k_y Y + (1-k_y)y_m$$

Поворот

$$X_1 = x_c + (X - x_c) \cos \theta + (Y - y_c) \sin \theta \quad M = \begin{pmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \det M = 1$$

$$Y_1 = y_c + (Y - y_c) \cos \theta - (X - x_c) \sin \theta$$

Перенос и поворот аддитивны, масштабирование мультипликативно.

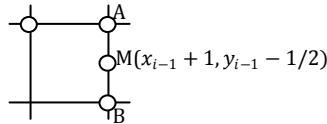
Коммутативны: перенос-перенос, масштабирование-масштабирование, поворот-поворот, масштабирование (однородное)-поворот.

Построение плоских кривых. Выбор шага изменения аргумента. Алгоритм построения эллипса и окружности по методу средней точки.

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \quad f_{np} = b^2x^2 + a^2y^2 - a^2b^2 \left\{ \begin{array}{l} = 0 \in \text{эллипсу} \\ > 0 \text{ вне} \\ < 0 \text{ внутри} \end{array} \right.$$

$$b^2x^2 + a^2y^2 - a^2b^2 = 0 \rightarrow \frac{dy}{dx} = \frac{-b^2x}{a^2y} \rightarrow b^2x = a^2y \text{ - условие границы}$$

Изучаем участок

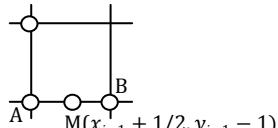


$$f_{np} = b^2(x_{i-1} + \frac{1}{2})^2 + a^2(y_{i-1} - 1)^2 - a^2b^2 \left\{ \begin{array}{l} = 0 \text{ A} \vee \text{B} \\ > 0 \text{ B} \\ < 0 \text{ A} \end{array} \right.$$

$$df = 2b^2x_{i-1} + b^2$$

Корректируем значение пробной точки для следующего шага.

Новый участок



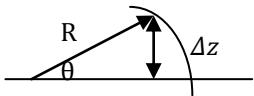
$$f_{np} = b^2(x_i + 1)^2 + a^2(y_{i-1} - \frac{1}{2})^2 - a^2b^2 \left\{ \begin{array}{l} = 0 \text{ A} \vee \text{B} \\ > 0 \text{ A} \\ < 0 \text{ B} \end{array} \right.$$

$$df = -2a^2y_{i-1} + a^2$$

Корректируем значение пробной точки для следующего шага.

Выбор шага

При достаточно большом радиусе, 2 соседние точки должны быть выбраны так, чтобы величина угла в радианах была не менее $1/R$



$$\sin\theta = \frac{\Delta z}{R}; \lim_{\theta \rightarrow 0} \frac{\sin\theta}{\theta} = 1; \theta = \frac{\Delta z}{R} = \frac{1}{R}$$

$$y = f(x); R = \frac{(1 + y'^2)^{\frac{3}{2}}}{|y''|}; x = x(t); y = y(t); R = \frac{(x'^2 + y'^2)^{\frac{3}{2}}}{|x'y'' - y'x''|}; R = \frac{(\rho^2 + \frac{d\rho^2}{d\varphi})^{\frac{3}{2}}}{|\rho^2 + 2\frac{d\rho^2}{d\varphi} - \rho\frac{d^2\rho}{d\varphi^2}|};$$

Требования, предъявляемые к алгоритмам вычерчивания отрезков. Пошаговый алгоритм разложения отрезка в растр. Разложение в растр по методу цифрового дифференциального анализатора.

Процесс нахождения пикселей, наилучшим образом аппроксимирующих заданный отрезок, называется *разложением отрезка в растр*.

Требования

1. отрезки должны выглядеть прямыми; начинаться и заканчиваться в заданных точках
2. яркость вдоль отрезка должна быть постоянной и не зависеть от длины и наклона
3. алгоритмы должны работать быстро

Первое требование в силу дискретной природы растрового дисплея выполнено всегда быть не может. Можно лишь добиться того, что визуально (человеческим глазом) отрезок будет восприниматься прямым. Решение этой задачи может достигаться путем увеличения разрешающей способности экрана дисплея и применения методов устранения ступенчатости.

Второму требованию удовлетворяют также только горизонтальные, вертикальные и наклоненные под углом в 45° отрезки. Однако вертикальные и горизонтальные отрезки по сравнению с отрезками, расположенными под 45°, будут выглядеть ярче, так как расстояние между соседними пикселями у них меньше, чем у наклонных отрезков. Обеспечение постоянной яркости вдоль отрезка требует вычисления очередного пикселя яркостью, зависящей от расстояния между пикселями, вычисление которого производится с использованием операций извлечения квадратного корня и умножения. Использование этих операций существенно замедляет работу алгоритма, поэтому второе требование остается, как правило, невыполненным.

Удовлетворение третьего требования достигается путем сведения к минимуму вычислительных операций, использования операций над целочисленными данными, а также реализацией алгоритмов на аппаратном или микропрограммном уровне.

Алгоритм ЦДА

1. Ввод исходных данных X_n, Y_n, X_k, Y_k .
2. Проверка вырожденности отрезка. Если отрезок вырожден, то вычисление точки и переход к п.7.
3. Вычисление $l = \begin{cases} |X_k - X_n| & \text{если } |X_k - X_n| > |Y_k - Y_n| \\ |Y_k - Y_n|, & \text{иначе} \end{cases}$
4. Вычисление $dX = (X_k - X_n) / l, dY = (Y_k - Y_n) / l$.
5. Задание координатам текущей точки начальных значений: $X = X_n, Y = Y_n$.
6. Цикл от $i=1$ до $i=l+1$ с шагом 1:
 1. вычисление точки с текущими координатами ($E(X), E(Y)$), где E - операция округления до ближайшего целого);
 2. вычисление координат следующей точки: $X = X + dX, Y = Y + dY$.
7. Конец

Алгоритмы Брезенхема разложения отрезков в растр. Простой алгоритм Брезенхема. Целочисленный алгоритм Брезенхема. Общий алгоритм Брезенхема.

Простой алгоритм Брезенхема

1. Ввод исходных данных X_n, Y_n, X_k, Y_k
2. Проверка вырожденности отрезка. Если отрезок вырожденный, то высвечивается точка и осуществляется переход к п.8
3. Вычисление приращений $dX = X_k - X_n$ и $dY = Y_k - Y_n$.
4. Вычисление модуля тангенса угла наклона отрезка: $m = dY/dX$
5. Инициализация начального значения ошибки: $f = m - 0,5$
6. Инициализация начальных значений координат текущего пикселя: $X = X_n, Y = Y_n$
7. Цикл от $i=1$ до $i=dX+1$ с шагом 1:
 1. Высвечивание точки с координатами (X, Y)
 2. Если $f < 0$, то
 - i) $Y++$
 - ii) $f = f + m$
 3. Вычисление ошибки $f = f + m$
 4. $X++$
8. Конец

Общий алгоритм Брезенхема

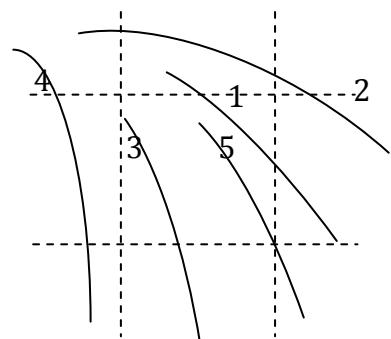
1. Ввод исходных данных X_n, Y_n, X_k, Y_k
2. Проверка вырожденности отрезка. Если отрезок вырожденный, то высвечивается точка и осуществляется переход к п.11
3. Вычисление приращений $dX = X_k - X_n$ и $dY = Y_k - Y_n$
4. Вычисление шага изменения каждой координаты пикселя: $SX = \text{sign}(dX), SY = \text{sign}(dY)$
5. Вычисление модулей приращения координат: $dX = |dX|, dY = |dY|$
6. Вычисление модуля тангенса угла наклона отрезка: $m = dY/dX$
7. Анализ вычисленного значения m и обмен местами dX и dY при $m > 1$ (fl - флаг обмена местами)
 1. если $m > 1$, то выполнить $W = dX, dX = dY, dY = W, m = 1/m, fl = 1$
 2. если $m < 1$, то $fl = 0$
8. Инициализация начального значения ошибки: $f = m - 0,5$ (для целочисленного $f = 2dY - dX$)
9. Инициализация начальных значений координат текущего пикселя: $X = X_n, Y = Y_n$
10. Цикл от $i=1$ до $i=dX+1$ с шагом 1:
 1. Высвечивание точки с координатами (X, Y)
 2. Вычисление координат и ошибки для следующего пикселя:
 3. Если $f > 0$, то
 - 1) если $fl = 1$, то $X = X + SX$
 - 2) иначе $Y = Y + SY$
 - 3) корректировка ошибки $f = f - 1$ (для целочисленного $f = f - 2dX$)
 4. Если $f < 0$, то
 - 1) если $fl = 1$, то $Y = Y + SY$
 - 2) иначе $X = X + SX$.
 5. Вычисление ошибки $f = f + m$ (для целочисленного $f = f + 2dY$)
11. Конец

Основные расчетные соотношения и алгоритм Брезенхема для генерации окружности.

$y = f(x)$ – монотонно убывающая

$$D = (x+1)^2 + (y-1)^2 - R^2 \begin{cases} < 0 & \text{внутри окружности (1 или 2)} \\ = 0 & \text{на окружности (5)} \\ > 0 & \text{вне окружности (3 или 4)} \end{cases}$$

1. $< 0; D1 = |(x+1)^2 + y^2 - R^2| - |(x+1)^2 + (y-1)^2 - R^2| > 0$ -диаг, ≤ 0 -горизонт



- 1) $(x+1)^2 + y^2 - R^2 > 0$; $D1 = 2(D+y) - 1$
- 2) < 0 ; $D1 = -2y + 1 < 0$
2. > 0 ; $D2 = |(x+1)^2 + (y-1)^2 - R^2| - |x^2 + (y-1)^2 - R^2| > 0$ -верт, ≤ 0 -диаг
- 3) < 0 ; $D2 = 2(D-x) - 1$
- 4) $x^2 + (y-1)^2 - R^2 > 0$; $D2 = 2x + 1 > 0$

Алгоритм

1. Ввод исходных данных R (радиус окружности) и при необходимости Xc, Yc (координаты центра окружности).
2. Задание начальных значений текущих координат пикселя $X=0, Y=R$, параметра $D=2(1-R)$, установка конечного значения ординаты пикселя $Yk=0$.
3. Высвечивание текущего пикселя (X, Y) .
4. Проверка окончания работы: если $Y < Yk$, то переход к п.11.
5. Анализ значения параметра D : если $D < 0$, то переход к п.6; если $D=0$, то переход к п.9; если $D > 0$, то переход к п.7.
6. Вычисление параметра $D1=2D+2Y-1$ и анализ полученного значения: если $D1 < 0$, то переход к п.8; если $D1 > 0$, то переход к п.9.
7. Вычисление параметра $D2=2D-2X-1$ и анализ полученного значения: если $D2 < 0$, то переход к п.9; если $D2 > 0$, то переход к п.10.
8. Вычисление новых значений X и D (горизонтальный шаг): $X=X+1; D=D+2X+1$. Переход к п.3.
9. Вычисление новых значений X, Y и D (диагональный шаг): $X=X+1; Y=Y-1; D=D+2(X-Y+1)$. Переход к п.3.
10. Вычисление новых значений Y и D (вертикальный шаг): $Y=Y-1; D=D-2Y+1$. Переход к п.3.
11. Конец.

Растровая развертка сплошных областей. Алгоритм с упорядоченным списком ребер.

Растровая развертка – генерация областей на основе простых описаний рёбер или вершин (закраска). Две категории методов – растровая развертка и затравочное заполнение.

Алгоритм с упорядоченным списком рёбер

1. Найти точки пересечения всех сканирующих строк со всеми рёбрами
2. Все точки в массив (список)
3. Сортировать их по убыванию y , а затем равные по возрастанию x
4. Разбить на пары и высветить пиксели между точками каждой пары

Заполнение многоугольников. Алгоритмы заполнения по ребрам, с перегородкой, со списком ребер и флагом.

Алгоритм по рёбрам

фон = закраска; закраска = фон. Для каждой сканирующей строки, пересекающей ребро многоугольника, дополнить все пиксели, лежащие правее точки пересечения. Недостаток – каждый пиксель может перекрашиваться много раз.

Алгоритм с перегородкой

Если точка пересечения сканирующей строки с ребром лежит левее перегородки, то дополнить все пиксели правее точки пересечения, но левее перегородки. Если правее – дополнить левее пересечения, но правее перегородки.

В обоих алгоритмах очерчивание границ не нужно

Алгоритм с флагом

1. Очертить контур, в результате чего на каждой строке образуются пары ограничивающих пикселей
2. Цикл по y от y_{max} до y_{min}
 - 1) $F=false$

- 2) Цикл по x от xmin до xmax
 а) (xy) – граница? да – F=-F
 б) F=true? да – закр, нет – фон

3. Конец

Алгоритм заполнения с затравкой, простой алгоритм заполнения с затравкой. Построчный алгоритм заполнения с затравкой.

Задать область и координаты точки внутри области. Рассматриваются пиксели, соседние с данным. Используется стек.

Области – 4-связные и 8-связные.

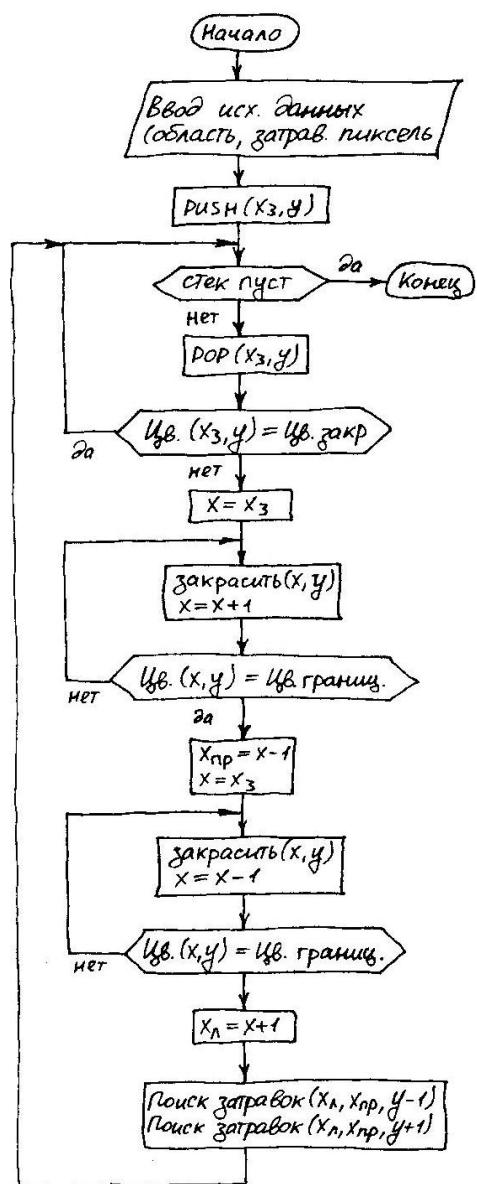
Простой алгоритм

1. Задание исходных данных
2. Поместить затравочный пиксель в стек
3. Пока стек не пуст
 1. Извлечь (xy) из стека
 2. Цвет(xy)≠Закраска → Цвет(xy):=Закраска
 3. Анализ 4 соседних пикселей. Цвет≠Закраска и Цвет≠Граница → пиксель в стек

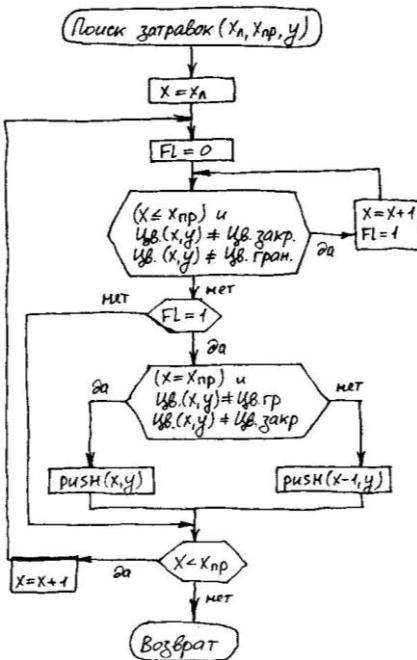
Построчный алгоритм

Основное отличие от простого – ограниченное количество пикселей в стеке. Непрерывный интервал пикселей – группа прилегающих друг к другу пикселей, незакрашенных и неграничных,

которые ограничены закрашенными или граничными пикселями.
В стек помещается 1 пиксель группы.



1. Задание исходных данных
2. Поместить затравочный пиксель в стек
3. Пока стек не пуст
 1. Извлечь (xy) из стека
 2. Заполнить влево от затравочного до граничного
 3. Запомнить Хлев
 4. Заполнить вправо до граничного
 5. Запомнить Хправ
 6. Поиск затравочных пикселей на соседних строках в пределах от Хлев до Хправ



Основы методов устранения ступенчатости. Алгоритм Брезенхема с устранением ступенчатости.

Принципиально устранить ступенчатость (лестничный эффект) невозможно. Однако применением специальных методов можно добиться того, что визуально ступеньки будут слабо заметны или

практически незаметны. Излагаемый способ рассматривает пиксел не как математическую точку, а как некоторую конечную область. Так же можно увеличить частоту выборки с помощью увеличения разрешения раstra.

$$\operatorname{tg} \alpha = m \text{ и } y_{i+m} = y_i + m$$

$$S = S_D + S_\Delta = y_i \cdot 1 + \frac{m \cdot 1}{2} = y_i + \frac{m}{2}$$

$$S = S_D + S_\Delta = y_i \cdot 1 + \frac{m \cdot 1}{2} = y_i + \frac{m}{2}$$



$$S_{i+1} = y_i \cdot 1 + m \cdot 1 + \frac{m \cdot 1}{2} = S_i + m$$

$$S_{i+1} = y_i \cdot 1 + m \cdot 1 + \frac{m \cdot 1}{2} - 1 \cdot 1 = S_i + m - 1 = S_i - (1 - m)$$

При наличии нескольких оттенков цвета внешний вид отрезка улучшается путем размывания его краев. Для этого интенсивность пикселя устанавливается пропорционально площади части пикселя, находящейся под отрезком.

Алгоритм

1. Ввод исходных данных X_n, Y_n, X_k, Y_k (координаты концов отрезка), I - количество уровней интенсивности
2. Проверка вырожденности отрезка. Если отрезок вырожден, то высвечивание отдельного пикселя и переход к п.13
3. Вычисление приращений $dX=X_k-X_n$ и $dY=Y_k-Y_n$
4. Вычисление шага изменения каждой координаты: $SX=\operatorname{sign}(dX)$, $SY=\operatorname{sign}(dY)$
5. Вычисление модулей приращения координат: $dX=|dX|$, $dY=|dY|$
6. Вычисление модуля тангенса угла наклона $m=dY/dX$
7. Анализ вычисленного значения m и обмен местами dX и dY при $m>1$
 1. если $m>1$, то выполнить $p=dX$; $dX=dY$; $dY=p$; $m=1/m$; $fl=1$
 2. если $m<1$, то $fl=0$
8. Инициализация начального значения ошибки $f=I/2$
9. Инициализация начальных значений координат текущего пикселя: $X=X_n$, $Y=Y_n$
10. Вычисление скорректированного значения тангенса угла наклона $m=mI$ и коэффициента $W=I-m$
11. Высвечивание пикселя с координатами (X, Y) интенсивностью $E(f)$
12. Цикл от $i=1$ до $i=dX$ с шагом 1
 1. Если $f < W$, то
 - 1) если $fl=0$, то $X=X+SX$
 - 2) если $fl=1$, то $Y=Y+SY$
 - 3) $f=f+m$
 2. Если $f > W$, то $X=X+SX$, $Y=Y+SY$, $f=f-W$
 3. Высвечивание пикселя с координатами (X, Y) интенсивностью $E(f)$
13. Конец

Двумерное отсечение. Простой алгоритм отсечения отрезка.

Отсечение – удаление изображение за пределами выделенной области.

Коды концов

1001	1000	1010
0001	0000	0010
0101	0100	0110

$$S = \sum T; P = \sum T_1 T_2$$

S=0 – точка видима

P≠0 – отрезок невидим

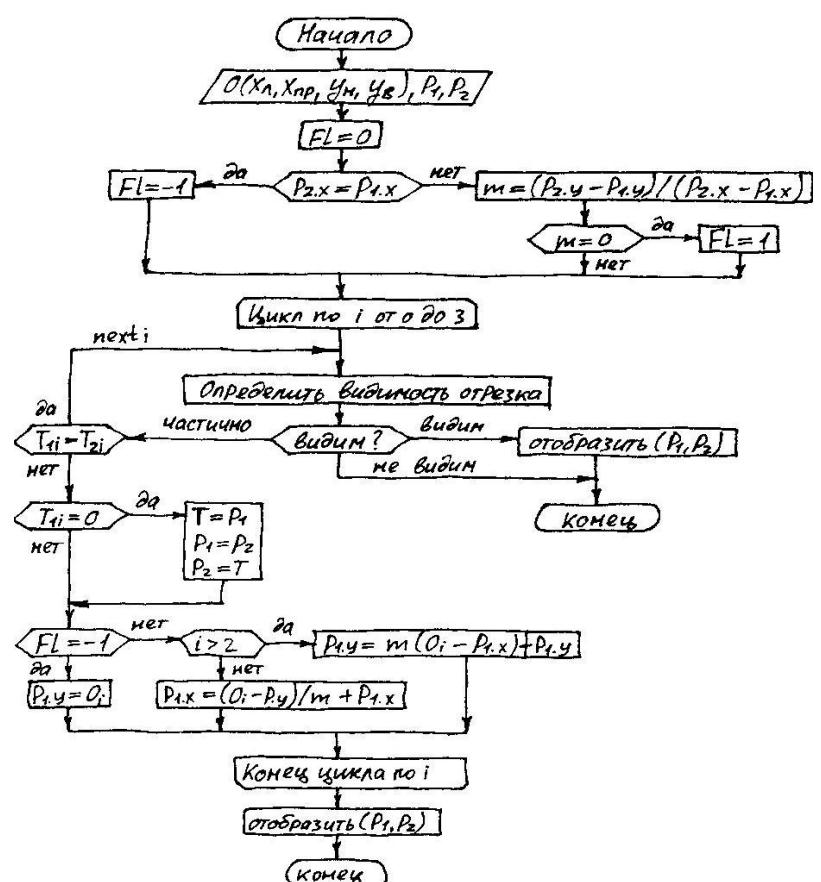
Алгоритм

1. Ввод координат отсекателя X_l, X_{pr}, Y_h, Y_b
2. Ввод координат концов отрезка $P_1(X_1, Y_1), P_2(X_2, Y_2)$
3. Вычисление кодов концов отрезка T_1, T_2 . Вычисление сумм кодов концов отрезка S_1, S_2
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, предполагается, что отрезок вертикальный)
6. Проверка полной видимости отрезка: если $(S_1=0) \& (S_2=0)=\text{true}$, 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 P
8. Проверка тривиальной невидимости отрезка: если $P=0$, 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 $S_2=0$ (первый 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 $Q=P_i$
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 $Q_x > X_l$ (пересечения нет), то переход к п.20
18. Вычисление ординаты точки пересечения отрезка с левой границей отсекателя: $Y_p=m(X_l-Q_x)+Q_y$
19. Проверка корректности найденного пересечения: если $(Y_p > Y_h) \& (Y_p < Y_b)=\text{true}$ (пересечение корректное), то выполнение следующих действий: $R_{i,x}=X_l, R_{i,y}=Y_p$ (занесение полученных координат в результат), переход к п.12
20. Проверка возможности пересечения отрезка с правой границей отсекателя: если $Q_x < X_n$ (пересечения нет), то переход к п.23.
21. Вычисление ординаты точки пересечения с правой границей: $Y_p=m(X_n-Q_x)+Q_y$
22. Проверка корректности найденного пересечения: если $(Y_p > Y_h) \& (Y_p < Y_b)=\text{true}$ (пересечение корректно), то выполнение следующих действий : $R_{i,x}=X_n, R_{i,y}=Y_p$ (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 $Q_y < Y_b$ (пересечения нет), то переход к п.27
25. Вычисление абсциссы точки пересечения с верхней границей: $X_p=(Y_b-Q_y)/m+Q_x$

26. Проверка корректности найденного пересечения: если $X_p > X_l \& (X_p < X_n) = \text{true}$ (пересечение корректно), то выполнение следующих действий: $R_{i,x} = X_p$; $R_{i,y} = Y_b$ (занесение полученных координат в результат); переход к п. 12
27. Проверка возможности пересечения с нижней границей отсекателя: если $Q_x > Y_h$ (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: $X_p = (Y_h - Q_y) / m + Q_x$
29. Проверка корректности найденного пересечения: если $(X_p > X_l) \& (X_p < X_n) = \text{true}$ (пересечение корректно), то выполнение следующих действий: $R_{i,x} = X_p$; $R_{i,y} = Y_h$ (занесение полученных координат в результат); переход к п. 12
30. Установка признака видимости $rg = -1$ (отрезок невидим полностью, так как ни одно пересечение не оказалось корректным)
31. Проверка признака видимости: если $rg = 1$, то вычерчивание отрезка R_1R_2
32. Конец

Отсечение. Алгоритм Сазерленда-Коэна отсечения отрезка.

1. Ввод координат отсекателя $X_l(01)$, $X_n(02)$, $Y_b(03)$, $Y_h(04)$.
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 0 до 3 отсечения отрезка по всем четырем сторонам отсекателя.
 - 1) 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
 - 2) Анализ полученного признака видимости: если $pr=-1$, то переход к п. 8; если $pr=1$, то переход к п. 7
 - 3) Проверка видимости обеих вершин отрезка относительно текущей i -ой стороны окна: если $T1(i)=T2(i)$, то переход к п.12)
 - 4) Проверка видимости первой вершины: если $T1(i)=0$ (вершина видима), то обмен местами вершин: $R=P1$; $P1=P2$; $P2=R$.
 - 5) Проверка вертикальности отрезка: если $Fl=-1$, то переход к п. 8)
 - 6) Анализ номера шага отсечения: если $i>3$, то переход к п. 8)
 - 7) Вычисление координат точки пересечения с i -ым ребром отсекателя (левым или правым): $P1.y = m(O_i - P1.x) + P1.y$; $P1.x = O_i$. Переход к п. 12)
 - 8) Проверка горизонтальности отрезка: если $Fl=1$, то переход к п. 12)
 - 9) Проверка вертикальности отрезка: если $Fl=-1$, то переход к п.11)
 - 10)Вычисление абсциссы точки пересечения отрезка общего положения со стороной отсекателя (верхней или нижней): $P1.x = (O_i - P1.y) / m + P1.x$



11) Присвоение ординате вершины отрезка ординаты стороны отсекателя: $P_1.y=O_i$

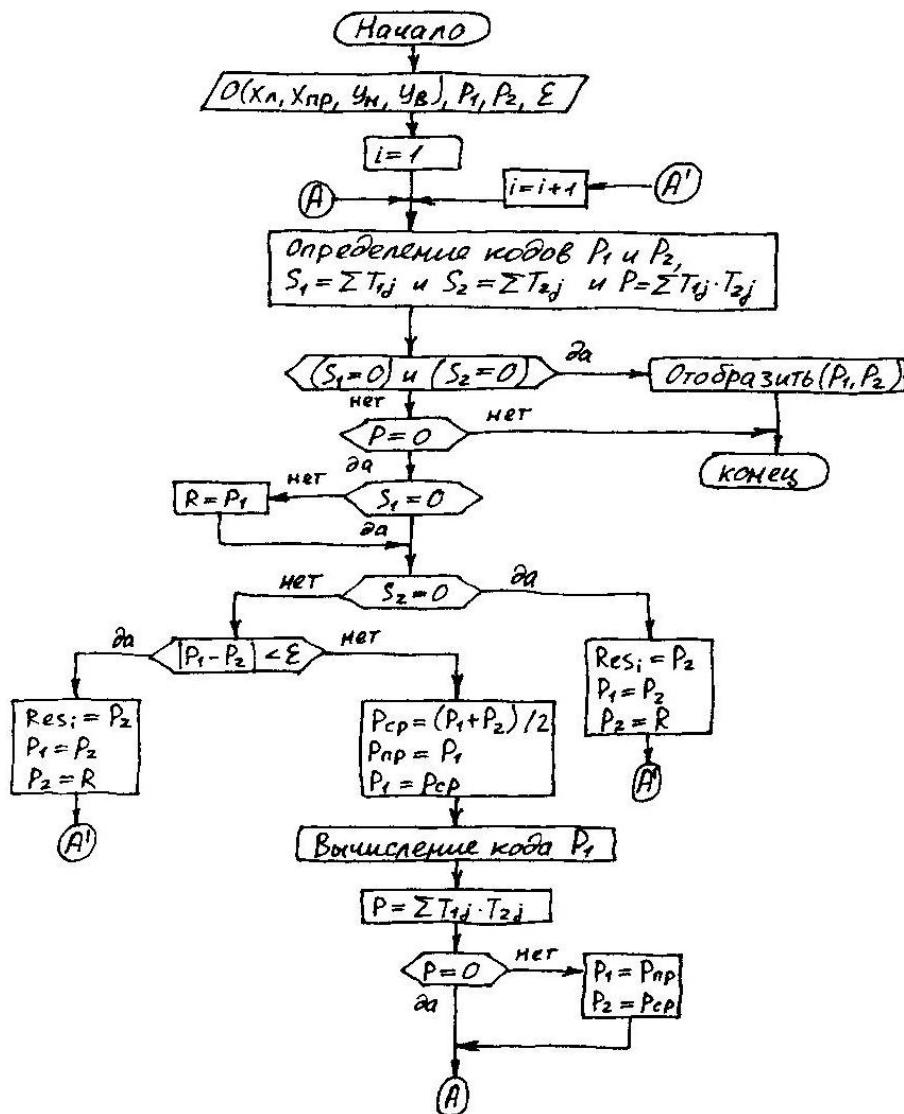
12) Конец цикла по i (вычисление нового значения параметра цикла $i=i+1$, анализ его значения и переход на повторное выполнение цикла или выход из цикла).

7. Вычерчивание отрезка P_1P_2 .

8. Конец.

Отсечение Алгоритм разбиения средней точкой при отсечении отрезка.

При реализации программно, работает медленнее простого. Аппаратно – быстрее в $\sqrt{2}$.



Отсечение. Алгоритм Кируса-Бека отсечения отрезка.

Отрезок задаётся в параметрическом виде $P(t)=P_1+(P_2-P_1)t$; $0 \leq t \leq 1$. Отрезок может пересекать выпуклое окно не более чем в двух точках.

Алгоритм использует понятие нормали к стороне многоугольника – окна отсечения.

Обозначим через N_i внутреннюю нормаль к i -той стороне окна, а через D – вектор, задающий ориентацию отсекаемого отрезка. $D = V_1 - V_0$

Скалярное произведение двух векторов: $P_i = N_i D = |N_i| * |D| * \cos$

1. Отрезок входит в окно через i -тое ребро многоугольника отсечения $P_i > 0$

2. Отрезок параллелен i -той стороне окна $P_i = 0$

3. Отрезок выходит из окна отсечения, пересекая i -тое ребро многоугольника $P_i < 0$

Алгоритм

1. $D=P_2-P_1$
2. $t_h=0, t_b=1$
3. Цикл по всем граням $i=1:N$
 - 1) Вычисление n_{vhi}
 - 2) $D_i=D \cdot n_{vhi}$
 - 3) $W_i=(P_1-f_i) \cdot n_{vhi}$
 - 4) $D_i=0?$
 - i) да. $W_i \geq 0$? нет - Конец. да - продолжаем
 - ii) нет. $t=-W_i/D_i; D_i > 0?$
 - (1) да. $t > 1?$
 - (a) нет. $t_h=\max(t_h, t)$
 - (b) да. Конец
 - (2) нет. $t < 0?$
 - (a) нет. $t_b=\min(t_b, t)$
 - (b) да. Конец
 4. $t_h \leq t_b?$ да – нарисовать($P(t_h), P(t_b)$)
 5. Конец

Внутреннее и внешнее отсечение. Определение выпуклости многоугольника; определение нормали; разбиение невыпуклых многоугольников.

Внутреннее – определяем части отрезка внутри отсекателя и вычерчиваем. *Внешнее* – определяем части вне отсекателя и вычерчиваем.

Определение выпуклости

Вычислить векторные произведения всех пар смежных сторон. Если все $=0$ – многоугольник вырожден. Все разных знаков – многоугольник невыпуклый. Все ≥ 0 – выпуклый, нормали ориентированы влево от рёбер. ≤ 0 – выпуклый, нормали вправо.

Разбиение

1. Перенести в $(0, 0)$ i -ую вершину
2. Повернуть вокруг $(0, 0)$, чтобы $i+1$ -ая вершина оказалась на +части оси x
3. Вычислить знаки ординат всех $i+2$ -ых вершин
 - 1) $=0$ – вырожден в отрезок
 - 2) одинаковы – выпуклый относительно этого ребра
 - 3) разные – невыпуклый, разрезаем на 2 части вдоль оси. далее работаем с двумя многоугольниками

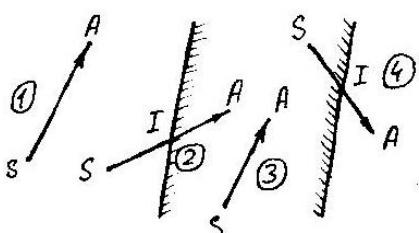
Определение внутренней нормали

$$\begin{aligned} C_{i+2} &\quad C_i C_{i+1} = C_{ix} i + C_{iy} j; n_i = n_{ix} i + n_{iy} j; \\ C_{i+1} &\quad C_i C_{i+1} n_i = 0; C_{ix} n_{ix} + C_{iy} n_{iy} = 0; \\ C_i &\quad n_{iy} = 1 \rightarrow n_{ix} = -C_{iy} / C_{ix}; \text{ если } C_{ix} = 0, \text{ то } n_{iy} = 0, n_{ix} = 1 \end{aligned}$$

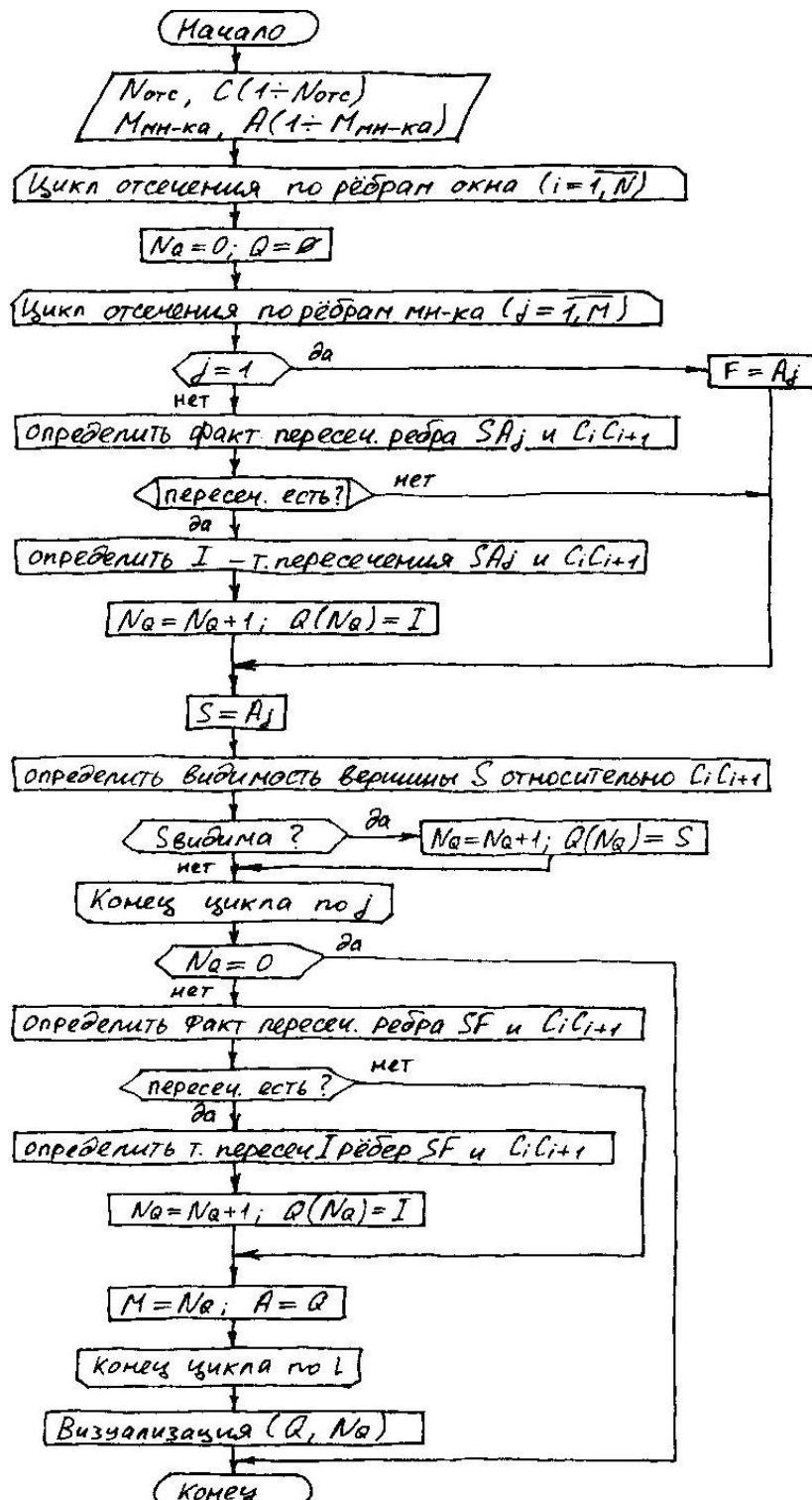
Отсечение многоугольников. Алгоритм Сазерленда-Ходжмена.

Исходный и каждый из промежуточных многоугольников отсекается последовательно относительно прямой.

1. Определить видимость вершин многоугольника
2. Определить точки пересечения рёбер исходного многоугольника и рёбер отсекателя
3. Заносить видимые вершины и найденные точки в результирующий список



- ① Ничего не заносить в список
- ② Занести т. I и т. A.
- ③ Занести т. A.
- ④ Занести т. I.



Отсечение многоугольников невыпуклыми областями. Алгоритм Вейлера-Азертона.

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.
12. Нахождение в списке вершин отсекаемого многоугольника очередной точки входа.
13. Просмотр списка вершин отсекаемого многоугольника до нахождения точки пересечения. Копирование просмотренных вершин в список внутренней принадлежности.
14. Переход к списку вершин отсекателя и поиск в нем одноименной точки пересечения.
15. Просмотр списка вершин отсекателя до нахождения точки пересечения. Копирование просмотренных вкшин в список внутренней принадлежности.
16. Сравнение найденной точки пересечения с первой точкой: если эти точки не совпадают, то переход к п. 13, иначе к п. 17.
17. 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.
18. Получение второй границы результирующего многоугольника. В этом случае можно использовать правило:
 - 1) 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.
 - 2) 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).
19. Конец цикла (выбор следующей точки входа из списка, если он не пуст и переход к п. 12, иначе выход из цикла).
20. Конец.

Модели трехмерных объектов. Требования, предъявляемые к моделям.

Геометрическая модель является машинным представлением формы и размера

Виды

1. Каркасная (не даёт представления о наличии отверстий)
2. Поверхностная (не даёт представления, по какую сторону материал, а по какую пустота)
3. Объёмная

Требования

1. Модель не должна противоречить исходному объекту
2. Модель должна допускать возможность конструирования тела целиком

3. Модель должна допускать вычисление геометрических характеристик тела
4. Модель должна позволять проводить расчеты: кинематические, сопротивления

Свойства

1. Однородность (тело заполнено изнутри)
2. Конечность (каждое тело занимает конечный объём)
3. Жёсткость (сплошное тело сохраняет форму независимо от положения)

Требования к ПО

1. Согласованность операций (любые операции над сплошными телами должны приводить к сплошным телам)
2. Возможность описания (любое тело должно представляться в машинном виде)
3. Непротиворечивость информации (любая точка принадлежит 1 телу, для любой точки можно сказать, принадлежит ли она телу)
4. Компактность модели
5. Открытость модели (разрабатываемая модель должна иметь применение при работе с разными алгоритмами)

Операции преобразования в трехмерном пространстве. Матрицы преобразований.

Перенос

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ dx & dy & dz & 1 \end{pmatrix}$$

Масштабирование

$$M = \begin{pmatrix} kx & 0 & 0 & 0 \\ 0 & ky & 0 & 0 \\ 0 & 0 & kz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Поворот

$$M_x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos & \sin & 0 \\ 0 & -\sin & \cos & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad M_y = \begin{pmatrix} \cos & 0 & -\sin & 0 \\ 0 & 1 & 0 & 0 \\ \sin & 0 & \cos & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad M_z = \begin{pmatrix} \cos & \sin & 0 & 0 \\ -\sin & \cos & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Перспективная проекция

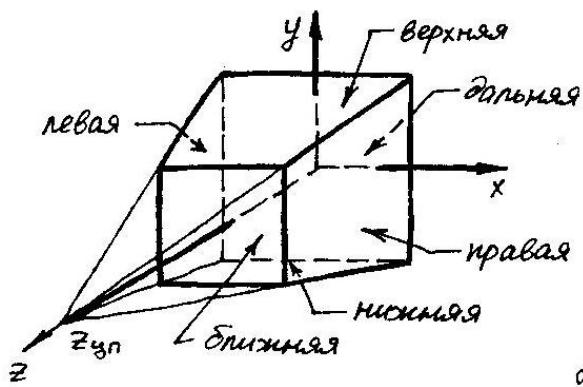
$$M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1/C \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Трехмерное отсечение. Виды отсекателей. Вычисление кодов концов отрезка для каждого типа отсекателей. Алгоритм отсечения отрезков средней точкой.

Виды

- Прямоугольный параллелепипед
- Усечённая пирамида видимости

Трёхмерный алгоритм аналогичен [двумерному](#), но к координатам добавляется третья компонента. Код видимости задаётся 6 битами. Если отсекатель – усечённая пирамида, то для составления кода используются пробные функции



Z_{cpr} – центр проекции

Уравнение плоскости, проходящей через правую грань:

$$x = (z - Z_{\text{cpr}}) \cdot x_{\text{pr}} / (Z_D - Z_{\text{cpr}})$$

$$x = Z\alpha_1 + \alpha_2, \quad \text{где}$$

$$\alpha_1 = x_{\text{pr}} / (Z_D - Z_{\text{cpr}}) \quad \text{и} \quad \alpha_2 = \alpha_1 \cdot (-Z_{\text{cpr}})$$

$$f_n = x - Z\alpha_1 - \alpha_2 \begin{cases} > 0 \Rightarrow P \text{ справа от плоскости *} \\ = 0 \Rightarrow P \text{ на плоскости} \\ < 0 \Rightarrow P \text{ слева от плоскости} \end{cases}$$

$$f_A = x - Z\beta_1 - \beta_2 \begin{cases} > 0 \Rightarrow P \text{ справа от плоскости} \\ = 0 \Rightarrow P \text{ на плоскости} \\ < 0 \Rightarrow P \text{ слева от плоскости *} \end{cases}$$

$$\beta_1 = x_{\text{pr}} / (Z_D - Z_{\text{cpr}}), \quad \beta_2 = -\beta_1 Z_{\text{cpr}}$$

$$f_B = y - Z\gamma_1 - \gamma_2 \begin{cases} > 0 \Rightarrow P \text{ выше плоскости *} \\ = 0 \Rightarrow P \text{ на плоскости} \\ < 0 \Rightarrow P \text{ ниже плоскости} \end{cases}$$

$$\gamma_1 = y_{\text{pr}} / (Z_D - Z_{\text{cpr}}), \quad \gamma_2 = -\gamma_1 Z_{\text{cpr}}$$

$$f_H = y - Z\delta_1 - \delta_2 \begin{cases} > 0 \Rightarrow P \text{ выше плоскости} \\ = 0 \Rightarrow P \text{ на плоскости} \\ < 0 \Rightarrow P \text{ ниже плоскости *} \end{cases}$$

$$\delta_1 = y_{\text{pr}} / (Z_D - Z_{\text{cpr}}), \quad \delta_2 = -\delta_1 Z_{\text{cpr}}$$

$$f_B = z - Z\epsilon_1 \begin{cases} > 0 \Rightarrow P \text{ ближе плоскости *} \\ = 0 \Rightarrow P \text{ на плоскости} \\ < 0 \Rightarrow P \text{ дальше плоскости} \end{cases}$$

$$f_D = z - Z_D \begin{cases} > 0 \Rightarrow P \text{ ближе плоскости *} \\ = 0 \Rightarrow P \text{ на плоскости} \\ < 0 \Rightarrow P \text{ дальше плоскости *} \end{cases}$$

Отсечение отрезков в трехмерном пространстве. Трехмерный алгоритм Кируса-Бека.

В трёхмерном варианте отсекатель может быть произвольным выпуклым телом. Алгоритм аналогичен [двумерному](#), только векторы имеют 3 компоненты. На каждом шаге ищем точку пересечения с i-ой гранью.

Определение факта выпуклости трехмерных тел. Разбиение тела на выпуклые многогранники.

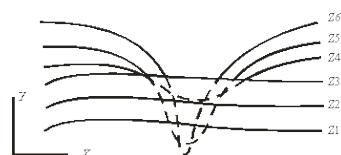
- Перенести тело так, чтобы одна из вершин совпала с центром координат
- Повернуть вокруг 0, чтобы одно из рёбер, выходящее из рассматриваемой вершины совпало с одной из координатных осей
- Поворот вокруг этой оси чтобы грань совпала с плоскостью
- Для всех вершин, не лежащих в плоскости определить знаки 3-ей координаты. Если одинаковы, тело выпукло относительно грани. Если не совпадают, то невыпукло. Равны 0 – вырождено.
- Если невыпукло, то разрезаем на части. Рассматриваемую процедуру применяем к каждой части.

Алгоритм плавающего горизонта.

$F(x, y, z)=0$ – поверхность в трёхмерном пространстве. Рассекаем $z=\text{const}$, $y=f(x, z=\text{const})$, $x=\varphi(y, z=\text{const})$

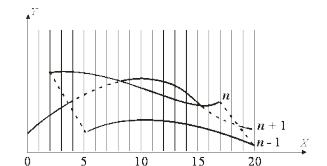


Для хранения максимальных значений y при каждом значении x используется массив, длина которого равна числу различных точек (разрешению) по оси x в пространстве изображения. Значения, хранящиеся в этом массиве, представляют собой текущие значения "горизонта". Поэтому по мере рисования каждой очередной кривой этот горизонт "всплывает". Фактически этот алгоритм удаления невидимых линий работает каждый раз с одной линией.

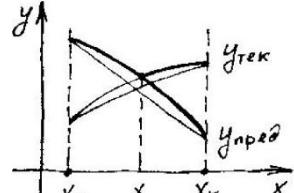


Алгоритм работает очень хорошо до тех пор, пока какая-нибудь очередная кривая не окажется ниже самой первой из кривых.

Нижняя сторона поверхности делается видимой, если модифицировать этот алгоритм, включив в него нижний горизонт, который опускается вниз по ходу работы алгоритма. Это реализуется при помощи второго массива, длина которого равна числу различных точек по оси x в пространстве изображения. Этот массив содержит наименьшие значения y для каждого значения x .



Проблема с зазубренностью боковых ребер решается включением в массивы верхнего и нижнего горизонтов ординат, соответствующих штриховым линиям. Это можно выполнить эффективно, создав ложные боковые ребра.



$$m_{\text{пред}} = \frac{y_{\text{пред}k} - y_{\text{пред}m}}{x_k - x_m}; m_{\text{тек}} = \frac{y_{\text{тек}k} - y_{\text{тек}m}}{x_k - x_m}$$

$$x_{\text{пер}} = x_m + \frac{\Delta x (y_{\text{тек}} - y_{\text{пред}})}{\Delta y_{\text{пред}} - \Delta y_{\text{тек}}}; y_{\text{пер}} = y_{\text{пред}} + m_{\text{пред}} (x_{\text{пер}} - x_m)$$

1. обработать левое боковое ребро
2. для каждой точки очередной кривой проверить условие видимости
3. если видимость изменилась, вычислить точку пересечения
4. изобразить видимую часть кривой
5. заполнить массивы горизонтов
6. обработать правое боковое ребро

Обработка боковых рёбер

Левое: если P_k – первая точка первой кривой, то запоминаем её в качестве предыдущей P_{k-1} , если на не первой – соединяем K и $K-1$ и запоминаем текущую как $K-1$. Аналогично правое.

В массивы верхнего и нижнего горизонтов заносим ординаты бокового ребра.

Задача удаления невидимых линий и поверхностей. Ее значение в машинной графике. Классификация алгоритмов по способу выбора системы координат (объектное пространство, пространство изображений). Задача удаления невидимых линий в объектном пространстве.

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

Единого и наилучшего способа решения задачи нет. Возникло множество алгоритмов, многие ориентированы на специализированные приложения. Все алгоритмы включают в себя сортировку. Главная – сортировка по расстоянию от точки наблюдения до тела-поверхности-ребра-точки. Так же есть сортировка по загораживанию тел друг другом. Эффективность алгоритмов в большой степени зависит от эффективности алгоритма сортировки.

Существует связь между скоростью работы алгоритма и детальностью его результата – ни один алгоритм не может достигнуть хороших показателей одновременно.

Алгоритмы делятся на:

1. Работающие в объектном пространстве (мировая ск, высокая точность). Сложность $\sim N^2$
2. Работающие в пространстве изображений (ск связана с дисплеем, точность ограничена разрешающей способностью дисплея). Сложность $\sim N^*M$

N – количество объектов, M – количество пикселей. $N^2 < N^*M$, но алгоритмы, работающие в пространстве изображений могут быть эффективнее в силу свойства когерентности при растровой реализации (близко расположенные пиксели чаще обладают одинаковыми свойствами).

Алгоритм Робертса.

Алгоритм удаления невидимых линий в объектном пространстве. Работает только с выпуклыми телами. Сортировка тел по мере их удалённости от наблюдателя. Сложность $\sim N^2$

Основные этапы

1. Подготовка исходных данных
2. Удаление линий, экранируемых самим телом
3. Удаление линий, экранируемых другими телами
4. Удаление линий пересечения тел, экранируемых самими телами, связанными отношением протыкания и другими телами

Формирование матрицы

Уравнение произвольной плоскости в трехмерном пространстве имеет вид $ax + by + cz + d = 0$. В матричной форме $[x \ y \ z \ 1][P]T = 0$, где $[P]T = [a \ b \ c \ d]$ представляет собой плоскость. Поэтому любое выпуклое твердое тело можно выразить матрицей тела, состоящей из коэффициентов уравнений плоскостей, т. е.

$[V] = \begin{bmatrix} a_1 & a_2 & \dots & a_n \\ b_1 & b_2 & \dots & b_n \\ c_1 & c_2 & \dots & c_n \\ d_1 & d_2 & \dots & d_n \end{bmatrix}$ Если же $[S]$ не лежит на плоскости, то знак этого скалярного произведения показывает, по какую сторону от плоскости расположена точка. В алгоритме Робертса предполагается, что точки, лежащие внутри тела, дают отрицательное скалярное произведение, т. е. нормали направлены наружу.

Матрица тела должна быть составлена так, чтобы в $[R] = [S][V]$, $r_i > 0$. Если это не так, матрица корректируется умножением соответствующего столбца на -1.

Коэффициенты можно определить из системы

$$\begin{cases} ax_1 + by_1 + cz_1 = -1 \\ ax_2 + by_2 + cz_2 = -1 \text{ или } [X][C]=[D], [C]=[X]^{-1}[D] \\ ax_3 + by_3 + cz_3 = -1 \end{cases}$$

Зная все координаты вершин грани, коэффициенты вычисляются

$$\begin{cases} a = \sum_{i=1}^n (y_i - y_j)(z_i + z_j) \\ b = \sum_{i=1}^n (x_i - x_j)(z_i + z_j), \text{ где } j = \begin{cases} i+1, & i < n \\ 1, & i = n \end{cases} \\ c = \sum_{i=1}^n (x_i - x_j)(y_i + y_j) \end{cases}$$

Зная уравнение нормали к плоскости $n = ai + bj + ck$, сразу видим a, b, c , а $d = -(ax_1 + by_1 + cz_1)$ для точки.

Можно преобразовать тело. $[VT] = [T]^{-1}[V]$.

Удаление нелицевых граней

$[E] = [0, 0, -1, 0]$ – вектор наблюдения. У вектора $[E_1] = [E][V]$ отрицательные компоненты будут соответствовать невидимым граням. $[E]n: >0$ – грань видима, <0 – невидима, $=0$ – на грани видимости. Невидимые рёбра образуются пересечением невидимых граней, поэтому удалив невидимые грани, мы удалили все невидимые рёбра. Других быть не может.

Удаление отрезков, экранируемых другими телами

Тело, рёбра которого проверяются – пробный объект, с которым проверяются – пробное тело.

Отсортировать все тела по максимальному z тела. Наиболее удалённое тело – более экранируемое.

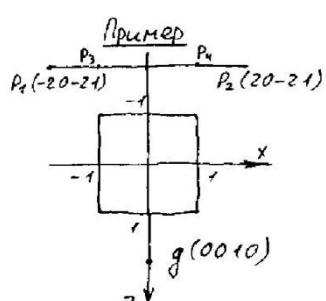
$Z_{\min p} > Z_{\max t}$ – тело не может загораживать ребро. $(X_{\max p} < X_{\min t}) \vee (X_{\min p} > X_{\max t}) \vee (Y_{\max p} < Y_{\min t}) \vee (Y_{\min p} > Y_{\max t})$ – тело экранировать ребро не будет.

Удаление экранируемых участков отрезка

P_1P_2 – исследуемый отрезок. $R(t) = P_1 + (P_2 - P_1)t$ – параметрическое уравнение. $\vec{v} = \vec{s} + t \vec{d}$. Нужно найти t , при котором изменяется видимость отрезка. Параметрически зададим отрезок от точки \vec{v} до \vec{g} – точки, где находится наблюдатель: $Q(\alpha, t) = \vec{v} + \alpha \vec{g}$, $0 \leq \alpha$. Наблюдатель в $+\infty$, а не $-\infty$. $g = [0, 0, 1, 0]$. $Q(\alpha, t) = \vec{s} + t \vec{d} + \alpha \vec{g}$ – фактически уравнение плоскости.

Скалярное произведение любой точки, расположенной внутри объекта и матрицы объекта, положительно (это утверждение справедливо и для преобразованной матрицы объекта). Точка, находящаяся внутри объекта, невидима. Следовательно, для проверки на экранирование вектор текущей точки отрезка умножают поочередно на матрицу каждого объекта и определяют положительное решение, соответствующее прохождению отрезка внутри объекта: $H = (\vec{s} + t \vec{d} + \alpha \vec{g})[V] > 0$. Значения t и α , для которых все значения вектора H положительны, соответствуют невидимой части отрезка. Пусть $\vec{p} = \vec{s}[V]$, $\vec{q} = \vec{d}[V]$, $\vec{w} = \vec{g}[V]$, тогда $h_j = \vec{p}_j + t \vec{q}_j + \alpha \vec{w}_j > 0, 0 \leq t \leq 1; 0 \leq \alpha; j$ – номер грани. То есть нужно решить задачу при n неизвестных и m ограничителях. Решая эту систему уравнений, находят все значения t и α , при которых изменяется значение видимости отрезка или части отрезка (число возможных решений при n плоскостях равно $n(n-1)/2$). Учитывая ограничения $0 \leq t \leq 1; 0 \leq \alpha$ – получим $(n+2)(n+3)/2$ возможных решений. Сначала находим для одной пары уравнений, а затем подставляем в остальные.

Затем определяют минимальное значение параметра t . Отрезок невидим при $t_{\min} < t < t_{\max}$



$$[V] = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

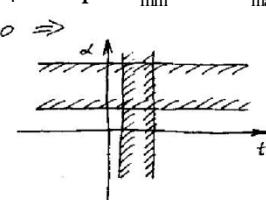
$$\vec{d} = (4000); \quad S = (-20-21)$$

$$P = \vec{s} \cdot [V] = (-1311-13)$$

$$W = \vec{g} \cdot [V] = (00001-1)$$

$$\alpha = \vec{d} \cdot [V] = (4-40000)$$

$$h_j = P_j + t q_j + \alpha w_j > 0 \Rightarrow \begin{cases} -1 + 4t > 0 \\ 3 - 4t > 0 \\ 1 > 0 \\ 1 > 0 \\ -1 + \alpha > 0 \\ 3 - \alpha > 0 \end{cases}$$



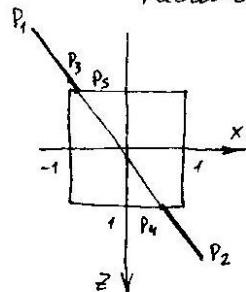
$$P_3 = P\left(\frac{1}{4}\right) = (-20-21) + (4000)\frac{1}{4} = (-10-21,$$

$$P_4 = P\left(\frac{3}{4}\right) = (-20-21) + (4000)\frac{3}{4} = (10-21)$$

$P_3 P_4$ – невидимая часть.

После определения частично видимых или полностью невидимых отрезков определяют пары объектов, связанных отношением протыкания (в случае протыкания объектов сцены ищутся решения на границе $\alpha = 0$), и вычисляют отрезки, которые образуются при протыкании объектами друг друга. Эти отрезки проверяют на скрытие всеми прочими объектами сцены. Видимые отрезки образуют структуру протыкания.

Рассмотрим случай, когда отрезок протыкает тело:

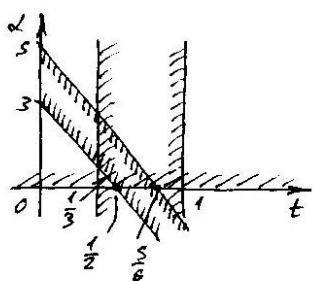


$$d = (3060) \Leftrightarrow P_1(-2-4) \text{ и } P_2(1021)$$

$$S = (-2-4), \rho = \bar{S}[V] = (-1311-35)$$

$$SE \quad q = \bar{d}[V] = (3-3006-6), w = \bar{g}[V] = (00001-1)$$

$$\begin{cases} -1+3t > 0 \\ 3-3t > 0 \\ 1 > 0 \\ 1 > 0 \\ -3+6t + \alpha > 0 \\ S-6t - \alpha > 0 \end{cases} \Rightarrow$$



Мы учли доп. условие $\alpha \leq 0$

$$P_3 = (-2-4) + \frac{1}{3}(3060) = (-10-21)$$

$$P_4 = (-2-4) + \frac{5}{6}(3060) = (0,50-11)$$

Точки протыкания находятся при $\alpha = 0$

$$t_{\text{пр1}} = \frac{1}{2}, \quad t_{\text{пр2}} = \frac{5}{6}.$$

$$P_5 = (-2-4) + \frac{1}{2}(3060) = (-0,50-11), \quad P_6 = P_4.$$

Таким образом, алгоритм подразделяется на следующие этапы.

1. Определение коэффициентов уравнения плоскости каждой грани, проверка правильности знака уравнения и формирование матрицы объекта визуализации.
2. Проведение видового преобразования матрицы объекта, вычисление прямоугольной охватывающей оболочки объекта.
3. Определение нелицевых граней, удаление их из списка граней и соответствующих ребер - из списка ребер.
4. Определение списка других объектов синтезируемой сцены, которые могут быть скрытыми для данного объекта визуализации на основании сравнений охватывающих оболочек объектов.
5. Формирование списка протыканий на основании сравнений охватывающих оболочек объектов.
6. Определение невидимых отрезков или участков отрезков.
7. Формирование списка возможных отрезков, соединяющих точки протыкания, для пар объектов, связанных отношением протыкания.
8. Проверка видимости полученных отрезков по отношению ко всем объектам сцены в соответствии с этапами 3 и 6.
9. Визуализация изображения.

Удаление невидимых линий и поверхностей в пространстве изображений. Алгоритм Варнока (разбиение окнами): последовательность действий и основные принципы. Типы многоугольников, анализируемых в алгоритме Варнока. Методы их идентификации.

Единственной версии алгоритма не существует. Окно разбивается на части, если оно не пусто. Это продолжается, пока не получим окно в 1 пиксель. В более сложных версиях, ставится вопрос о том, что изображается в окне. Окно делим пока нельзя сказать, что изображено в окне.

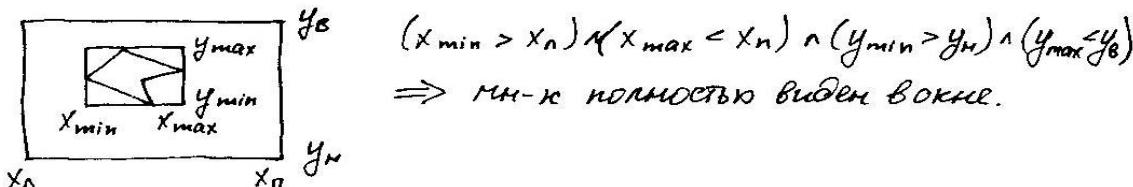
1. Все многоугольники являются внешними – окно закрасить цветом фона
2. Многоугольник, связанный с окном является внутренним – закрасить окно цветом фона, растровая развёртка многоугольника
3. Многоугольник, связанный с окном является пересекающим – отсечение по границе окна и п. 2
4. С окном связан один охватывающий многоугольник – закрасить цветом многоугольника
5. С окном связаны внутренние пересекающие многоугольники, и есть хотя бы один охватывающий, расположенный ближе всех остальных к наблюдателю.

Рекомендуемая последовательность

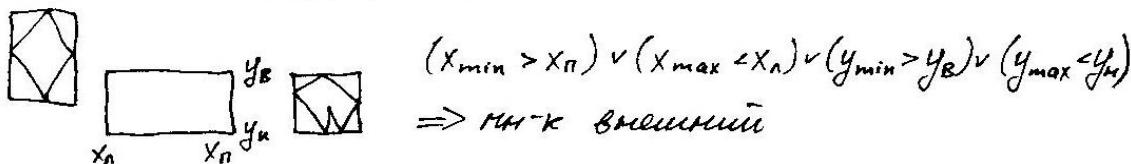
1. Проводится простейший габаритный тест с прямоугольной оболочкой, определяется как можно большее количество пустых окон и окон с единственным внутренним многоугольником
2. Выполнение теста с целью определения окон, пересекаемых единственным многоугольником
3. Тест с целью распознавания внешних и охватывающих многоугольников. Получаем новые пустые окна и окна, охватываемые пустым многоугольником
4. Можно проводить разбиение окна на подокна или проводить ещё тест на обнаружение охватывающего многоугольника, лежащего ближе к наблюдателю

Определение многоугольников

Определение внутреннего мн-ка



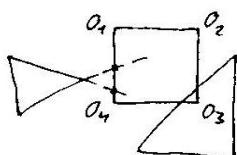
Определение некоторых внешних мн-ков



С помощью этого теста мы не распознаем такой случай:



Определение пересекающих



Составить ур-я прямых, проходящих
через каждое ребро мн-ка:
 $f_{\text{пр}i} = A_i x + B_i y + C_i$

* Нужно еще проверить правильность точки пересечения ребра.

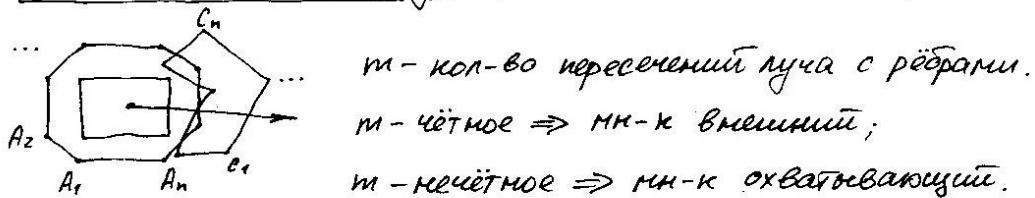
Ур-я прямой, проходящей через две вершины $P_1(x_1, y_1), P_2(x_2, y_2)$: $y = mx + b \Rightarrow f_{\text{пр}} = y - mx - b$, где $m = \frac{y_2 - y_1}{x_2 - x_1}$, $x_2 - x_1 \neq 0$
 $b = y_1 - mx_1$.

$f_{\text{пр}} = x - x_1$, если $x_2 - x_1 = 0$

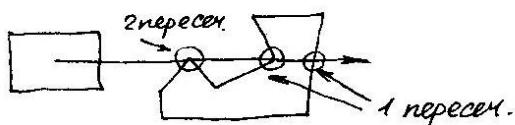
Координаты вершин окна подставим в $f_{\text{пр}i}$. Если знак $f_{\text{пр}i}$ не зависит от выбора вершины окна, то все его вершины лежат по одному направлению от несущей прямой и точек пересечения нет. Если знаки различаются, то мн-к пересекает окно.

Если ни одно из рёбер не пересекает окно, то он либо внешний, либо охватывающий.

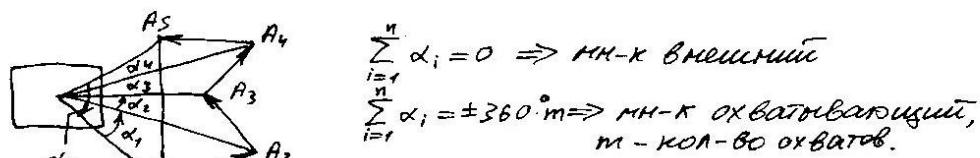
Тест с бесконечным лучом (внеш. или охватывающий.)



Частный случай:

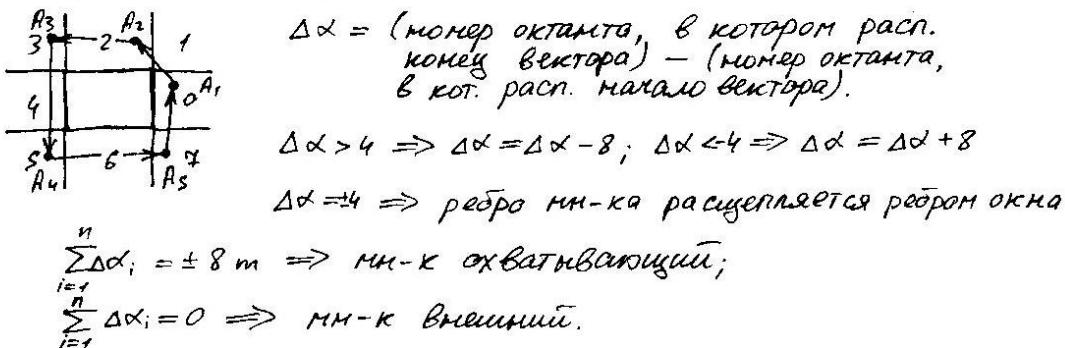


Тест, основанный на вычислении угла (внеш. или охват.)



Сумму $\sum \alpha_i$ считать с учётом направления.

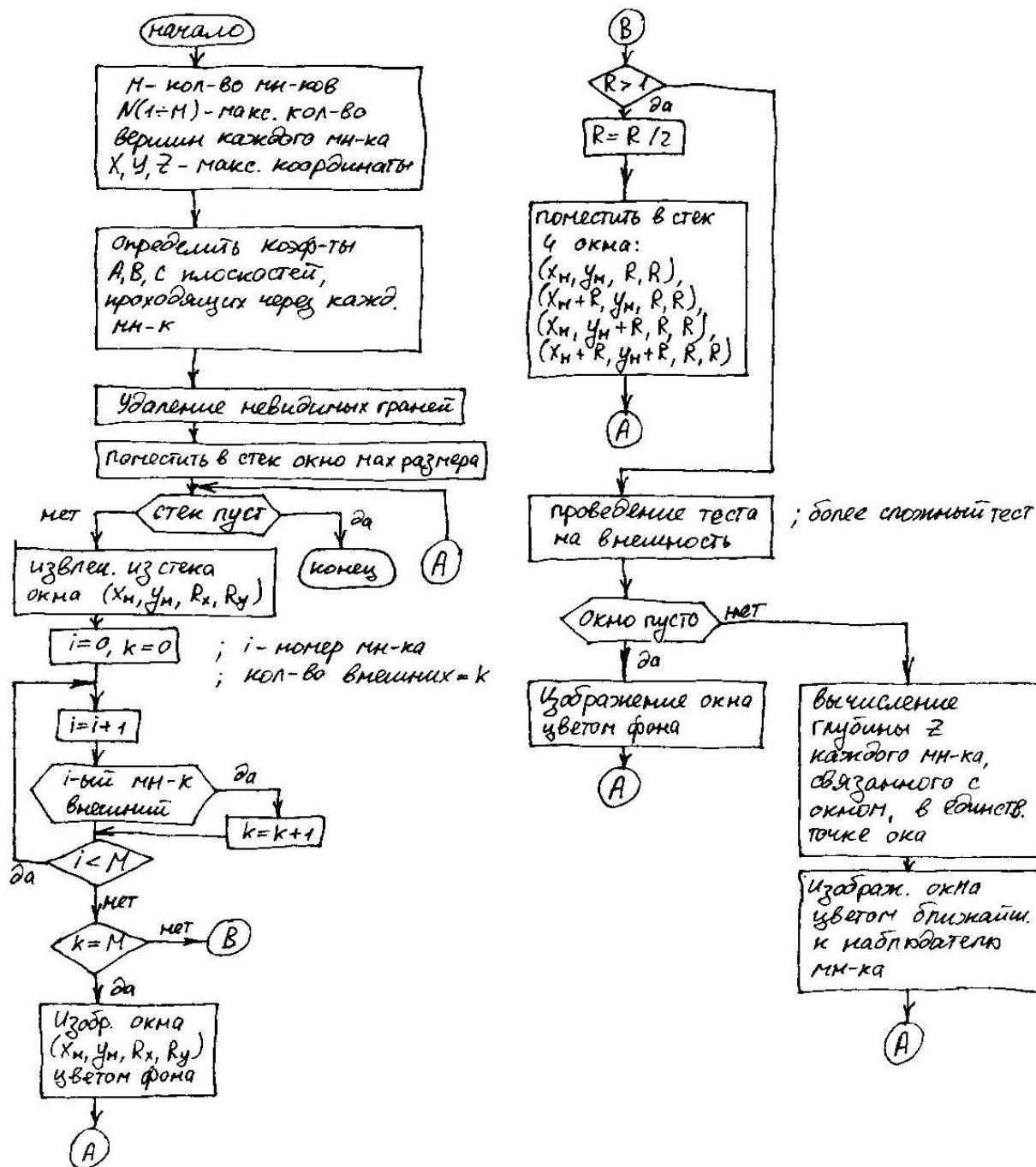
Тест, основанный на октантах (внеш. или охват.)



$Ax + By + Cz + D = 0$ - уравнение плоскости, проходящей через мн-ка.

Если глубина окн. мн-ка во всех 4-х узлах окна больше глубин в мн-ка, то он выреди всех (ближе к набл.)

Алгоритм Варнока



Алгоритм Вейлера-Азертонна удаления невидимых линий и поверхностей.

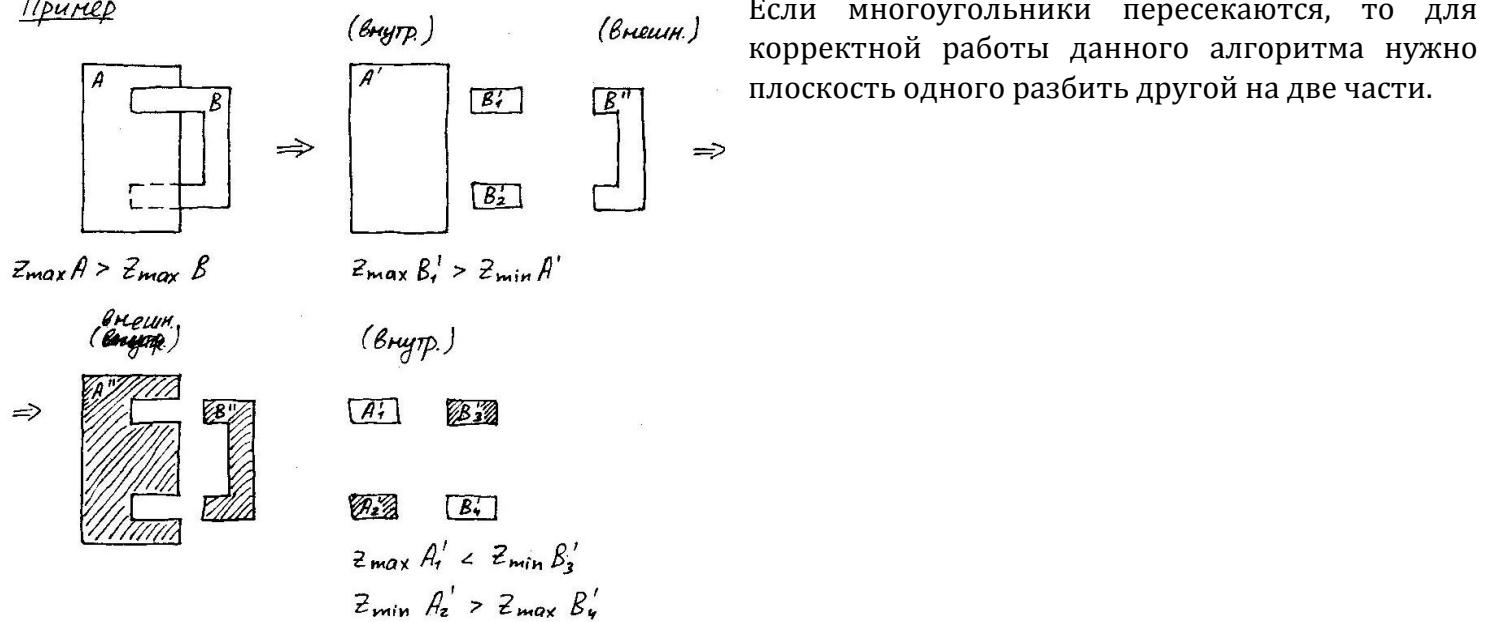
Алгоритм – попытка минимизировать количество шагов в алгоритме Варнока путем разбиения окна вдоль границ многоугольника.

1. Предварительная сортировка по глубине (для формирования списка приблизительных приоритетов)
2. Отсечение по границе ближайшего к наблюдателю многоугольника, называемое сортировкой многоугольников на плоскости (в качестве отсекателя используется копия первого многоугольника из списка приблизительных приоритетов. отсекаться будут все многоугольники в этом списке, включая первый. формируется 2 списка – внутренний (для каждого отсекаемого

многоугольника то часть, которая оказывается внутри отсекателя) и внешний (оставшаяся часть))

3. Удаление многоугольников внутреннего списка, которые экранируются отсекателем
4. Если глубина многоугольника из внутреннего списка больше, чем Z_{min} отсекателя, то такой многоугольник частично экранирует отсекатель. Нужно рекурсивно разделить плоскость, используя многоугольник, нарушивший порядок в качестве отсекателя (нужно использовать копию исходного, а не остаток после предыдущего отсечения). Отсечению подлежат все многоугольники из внутреннего списка
5. По окончанию отсечения или рекурсивного разбиения изображаются многоугольники из внутреннего списка (те, которые остались после удаления всех экранируемых на каждом шаге многоугольников – остаются только отсекающие многоугольники)
6. Работа продолжается с внешним списком (шаги 1-5)

Пример



Алгоритм, использующий Z-буфер.

Алгоритм работает в пространстве изображения.

Буфер кадра (регенерации) используется для заполнения атрибутов (интенсивности) каждого пикселя в пространстве изображения. Для него требуется буфер регенерации, в котором запоминаются значения яркости, а также Z-буфер (буфер глубины), куда можно помещать информацию о координате z для каждого пикселя. Вначале в Z-буфер заносятся максимально возможные значения z, а буфер регенерации заполняется значениями пикселя, описывающими фон. Затем каждый многоугольник преобразуется в растровую форму и записывается в буфер регенерации, при этом, однако, не производится начального упорядочения.

Достоинства

1. Простота. Сцены могут быть любой сложности
2. Элементы сцены не нужно сортировать

Недостатки

1. Большой объём памяти
2. Трудоёмкость устранения лестничного эффекта
3. Трудоёмкость реализации эффектов прозрачности

Вычисление Z

Уравнение плоскости, несущей многоугольник записывают в виде $Ax + By + Cz + D = 0$.

$Z = -(Ax + By + D)/C$ при $C \neq 0$. Если $C = 0$, то плоскость многоугольника параллельна оси Z . (Для наблюдателя такой многоугольник вырождается в линии).

Для сканирующей строки $y = const$, глубина пикселя, для которого $x_1 = x + \Delta x$, равна:
$$z_1 = \frac{Ax + By + D}{C} - \frac{A\Delta x}{C} = z - \frac{A}{C}\Delta x.$$
 Поскольку $\Delta x = 1$, то $z_1 = z - A/C.$

Алгоритм

1. Заполнение буфера кадра фоновый значением интенсивности (цвета).
2. Заполнение Z - буфера минимальным значением Z .
3. Преобразование каждого многоугольника в растровую форму в произвольном порядке.
4. Вычисление для каждого пикселя с координатами (x, y), принадлежащего многоугольнику, его глубины $Z(x, y)$.
5. Сравнение глубины $Z(x, y)$ со значением $Z_{\delta y \phi}(x, y)$, хранящимся в Z -буфере для пикселя теми же координатами(x, y). Если $Z(x, y) > Z_{\delta y \phi}(x, y)$, то записать атрибут очередного многоугольника в буфер кадра и $Z_{\delta y \phi}(x, y)$ заменить на значение $Z(x, y)$.

Замечание

Алгоритм, использующий Z -буфер, можно применять для построения разрезов поверхностей. В этом случае изменяется только операция сравнения глубины пикселя со значением, занесенным в буфер: $[Z(x, y) > Z_{\delta y \phi}(x, y)]$ и $(Z(x, y) \leq Z_{разр})$, где $Z_{разр}$ - глубина искомого разреза.

Алгоритм, использующий список приоритетов (алгоритм Художника).

В основе алгоритма способ изображения сцены от дальних объектов к ближним.

Алгоритм

1. Отсортировать многоугольники сцен в кадре в порядке возрастания Z
2. Построить самый дальний многоугольник, если он не экранирует другие другие многоугольники ($Z_{max}(A) < Z_{min}(B)$)
3. Проверить, экранирует ли многоугольник P многоугольник Q при $Z_{max}(P) > Z_{min}(Q)$. Если на один из тестов даётся + ответ, P заносится в буфер кадра (P не экранирует Q). Если все -, то P и Q меняются местами, позиция Q помечается, тесты повторяются снова. Если вновь попытка менять местами, то P разрезается плоскостью, несущей Q, исходный многоугольник удаляется из списка, а его части заносятся в список. Тесты повторяются снова.
 1. верно ли, что прямоугольные объемлющие оболочки P и Q не перекрываются по X? по Y?
 2. верно ли, что P целиком лежит по ту сторону плоскости, несущей Q, которая расположена дальше от точки наблюдения
 3. верно ли, что Q целиком лежит по ту сторону плоскости, несущей P, которая расположена ближе к точке наблюдения
 4. верно ли, что проекции P и Q не перекрываются

Для 3,4 можно проверять функцией $f = Ax + By + Cz + D$, где A, B, C – коэффициенты пробной плоскости U. В f подставляются координаты вершин испытуемого многоугольника W. Если знаки f для вершин совпадают и + или =0, то W находится с дальней стороны от плоскости U. Если знаки совпадают и отрицательны или равны 0, то W расположен с ближней стороны от U.

Алгоритм построчного сканирования, использующий Z-буфер. Интервальные методы построчного сканирования (основные предпосылки).

Алгоритмы Варнока, Z-буфера и строящего список приоритетов обрабатывают элементы сцены в порядке, который не связан с процессом визуализации. Алгоритмы построчного сканирования обрабатывают сцену в порядке прохождения сканирующей строки.

Алгоритм

1. Подготовка информации

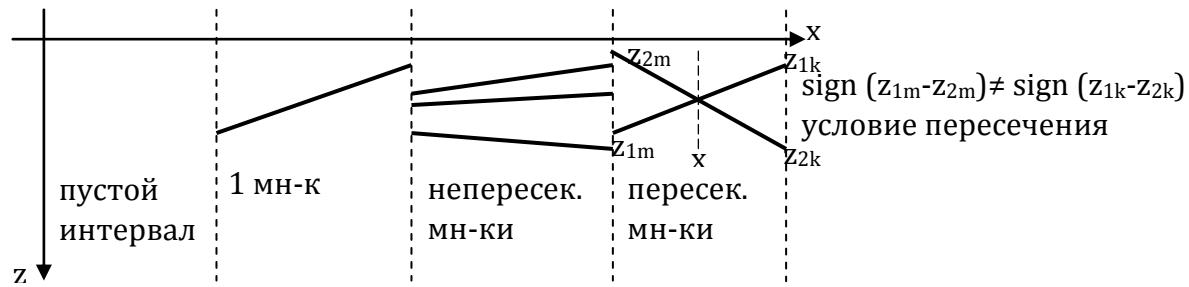
- a) Для каждого многоугольника определить самую верхнюю сканирующую строку, которую он пересекает.
- b) Занести многоугольник в группу у, соответствующую этой сканирующей строке
- c) Запомнить для каждого многоугольника: Δu – число строк, пересекающих этот многоугольник, список рёбер многоугольника, коэффициенты A, B, C, D уравнения плоскости многоугольника, визуальные атрибуты многоугольника

2. Решение задачи удаления невидимых поверхностей

- a) Инициализировать буфер кадра дисплея
- b) Для каждой сканирующей строки
 - 1) Инициализировать буфер кадра размером с одну сканирующую строку, заполнив его фоновым изображением
 - 2) Инициализировать Z-буфер размером с одну сканирующую строку значением Z_{min}
 - 3) Проверить необходимость добавления в список активных многоугольников (САМ) новых многоугольников
 - 4) Если было добавление многоугольника в САМ, то добавить в САР соответствующие рёбра новых многоугольников
 - 5) Если произведено удаление какого-либо элемента из пары рёбер САР, то проверить необходимость удаления всего многоугольника из САМ. Если он остаётся, то проверить необходимость удаления другого ребра из этой пары – если его удалять не нужно, то доукомплектовать пару (добавив недостающее левое или правое ребро)
- c) В САР должна храниться следующая информация:
 - 1) X_l – пересечение левого ребра с текущей сканирующей строкой
 - 2) ΔX_l – приращение X_l в интервале между соседними сканирующими строками
 - 3) ΔY_l – число сканирующих строк, пересекаемых левым ребром
 - 4) $X_p, \Delta X_p, \Delta Y_p$
 - 5) $\Delta Z_x = -A/C$ для $C \neq 0$ (иначе $\Delta Z_x = 0$) – приращение по Z вдоль сканирующей строке
 - 6) $\Delta Z_y = -B/C$ для $C \neq 0$ (иначе $\Delta Z_y = 0$) – приращение по Z в интервале между соседними сканирующими строками
- d) Для каждой пары ребёр многоугольника из САР выполнить:
 - 1) Извлечь
 - 2) Инициализировать Z значением Z_l
 - 3) Для каждого пикселя $X_l \leq X \leq X_p$ вычислить $Z(x, y = \text{const})$. $Z_1 = Z_l, \dots, Z_k = Z_{k-1} + \Delta Z_x$
 - 4) Если $Z() > Z_{\text{буй}}()$, то $Z_{\text{буй}} = Z$ и занести атрибуты многоугольника в буфер кадра
- e) Записать буфер кадра сканирующей строки в буфер кадра дисплея
- f) Скорректировать САР
 - 1) $\Delta Y_l--, \Delta Y_p--$
 - 2) $X_l = X_l + \Delta X_l, X_p = X_p + \Delta X_p$
 - 3) $Z_l = Z_l + \Delta Z_x \Delta X + \Delta Z_y$
 - 4) Если $\Delta Y_l < 0$ или $\Delta Y_p < 0$, то удалить соответствующее ребро из списка, пометив положение обоих рёбер в списке и породивший их многоугольник
- g) Скорректировать САМ
 - 1) $\Delta Y--$
 - 2) Если $\Delta Y < 0$, то удалить многоугольник из САМ

Интервальные методы построчного сканирования

В алгоритме построчного сканирования с использованием Z-буфера глубина многоугольника вычисляется для каждого пикселя на сканирующей строке. Количество вычислений можно сократить, если использовать понятие интервалов. Решение задачи удаления невидимых поверхностей сводится к выбору видимых отрезков в каждом интервале, полученном путём деления сканирующей строки проекциями точек пересечения ребёр



1. Изобразить фон
2. Изобразить атрибуты многоугольника, соответствующие этому отрезку
3. Изобразить атрибуты многоугольника, соответствующие отрезку с MAX Z
4. $\text{sign}(z_{1m}-z_{2m}) \neq \text{sign}(z_{1k}-z_{2k})$ – разбить интервал точкой пересечения

Алгоритм определения видимых поверхностей путем трассировки лучей.

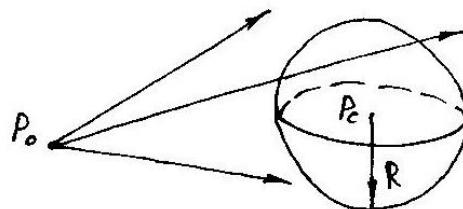
Наблюдатель видит объект посредством испускаемого неким источником света, который падает на поверхность объекта и затем как-то доходит до наблюдателя. В алгоритме отслеживаются лучи в обратном направлении: от наблюдателя к объекту.

Тест со сферической оболочкой:

(x_1, y_1, z_1) — координаты очередного пикселя.

Уравнение луча:

$$\begin{cases} x(t) = x_0 + (x_1 - x_0)t = x_0 + at \\ y(t) = y_0 + (y_1 - y_0)t = y_0 + bt \\ z(t) = z_0 + (z_1 - z_0)t = z_0 + ct \end{cases}$$



Минимальное расстояние l от луча до P_c :

$$l^2 = (x_0 + at - x_c)^2 + (y_0 + bt - y_c)^2 + (z_0 + ct - z_c)^2$$

$$\frac{\partial}{\partial t} (l^2) = 2(a(x_0 + at - x_c) + b(y_0 + bt - y_c) + c(z_0 + ct - z_c))$$

$$t_{min} = - (a(x_0 - x_c) + b(y_0 - y_c) + c(z_0 - z_c)) \cdot (a^2 + b^2 + c^2)^{-1}$$

$l^2(t_{min}) > R^2 \Rightarrow$ пересечения с оболочкой нет.

Тест с прямоугольной оболочкой (труднее):

Применить преобразование, которое совместит луч с осью Z :

$(x'_{min} < 0) \wedge (x'_{max} > 0) \wedge (y'_{min} < 0) \wedge (y'_{max} > 0) \Rightarrow$ пересечение есть.

Упрощение вычисления пересечения:

$$Q(x, y, z) = a_1 x^2 + a_2 y^2 + a_3 z^2 + b_1 yz + b_2 xz + b_3 xy + c_1 x + c_2 y + c_3 z + d = 0$$

Применить преобразование, которое совместит луч с осью Z :

$$x = a, y = 0, z = (-2c_3 \pm \sqrt{c_3'^2 - 4a_3'd}) \cdot (2a_3')^{-1}$$

$(c_3'^2 - 4a_3'd) < 0 \Rightarrow$ пересечения нет.

Для получения т. пересечения в исходной СК применить обратное преобразование для (x, y, z) .

Алгоритм:

1. Подготовка данных:

Создать список объектов и для каждого запомнить:

- а) коэф-ты уравнений поверхностей;
- б) сферич. оболочку и прямоуг. оболочку.

2. Для каждого луча выполнить:

2.1) Для каждого тела выполнить тест со сферич. оболочкой.

Пересечение луча и оболочки есть \Rightarrow тело в список (CAT).

2.2) CAT пуст \Rightarrow изобразить данный пиксель цветом фона.

CAT не пуст \Rightarrow найти, запомнить преобразование, которое совместит луч с осью Z.

2.3) Для каждого объекта из CAT выполнить:

2.3.1) Выполнить тест с прямоуг. оболочкой тела.

2.3.2) Пересечения луча и оболочки нет \Rightarrow перейти к ф. объекту.

Пересечение луча и оболочки есть \Rightarrow преобразовать тело, используя запомненное преобразование; найти в новой СК пересечение с лучом; Если пересечение есть \Rightarrow в список (СП).

2.4) Вывод:

2.4.1) СП пуст \Rightarrow изобразить данный пиксель цветом фона.

2.4.2) СП не пуст \Rightarrow найти пересечение с MAX Z.

2.4.3) Освещение не используется \Rightarrow изобразить данный пиксель цветом поверхности, соответствующей этому пересечению.

2.4.4) Освещение используется \Rightarrow выполнить для точки пересечения преобразование, обратное запомненному; вычислить интенсивность для данного пикселя в полученной точке.

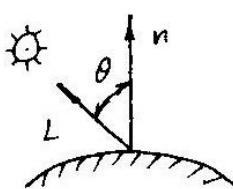
Построение реалистических изображений. Физические и психологические факторы, учитываемые при создании реалистичных изображений. Простая модель освещения.

При построении реалистического изображения необходимо:

- 1) Учитывать оптические свойства поверхности.
- 2) Воспроизводить рисунок на поверхности.
- 3) Воспроизводить неровности.
- 4) Учитывать, что поверхности отбрасывают тени.
- 5) Учитывать восприятие окружающего мира человеком.

Простая модель освещения

Отраженный от объекта свет может быть диффузным и зеркальным.



Диффузное отражение света происходит, когда свет как бы проникает под поверхность объекта, поглощается, а затем вновь испускается. Диффузно отраженный свет рассеивается равномерно во всем направлении, значит, положение наблюдателя не имеет значения.

$$I = I_l \cdot k_d \cdot \cos\theta, 0 \leq \theta \leq \frac{\pi}{2}$$

I - интенсивность отраженного света; I_l - интенсивность точечного источника; k_d - коэф-т диффузного отражения ($0 \leq k_d \leq 1$); θ - угол между направлением света и нормалью к поверхности. k_d зависит от материала и длины волны света, но в простых моделях освещения считается постоянным.

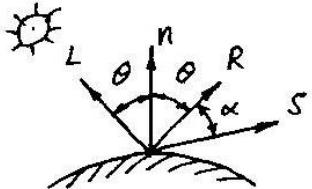
На объекты реальных сцен падает еще и рассеянный свет:

$$I = I_a \cdot k_a + I_l \cdot k_d \cdot \cos\theta$$

I_a - интенсивность рассеянного света; k_a - коэф-т диффузного отражения рассеянного света ($0 \leq k_a \leq 1$).

Опытным путем: $I = I_a \cdot k_a + I_l \cdot k_d \cdot \cos\theta / (d + K)$

d - расстояние от центра проекции до объекта; K - константа.



Интенсивность зеркального отражения
света зависит от угла падения; длины
волны падающего света; свойств вещества.

Зеркальное отражение света является направленным. Угол отражения от идеальной отражающей поверхности (зеркала) равен углу падения, в любом другом положении наблюдатель не видит зеркально отраженного света. Это означает, что $\alpha = 0$ и $\vec{S} = \vec{R}$ (S - вектор наблюдения, R - вектор отражения).

Благодаря зеркальному отражению на блестящих предметах появляются блики. У гладких поверхностей распределение зеркального отраженного света узкое или сформулированное, у шероховатых - более широкое.

В простых моделях освещения пользуются формулой:

$$I_s = I_i \cdot w(i, \lambda) \cdot \cos^n \alpha$$

I_s - интенсивность отраженного зеркального света;
 $w(i, \lambda)$ - кривая отражения, представляющая отношение зеркально отраженного света к падающему, как функцию ^{угла} падения i и длины λ ; n - степень, аппроксимирующая пространственное распределение зеркально отраженного света.

$w(i, \lambda)$ - можноная ф-ция \Rightarrow заменяется константой k_s .

Ф-ция закраски, применяемая для расчёта интенсивности
или тона точек объекта или пикселов изображения:

$$I = I_a \cdot k_a + \frac{I_l}{d+k} (k_d \cdot \cos \theta + k_s \cdot \cos^n \alpha)$$

$$I = I_a \cdot k_a + \sum I_{lj} (k_d \cdot \cos \theta_j + k_s \cdot \cos^{n_j} \alpha_j) / (d+k) - \text{для многих источников}$$

$$\cos \theta = \frac{n \cdot L}{|n| \cdot |L|} = \hat{n} \cdot \hat{L} \quad \text{и} \quad \cos \alpha = \frac{R \cdot S}{|R| \cdot |S|} = \hat{R} \cdot \hat{S}$$

$$I = I_a \cdot k_a + \frac{I_l}{K+d} (k_d \cdot (\hat{n} \cdot \hat{L}) + k_s \cdot (\hat{R} \cdot \hat{S})^n)$$

Построение реалистических изображений. Метод Гуро закраски поверхностей (получение сглаженного изображения).

Методы закраски:

- 1) простая (на граних резкий переход оттенков);
- 2) по Гуро (сглаживание на основе интерполяции интенсивности);
- 3) по Фонту (сглаживание на основе интерполяции нормалей).

Однотонной закраской можно пользоваться при выполнении трёх условий:

- 1) источник света находится в бесконечности $\Rightarrow \bar{n} \cdot \bar{L} = \text{const}$;
- 2) наблюдатель находится в бесконечности $\Rightarrow \bar{R} \cdot \bar{S} = \text{const}$;
- 3) заштрихованный мн-к является реально существующим мн-ком, а не результатом аппроксимации поверхности.

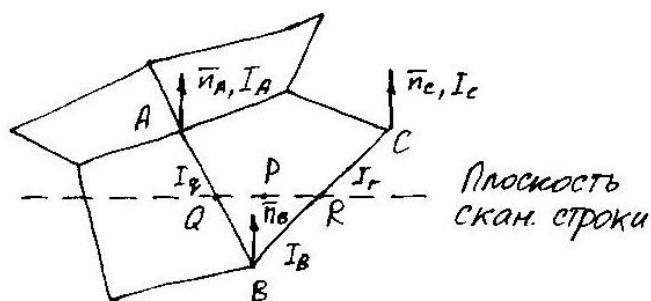
Билинейная интерполяция интенсивностей - это определение значений ф-ции в точке, лежащей внутри какого-то интервала.

Экстраполяция - для точки вне интервала.

Метод Гуро

Если при построении полигональной поверхности для каждой грани используется одна нормаль, то модель освещения создаёт изображение, состоящее из отдельных мн-ков.

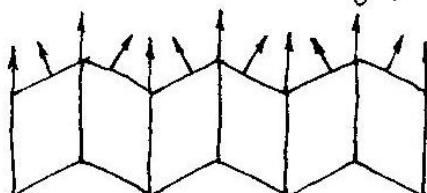
Методом Гуро можно получить слаженное изображение.



1. Вычисление векторов нормалей к каждой грани.
 2. Вычисление векторов нормалей к каждой ^{вершине} грани (путём усреднения нормалей к граням).
 3. Вычисление интенсивностей в вершинах граней.
 4. Интерполяция интенсивности вдоль рёбер граней:
 $I_q = u I_A + (1-u) I_B, 0 \leq u \leq 1, u = \frac{AQ}{AB};$
 $I_r = w I_B + (1-w) I_C, 0 \leq w \leq 1, w = \frac{BR}{BC}.$
 5. Линейная интерполяция интенсивности вдоль скан. строк:
- $$I_p = t I_q + (1-t) I_r, 0 \leq t \leq 1, t = \frac{QP}{QR}.$$

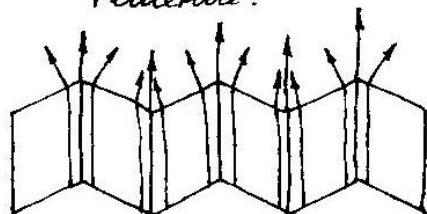
Закраска по Гуро хорошо сочетается с диффузными отражениями. Данный метод интерполяции обеспечивает лишь непрерывность значений интенсивности вдоль граней мн-ков, но не обеспечивает непрерывности изменения интенсивности. Значит, возможно проявление полос Маха.

Недостаток: усреднение нормалей.



Поверхность закрашивается с одной интенсивностью. Будет выглядеть плоской.

Решение:



Если нужно сохранить острый переход, то не делается усреднение.

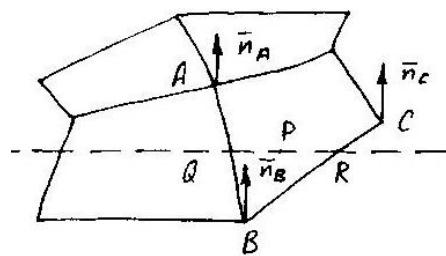
Если нужно, чтобы ребро сгладилось, вводят дополнительные мн-ки.

Инкрементальное вычисление вдоль скан. строки:

$$I_{P_2} = t_2 I_q + (1-t_2) I_r \text{ и } I_{P_1} = t_1 I_q + (1-t_1) I_r \Rightarrow \\ \Rightarrow I_{P_2} = I_{P_1} + (I_q - I_r)(t_2 - t_1) = I_{P_1} + \Delta I \cdot \Delta t.$$

Построение реалистических изображений. Закраска Фонга (улучшение аппроксимации кривизны поверхности).

Закраска Фонга требует больших вычислительных затрат, однако она позволяет решить многие проблемы метода Гуро. При закраске Гуро вдоль скан. строки интерполируется значение интенсивности, а при закраске Фонга — вектор нормали. Затем он используется в модели освещения для вычисления интенсивности. При этом достигается лучшая локальная аппроксимация кривизны поверхности. Изображение выходит более реалистичным. В частности, правдоподобнее выглядят зеркальные блики.



Трудность метода: (при создании кинокадра) закраска может значительно меняться от кадра к кадру (правило закраски зависит от ориентации объекта).

1. Вычисление векторов нормалей к каждой грани.
2. Вычисление векторов нормалей к каждой вершине грани.
3. Интерполяция векторов нормалей вдоль рёбер грани:

$$n_q = u \cdot n_A + (1-u) \cdot n_B, \quad 0 \leq u \leq 1; \quad u = \frac{AQ}{AB};$$

$$n_r = w \cdot n_B + (1-w) \cdot n_C, \quad 0 \leq w \leq 1; \quad w = \frac{BR}{BC}.$$

4. Линейная интерполяция векторов нормалей вдоль скан. строки:

$$n_p = t \cdot n_q + (1-t) \cdot n_r, \quad 0 \leq t \leq 1; \quad t = \frac{QP}{QR}.$$

5. Вычисление интенсивности в очередной точке скан. строки.

Инкрементальное вычисление вдоль скан. строки:

$$\bar{n}_{p_2} = \bar{n}_{p_1} + (\bar{n}_q - \bar{n}_r)(t_2 - t_1) = \bar{n}_{p_1} + \Delta \bar{n} \cdot \Delta t.$$

$$t_2 - t_1 = \frac{QP_2}{QR} - \frac{QP_1}{QR} = \frac{\Delta QR}{QR} = \frac{1}{QR} \Rightarrow \bar{n}_{p_2} = \bar{n}_{p_1} + \frac{\Delta \bar{n}}{RQ}$$

Определение нормали к поверхности и вектора отражения в алгоритмах построения реалистических изображений. Определение направления отраженного и преломленного лучей.

Определение нормали к поверхности зависит от формы представления поверхности, если поверхность задана аналитически уравнение вида $F(x, y, z) = 0$, то нормаль можно вычислить как градиент функции в заданной точке: $\nabla F(x, y, z)$.

Для полигональных поверхностей нормаль совпадает с нормалью соответствующего полигона и вычисляется как векторное произведение двух его ребер.

Направление отражённого и преломлённого лучей соответственно:

$$\begin{aligned}\vec{r}' &= \vec{i} - 2 \cos \theta_i \vec{n} \\ \vec{t}' &= \frac{\eta_1}{\eta_2} \vec{i} - \left(\frac{\eta_1}{\eta_2} \cos \theta_i + \sqrt{1 - \sin^2 \theta_t} \right) \vec{n}\end{aligned}$$

где

- i – направление падающего луча
- n – нормаль к поверхности в точке падения луча
- η – абсолютные показатели преломления двух сред (1- из которой падает луч, 2 – в которую попадает после преломления)

$$\cos \theta_i = \vec{i} \cdot \vec{n}$$

$$\sin^2 \theta_t = \left(\frac{\eta_1}{\eta_2} \right)^2 (1 - \cos^2 \theta_i)$$

•

Если $\sin^2 \theta_t \leq 1$ то имеет место полное внутреннее отражение.

Построение теней при создании реалистических изображений. Учет теней в алгоритмах удаления невидимых поверхностей.

Если положение наблюдателя и источника света совпадают, то теней не видно, но они появляются при перемещении в любую другую точку. Тень состоит из 2 частей – **полная тень** (центральная, тёмная, резко очернённая часть) и **полутень** (окружающая полную более светлая часть).

Собственная тень – объект, препятствует попаданию тени на некоторые свои грани. **Проекционная тень** – один объект препятствует попаданию света на другой.

Процесс построения тени

1. Удалить невидимые поверхности для положения каждого источника
2. Удалить невидимые поверхности для положения наблюдателя

Построение собственной тени

Удаление неблицевых граней, если точку наблюдения совместить с источником света (алгоритм Робертса)

Построение проекционной тени

Построить проекции всех неблицевых граней на сцену, если точку наблюдения совместить с источником света. Точки пересечения проецируемой грани со всеми другими плоскостями образуют многоугольники, которые помечают как теневые и заносят в список.

Интервальный метод построчного сканирования

1. Для каждого источника и многоугольника определить собственные тени и проекционные тени. Они записываются в виде двоичной матрицы – строки – многоугольники, отбрасывающие тень, столбцы – затеняемые многоугольники
2. Обработка сцены относительно положения наблюдателя
 1. Определить видимые отрезки на интервале
 2. С помощью списка теневых многоугольников определить, падает ли тень на многоугольник, которые создаёт видимый отрезок на данном интервале
 - нет теневых многоугольников – изобразить отрезок
 - есть теневые многоугольники, но они не пересекают и не покрывают данный интервал – изобразить видимый отрезок

- c) интервал полностью покрывается хотя бы одним многоугольником – интенсивность изображаемого отрезка определяется с учётом интенсивностей этих многоугольников и самого отрезка
- d) хотя бы один теневой многоугольник частично покрывает интервал – рекурсивно разбивать интервал и работать с подынтервалами

Алгоритм, использующий Z-буфер

1. Строится сцена, где точка наблюдения совпадает с источником. Запомнить значения Z в отдельном теневом Z-буфере
2. Строится сцена из точки наблюдателя. При обработке каждой поверхности глубина каждого пикселя сравнивается с глубиной Zбуф. Если $>$, то xyz из вида наблюдателя преобразуются в x'y'z' на виде из источника
 1. $Z' \geq Z_{\text{тенбум}} \rightarrow xyz$ в буфер кадра
 2. $< - (xyz)$ в тени и изображается согласно правилу рассчёта интенсивности с учётом затенения. $Z_{\text{буф}} = Z$

Учет прозрачности в модели освещения. Учет прозрачности в алгоритмах удаления невидимых поверхностей.

Чтобы включить преломление в модель освещения нужно при построении видимых поверхностей учитывать не только падающий, но и отражённый и пропущенный свет. Эффективнее это делается с помощью глобальной модели освещения в сочетании с алгоритмом трассировки лучей для выделения видимых поверхностей. Обычно рассматриваются только зеркально отражённые и пропущенные лучи, т.к. диффузное отражение от просвечивающих поверхностей порождает бесконечное количество беспорядочно ориентированных лучей.

Формула для расчета интенсивности для прозрачных тел

$$I = k_a I_a + k_d I_d + k_s I_s + k_t I_t \quad (a, d, s, t - \text{рассеянный, диффузный, зеркальный и пропущенный свет})$$

Пропускание

1. Зеркальное (направленное) – свойственно прозрачным телам. Если смотреть на объект сквозь такое вещество, то за исключением контурных линий криволинейных поверхностей, искажения происходить не будет. Нелинейная аппроксимация коэффициента прозрачности: $t = t_{\min} + (t_{\max} - t_{\min})(1 - (1 - |n_z|)^p)$. t_{\min} , t_{\max} – макс и мин прозрачность объекта, n_z - z-составляющая единичной нормали к поверхности, p – коэффициент прозрачности, t – прозрачность пикселя
2. Диффузное (рассеянное) – вещества кажутся полупрозрачными или матовыми

Простое пропускание света можно встроить в любой алгоритм удаления невидимых поверхностей, кроме z-буфера. Прозрачные поверхности помечаются и если видимая грань прозрачна, то в буфер кадра заносится линейная комбинация двух ближайших поверхностей $I = tI_1 + (1-t)I_2$; 1-видимая поверхность, 2 – за нею

Алгоритм с z-буфером

1. Для каждого многоугольника
 1. Если прозрачен – занести в список прозрачных
 2. Если нет и $Z > Z_{\text{буф}}$, то записать в буфер кадра для непрозрачных многоугольников и скорректировать Zбуф
2. Для каждого многоугольника из списка прозрачных
 1. Если $Z \geq Z_{\text{буф}}$, то прибавить его r к значению в буфере весовых коэффициентов прозрачности, прибавить его I к значению в буфере интенсивности прозрачности по правилу: $I_{\text{вн}} = I_{\text{вн}} t_{\text{бо}} + I_{\text{с}} t_{\text{с}}$ (вн – новое, бо – старое, с – для многоугольника)
3. Объединим буфера интенсивности для прозрачных и непрозрачных многоугольников $I = t_{\text{бо}} I_{\text{бо}} + (1-t_{\text{бо}}) I_{\text{фбо}}$ ($I_{\text{фбо}}$ – старое значение интенсивности в этом буфере)

Учет фактуры при создании реалистических изображений.

Под фактурой, или текстурой, в компьютерной графике понимается детализация строения поверхности. Как правило, рассматривают два вида детализации. В первом случае на гладкую поверхность объекта наносят заданный рисунок (узор). В этом случае поверхность остается гладкой. Во втором случае ставится задача создания неровностей на поверхности, т.е. создания шероховатой поверхности.

Проективные текстуры

Задается в некоторой (обычно двумерной) системе координат рисунок, подлежащий нанесению на трехмерную поверхность. Тогда для наложения заданного рисунка на поверхность необходимо найти функцию отображения или, другими словами, произвести преобразование координат.

Проективным текстурам присущи определенные недостатки. Во-первых, для хранения используемых изображений требуется достаточно большой объем памяти. Во-вторых, они не обладают гибкостью, а, в-третьих, возможны большие сложности при подборе способа проектирования при нанесении рисунка на объекты сложной формы. На практике используют ограниченное количество вариантов проектирования: плоское (параллельное проектирование), цилиндрическое и сферическое.

Процедурные текстуры

Необходимо найти функцию $C(X,Y,Z)$, определяющую для каждой точки поверхности цвет таким образом, чтобы он соответствовал цвету моделируемого материала (наносимого рисунка). Такой подход не требует больших затрат памяти и хорошо работает с поверхностями любой сложности. Но поскольку используемая функция зависит от большого количества параметров, то возникают сложности в подборе этой функции, с другой стороны, изменение этих параметров позволяет легко изменять свойства текстуры.

Неровности

В первом случае можно оцифровать фотографию нерегулярной фактуры и отобразить ее на поверхность. Однако при этом неровности будут восприниматься как нарисованные на гладкой поверхности, т.е. изображение будет не полностью реалистическим. Это связано с тем, что векторе нормали к реальной шероховатой поверхности, а следовательно, в направлении вектора отражения, есть небольшая случайная составляющая. Этот факт и лег в основу способа моделирования неровностей на отображаемой поверхности.

Пусть $Q(X,Y)$ – уравнение поверхности, т.е. функция Q позволяет для каждой точки поверхности определить ее третью координату Z . В произвольной точке поверхности частные производные по направлениям X, Y лежат в плоскости, касательной к поверхности в этой точке. Нормаль в точке поверхности может определяться векторным произведением этих производных $\mathbf{Q}'_x, \mathbf{Q}'_y: \mathbf{N} = \mathbf{Q}'_x \otimes \mathbf{Q}'_y$

Для создания шероховатой поверхности можно создать новую поверхность путем внесения возмущения в направлении нормали в точках поверхности. Пусть $P(X,Y)$ – функция возмущения, тогда радиус-вектор точки на новой поверхности будет определяться из $\mathbf{Q}_n(X,Y) = \mathbf{Q}(X,Y) + P(X,Y) \frac{\mathbf{N}}{|\mathbf{N}|}$ Нормаль к новой возмущенной поверхности будет иметь вид: $\mathbf{N}_n = \mathbf{Q}'_{nx} \otimes \mathbf{Q}'_{ny}$

Частные производные $\mathbf{Q}'_{nx}, \mathbf{Q}'_{ny}$ вычисляются из следующих выражений:

$$\mathbf{Q}'_{nx} = \mathbf{Q}'_x + P'_x(X,Y) \frac{\mathbf{N}}{|\mathbf{N}|} + P(X,Y) (\mathbf{N}/|\mathbf{N}|)_x$$

$$\mathbf{Q}'_{ny} = \mathbf{Q}'_y + P'_y(X,Y) \frac{\mathbf{N}}{|\mathbf{N}|} + P(X,Y) (\mathbf{N}/|\mathbf{N}|)_y$$

Последними слагаемыми можно пренебречь, так как функция возмущения $P(X,Y)$ - очень маленькая величина.

Теперь можно записать выражение для вычисления нормали к возмущенной поверхности:

$$\mathbf{N}_n = \mathbf{Q}'_{nx} \otimes \mathbf{Q}'_{ny} + P'_y(X,Y) (\mathbf{Q}'_x \otimes \mathbf{N}/|\mathbf{N}|) + P'_x(X,Y) (\mathbf{N}/|\mathbf{N}| \otimes \mathbf{Q}'_y) + P'_x(X,Y) P'_y(X,Y) (\mathbf{N} \otimes \mathbf{N}/|\mathbf{N}|^2)$$

Учитывая, что первое слагаемое представляет собой нормаль к исходной поверхности, а последнее слагаемое равно нулю как векторное произведение коллинеарных векторов, окончательно получим:

$$\mathbf{N}_h = \mathbf{N} + P'_y(X, Y) (\mathbf{Q}'_x \otimes \mathbf{N}) / |\mathbf{N}| + P'_x(X, Y) (\mathbf{N} / |\mathbf{N}| \otimes \mathbf{Q}'_y)$$

Для моделирования искажений вводится шумовая функция $R(X, Y, Z)$, которая должна удовлетворять следующим требованиям: 1) она должна быть непрерывной; 2) принимать значения из интервала $[0, 1]$; 3) вести себя аналогично равномерно распределенной случайной величине.

Одним из способов задания такой функции является задание случайных значений в узлах некоторой регулярной сетки (в точках i, j, k , принадлежащих множеству целых чисел) и последующей интерполяции на все остальные точки. Для отыскания значения функции в произвольной точке сначала надо определить параллелепипед, содержащий данную точку внутри себя, затем, используя значения функции в вершинах параллелепипеда и проводя интерполяцию, найти значений функции в исходной точке. Для использования целочисленной решетки может быть предложено следующее задание функции:

$$R(X, Y, Z) = \sum_{i=|x|}^{|x|+1} \sum_{j=|y|}^{|y|+1} \sum_{k=|z|}^{|z|+1} w(|x - i|) w(|y - j|) w(|z - K|) A_{ijk}$$

$W(t)$ – одномерная весовая функция, в простейшем случае $w(t) = t$, $t \in [0, 1]$.

Использование приведенной трилинейной интерполяции дает не очень хорошие результаты, так как на границе параллелепипеда происходит разрыв первых производных, т.е. функция не является гладкой. Для достижения гладкости на функцию следует наложить условие $W(0) = W(1) = 0$. Простейшим вариантом функции, удовлетворяющей этому условию, является многочлен Эрмита: $W(t) = 3t^2 - 2t^3$, $t \in [0, 1]$.

Если функция возмущения не описывается аналитически, ее можно задать двумерной таблицей цветов. Промежуточные значения вычисляются билинейной интерполяцией табличных величин, а производные вычисляются методом конечных разностей.

Глобальная модель освещения. Алгоритм трассировки лучей с использованием глобальной модели освещения.

Алгоритм трассировки лучей может быть расширен использованием глобальной модели освещения для улучшения качества получаемого изображения. Популярным расширением является метод фотонных карт состоящий из трёх этапов:

1) Трассировка фотонов: Фотоны в данном методе — это частицы, переносящие некоторую дискретную порцию световой энергии. На начальном этапе фотоны испускаются из источника света в соответствии с распределением световой энергии у данного источника. Например известно, что точечный или сферический источник света (например, Солнце) испускают свет изотропно во всех направлениях. Площадные источники света имеют косинусоидальное распределение, имеющее максимум по направлению, совпадающему с нормалью к плоскости источника и нуль по направлениям, лежащим в этой плоскости.

В процессе трассировки, фотоны ударяются о различные поверхности модели (сцены). В зависимости от свойств материала, с ними могут происходить разные события: фотон может отразиться рассеянно (то есть в случайному направлении), зеркально, преломиться через поверхность или поглотиться. Решение о том, какое из событий происходит с фотоном, принимается на основании свойств материала поверхности и статистического метода «Русской рулетки». Лишь при рассеянном отражении, запись о фотоне сохраняется в список.

2) Построение фотонной карты. На основе списка фотонов, полученного после этапа трассировки фотонов, производится построение фотонной карты в виде некоторой структуры

пространственного разбиения. Этот этап вспомогательный и служит для того, чтобы на следующем шаге в произвольной точке пространства можно было найти k ближайших фотонов. Если бы фотоны просто оставались в списке, то k ближайших каждый раз пришлось бы вычислять перебором, что часто нецелесообразно. Для хранения фотонов как правило используется специализированный вариант k -мерного дерева (k D-tree).

3) Сбор глобального освещения. Применяется обычная трассировка лучей из камеры, но в местах пересечения лучей с объектами находятся k ближайших фотонов, их энергия суммируется и делится на площадь сферы с радиусом равным расстоянию до самого дальнего из этих фотонов. Полученная величина добавляется к общей энергии собранной лучом