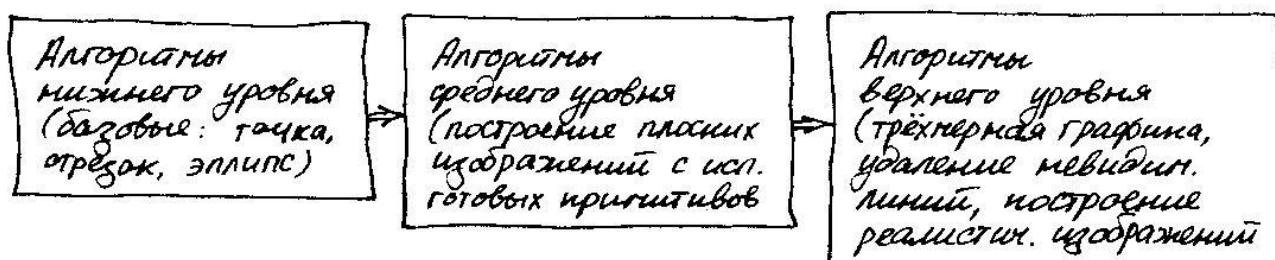


#1 Задача синтеза сложного динамического изображения. Этапы синтеза изображения.

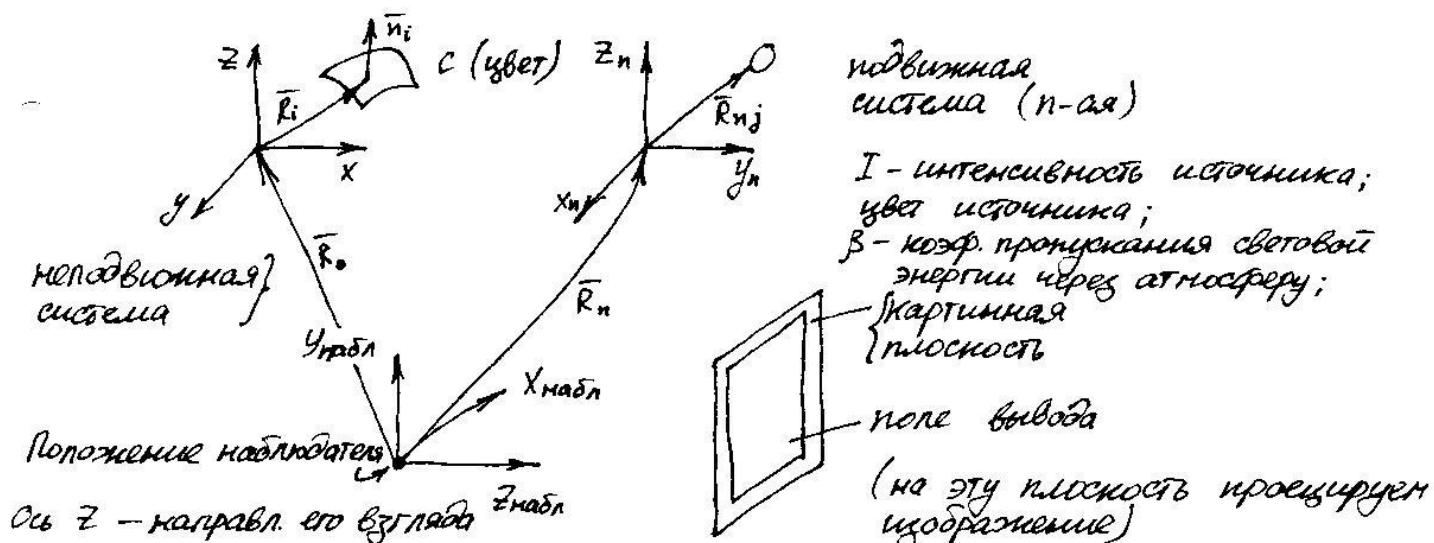
опр Маш.граф. – это совокупность методов и средств преобразования информации в графич. формат и из гр. формы с помощью ЭВМ.

1. Данные графика (графики, диаграммы)
2. Научная графика (чертежи, документация, поддер. технологии)
3. Иллюстративная графика (карты местности, генерация символов и рез-ов операций)
4. Инженерная графика (иллюстрации и тексты)

Три уровня алгоритмов:



Синтез сложного динамического изображения



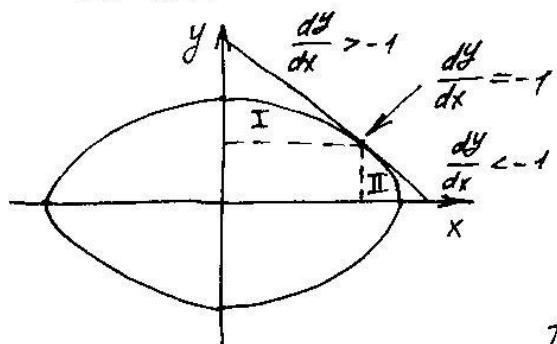
Частота вывода изображения: $25 \div 30$ Гц

Г – интервал вывода одного изображения.

Этапы синтеза

1. Разработка трехмерной математической модели синтезирующей визуальной обстановки.
2. Задание: положения наблюдателя, картинной плоскости (её положение), размеров окна вывода, значений управляющих сигналов.
3. Определение операторов, определяющих пространственное перемещение объектов визуализации.
4. Преобразование координат объектов в координаты наблюдателя.
5. Отсечение объектов видимого пр-ва в пределах пирамиды видимости.
6. Вычисление 2-мерных перспективных проекций синтезированных объектов на картинную плоскость.
7. Исключение из сцены невидимых элементов при заданном положении наблюдателя (удаление невидимых линий и несуществующих поверхностей); закрашивание и затенение объектов сцены.
8. Вывод полутонового изображения на экран растрового дисплея.

#2 Алгоритм построения эллипса и окружности по методу средней точки.



$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

$$f_{np} = b^2x^2 + a^2y^2 - a^2b^2$$

$$f_{np}(x,y) = \begin{cases} = 0, & (x,y) \in \text{эллипс} \\ > 0, & (x,y) \text{ вне эллипса} \\ < 0, & (x,y) \text{ внутри эллипса} \end{cases}$$

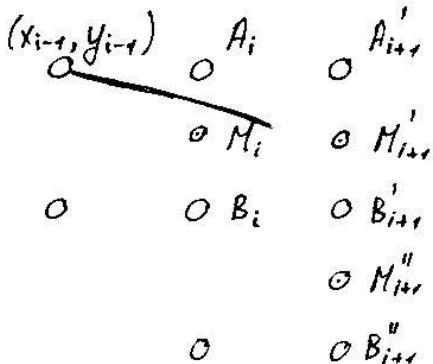
Поиск граничной точки:

$$b^2x^2 + a^2y^2 - a^2b^2 = 0$$

$$2b^2xdx + 2a^2ydy - a^2b^2 = 0$$

$$\frac{a^2ydy}{b^2xdx} = -1 \Rightarrow \frac{dy}{dx} = \frac{-b^2x}{a^2y} \Rightarrow \text{т.к. для граничной точки } \frac{dy}{dx} = -1, \text{ то } b^2x = a^2y \text{ - условие границы.}$$

I. Цикл по x слева направо:



$$f_{np,i} = b^2(x_{i-1} + 1)^2 + a^2(y_{i-1} - \frac{1}{2})^2 - a^2b^2$$

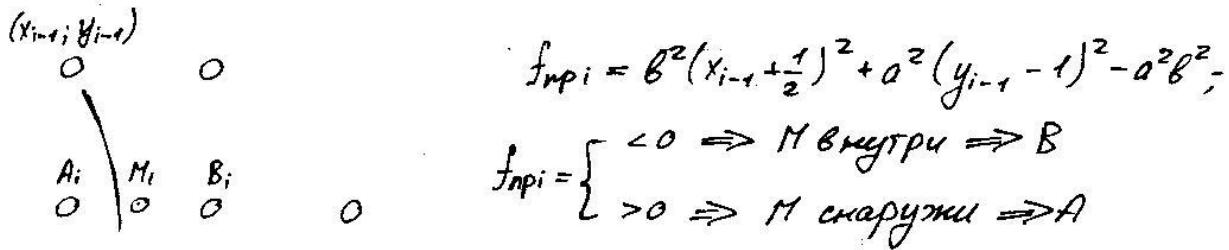
$$f_{np,i} \leftarrow f_{np,i} \quad f_{np,i} = \begin{cases} < 0 \Rightarrow \text{т.м. внутри} \Rightarrow A \\ > 0 \Rightarrow \text{т.м. снаружи} \Rightarrow B \end{cases}$$

$$df = f_i - f_{i-1} = b^2(x_{i-1} + 1)^2 + a^2(y_{i-1} - \frac{1}{2})^2 - a^2b^2 - b^2x_{i-1}^2 - a^2(y_{i-1} - \frac{1}{2})^2 - a^2b^2 = 2b^2x_{i-1} + b^2$$

Если выбрана т.в. ($y_i = y_{i-1} - 1$), то значение пробной ф-ции надо скорректировать, т.к. для случая А пробная точка была бы: $(x_{i-1}, y_{i-1} + \frac{1}{2})$, для случая В: $(x_{i-1}, y_{i-1} - \frac{1}{2})$.

$$\Delta f = b^2(x_{i-1} + 1)^2 + a^2(y_{i-1} - \frac{1}{2})^2 - a^2b^2 - b^2(x_{i-1} + 1)^2 - a^2(y_{i-1} + \frac{1}{2})^2 + a^2b^2 = -2a^2y_{i-1}$$

II. Цикл по Y сверху вниз



$$f_{\text{нр}i} = b^2(x_{i-1} + \frac{1}{2})^2 + a^2(y_{i-1} - 1)^2 - a^2b^2,$$

$$f_{\text{нр}i} = \begin{cases} < 0 \Rightarrow M \text{ внутри} \Rightarrow B \\ > 0 \Rightarrow M \text{ снаружи} \Rightarrow A \end{cases}$$

$$A'_i, M'_i, B'_i, M''_i, B''_i, \dots$$

$$0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0$$

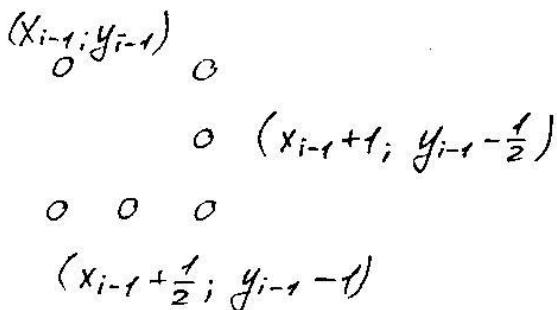
$$\Delta f = f_i - f_{i-1} = b^2(x_{i-1} + \frac{1}{2})^2 + a^2(y_{i-1} - 1)^2 - a^2b^2 - b^2(x_{i-1} + \frac{1}{2})^2 - a^2y_{i-1}^2 + a^2b^2 = -2a^2y_{i-1} + a^2$$

Если восбрался т.в., то проблему нужно надо корректировать, т.к. для случая А проблема точки общая бы:

$$(x_{i-1} - \frac{1}{2}; y_{i-1} - 1), \text{ а для } B: (x_{i-1} + \frac{1}{2}; y_{i-1} - 1).$$

$$\Delta f = b^2(x_{i-1} + \frac{1}{2})^2 + a^2(y_{i-1} - 1)^2 - a^2b^2 - b^2(x_{i-1} - \frac{1}{2}) - a^2(y_{i-1} - 1) + a^2b^2 = 2b^2x_{i-1}$$

Переход от I к II



$$\Delta f = b^2(x_{i-1} + \frac{1}{2})^2 + a^2(y_{i-1} - 1)^2 - a^2b^2 - b^2(x_{i-1} + 1)^2 - a^2(y_{i-1} - \frac{1}{2})^2 + a^2b^2 = b^2(x_{i-1}^2 + x_{i-1} + \frac{1}{4}) + a^2(y_{i-1}^2 - 2y_{i-1} + 1) - b^2(x_{i-1}^2 + 2x_{i-1} + 1) - a^2(y_{i-1}^2 - y_{i-1} + \frac{1}{4}) = b^2(-x_{i-1} - \frac{3}{4}) + a^2(-y_{i-1} + \frac{3}{4}) = \frac{3}{4}(a^2 - b^2) - (b^2x_{i-1} + a^2y_{i-1})$$

Первый шаг

$$i-1: x=0, b=y \Rightarrow i: f_{\text{нр}i} = b^2 + a^2(b - \frac{1}{2})^2 - a^2b^2 = b^2 - a^2b + \frac{a^2}{4}, \dots$$

$$f_i - f_{i-1} = 2b^2x_{i-1} + b^2 \Rightarrow f_i = f_{i-1} + 2b^2x_{i-1} + b^2$$

#3 Требования, предъявляемые к алгоритмам вычерчивания отрезков. Помаговой алгоритм разложения отрезка в растр. Разложение в растр по методу цифрового дифференциального аналита.

Процесс, определяющий пиксели, наилучшим образом аппроксимирующих отрезок, называется разложением отрезка в растр.

Требования:

1. Отрезок должен выглядеть прямым.
2. Отрезок должен начинаться и заканчиваться в заданных точках.
3. Яркость отрезка вдоль его длины должна быть постоянной и не зависеть от угла наклона.
4. Вычерчивание отрезков должно осуществляться быстро.

— пестринный эффект

Практически все алгоритмы разложения отрезков в растр работают в пошаговом режиме (выполняются вычисления для следующего шага).



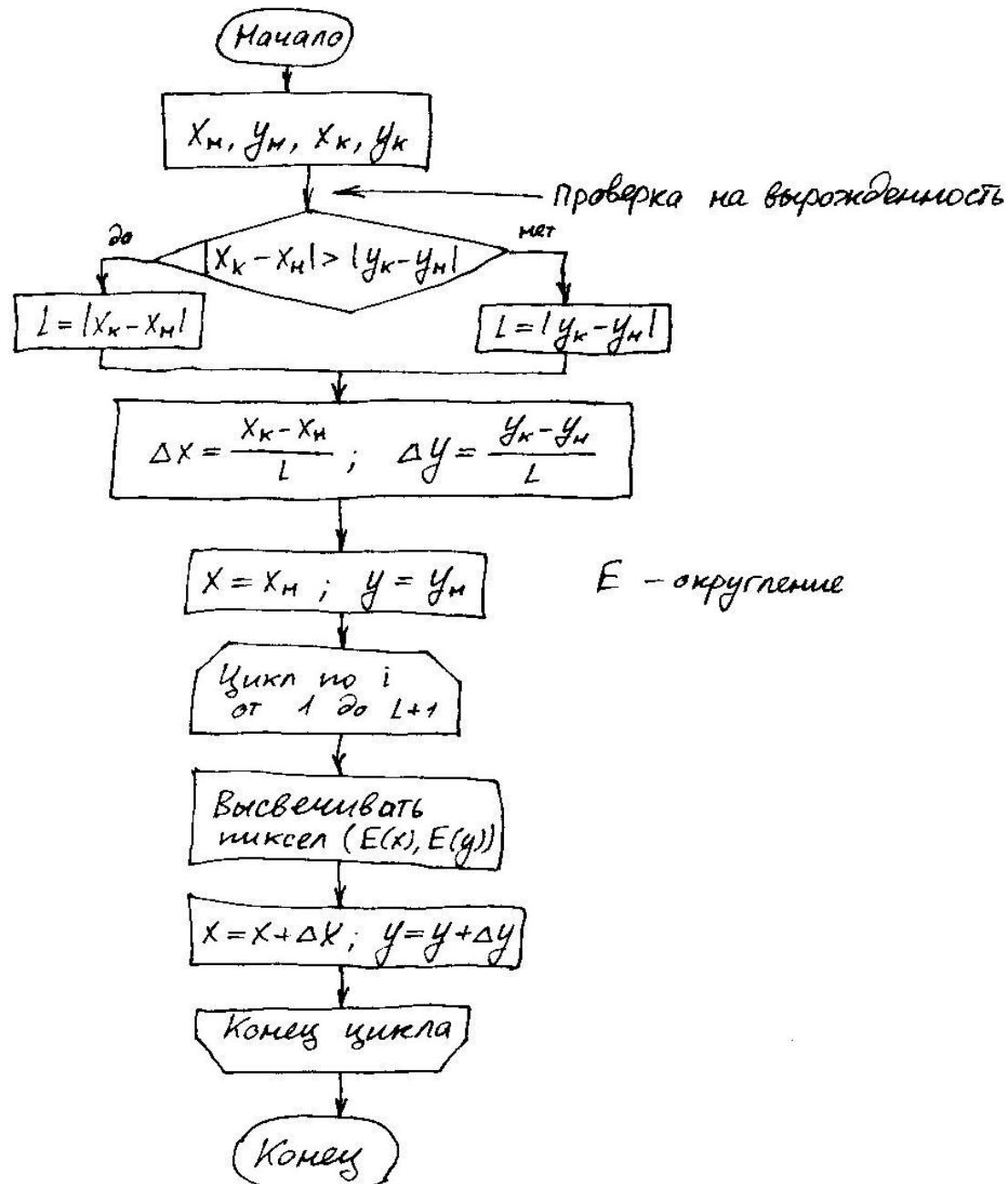
За интервал, по которому будет рассматривать приращения (Δx или Δy) возьмём интервал большей длины: $(x_n \dots x_k)$ или $(y_n \dots y_k)$.

Алгоритм цифрового дифференциального анализатора

$$Ax + By + C = 0$$

$$\frac{dy}{dx} = \text{const}; \quad \frac{\Delta Y}{\Delta X} = \frac{Y_k - Y_H}{X_k - X_H}$$

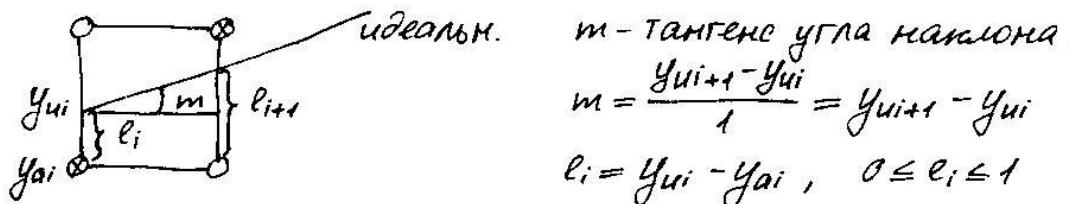
$$\Delta X = 1 \Rightarrow Y_{i+1} = Y_i + \Delta Y = Y_i + \frac{Y_k - Y_H}{X_k - X_H} \Delta X$$



- #4 Алгоритм Брезенхема разложения отрезков в растр. Простой алгоритм Брезенхема.
Численный алгоритм Брезенхема.
Общий алгоритм Брезенхема

Работа алгоритма строится на понятии "шага".
Шаг - расстояние между действительным положением отрезка и пикселям, аппроксимирующим отрезок на данном шаге.

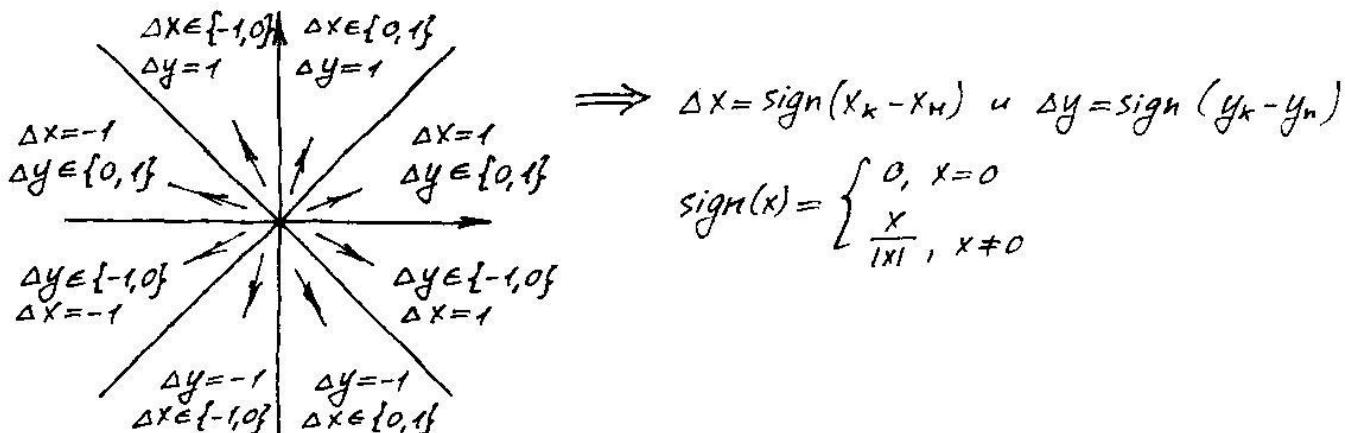
Простой алгоритм позволяет строить отрезок только в I октанте ($dx \geq 0, dy \geq 0, m \leq 1$)



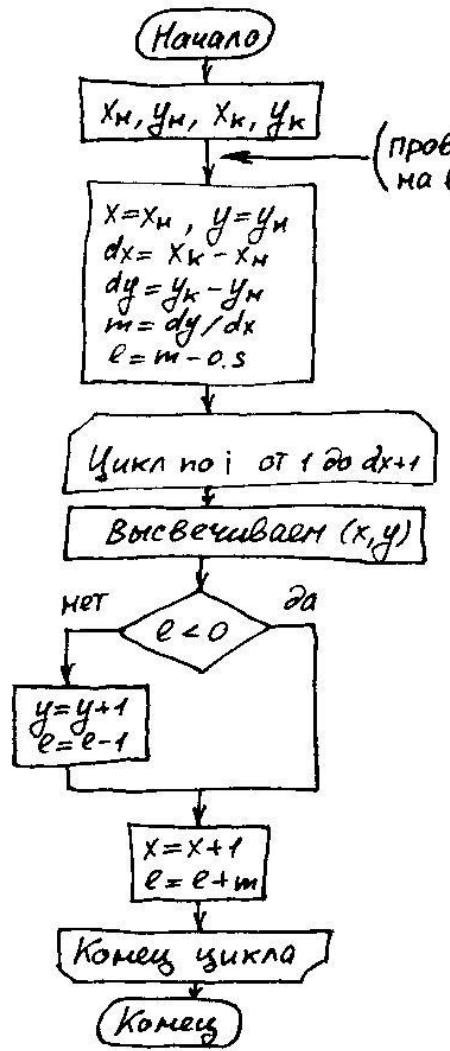
$$e_i < \frac{1}{2} \Rightarrow y_{i+1} = y_i; \quad e_i \geq \frac{1}{2} \Rightarrow y_{i+1} = y_i + 1$$

$$e_{i+1} = y_{i+1} - y_{i+1} = y_i + m - y_{i+1} = \begin{cases} y_{i+1} + m - y_i = e_i + m \\ y_{i+1} + m - y_i - 1 = e_i + m - 1 \end{cases}$$

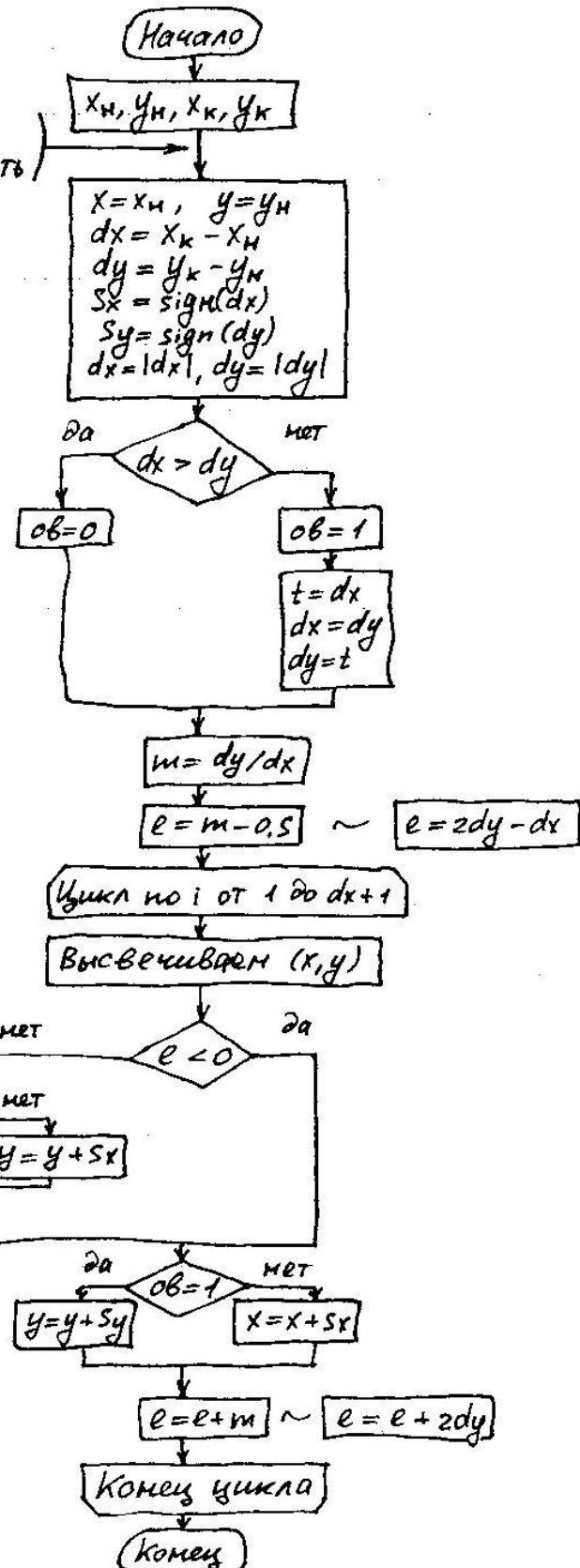
Общий алгоритм Брезенхема позволяет строить отрезок в любой октанте.



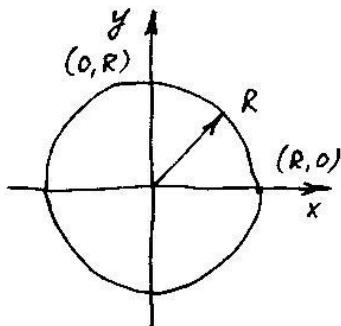
Простой алгоритм Брезенхема



Общий алгоритм Брезенхема

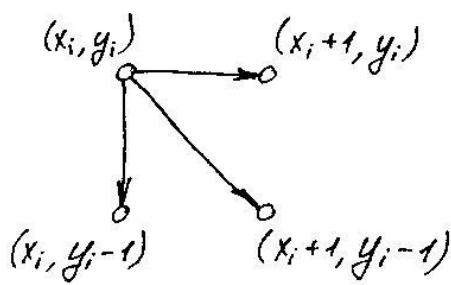


#5 Основные расчётные соотношения и алгоритм брезенхема для генерации окружности.



Построение в I четверти.

$y = f(x)$ — монотонно убывающая.



$$\Delta_i = (x_{i+1})^2 + (y_{i-1})^2 - R^2$$

1) $\Delta_i < 0 \Rightarrow$ пиксель лежит внутри окр.

Выбираем диаг. или горизонт. пикс. (случаи 1 или 2)

2) $\Delta_i = 0 \Rightarrow$ диаг. пиксель лежит на окр. (шаг по диагонали)

3) $\Delta_i > 0 \Rightarrow$ диаг. пиксель лежит вне окр.

Выбираем диаг. или вертик. пиксель (случаи 3 или 4)

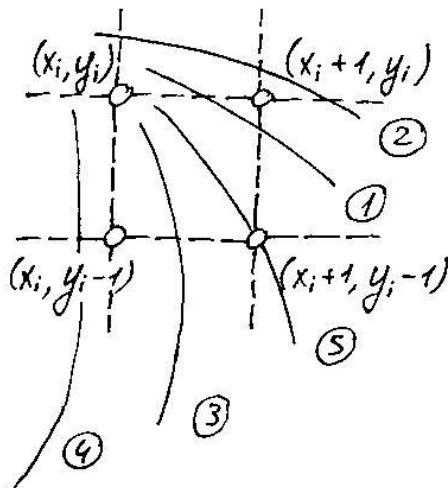
Рассмотрим наши случаи:

1) $\Delta_i < 0$

$$\delta_i = |(x_{i+1})^2 + y_i^2 - R^2| - |(x_{i+1})^2 + (y_{i-1})^2 - R^2|$$

$\delta_i \leq 0 \Rightarrow$ расст. до диаг. пикс. больше, чем расст. до гориз. пикселя (выбор горизонт. пикселя)

$\delta_i > 0 \Rightarrow$ расст. до гориз. пикс. больше, чем расст. до диаг. пикселя (выбор диаг. пикселя)



Случай 1

$$(x_i + 1)^2 + y_i^2 - R^2 \geq 0$$

$$(x_i + 1)^2 + (y_i - 1)^2 - R^2 < 0$$

$$\delta_1' = (x_i + 1)^2 + y_i^2 - R^2 + (x_i + 1)^2 + (y_i - 1)^2 - R^2 =$$

$$= 2 [(x_i + 1)^2 + (y_i - 1)^2 - R^2] + 2y_i - 1 = 2(\Delta_i + y_i) - 1$$

Случай 2

$$(x_i + 1)^2 + y_i^2 - R^2 < 0$$

$$(x_i + 1)^2 + (y_i - 1)^2 - R^2 < 0$$

$$\delta_1' = -(x_i + 1)^2 - y_i^2 + R^2 + (x_i + 1)^2 + (y_i - 1)^2 - R^2 = -2y_i + 1 < 0$$

2) $\Delta_i > 0$

$$\delta_2' = |(x_i + 1)^2 + (y_i - 1)^2 - R^2| - |x_i^2 + (y_i - 1)^2 - R^2|$$

$\delta_2' \leq 0 \Rightarrow$ расст. до верт. пикселя больше, чем до диаг.
(выбираем диагональный).

$\delta_2' > 0 \Rightarrow$ расст. до диаг. пикселя больше, чем до верт. \Rightarrow
 \Rightarrow выбираем вертикальный.

Случай 3

$$(x_i + 1)^2 + (y_i - 1)^2 - R^2 \geq 0$$

$$x_i^2 + (y_i - 1)^2 - R^2 < 0$$

$$\delta_2' = (x_i + 1)^2 + (y_i - 1)^2 - R^2 + x_i^2 + (y_i - 1)^2 - R^2 =$$

$$= 2[(x_i + 1)^2 + (y_i - 1)^2 - R^2] - 2x_i - 1 = 2(\Delta_i - x_i) - 1$$

Случай 4

$$(x_i + 1)^2 + (y_i - 1)^2 - R^2 > 0$$

$$x_i^2 + (y_i - 1)^2 - R^2 \geq 0$$

$$\delta_2' = (x_i + 1)^2 + (y_i - 1)^2 - R^2 - x_i^2 - (y_i - 1)^2 + R^2 = 2x_i + 1 > 0$$

Случай 5

$$\Delta_i = 0$$

$$\delta_1' = |(x_i + 1)^2 + y_i^2 - R^2| > 0$$

$$\delta_2' = -|x_i^2 + (y_i - 1)^2 - R^2| < 0$$

1. Горизонтальный шаг

$$x_{i+1} = x_i + 1, \quad y_{i+1} = y_i$$

$$\begin{aligned}\Delta_{i+1} &= (x_{i+1} + 1)^2 + (y_{i+1} - 1)^2 - R^2 = (x_i + 2)^2 + (y_i - 1)^2 - R^2 = \\ &= (x_i + 1)^2 + (y_i - 1)^2 - R^2 + 2x_i + 3 = \Delta_i + 2x_{i+1} + 1\end{aligned}$$

2. Диагональный шаг

$$x_{i+1} = x_i + 1, \quad y_{i+1} = y_i - 1$$

$$\begin{aligned}\Delta_{i+1} &= (x_{i+1} + 1)^2 + (y_{i+1} - 1)^2 - R^2 = (x_i + 2)^2 + (y_i - 2)^2 - R^2 = \\ &= (x_i + 1)^2 + (y_i - 1)^2 - R^2 + 2x_i + 3 - 2y_i + 3 = \Delta_i + 2x_{i+1} - 2y_{i+1} + 2\end{aligned}$$

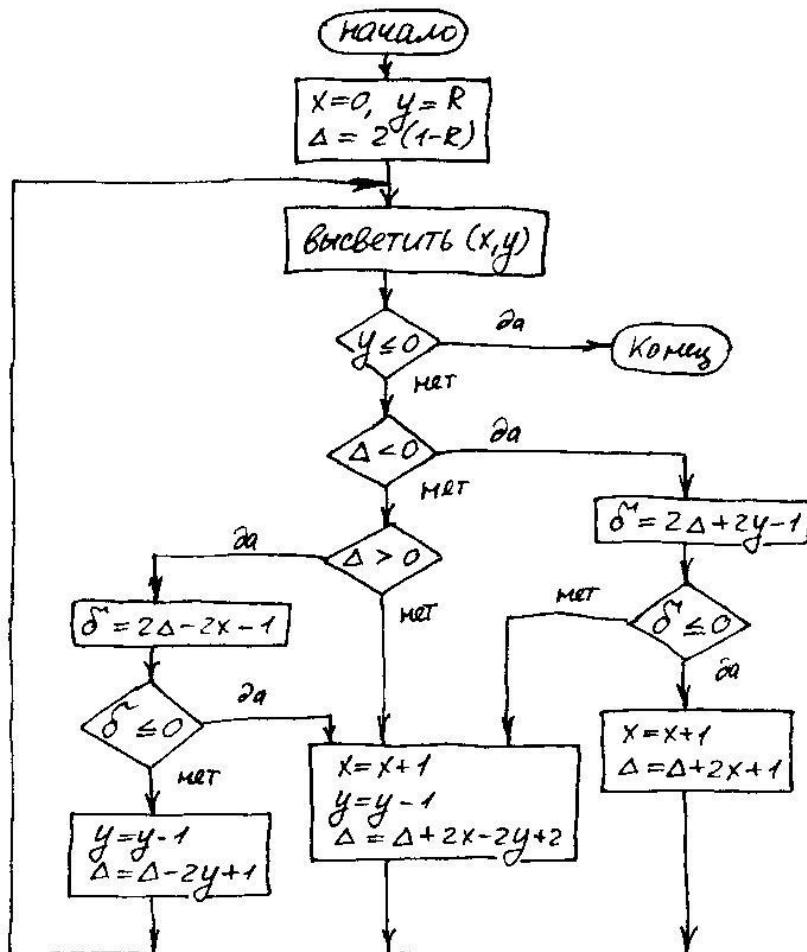
3. Вертикальный шаг

$$x_{i+1} = x_i, \quad y_{i+1} = y_i - 1$$

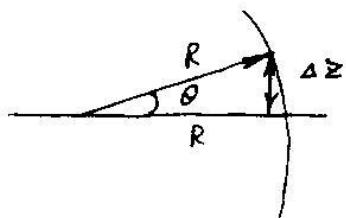
$$\begin{aligned}\Delta_{i+1} &= (x_{i+1} + 1)^2 + (y_{i+1} - 1)^2 - R^2 = (x_i + 1)^2 + (y_i - 2)^2 - R^2 = \\ &= (x_i + 1)^2 + (y_i - 1)^2 - R^2 - 2y_i + 3 = \Delta_i - 2y_{i+1} + 1\end{aligned}$$

Например: $\Delta_1 = (0+1)^2 + (R-1)^2 - R^2 = 2(1-R)$

Блок-схема



Построение кривых. Выбор шага изменения аргумента



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

$$\text{Тогда } \theta = \frac{\Delta z}{R} = \frac{1}{R}$$

При достаточно большом радиусе кривизны 2 соседние точки должны быть выбраны таким образом, чтобы величина угла, выраженная в радианах, образованная радиусами была не менее $\theta = \frac{1}{R}$.

$$y = f(x)$$

$$R = \frac{(1 + (y')^2)^{\frac{3}{2}}}{|y''|}$$

$$x = x(t), \quad y = y(t)$$

$$R = \frac{(x'^2 + y'^2)^{\frac{3}{2}}}{|x'y'' - y'x''|}$$

Для полярн. коорд:

$$R = \frac{\left(\rho^2 + \left(\frac{d\rho}{d\varphi}\right)^2\right)^{\frac{3}{2}}}{\left(\rho^2 + 2\left(\frac{d\rho}{d\varphi}\right)^2 - \rho \frac{d^2\rho}{d\varphi^2}\right)}$$

(#6) Способы генерации растровых изображений.
Развертка в реальном времени (организация списка активных рёбер).

Способы генерации изображений:

1. Планшет буфера кадра.
2. Растровая развертка в реальном времени.
3. Групповое кодирование.
4. Клеточное кодирование.

Растровая развертка в реальном времени

- геометр. характеристики (коорд. x, y, учит наклона, текст);
- информация о визуальных хар-ках (цвет, интенсивность).

В буфере мы храним информацию о списках акт. рёбер.

САР хранит инф-цию о тех активных рёбрах, которые пересекают текущ. скан. строку.

Для организации САР необходимо:

1. Упорядочить рёбра по убыванию их координаты y вершин рёбер.
2. Если на одной сканирующей строке начинается несколько рёбер, то их надо упорядочить по убыванию y-координаты вторых вершин.

x_A - абсцисса вершины A.

Δx_A - изменение абсциссы точки пересечения ребра при переходе к след. скан. строке.

Δy - кол-во строк, пересекаемых ребром.

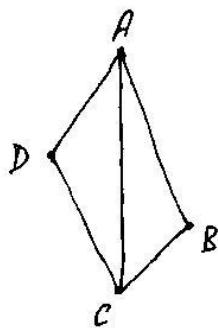
$$\Delta x_A = \frac{x_B - x_A}{\Delta y}$$

Для построения ребра достаточно знать:

$$x_A, \Delta x, \Delta y, \text{ где } \Delta x = \frac{x_k - x_n}{\Delta y}, \Delta y = y_k - y_n.$$

3. Ввести указатели на начало и конец списка:

Например



$y_A: \overbrace{AD, AB, AC}^{\downarrow}, DC, BC$

$y_D: AD, \overbrace{AB, AC, DC}^{\downarrow}, BC$

При рисовании пологих отрезков может получаться, что он равной:



$y_{max} AD$	x_{AD}	Δx_{AD}	Δy_{AD}
$y_{max} AB$	x_{AB}	Δx_{AB}	Δy_{AB}

$y_{tek} > y_{max} в 1ом эл-те$ — нет акт. рёбер

$\Delta y_{AD} < 0$ — эл-т из списка удаляется

I способ: (y -группы)

$X_1, \Delta X_1, \Delta Y_1$
$X_2, \Delta X_2, \Delta Y_2, \dots$
$X_k, \Delta X_k, \Delta Y_k$
—
—
$X_{k+1}, \Delta X_{k+1}, \Delta Y_{k+1}$
—

← инф-ция об отрезке начинается на ст. строке

Недобективность метода.

II способ:

1	1	1
—	2	X_{AD}
—	3	ΔX_{AD}
11	4	ΔY_{AD}
	5	X_{AB}
	6	ΔX_{AB}
	7	ΔY_{AB}
	8	X_{AC}
	9	ΔX_{AC}
	10	ΔY_{AC}
	—	
	11	X_{DC}
	12	ΔX_{DC}
	13	ΔY_{DC}
	14	—

III способ:

y_A	1
	...
y_D	13
	...
y_B	17
	...

1	X_{AD}
2	ΔX_{AD}
3	ΔY_{AD}
4	—
5	X_{AB}
6	ΔX_{AB}
7	ΔY_{AB}
8	—
9	X_{AC}
10	ΔX_{AC}
11	ΔY_{AC}
12	—
13	X_{DC}
14	ΔX_{DC}
15	ΔY_{DC}
16	—
17	X_{BC}
18	ΔX_{BC}
19	ΔY_{BC}
20	—

— след. отрезок

(#7) Способы генерации изображений:

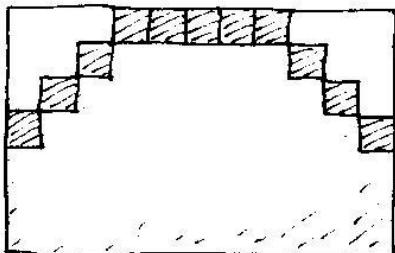
групповое кодирование, клеточное кодирование,
адресация растра.

Групповое кодирование

интенсивность | длина участка | - информационная ячейка

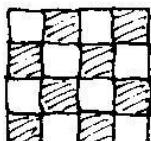
интенсивность красного | интенсивн. зелёного | интенсивн синего | длина участка | - с учётом цвета

Например:



4 3	8 5	4 3	6
4 2	8 7	4 2	6
4 1	8 9	4 1	6
8 11			2
8 11			2
8 11			2
8 11			2

$7 \cdot 11 = 77 > 26$ - т.е. мы получили хорошее сжатие

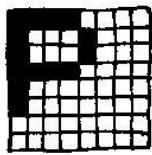


- здесь этот метод проигрывает пиксельному хранению в 2 раза.

Добавление или удаление отрезков или текста в изобр. является трудоёмкой операцией и занимает много времени из-за последовательного хранения этих участков.

Клеточное кодирование

Область экрана разбивается на квадраты площадью $n \times n$ (например, квадрат 8×8 хватит для вывода латеры). Для вывода изображения с использованием этого метода в ПЗУ нужно хранить шаблоны сегментов изображения. Тогда комбинации таких сегментов, расположенных в соседних квадратах реализуют построение изображения.



Для произвольной клетки размером $n \times n$ существует 2^{n^2} возможных шаблонов, составленных из пикселей.

Хотя показано, что для алгоритма Брезенхема для отрезков с тангенсом угла наклона между 0 и 1 существует не более $2^n - 1$ шаблонов, представляющих сегменты.

Жордан и Баррет показали, что для клетки 8×8 при использовании переноса, отражения и маскирования требуется только $108^{\text{шаблонов}}$ сегментов отрезков.

Адресация растра

Пусть пиксель в растре имеет координаты (x, y) .

Цифровая память организована в линейный список адресатов.

Тогда: Адрес = $(x_{\max} - x_{\min})(y - y_{\min}) + (x - x_{\min}) + \text{базовый адрес.}$

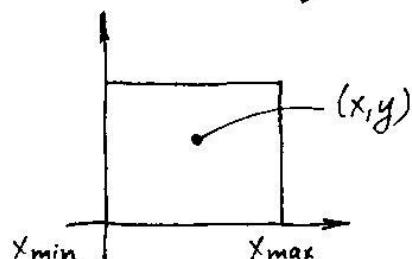
Адрес = $K_1 + K_2 y + x$, где $K_1 = \text{базовый адрес} - K_2 y_{\min} - x_{\min}$;

$$K_2 = x_{\max} - x_{\min}$$

$$\text{Адрес}(x \pm 1, y) = K_1 + K_2 y + x \pm 1 = \text{Адрес}(x, y) \pm 1$$

$$\text{Адрес}(x, y \pm 1) = K_1 + K_2(y \pm 1) + x = \text{Адрес}(x, y) \pm K_2$$

$$\text{Адрес}(x \pm 1, y \pm 1) = \text{Адрес}(x, y) \pm K_2 \pm 1$$

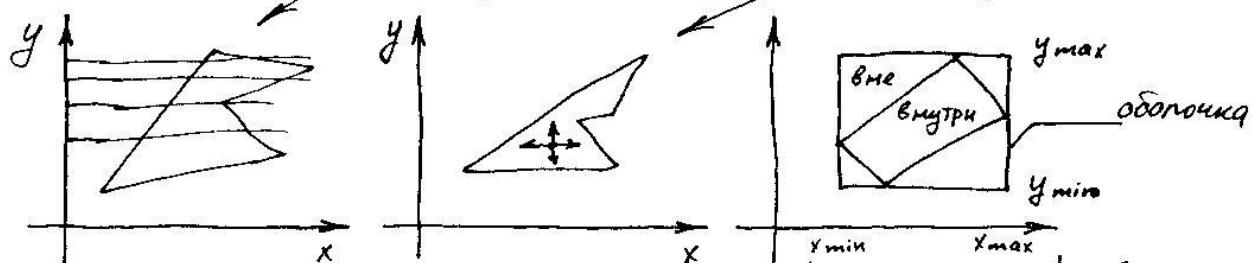


#8) Растровая развертка сплошных областей.
Алгоритм с упорядоченным списком рёбер.

Под растровой разверткой сплошных областей понимается генерация областей на основе простых описанных рёбер или вершин многоугольников (зиграска).

Для этого используются две категории методов:

растровая развертка и затравочное заполнение.



В порядке сканирования строк определяют, лежит ли точка внутри многоугольника. Эти алгоритмы методы применимы для растровых и векторных дисплеев.

Известна точка (затравка) внутри замкнутого контура. Ищут точки, соседние с затравочной и расположенные внутри контура. Если точка не внутри, то обнаружена граница. Поиск рекурсивный. Эти методы применимы только к растровым дисплеям.

$$(x_{\min} \leq x \leq x_{\max}) \wedge (y_{\min} \leq y \leq y_{\max})$$

Алгоритм заполнения с упорядоченным списком рёбер

1. Найти точки пересечения всех скан. строк со всеми рёбрами.
2. Все точки в массив (список).
3. Сортировать точки по убыванию y .
4. Сортировать интервалы точек разных y по возрастанию x : $((x_k, y), (x_l, y))$, если $y_1 > y_2$ или $x_k \leq x_l$ и $y_1 = y_2$.
5. Работать на пары и выставлять все пиксели между точками каждой пары.

#9 Заполнение многоугольников.

Алгоритм заполнения по рёбрам, с перегородкой, со списком рёбер и флагом.

Алгоритм заполнения по рёбрам

фон = цв. закраски

цв. закраски = фон

Для каждой сканирующей строки, пересекающей ребра ми-ка дополнить все пиксели, лежащие правее точек пересечения.

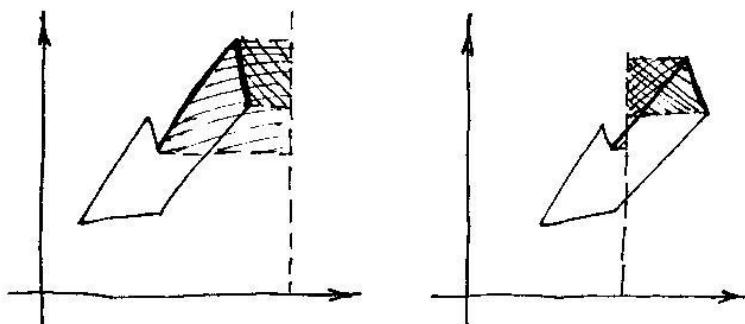
Недостаток: каждый пиксель может перекрашиваться много раз.

Алгоритм заполнения по рёбрам с перегородкой

фон = цв. закр.

цв. закр. = фон

Если т. пересечения скан. строки с ребром ми-ка лежит левее перегородки (вертикальная линия между x_{min} и x_{max}), то дополнить все пиксели, лежащие правее т. пересечения, но левее перегородки. Если правее перегородки - то дополнить все пиксели левее пересечения, но правее перегородки.

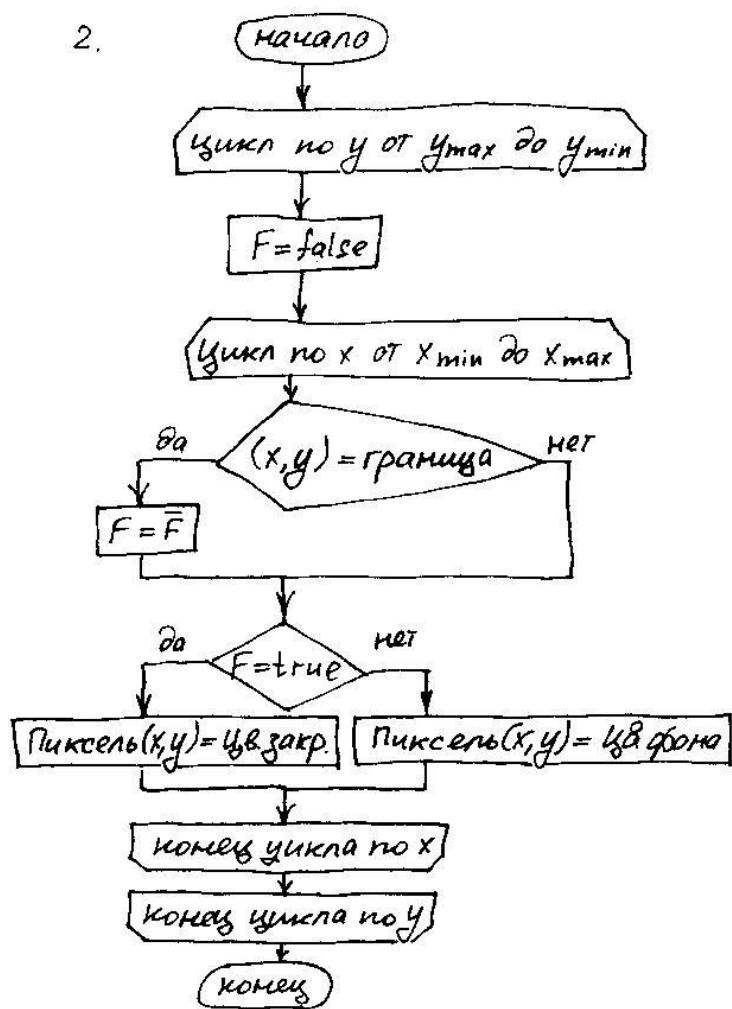


В обоих алгоритмах очертывания границы заливаемой области не нужно.

Алгоритм со списком рёбер и флагом

1. Очергить контур, в результате чего на каждой строке образуются пары ограничивающих пикселов.

2.



(#10) Алгоритм заполнения с затравкой,
простой алгоритм заполнения с затравкой

Алгоритм заполнения с затравкой

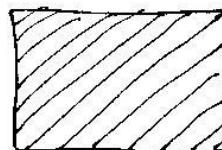
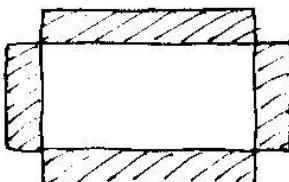
Необходимо:

- 1) Задать область.
- 2) Знать координаты точки, лежащей внутри.

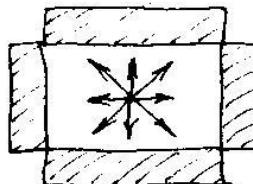
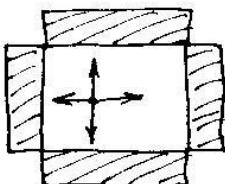
Рассматриваются пиксели, соседние с данным.

Используется стек.

Границно-опред. области: Внутренне-определенные



Области бывают 4-связные и 8-связные:



Простой алгоритм

1. Задание исх. данных
(область и затрав. пиксель)
2. Поместить затрав. пиксель в стек
3. Пока стек не пуст:
 - a) извлечь пиксель (x, y) из стека
 - b) цвет $(x, y) \neq$ цвет закраски \Rightarrow
 \Rightarrow цвет $(x, y) =$ цвет закраски
 - c) Анализ 4-х соседних пикселей:
 $(x-1, y), (x, y+1), (x+1, y), (x, y-1)$

Цвет пикс. \neq Цв.закр. и \neq Цв.границы \Rightarrow Пиксель в стек

Недостатки:

1. Стек увеличивается.
2. Незакраин. пиксели могут попадать в стек не один раз.

(#11) Алгоритмы заполнения с затравкой.

Построенный алгоритм заполнения с затравкой.

Основное отличие от простого: ограниченное количество пикселей в стеке.

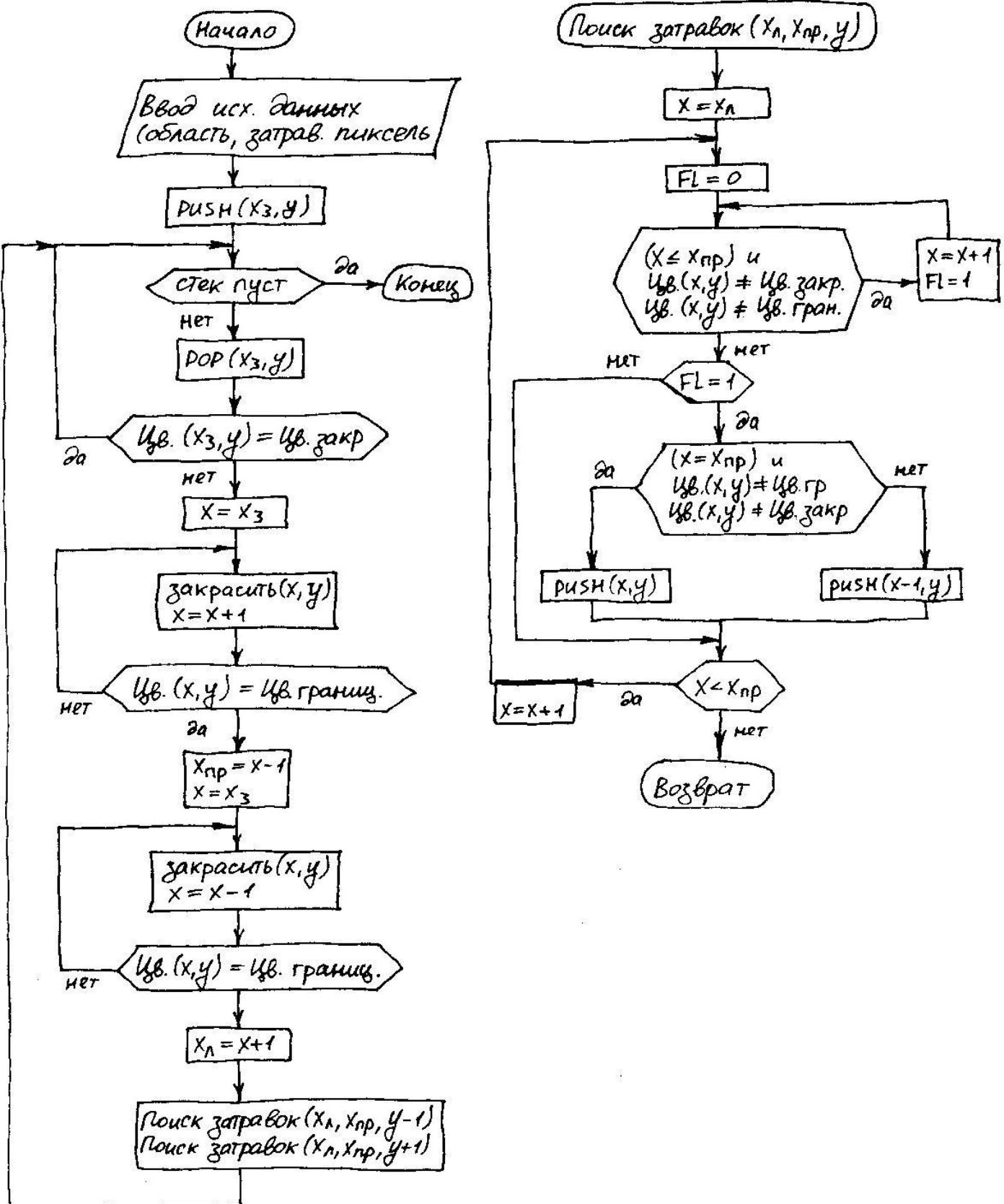
Непрерывность интервал пикселей - группа прилегающих друг к другу пикселей, незакрашенных и неграницевых, которые ограничены закрашенными или границевыми пикселями.

В стек помещается один из пикселей этой группы.

Алгоритм:

1. Задание исходных данных (область, затрав. пиксл).
2. Поместить в стек затрав. пиксл.
3. Пока стек не пуст, выполнять:
 - a) Извлечь затравочный пиксл из стека.
 - b) Заполнение пикселей, лежащих левее затравочного (в пределах строки), до встречи с границным.
 - c) Запомнить самый левый закрашенный пиксл в $X_{лев}$.
 - d) Заполнение пикселей, лежащих правее затравочного, до встречи с границным.
 - e) Запомнить самый правый закрашенный пиксл в $X_{прав}$.
 - f) Поиск затравочных пикселей на соседних строках ($y+1, y-1$) в интервалах $X_{лев} \leq X \leq X_{прав}$ (для интервала незакрашенных и неграницевых пикселей самый правый - затравочный - поместить его в стек).

Блок-схема алгоритма:



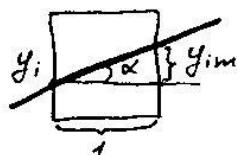
#12 Основы методов устранения ступенчатости.

Алгоритм брезенхема с устранением ступенчатости.

Основная причина появления лестничного эффекта заключается в том, что отрезки имеют непрерывную природу, тогда как растровое устройство дискретно.

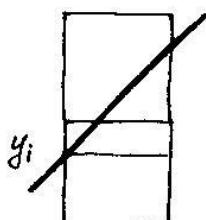
Существует 2 метода устранения искажений изображения:

- 1) увеличить частоту выборки, что достигается с помощью увеличения разрешения растра \Rightarrow учитываются более мелкие детали.
- 2) трактовать пиксели не как точку, а как конечную область (высвечивать пиксели интенсивностью, пропорциональной площади части пикселя, находящегося под отрезком).



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

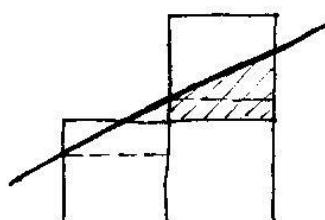
$$S = S_{\square} + S_{\Delta} = y_i \cdot 1 + \frac{m \cdot 1}{2} = y_i + \frac{m}{2}$$



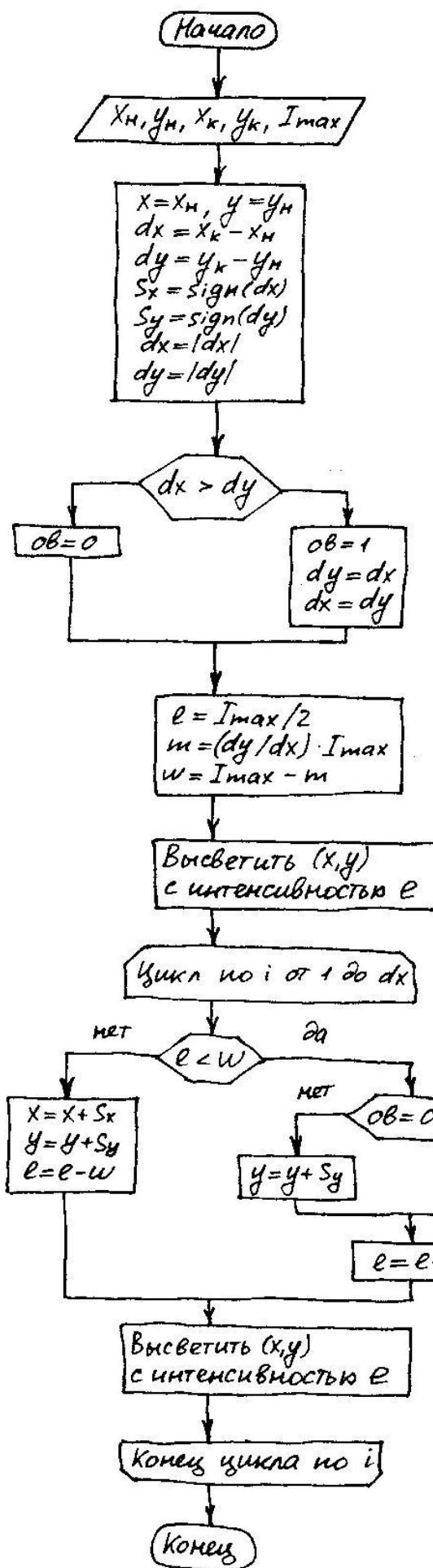
$$S = S_{\square} + S_{\Delta} = y_i \cdot 1 + \frac{m \cdot 1}{2} = y_i + \frac{m}{2}$$



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



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



#13 Двумерное отсечение. Простой алгоритм отсечения отрезка.

Отсечение — удаление изображения за пределами выделенной области (окна).

Для того, чтобы отсечь, необходимо знать:

1. Геометрические характеристики отсекаемых объектов.
2. Характеристики отсекателя.

Объект, целиком лежащий в области, — видимый.

Невидимый — если целиком расположен за пределами отсекателя.

Частично видимый — часть в пределах, часть за пределами.

1001	1000	1010
0001	0000	0010
0101	0100	0110

$$\left. \begin{array}{l} T_{i0} = 1, \text{ если } x < x_n, \text{ иначе } T_{i0} = 0 \\ T_{i1} = 1, \text{ если } x > x_{pr}, \text{ иначе } T_{i1} = 0 \\ T_{i2} = 1, \text{ если } y < y_n, \text{ иначе } T_{i2} = 0 \\ T_{i3} = 1, \text{ если } y > y_b, \text{ иначе } T_{i3} = 0 \end{array} \right\} \begin{array}{l} \text{код} \\ \text{точки} \\ P_i \end{array}$$

$$S_j = \sum_{i=0}^3 T_{ji} \quad \text{и} \quad P = \sum_{i=0}^3 T_{i1} \cdot T_{i2}$$

$S_j \neq 0 \Rightarrow$ точка невидима.

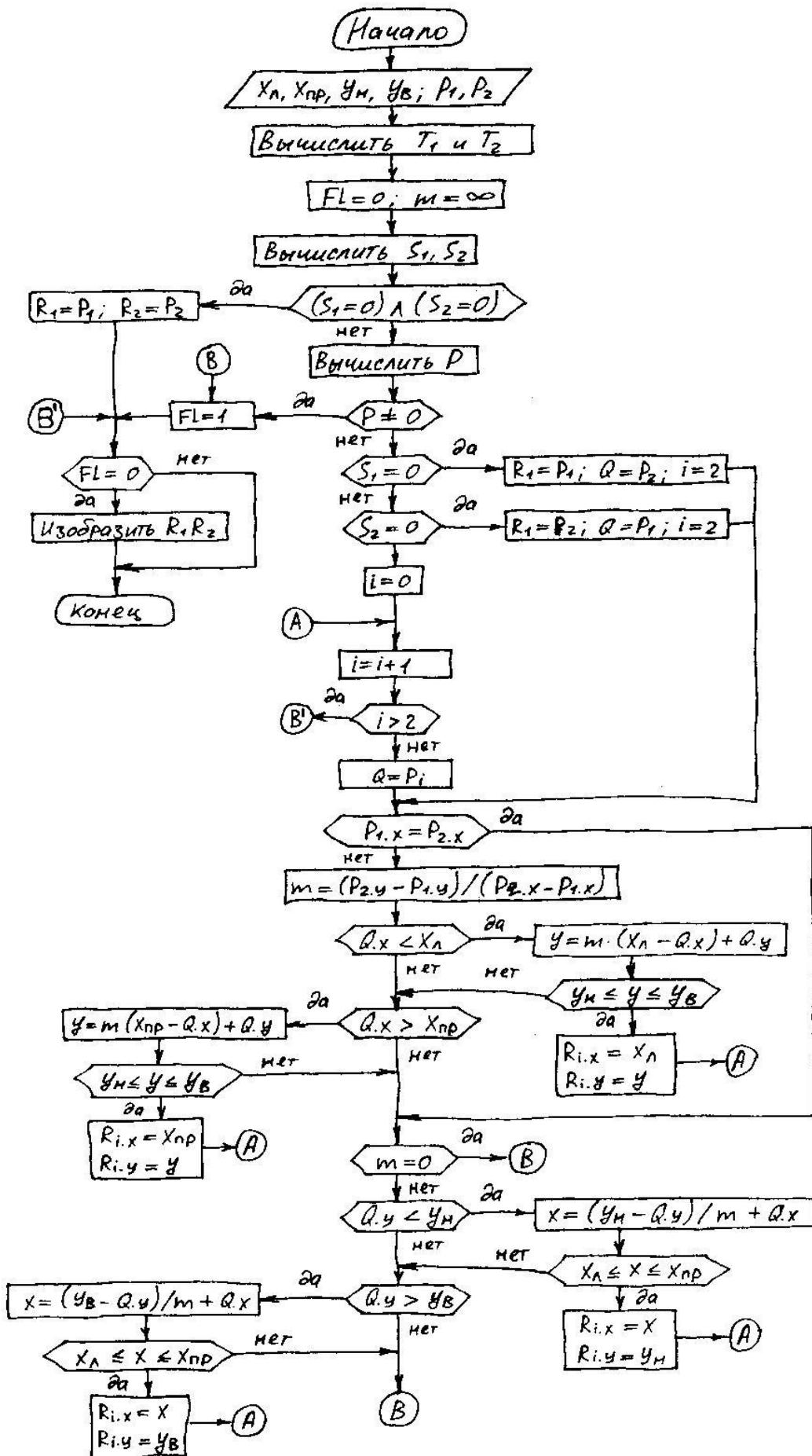
$S_j = 0 \Rightarrow$ точка видима.

$P \neq 0 \Rightarrow$ отрезок лежит по одному из сторон окна (невидим).

$(S_1 = 0) \wedge (S_2 = 0) \Rightarrow$ отрезок полностью видим.

$((S_1 = 0) \wedge (S_2 \neq 0)) \vee ((S_1 \neq 0) \wedge (S_2 = 0)) \Rightarrow$ частично видим.

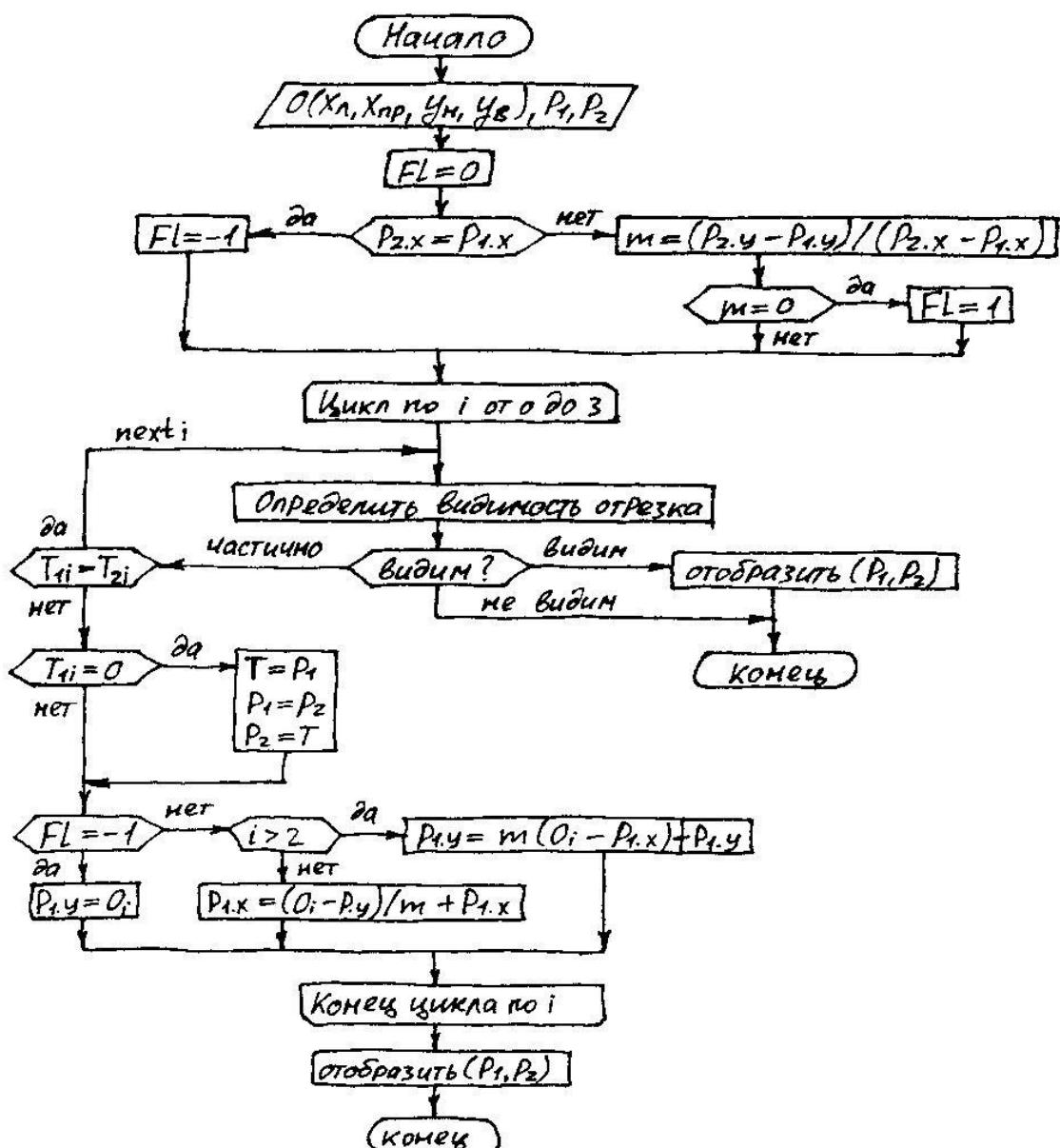
Блок-схема простого алгоритма отсечения



(#14) Отсечение. Алгоритм Сандерленда-Кээна отсечения отрезка.

Алгоритм отличается от простого способом нахождения точки пересечения. Отрезок на каждом шаге разбивается на 2 части (если сторона отсекателя или её продолжение пересекают отрезок). Одна часть отрезка будет видимой (или частично видимой), а другая будет невидимой - её мы отбрасываем.

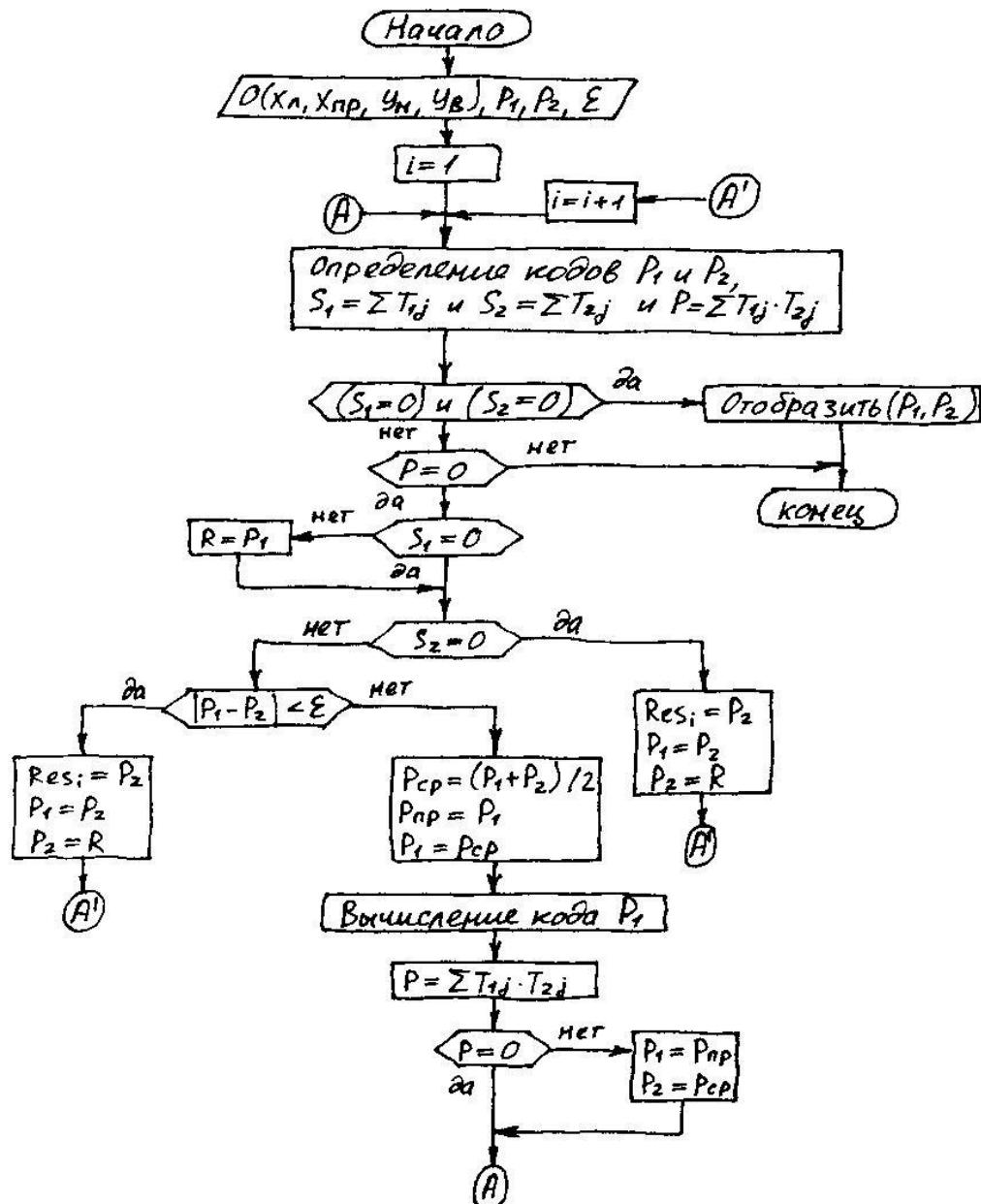
Блок-схема:



(#15) Отсечение. Алгоритм разбиения средней точкой при отсечении отрезка.

При реализации программным путём этот алгоритм работает медленнее простого, при реализации аппаратным путём — быстрее. $\varepsilon \approx \sqrt{2}$.

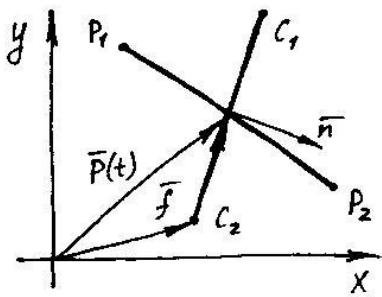
Блок-схема:



(#16) Отсечение. Алгоритм Кируса-Бека отсечения отрезка.

Параметрическое задание отрезка:

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



$(P(t) - f_i) \cdot \bar{n}_i = 0$ при таком t , которое соответствует $P(t)$ — точке пересечения $P_1 P_2$ и $C_1 C_2$ (f_i — в точка на $C_1 C_2$, \bar{n} — нормаль к $C_1 C_2$).

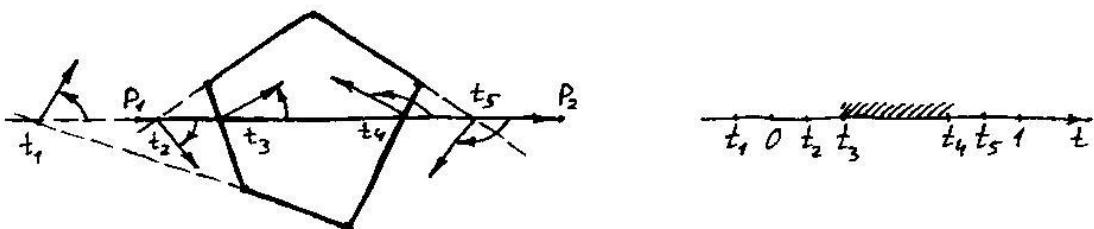
$$(P_1 + (P_2 - P_1) \cdot t - f_i) \cdot \bar{n}_i = 0$$

$$t = -\frac{W_{CKi}}{D_{CKi}}, \text{ где } W_{CKi} = (P_1 - f_i) \cdot \bar{n}_i \text{ и } D_{CKi} = (P_2 - P_1) \cdot \bar{n}_i$$

$$D_{CKi} = 0 \Rightarrow C_1 C_2 \parallel P_1 P_2$$

Отрезок может пересекать выпуклое окно не более, чем в двух точках.

Пусть $\bar{n}_i = \bar{n}_{BHi}$ — вектор внутренней нормали выпуклого окна.



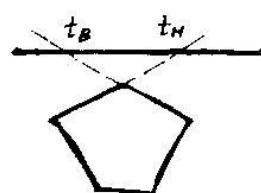
$D_{CKi} > 0 \Rightarrow t$. пересечения с i -ой гранью лежит ближе к видимой части отрезка (ближе к началу). Если $D_{CKi} > 0$, то t_i принадлежит нижней группе видимости (в примере это: t_1, t_2, t_3).

$D_{CKi} < 0 \Rightarrow t$. пересечения с i -ой гранью лежит ближе к концу видимой части отрезка. Если $D_{CKi} < 0$, то t_i принадлежит верхней группе видимости (в примере это: t_4, t_5).

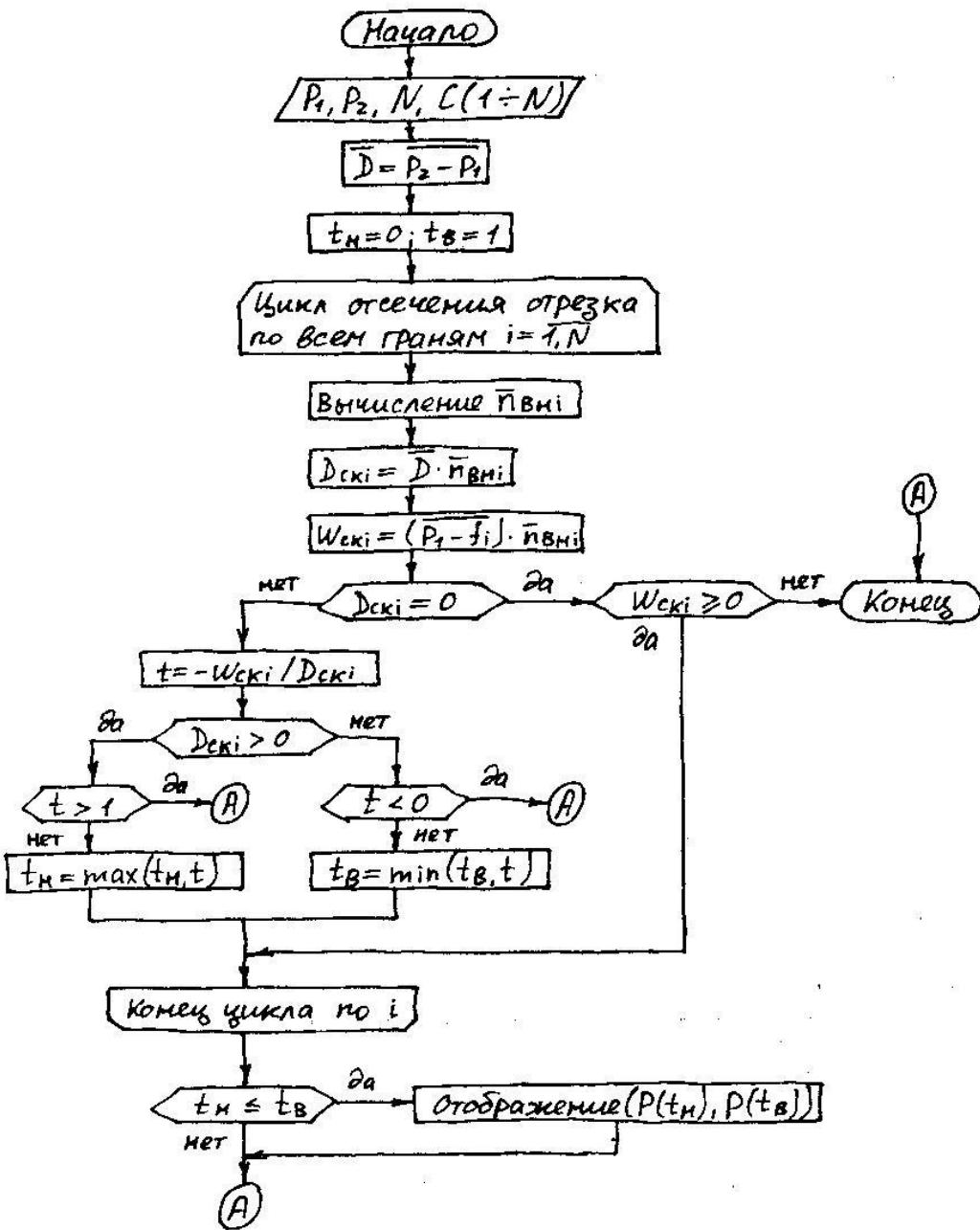
$$t_u = \max t_{Bi} \text{ и } t_b = \min t_{Bi}$$

Необходимо проверить, чтобы $t_u \leq t_b$.

$$W_{CKi} = \begin{cases} > 0 & \Rightarrow P_1 \text{ видима} \\ = 0 & \Rightarrow P_1 \text{ на грани} \\ < 0 & \Rightarrow P_1 \dots \text{и т.д.} \end{cases}$$



БЛОК-СХЕМА:

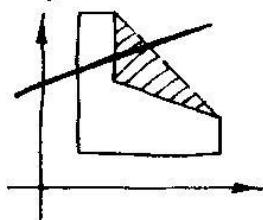


#17 Внутреннее и внешнее отсечение. Определение выпуклости многоугольника; определение нормали; разбиение невыпуклых многоугольников.

Внутреннее отсечение: определяются части отрезков, лежащие внутри отсекателя, и вычерчиваются.

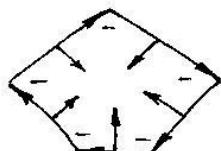
Внешнее отсечение: определяются части отрезков, лежащие вне отсекателя, и вычерчиваются.

Внешнее отсечение играет важную роль в дисплеях, допускающих работу с несколькими окнами.



Внешним отсечением можно воспользоваться при работе с волнистым окном.
(для рис.: относительно треуг-ка сделать внешнее отсечение).

Определение факта выпуклости



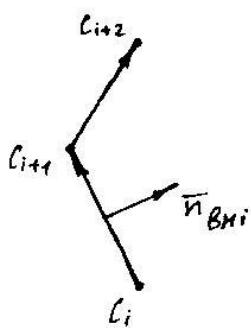
Нужно вычислить векторные произведения всех пар смежных сторон:

- 1) если все равны нулю \Rightarrow мн-к выпуклен в отрезок;
- 2) если есть неравные нулю \Rightarrow мн-к невыпуклен;
- 3) если есть и положительные и отрицательные \Rightarrow мн-к невыпуклый;
- 4) если все неотрицательные \Rightarrow мн-к выпуклой, внутренние нормали ориентированы влево от рёбер-векторов;
- 5) если все неположительные \Rightarrow мн-к выпуклы, внутренние нормали ориентированы вправо от рёбер-векторов.

Определение факта выпуклости и разбиение невыпуклого

- 1) Перенести мн-к так, чтобы i -ая вершина совпала с $(0,0)$.
- 2) Повернуть мн-к вокруг $(0,0)$ так, чтобы его $(i+1)$ -ая вершина оказалась на положительной части оси абсцисс.
- 3) Внимательно ознакомт всех $(i+2)$ -ых вершин:
 - 3.1) если все равны нулю \Rightarrow мн-к вырожден в отрезок;
 - 3.2) если все знаки одинаковы \Rightarrow мн-к выпуклой относительно i -го ребра; в повернутой системе координат внутренняя нормаль имеет ту же самую абсциссу и ординату, равную знаку ординаты $(i+2)$ -ой вершины;
 - 3.3) если есть и положительные и отрицательные знаки \Rightarrow \Rightarrow мн-к невыпуклой; мн-к разрезается на две части вдоль оси абсцисс, далее работа ведётся уже с двумя мн-ками.

Определение внутренней нормали



$$\overline{c_i c_{i+1}} = c_{ix} \cdot \bar{i} + c_{iy} \cdot \bar{j}$$

$$\overline{n_{ВН}} = n_{ix} \cdot \bar{i} + n_{iy} \cdot \bar{j}$$

$$(\overline{c_i c_{i+1}}) \cdot \overline{n_{ВН}} = 0$$

$$c_{ix} \cdot n_{ix} + c_{iy} \cdot n_{iy} = 0$$

Пусть $n_{iy} = 1 \Rightarrow n_{ix} = -\frac{c_{iy}}{c_{ix}}$ для $c_{ix} \neq 0$;

$n_{iy} = 0$ и $n_{ix} = 1$ для $c_{ix} = 0$.

(#18) Отсечение мн-ков. Алгоритм Сандерленда - Ходжмена.

В этом алгоритме исходной и каждый из промежуточных мн-ков отсекается последовательно относительно прямой.

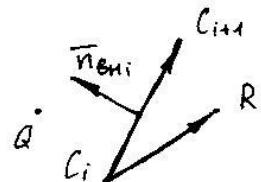
Суть алгоритма:

1. Определить видимость вершин мн-ка.

a) $f_{\text{пр}}(x, y) = Ax + By + C$ - пробная ф-ция, где A, B, C - коэф-ты уравнения прямой, проходящей через i -ое ребро. Нужно сравнить знаки $f_{\text{пр}}(Q_x, Q_y)$ и $f_{\text{пр}}(R_x, R_y)$, где (Q_x, Q_y) - точка с известным положением, (R_x, R_y) - испытываемая точка.

б) Использовать $(\bar{n}_{\text{вн}}, \bar{C}_i R)$.

в) Использовать $\overline{C_i C_{i+1}} \otimes \overline{C_i R}$.



2. Определить точки пересечения рёбер исходного многоугольника и рёбер отсекателя.

Прежде чем искать т. пересечения, нужно убедиться, что она существует. Если т. пересечения есть, то решить систему:

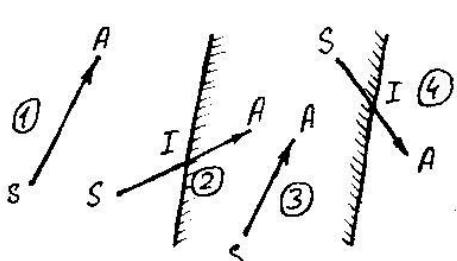
$$P(t) = P_1 + (P_2 - P_1) \cdot t, \quad 0 \leq t \leq 1 \quad \text{- ребро мн-ка.}$$

$$Q(s) = Q_1 + (Q_2 - Q_1) \cdot s, \quad 0 \leq s \leq 1 \quad \text{- ребро отсекателя.}$$

$$\begin{cases} P_{1,x} + (P_{2,x} - P_{1,x}) \cdot t = Q_{1,x} + (Q_{2,x} - Q_{1,x}) \cdot s \\ P_{1,y} + (P_{2,y} - P_{1,y}) \cdot t = Q_{1,y} + (Q_{2,y} - Q_{1,y}) \cdot s \end{cases}$$

$$\begin{cases} P_{1,x} + (P_{2,x} - P_{1,x}) \cdot t = Q_{1,x} + (Q_{2,x} - Q_{1,x}) \cdot s \\ P_{1,y} + (P_{2,y} - P_{1,y}) \cdot t = Q_{1,y} + (Q_{2,y} - Q_{1,y}) \cdot s \end{cases}$$

3. Заносить видимые вершины и найденные т. пересечения в результирующий список.



① Ничего не заносить в список.

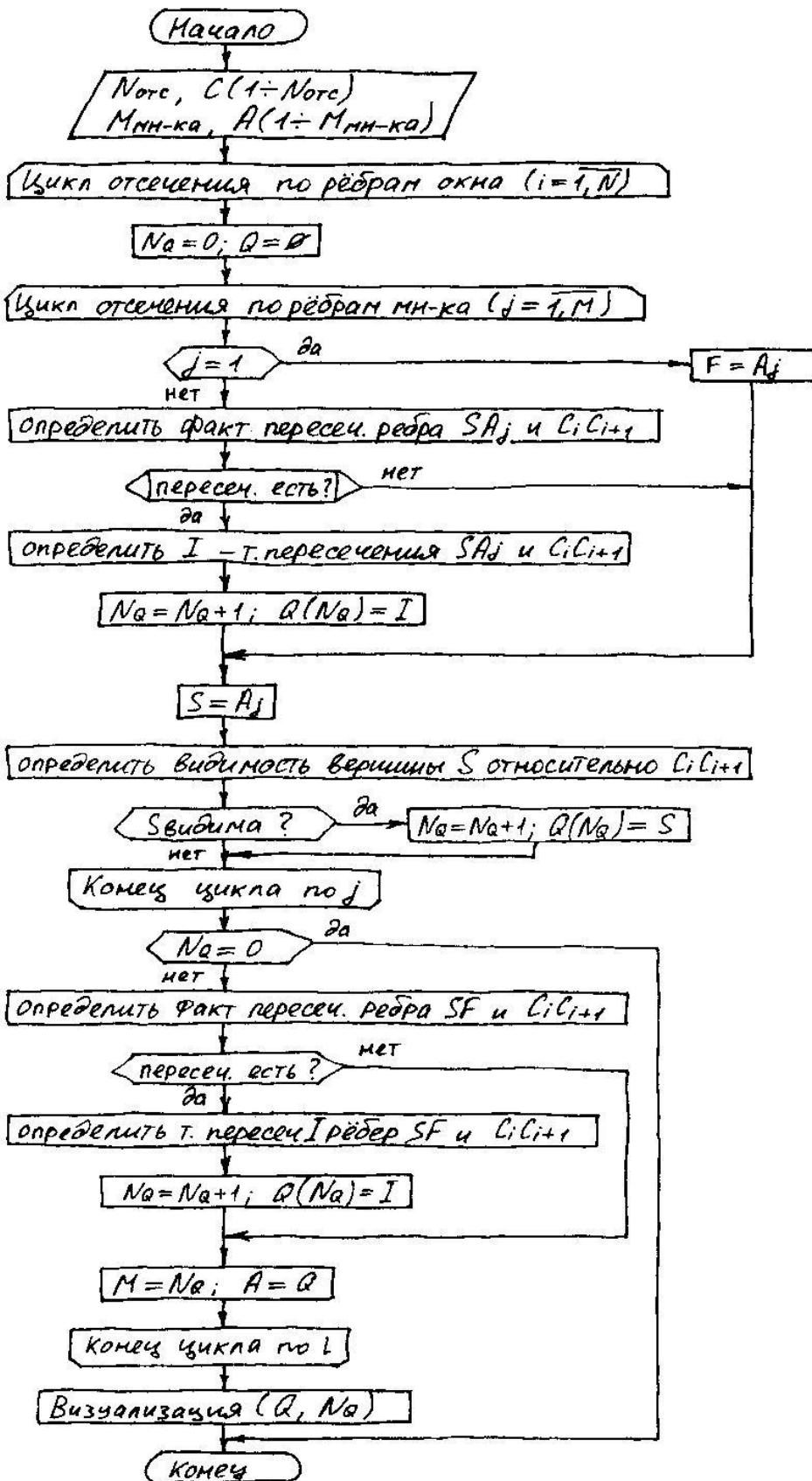
② Занести т. I и т. A.

③ Занести т. A.

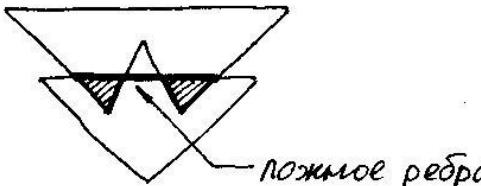
④ Занести т. I.

(возможные положения рёбер отсекателя и мн-ка).

БЛОК-СХЕМА АЛГОРИТМА:



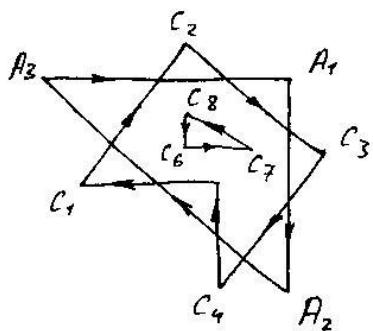
Недостаток алгоритма:



(#19) Отсечение мн-кв невыпуклыми областями.

Алгоритм Вейлера-Азертона.

Внешние границы мн-ка или отсекателя задаются по часовой стрелке, внутренние - против часовой.



Результат отсечения - мн-к, рёбра которого либо совпадают с рёбрами исходного мн-ка, либо с рёбрами отсекателя.
Новые рёбра не появляются.

Точной входа называется т. пересечения, если в этой точке ребро мн-ка входит в отсекатель. Точной выхода называется т. пересечения, если в ней ребро мн-ка выходит из отсекателя.

Чтобы получить внутренний мн-к в результате отсечения, нужно просматривать список с вершинами входного типа.

Чтобы получить внешний мн-к в результате отсечения, нужно просматривать список с вершинами выходного типа.

Описание алгоритма:

1. Ввод исходных данных.
2. Сформировать двунаправленные циклические списки вершин.
3. Найти все точки пересечения рёбер отсекаемого мн-ка и отсекателя.
4. Добавить точки пересечения в списки в соответствующие места.
5. Разделить точки пересечения на две группы: точки входа и точки выхода.
6. ...

Внутренне отсечение:

1. Взять очередную точку из списка точек входа.
Если этот список исчерпан, то поиск внутренних
мн-ков завершён.
2. Просматривать список вершин отсекаемого мн-ка
пока не встретится т. пересечения.
3. Просмотренные вершины скопировать в список
вершин редуцирующего мн-ка.
4. В т. пересечения перейти к списку вершин отсекателя.
Просматривать список вершин отсекателя, пока не встретит-
ся т. пересечения.
5. Просмотренные вершины добавить в список вершин
редуцирующего мн-ка.
6. В точке пересечения вернуться назад к списку вершин
отсекаемого мн-ка.
7. Повторять эти переходы и ходы по спискам пока не будет
достигнута исходная вершина. При достижении исходной
вершины внутр. мн-к замыкается. Для поиска следующего
мн-ка надо взять след. вершину из входного списка.

Внешнее отсечение:

Аналогично внутреннему, но:

1. В качестве начальной вершины нужно брать точку
из выходного списка.
2. Список вершин отсекателя нужно просматривать
в обратном порядке.

(#20) Модели трёхмерных объектов. Требования, предъявляемые к объектам.

Геометрическая модель тела является машинным представлением его формы и размеров.

Виды геометрических моделей:

- 1) Каркасная модель ("проволочная").

Информационная часть: координаты вершин и рёбер. Не даёт представления о наимен отверстий.

- 2) Поверхностная модель.

Не даёт представления о том, на какую сторону материал, а на какую пустота.

- 3) Объёмная модель (сплошное тело).

Добавляется информация о том, где материал, а где пустота. Самая информативная.

Требования к 3D-модели:

- 1) Модель не должна противоречить исходному объекту.
- 2) Модель должна допускать возможность конструирования тела целиком (полнота модели).
- 3) Модель должна позволять вычисление геометрических характеристик тела.
- 4) Модель должна позволять проводить расчёты:
кинематические,
сопротивление материала.

Свойства объёмных моделей:

- 1) Однородность (тело заполнено изнутри).
- 2) Конечность (каждое тело занимает конечный объём).
- 3) Жёсткость (сплошное тело должно сохранять форму независимо от положения в пространстве).

Требования к программам геометрического моделирования:

- 1) Согласованность операций (все операции, проводимые над сплошными телами, должны приводить к сплошным телам).
- 2) Возможность описания (в тело должно представляться в машинном виде).
- 3) Непротиворечивость информации (в точка пространства должна принадлежать только одному телу; для в точки можно сформулировать: принадлежит ли она какому телу или нет).
- 4) Компактность модели (определяется количеством информации).
- 5) Открытость модели (разрабатываемая модель должна иметь применение при работе с различными алгоритмами).

#21 Операции преобразования в трёхмерном пространстве. Матрицы преобразований.

1. Перенос

$$M_{\text{пер}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ dx & dy & dz & 1 \end{pmatrix} \quad \begin{cases} x_1 = x + dx \\ y_1 = y + dy \\ z_1 = z + dz \end{cases}$$

2. Масштабирование

(x_m, y_m, z_m) - центр масштабирования.

$$M_{\text{масшт}} = \begin{pmatrix} k_x & 0 & 0 & 0 \\ 0 & k_y & 0 & 0 \\ 0 & 0 & k_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \begin{cases} x_1 = k_x \cdot x + (1-k_x)x_m \\ y_1 = k_y \cdot y + (1-k_y)y_m \\ z_1 = k_z \cdot z + (1-k_z)z_m \end{cases}$$

3. Поворот

(x_c, y_c, z_c) - центр поворота

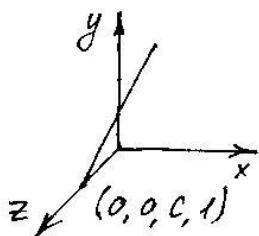
$$M_{\text{повор}_x} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \begin{cases} x_1 = x \\ y_1 = y_c + (y-y_c)\cos\theta - (z-z_c)\sin\theta \\ z_1 = z_c + (z-z_c)\cos\theta + (y-y_c)\sin\theta \end{cases}$$

$$M_{\text{повор}_y} = \begin{pmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \begin{cases} x_1 = x_c + (x-x_c)\cos\theta + (z-z_c)\sin\theta \\ y_1 = y \\ z_1 = z_c - (x-x_c)\sin\theta + (z-z_c)\cos\theta \end{cases}$$

$$M_{\text{повор}_z} = \begin{pmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \begin{cases} x_1 = x_c + (x-x_c)\cos\theta - (y-y_c)\sin\theta \\ y_1 = y_c + (x-x_c)\sin\theta + (y-y_c)\cos\theta \\ z_1 = z \end{cases}$$

4. Матрица проецирования (перспект.)

$$M_{\text{персп}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1/c \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

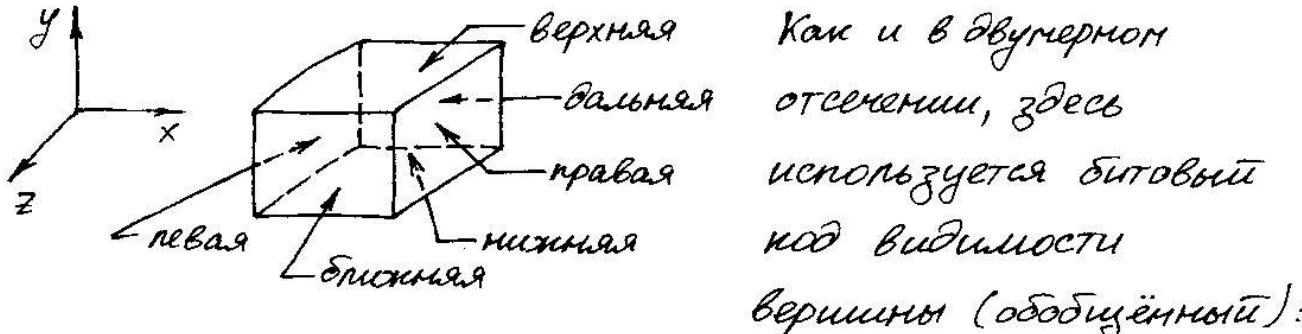


(#22) Отсечение отрезков в трёхмерном пространстве.

Алгоритм отсечения средней точкой.

Виды отсекателей:

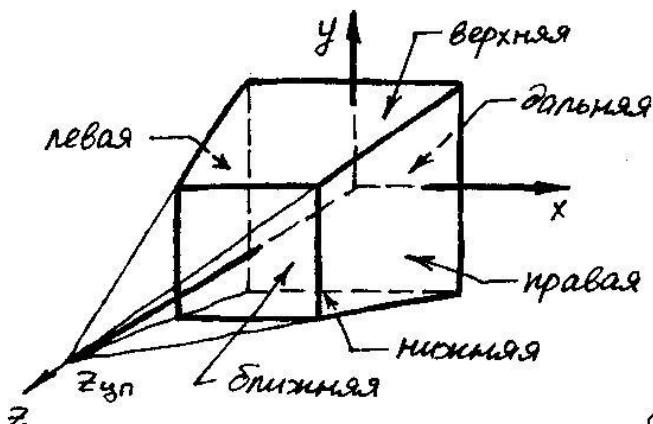
1. Прямоугольный параллелепипед.



$$T_{14} = 1, \text{ если } Z > Z_B, \text{ иначе } T_{14} = 0$$

$$T_{15} = 1, \text{ если } Z < Z_D, \text{ иначе } T_{15} = 0$$

2. Усеченная пирамида видимости.



$z_{цп}$ - центр проекции

Уравнение плоскости, проходящей через правую грань:

$$x = (Z - z_{цп}) \cdot X_{pr} / (Z_D - z_{цп})$$

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

$$\alpha_1 = X_{pr} / (Z_D - z_{цп}) \text{ и } \alpha_2 = \alpha_1 \cdot (-z_{цп})$$

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

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

$$\beta_1 = X_{pr} / (Z_D - Z_{цп}), \beta_2 = -\beta_1 \cdot Z_{цп}$$

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

$$\gamma_1 = y_B / (z_D - z_{\text{уп}}), \quad \gamma_2 = -\gamma_1 z_{\text{уп}}$$

$$f_H = y - z \tilde{\delta}_1 - \tilde{\delta}_2 \begin{cases} > 0 \Rightarrow P \text{ выше плоскости} \\ = 0 \Rightarrow P \text{ на плоскости} \\ < 0 \Rightarrow P \text{ ниже плоскости} \end{cases} *$$

$$\tilde{\delta}_1 = y_H / (z_D - z_{\text{уп}}), \quad \tilde{\delta}_2 = -\tilde{\delta}_1 z_{\text{уп}}$$

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

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

Трёхмерный алгоритм разбиения средней точкой

Аналогично двумерному, но к координатам вершин добавляется третья компонента, код видимости вершины задаётся 6-ю битами. Если отсекатель - усечённая пирамида, то для составления 6-битного кода используются промежуточные:

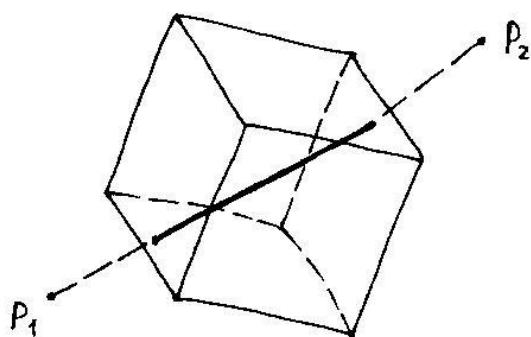
$f_L, f_R, f_B, f_H, f_B, f_D$.

#23 Отсечение отрезков в трёхмерном пространстве.
Трёхмерный алгоритм Кируса - Бека.

В трёхмерном варианте отсекатель может быть произвольным выпуклым телом.

Трёхмерный алгоритм аналогичен двумерному, только теперь векторы имеют по 3 компоненты: x, y, z .

На каждом шаге ищем точку пересечения i -ой грани и отрезка.



(#24) Определение факта выпуклости трёхмерных тел. Разбиение тела на выпуклые многоугольники.

1. Перенести тело так, чтобы одна из вершин грани совпала с началом координат.
2. Повернуть тело вокруг начала координат таким образом, чтобы одно из рёбер, выходящих из рассматриваемой вершины, совместилось с одной из коорд. осей.
3. Поворот вокруг выбранной оси коорд. такой, чтобы рассматриваемая грань тела совпала с коорд. плоскостью.
4. Для всех вершин многогранника, не лежащих в рассматриваемой плоскости, определить знаки координаты, которая \perp на рассматриваемой плоскости.

Если знаки для всех вершин одинаковы (или равны нулю), то тело выпуклое относительно рассматриваемой грани.

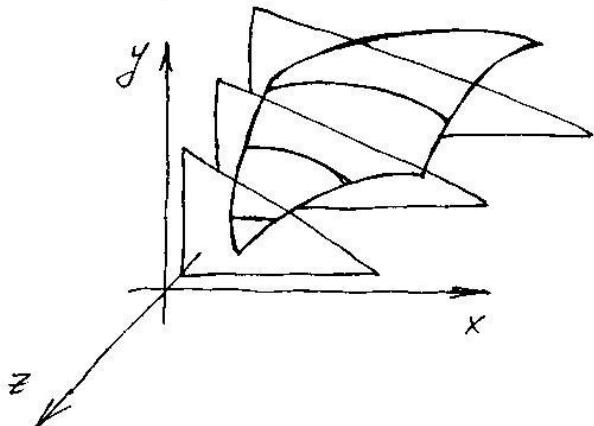
Если тело выпукло относительно всех своих граней, то оно и в целом выпуклое.

Если знаки в целом не совпадают, то тело невыпукло. Если все эти знаки равны нулю, то тело вырожденное. Если тело не выпуклое, то плоскость, проходящей через рассматриваемую грань, мы разделяем тело на составляющие части. Рассматривая эту процедуру применяем к каждой из полученных частей.

5. Вектор внутр. нормали к рассм. грани (для вып. тела) будет иметь 2 ненулевые компоненты (те, кот. лежат в пл-ти этой грани), а третья будет совпадать с осью, 1-ной рассм. грани ($u \neq 0$). Знак этой компоненты совпадает со знаком любой вершины, не лежащей в данной пл-ти. Получим вектор внутренней нормали в повернутой системе координат. Для вычисления его в исх. системе координат надо выполнить обратные преобразования.

#2S Алгоритм плавающего горизонта

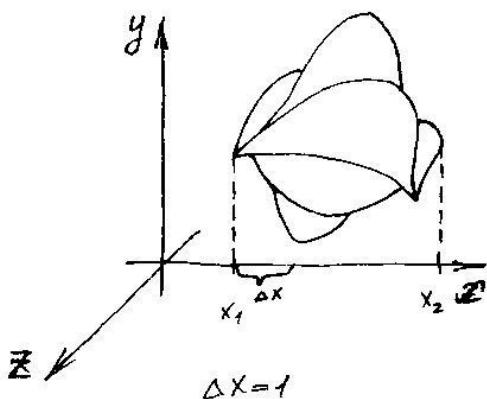
$F(x, y, z) = 0$ — поверхность в 3-мерном пространстве.



$$z = \text{const}$$

$$y = f(x, z = \text{const})$$

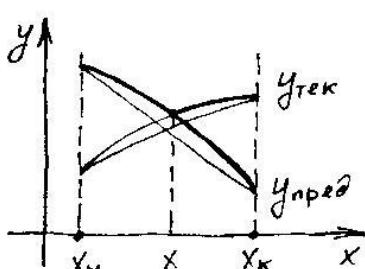
$$x = \varphi(y, z = \text{const})$$



y_{\max} — массив значений верхнего горизонта.

y_{\min} — массив значений нижнего горизонта.

$y_{\min}(x_i) < y(x_i) < y_{\max}(x_i)$ — точка невидимая, иначе — видима.



$$m_{\text{пред}} = \frac{y_{\text{пред}k} - y_{\text{пред}H}}{x_k - x_H}$$

$$m_{\text{тек}} = \frac{y_{\text{тек}k} - y_{\text{тек}H}}{x_k - x_H}$$

$$y_{\text{пред}H} + m_{\text{пред}}(x - x_H) = y_{\text{тек}H} + m_{\text{тек}}(x - x_H)$$

$$x_{\text{неп}} = x_H + \frac{y_{\text{тек}H} - y_{\text{пред}H}}{m_{\text{пред}} - m_{\text{тек}}} = x_H + \frac{\Delta x (y_{\text{тек}H} - y_{\text{пред}H})}{\Delta y_{\text{пред}} - \Delta y_{\text{тек}}}$$

$$y_{\text{неп}} = y_{\text{пред}H} + m_{\text{пред}}(x_{\text{неп}} - x_H)$$

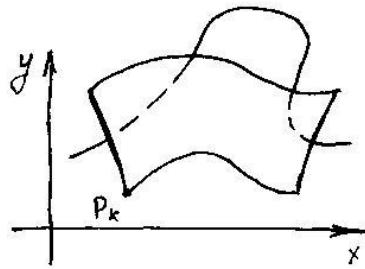
Обработка базовых рёбер

1. Левое: если P_k — первая точка первой кривой, то эту точку запоминаем в качестве предыдущей точки: P_{k-1} .
Если P_k — первая точка на непрвой кривой, то ~~запоминаем~~

создаём P_k и P_{k-1} , т.е. создаёт боковое ребро и запоминаем текущую точку P_k в качестве P_{k-1} .

Правое: аналогично.

В массивах верхнего и нижнего горизонтов занесём ординаты бокового ребра.



Суть Алгоритма

Для каждой текущей плоскости $z = \text{const}$ выполнить:

- обработать левое боковое ребро;
- для каждой точки, лежащей на очередной ~~прямой~~ кривой, проверить условие: $y_{\min}(x_i) < y(x_i) < y_{\max}(x_i)$. Если оно истинно, то кривая невидима в этой точке, иначе — видима;
- если на участке от $x_{\text{лев}}$ до $x_{\text{тек}}$ видимость кривой изменилась, то вычислить $x_{\text{пересеч}}$;
- если на текущем сегменте от $x_{\text{лев}}$ до $x_{\text{тек}}$ участок кривой полностью видим, то изобразить его цветом;
- Если сегмент кривой стал невидим, то изобразить участок от $x_{\text{лев}}$ до $x_{\text{пересеч}}$;
- Если кривая стала видимой, то изобразить участок от $x_{\text{пересеч}}$ до $x_{\text{тек}}$;
- заполнить массивы горизонтов;
- обработать правое бок. ребро.

$$y_{\max}(x_i) = \max(y_{\max}(x_i), y(x_i))$$

$$y_{\min}(x_i) = \min(y_{\min}(x_i), y(x_i))$$

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

Основные этапы Алгоритма Роберта.

Задача удаления невидимых линий и поверхностей является одной из наиболее сложных в машинной графике. Алгоритмы удаления невидимых линий и поверхностей служат для определения линий рёбер, поверхностей или объёмов, которые видны или невидимы для наблюдателя, находящегося в заданной точке пр-ва.

Единого и наилучшего способа удаления не существует для этой, одной из центральных задач машинной графики. Сложность задачи привела к появлению большого числа различных способов её решения. Многие из них ориентированы на специализированные приложения.

Все алгоритмы удаления невидимых линий (поверхностей) включают в себя сортировку. Главная сортировка ведётся по геометрическому расположению от тела, поверхности, ребра или точки до точки наблюдения. Эффективность этих алгоритмов зависит в большей мере от эффективности процесса сортировки.

Существует тесная взаимосвязь между скоростью работы алгоритма и детальностью его результата.

Ни один из алгоритмов не может достичнуть хороших оценок для двух этих показателей одновременно.

(Ещё есть сортировка по загораживанию тел друг другом)

Алгоритмы удаления делятся на две группы:

- 1) работающие в объектном пр-ве (мировая система координат; высокая точность);
- 2) работающие в пр-ве изображений (система координат связана с дисплеем; точность ограничена разрешающей способностью экрана; например, при увеличении изображения во много раз могут не совпасть концы отрезков).

Сложность для алгоритма, работающего в объектном пр-ве, $\sim N^2$, где N - количество объектов;

для алгоритма, работающего в пр-ве изображений, $\sim N \cdot M$, где M - число пикселов.

$N^2 < N \cdot M$, но алгоритмы, работающие в пр-ве изображений на практике могут быть эффективнее в силу свойства когерентности при растровой реализации (рядом расположенные пиксели чаще всего будут обладать одинаковыми свойствами).

Алгоритм Роберса. Основные этапы и матем. предпосылки

Удаление невидимых линий (в объектном пр-ве).

Работает только с выпуклыми телами.

Идея сортировки тел по мере их удалённости от наблюдателя. Сложность $\sim N^2$.

Этапы работы:

I. Подготовка исх. данных.

II. Удаление линий, экранируемых самим телом (для одного тела на этом работа заканчивается).

III. Удаление линий, экранируемых другими телами.

IV. Удаление линий пересечения тел, экранируемых самими телами, связанными отношением прорывания, и др. телами.

#27 Алгоритм Роберта. Формирование матрицы тела. Удаление ненужных плоскостей.

Требуется, чтобы все изображаемые тела были выпуклыми.
Невыпуклые должны быть разбиты на выпукл. части.

Уравнение плоскости: $ax + by + cz + d = 0$.

Выпуклое многогранное тело можно представить, как пересечение наборов пересекающихся плоскостей.

$(x, y, z, 1) \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = 0$ — уравнение плоскости в матр. форме.

$[V] = \begin{bmatrix} a_1 & \dots & a_n \\ b_1 & \dots & b_n \\ c_1 & \dots & c_n \\ d_1 & \dots & d_n \end{bmatrix}$ — тогда это матрица $4 \times n$ — матрица тела, состоящая из коэф-тов.

$[S] = [x \ y \ z \ 1]$ — точка, лежащая внутри тела.

Матрица тела должна быть составлена так, чтобы для $[R] = [S][V]$ — для $[R]$ каждая его компонента $r_i > 0$.

Точку, лежащую внутри тела, можно получить умножением координат тела.

Если мат в $[R]$ какой-то компонент $r_i < 0$, то матрица тела $[V]$ корректируется умножением i -го столбца на -1 .

Если для $ax + by + cz + d$ коэф-ты неизвестны, то их (зная координаты 3-х ее точек плоскости) можно найти так:

$$\begin{cases} ax_1 + by_1 + cz_1 + d_1 = 0 \\ ax_2 + by_2 + cz_2 + d_2 = 0 \\ ax_3 + by_3 + cz_3 + d_3 = 0 \end{cases} \sim a'x_1 + b'y_1 + c'z_1 = -1 \quad (a' = \frac{a}{d} \text{ и т.д.})$$

$$\begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} \Rightarrow [X][C] = [D] \Rightarrow [C] = [X]^{-1}[D]$$

Зная все координаты вершин соотв. грани, коэф-ты вычисл.

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

Зная уравнение нормали для плоскости $\bar{n} = a\bar{i} + b\bar{j} + c\bar{k}$, сразу видим a, b, c , а $d = -(ax_1 + by_1 + cz_1)$ для извешн. точки.

Если мы преобразуем тело матрицей преобразования $[T]$:

$[B]$ — матрица вершин ($m \times 4$)

$[V]$ — матрица тела (коэф-тov, $n \times n$)

$$[B][V] = [D]$$

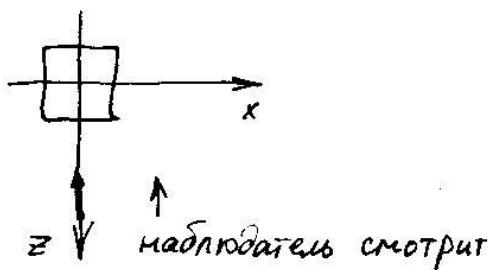
$[BT] = [B][T]$ — матрица вершин преобр. тела.

$[VT]$ — матрица коэф-тov преобр. тела.

$$[BT][VT] = [D]$$

$$[B][T][VT] = [B][V] \Rightarrow [VT] = [T]^{-1}[V]$$

Удаление невидимых плоскостей



$[E] = [0, 0, -1; 0]$ — вектор наблюдения.

У вектора $[E_i] = [E] \cdot [V]$ отрицательные компоненты будут соответствовать невидимым граням.

$[E] \cdot \bar{n}_{\text{внутрi}} > 0$ — i-ая грань видима.

$[E] \cdot \bar{n}_{\text{внутрi}} < 0$ — i-ая грань невидима.

$[E] \cdot \bar{n}_{\text{внутрi}} = 0$ — i-ая грань на грани видимости.

Невидимые рёбра обраузутся пересечением невидимых граней. Поэтому, удалив невидимые грани, мы удалим невидимые рёбра. Других невид. рёбер быть не может.

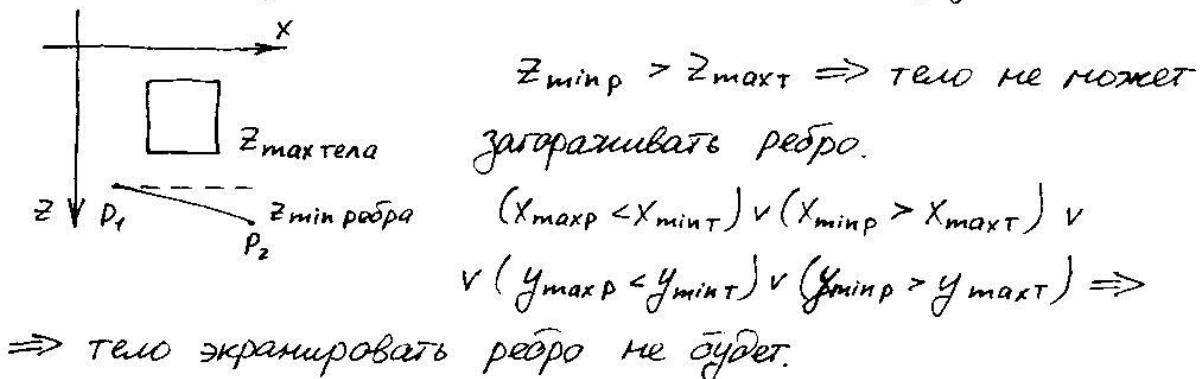
#28 Алгоритм Робертса. Удаление отрезков, экранируемых другими телами.

Тело, рёбра которого проверяются на предмет экранирования другими телами, — промежуточный объект. С которым сравнивается — прямое тело.

Отсортировать все тела по максимальному Z тела:

$$Z_{\max 1} < Z_{\max 2} < \dots < Z_{\max n}; \quad Z - \text{расст. до наблюдателя.}$$

Наиболее удалённое тело — более экранирующее.



Удаление экранируемых участков тела отрезка

P_1, P_2 — исходящий отрезок.

$$P(t) = P_1 + (P_2 - P_1)t; \quad 0 \leq t \leq 1 \quad \text{— параметрическое уравнение отрезка}$$

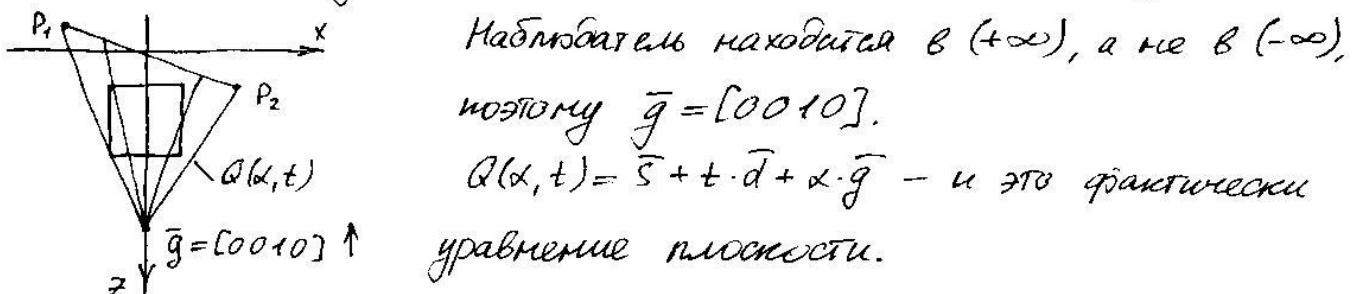
$$\bar{V} = \bar{s} + t \cdot \bar{d}, \quad \text{где } \bar{V} \text{ — вектор точки на отрезке,}$$

\bar{s} — начальная точка, \bar{d} — направление.

Нужно найти t , при которых изменяется видимость отрезка.

Параметрически зададим отрезок от точки

$$\bar{V} \text{ до точки } \bar{g} \text{ — точки наблюдения: } Q(x, t) = \bar{V} + x \cdot \bar{g}, \quad 0 \leq x$$



Скалярное произведение \bar{V} точки, лежащей внутри тела, на матрицу коэф-тов ненулевым:

Найдём такие α и t , для которых:

$$\bar{H} = (\bar{S} + t \cdot \bar{d} + \alpha \cdot \bar{g}) \cdot [V] > 0$$

$$\bar{H} = \bar{S} \cdot [V] + t \cdot \bar{d} \cdot [V] + \alpha \cdot \bar{g} \cdot [V] > 0$$

Если все компоненты \bar{H} для некоторых t и α неотрицательны, то отрезок при этих t эквивалентен галом.

Обозначим: $\bar{P} = \bar{S} \cdot [V]$; $\bar{q} = \bar{d} \cdot [V]$; $\bar{w} = \bar{g} \cdot [V]$.

Тогда: $h_j = p_j + tq_j + \alpha w_j > 0$, $0 \leq t \leq 1$, $0 \leq \alpha$, а j - номер грани.

Т.е. нужно решить задачу линейного программирования при n неизвестных и m ограничениях (р-ва, неравенства)

$$\left\{ \begin{array}{l} \sum_{i=1}^n a_{ri} x_i \leq b_r, \\ \dots \\ \sum_{i=1}^n a_{mi} x_i \leq b_m \end{array} \right. \quad L = \sum_{i=1}^n c_i x_i \rightarrow \min$$

$h_j > 0$, t - целевая ф-ция

Решать $h_j = 0$. Получить все возможные пары систем 2-х уравнений (n граней) \Rightarrow число решений $\frac{n(n-1)}{2}$.

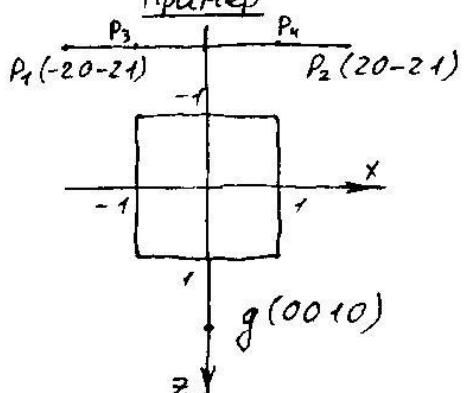
Учитывая ограничения: $0 \leq \alpha$, $1 \geq t$, $t \geq 0$ - получим $\frac{(n+3)(n+2)}{2}$ возможных решений.

Сначала мы находим α и t для одной пары ур-ий и подставляем в остальные.

Затем найдём t_{\max} - максимальное среди максимальных и t_{\min} - минимальное среди максимальных.

Отрезок невидим при $t_{\max} < t < t_{\min}$.

Пример



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

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

$$P = \bar{S} \cdot [V] = (-13 11 -13)$$

$$w = \bar{g} \cdot [V] = (00001-1)$$

$$q = \bar{d} \cdot [V] = (4-40000)$$

$$h_j = p_j + t q_j + \alpha w_j > 0 \Rightarrow$$

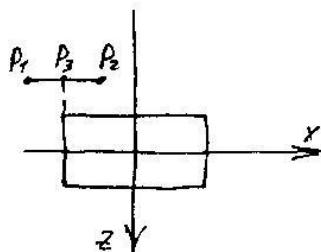
$$\Rightarrow \begin{cases} -1 + 4t > 0 \\ 3 - 4t > 0 \\ 1 > 0 \\ 1 > 0 \\ -1 + \alpha > 0 \\ 3 - \alpha > 0 \end{cases}$$

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

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

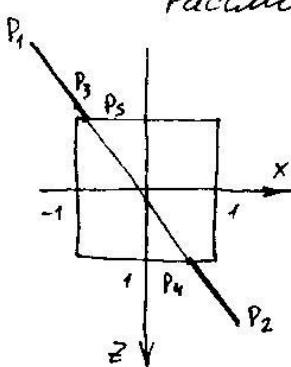
$P_3 P_4$ — невидимая часть.

Если ситуация такая:



то нужно будет учесть $t \leq 1$. В др. случаях ост. И здесь $P_4 = P_2$.

Рассмотрим случай, когда отрезок протыкает тело:

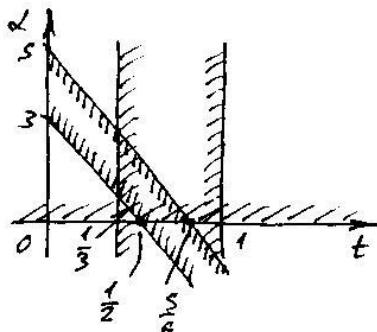


$$d = (3060) \Leftrightarrow P_1(-20-41) \text{ и } P_2(1021)$$

$$S = (-20-41), \quad p = \bar{S}[V] = (-1311-35)$$

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

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



Мы учли доп. условие $\alpha \leq 0$

$$P_3 = (-20-41) + \frac{1}{3}(3060) = (-10-21)$$

$$P_4 = (-20-41) + \frac{5}{6}(3060) = (0,50-11)$$

Точки протыкания находятся при $x = 0$

$$t_{np1} = \frac{1}{2}, \quad t_{np2} = \frac{5}{6}.$$

$$P_5 = (-20-41) + \frac{1}{2}(3060) = (-0,50-11), \quad P_6 = P_4.$$

Можно заранее (не решая систему) определить факт полной видимости отрезка:

$P_j < 0$ - 1-ая вершина отрезка лежит по внешнюю сторону от j -грани.

$P_j + q_j < 0$ - 2-ая вершина отр. лежит по внеш. сторону от j -грани.

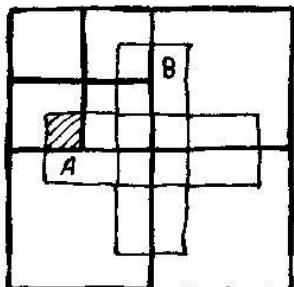
$w_j \leq 0$ - ~~точка наблюдения~~ лежит j -ая грань видима.

$w_j \leq 0$ и $P_j \leq 0$ - один конец отрезка лежит или на видимой плоскости или между видимой плоскостью и точкой наблюдения.

$w_j \leq 0$ и $P_j \leq 0$ и $P_j + q_j \leq 0$ для $\forall j$ гарантирует, что отрезок полностью видим.

Можно определить, что оба конца отрезка позади невидимой плоскости, но невозможно определить, будут ли при этом концы отрезка позади тела.

#29 Удаление невидимых линий и поверхностей в пространстве изображений. Алгоритм Варнока (разбиение окнами): последовательность действий и основные принципы.



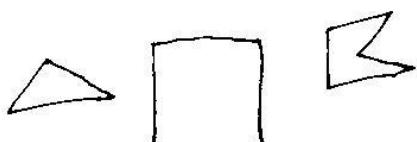
Одной единственной версии этого алгоритма не существует.

Окно разбивается на части, если оно не пусто. Это продолжается до тех пор, пока не получим окно в 1 пикселе (в простейшем случае алгоритма Варнока).

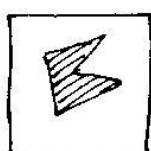
В более сложных версиях этого алгоритма на этапах, когда окно имеет размер > 1 пикселя, ставится вопрос о том, что изображается в этом окне.

Следует дать на подокна, если можно сказать, что изображать в окне.

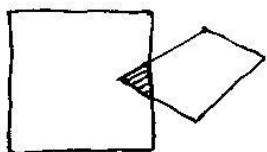
1. Все многоугольники являются внешними \Rightarrow окно закрасить цветом фона.



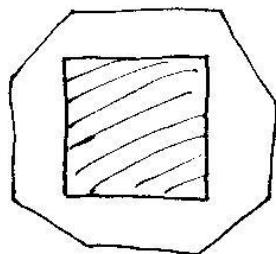
2. Ми-к, связанный с окном, является внутренним:
закрасить окно цветом фона;
растровая развертка ми-ка.



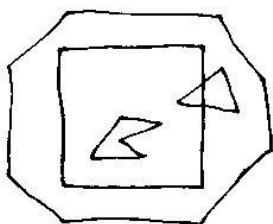
3. Ми-к, связанный с окном, является пересекающим:
отсечение ми-ка по границе окна;
далее работа, как в п.2.



4. С окном связан один охватывающий ми-к \Rightarrow
 \Rightarrow всё окно закрасить цветом ми-ка.



5. С окном связаны внутренние, пересекающие ми-ки,
и есть хотя бы один охватывающий, расположенный ближе
всех остальных по отношению к наблюдателю.

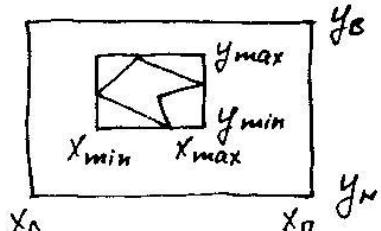


Рекомендуемая последовательность действий:

1. Проводится простейший габаритный тест с правоцнг. обложкой;
определяется как можно большее кол-во пустых окон и окон
с единств. внутр. ми-ком. Найденные окна сразу изображаются;
2. Выполнение теста с целью определения окон, пересекаемых
единственным ми-ком;
3. Тест с целью распознавания внешних и охвач. ми-ков.
Получают новые пустые окна и окна, охватываемые евр. ми-ком;
4. Можно либо проводить разбиение окна на подокна, либо
проводить ещё один тест на обнаружение ми-ка (охватываю-
щего), лежащего ближе к наблюдателю.

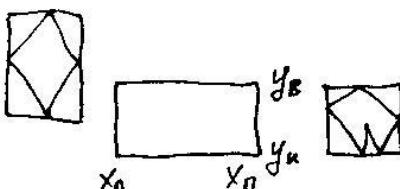
(#30) Типы многоугольников, анализируемых в алгоритме Варнока. Методы их идентификации.

Определение внутреннего мн-ка



$$(x_{\min} > x_n) \wedge (x_{\max} < x_n) \wedge (y_{\min} > y_n) \wedge (y_{\max} < y_B) \\ \Rightarrow \text{мн-к полностью виден снаружи.}$$

Определение некоторых внешних мн-ков

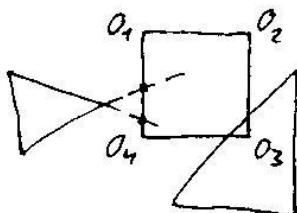


$$(x_{\min} > x_n) \vee (x_{\max} < x_n) \vee (y_{\min} > y_B) \vee (y_{\max} < y_n) \\ \Rightarrow \text{мн-к внешний}$$

С помощью этого теста мы не распознаем такой случай:



Определение пересекающих



Составить ур-я прямых, проходящих через каждое ребро мн-ка:

$$f_{np_i} = A_i x + B_i y + C_i$$

* Нужно еще проверить принадлежность точки пересечения ребру.

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

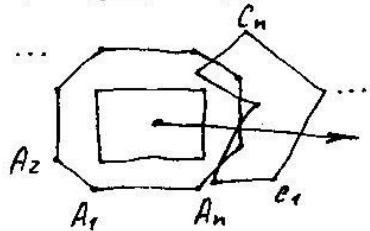
$$b = y_1 - m x_1$$

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

Координаты вершин окна подставим в f_{np_i} . Если знак f_{np_i} не зависит от выбора вершин окна, то все его вершины лежат на одну сторону от несущей прямой и точек пересечения нет. Если знаки различаются, то мн-к ^{с чистотой} * пересекает окно.

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

Тест с бесконечным лучом (внеш. или охватывающий)

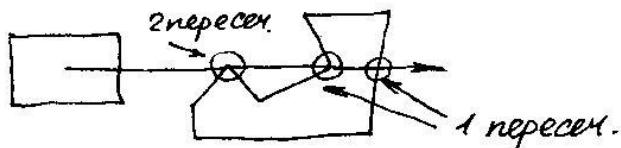


m - кол-во пересечений луча с рёбрами.

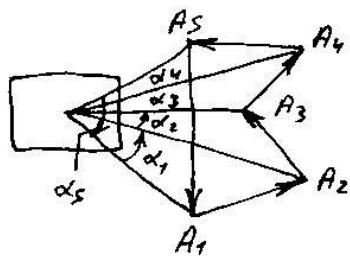
m - чётное \Rightarrow мн-к внешний;

m - нечётное \Rightarrow мн-к охватывающий.

Частный случай:



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



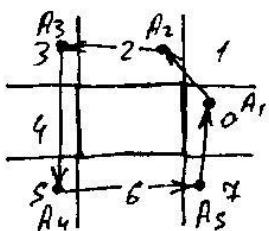
$$\sum_{i=1}^n \alpha_i = 0 \Rightarrow \text{мн-к внешний}$$

$$\sum_{i=1}^n \alpha_i = \pm 360^\circ m \Rightarrow \text{мн-к охватывающий},$$

m - кол-во охватов.

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

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



$\Delta\alpha$ = (номер октанта, в котором расп. конец вектора) - (номер октанта, в кот. расп. начало вектора).

$$\Delta\alpha > 4 \Rightarrow \Delta\alpha = \Delta\alpha - 8; \Delta\alpha < -4 \Rightarrow \Delta\alpha = \Delta\alpha + 8$$

$\Delta\alpha = 4 \Rightarrow$ ребро мн-ка расщепляется ребром окна

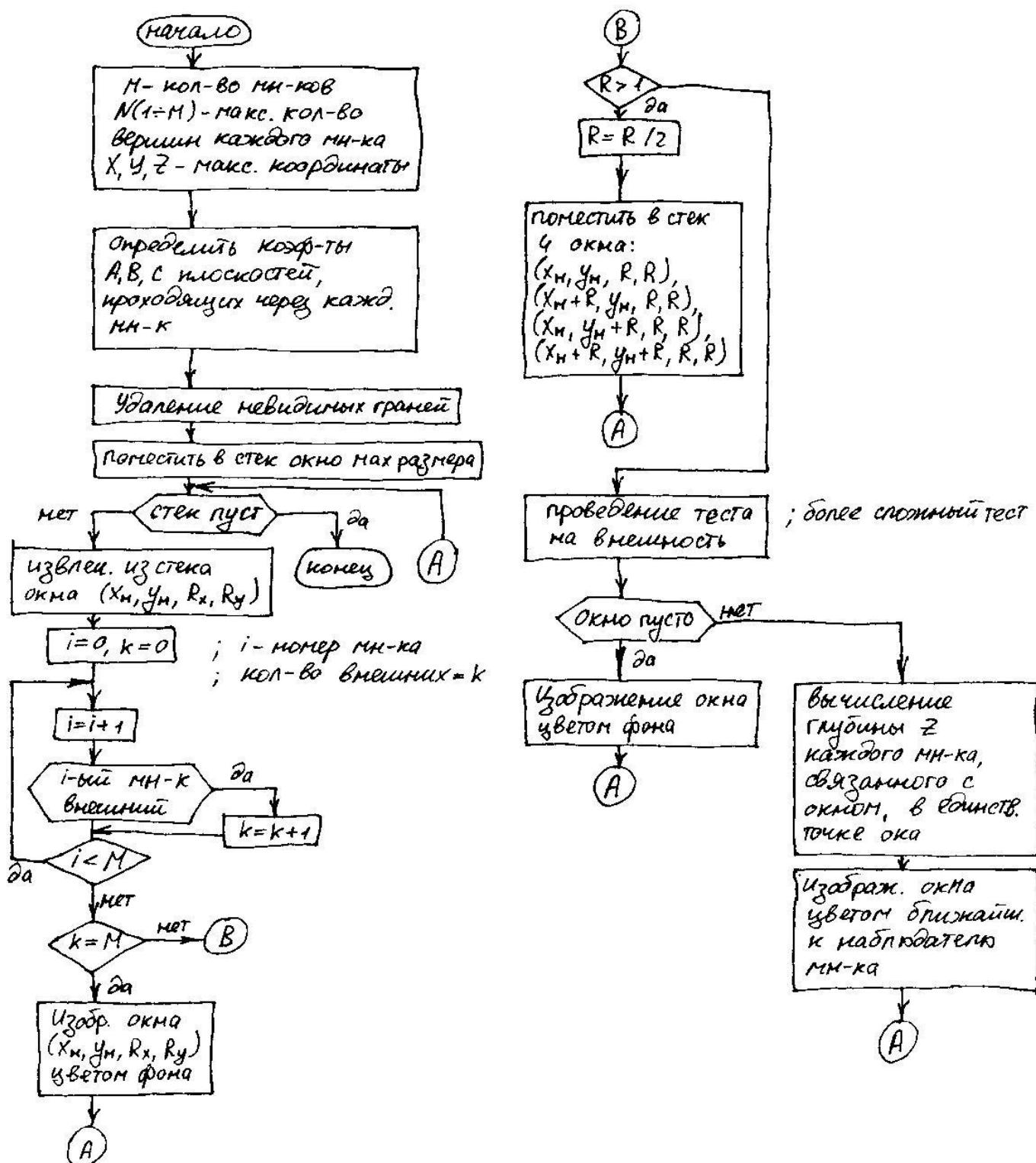
$$\sum_{i=1}^n \Delta\alpha_i = \pm 8m \Rightarrow \text{мн-к охватывающий};$$

$$\sum_{i=1}^n \Delta\alpha_i = 0 \Rightarrow \text{мн-к внешний}.$$

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

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

Алгоритм Варнока



#31 Алгоритм Вейпера-Азертсона удаления невидимых линий и поверхностей.

Вейпер и Азертон попытались минимизировать кол-во шагов в алгоритме разбиения типа алгоритма Варноса путём разбиения окна вдоль границы мн-ка. Основой этого алгоритма послужил алгоритм отсечения мн-ков. Выходные данные - мн-ки.

Алгоритм:

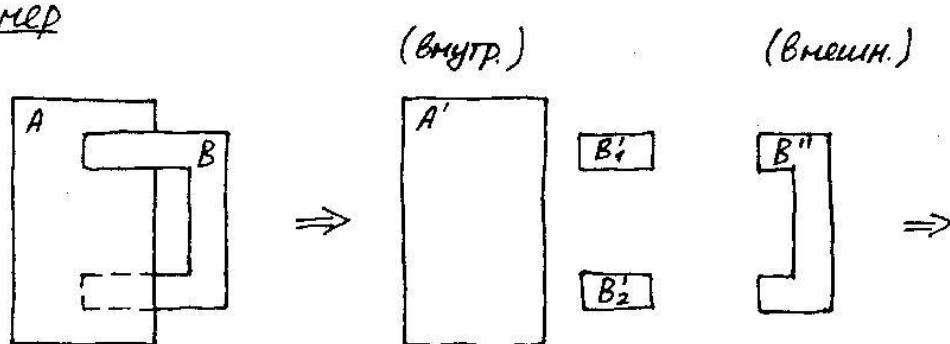
1. Предварительная сортировка по глубине (для формирования списка приближенных приоритетов).
2. Отсечение по границе ближайшего к наблюдателю мн-ка, называемое сортировкой мн-ков на плоскости. (В качестве отсекателя используется копия первого мн-ка из списка приближенных приоритетов). Отсекаться будут все мн-ки в этом списке, включая первый (в отсечении участвуют проекции мн-ков). Формируются 2 списка: внутренний (для каждого отсекаемого мн-ка та часть, которая оказывается внутри отсекателя) и внешний (для каждого мн-ка оставшаяся часть)).
3. Удаление мн-ков внутреннего списка, которые экранируются отсекателем.
4. Если глубина мн-ка из внутреннего списка больше, чем ϵ_{min} отсекателя, то такой мн-к частично экранирует отсекатель. Тогда нужно рекурсивно подразделить плоскость (x, y) , используя мн-к, нарушивший порядок, в качестве отсекателя (для отсекателя нужно использовать копию исходного

мн-ка, а не остаток после предыдущего отсечения).
Отсечения подлежат мн-ки из внутреннего списка.

5. По окончании отсечения или рекурсивного подразбиения изображаются мн-ки из внутреннего списка (те, которые остались после удаления всех экранируемых на каждом шаге мн-ков — остаются только отсекающие мн-ки).

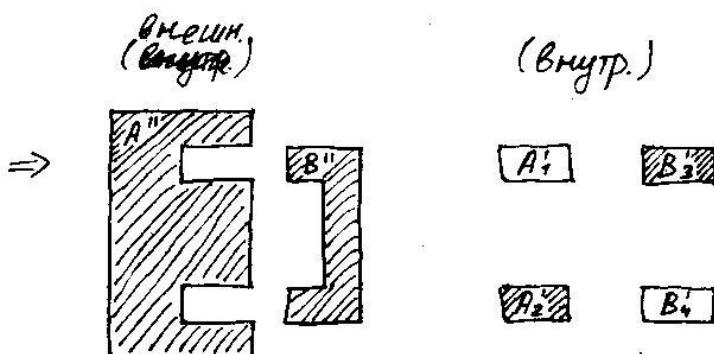
6. Работа продолжается с внешним списком (шаги 1-5).

Пример



$$z_{\max} A > z_{\max} B$$

$$z_{\max} B'_1 > z_{\min} A'$$



$$z_{\max} A'_1 < z_{\min} B'_3$$

$$z_{\min} A'_2 > z_{\max} B'_4$$

Если мн-ки пересекаются, то для корректной работы этого алгоритма нужно тяжкостью одного мн-ка разбить другой на две части.

(#32) Алгоритм, использующий Z-буфер.

Одни из простейших алгоритмов удаления невидимых поверхностей.

Буфер кадра используется для запоминания атрибутов (интенсивности) каждого пикселя в пространстве изображения.

Z-буфер - это отдельный буфер глубины, используемый для запоминания координаты Z каждого видимого пикселя в пространстве изображения.

Достоинства:

1. Простота.

Сцены могут быть любой сложности.

2. Элементы сцены не нужно сортировать, т.к. их можно заносить в буфер в любом порядке. Поэтому экономится вычислительное время.

Недостатки:

1. Большой объём памяти.

Буфер кадра $512 \times 512 \times 24$ бит требует в комбинации с Z-буфером размера $512 \times 512 \times 20$ бит около 1,5 Мб памяти.

2. Трудоёмкость и высокая стоимость устранения листингового эффекта.
3. Трудоёмкость реализации эффектов прозрачности и просвечивания.

Вычисление Z:

Уравнение плоскости: $Ax + By + Cz + D = 0 \Rightarrow$

$$\Rightarrow z = -\frac{Ax + By + D}{C} \text{ для } C \neq 0 \text{ и } z = 0 \text{ для } C = 0.$$

Для сканирующей строки: ($y = \text{const}$) $z_2 - z_1 = \frac{-Ax_2}{C} + \frac{Ax_1}{C} = -\frac{A}{C} \times 1 \times v - 1 \Rightarrow z_2 - z_1 = \dots \dots z_{i-1} - z_i = -\frac{A}{C}$

Алгоритм:

1. Заполнить буфер кадра фоновым цветом.
2. Заполнить Z -буфер минимальным значением глубины.
3. Преобразовать каждый ми-к в растровую форму в произвольном порядке:
 - a) для каждой точки экрана, покрытой ми-ком вычислить значение $z(x, y)$;
 - b) Если $z(x, y) > z_{\text{буф}}(x, y) \Rightarrow z_{\text{буф}}(x, y) = z(x, y)$,
 $\text{цвет}(x, y) = \text{цвет текущего ми-ка}.$

Замечание:

Этот алгоритм можно применять для построения сечений поверхностей:

$(z(x, y) > z_{\text{буф}}(x, y) \text{ и } z(x, y) \leq z_{\text{сечения}})$ — условие.

$z_{\text{сечения}}$ — глубина сечущей плоскости. Эффект заключается в том, что остаются только такие элементы поверхности, которые лежат на самом сечении или позади него.

(#33) Алгоритм, использующий список приоритетов.

В основе алгоритма лежит способ изображения сцен от дальних объектов к ближним.

Алгоритм:

1. Отсортировать мн-ки сцен в порядке возрастания:

$$Z_{\min}(A) < Z_{\min}(B) < \dots < Z_{\min}(P) < Z_{\min}(Q).$$

2. Построить самый дальний мн-к, если он не экранирует другие мн-ки (если $Z_{\max}(A) < Z_{\min}(B)$).
3. Проверить, экранирует ли мн-к Р мн-к Q при $Z_{\max}(P) > Z_{\min}(Q)$:

а) Верно ли, что прямоугольные объемлющие оболочки P и Q не перекрываются по X?

б) Верно ли, что прямоугольные объемлющие оболочки P и Q не перекрываются по Y?

в) Верно ли, что Р целиком лежит по ту сторону плоскости, несущей Q, которая расположена дальше от точки наблюдения?

г) Верно ли, что Q целиком лежит по ту сторону плоскости, несущей Р, которая расположена ближе к точке наблюдения?

д) Верно ли, что проекции Р и Q не перекрываются?

Как только какой-либо тест даёт положительный ответ, Р заносится в буфер кадра (Р не экранирует Q). Если все ответы отрицательны, то Р и Q меняются местами в списке, позиция Q поменяется. Тесты нужно повторить с новым списком.

Если сделана попытка вновь представить Q, значит, обнаружена ситуация циклического экранирования.

В этом случае Р разрезается плоскостью, несущей Q, исходный мн-к Р удаляется из списка, а его части заносятся в список. Тесты нужно повторить с новым списком.

Замечание:

Проверку для пунктов 3.8 и 3.9 можно проводить с помощью пробной ф-ции:

$$f_{\text{пр}}(x, y, z) = Ax + By + Cz + D, \text{ где } A, B, C - \text{коэф-ты}$$

пробной плоскости U.

В $f_{\text{пр}}$ подставляются координаты вершин используемого мн-ка W.

Если знаки $f_{\text{пр}}$ для всех вершин W совпадают и положительны или равны нулю, то мн-к W находится с дальней (невидимой) стороны от ~~одной~~ плоскости U.

Если знаки $f_{\text{пр}}$ для всех вершин W совпадают и неположительны, то мн-к W расположен с ближней (видимой) стороны от плоскости U.

#34 Алгоритм построчного сканирования, использующий Z-буфер. Интервальные методы построчного сканирования.

Алгоритмы Варнока, Z-буфера и отрицательного списка приоритетов обрабатывают элементы сцены или мн-ка в порядке, который не связан с процессом визуализации. Алгоритмы построчного сканирования обрабатывают сцену в порядке прохождения сканирующей строки.

Алгоритм:

1. Подготовка информации:

- Для каждого мн-ка определить самую верхнюю сканирующую строку, которую он пересекает.
- Занести мн-к в группу U , соответствующую этой сканирующей строке.
- Заполнить для каждого мн-ка:

ΔU - число скан. строк, пересекающих этот мн-к;
список рёбер этого мн-ка;
коэф-ты A, B, C, D уравнения плоскости мн-ка;
визуальные атрибуты мн-ка.

2. Решение задачи удаления невидимых поверхностей:

- Инициализировать буфер кадра дисплея.
- Для каждой сканирующей строки:
 - Инициализировать буфер кадра размером с одну скан. строку, заполнив его фоновым значением;
 - Инициализировать Z-буфер размером с одну скан. строку значением Z_{min} .
 - Проверить необходимость добавления в список активных мн-ков (SAM) новых мн-ков.

- 2) Если было добавление ми-ка в САМ, то добавить в САР соотв. ребра новых ми-ков.
- 3) Если произведено удаление какого-либо элемента из пары рёбер САР, то проверить необходимость удаления всего ми-ка из САМ. Если ми-к остается, то проверить необходимость удаления другого ребра из этой пары — если его удалять не нужно, то дополнить пару (добавив недостающее левое или правое ребро).

2.3) В САР должна храниться следующая инф-ция:

x_l — пересечение левого ребра пары с текущ. скан. строкой;

Δx_l — приращение x_l в интервале между соседними скан. строками;

Δy_l — число скан. строк, пересекаемых левым ребром;

$x_n; \Delta x_n; \Delta y_n;$

$\Delta z_x = -\frac{A}{C}$ для $C \neq 0$ ($\Delta z_x = 0$ для $C=0$) — приращение по Z вдоль скан. строки;

$\Delta z_y = -\frac{B}{C}$ для $C \neq 0$ ($\Delta z_y = 0$ для $C=0$) — приращение по Z в интервале между соседними скан. строками.

2.4) Для каждой пары рёбер ми-ка из САР выполнить:

- а) извлечь эту пару из САР;
- б) инициализировать Z со значением z_n ;
- в) Для каждого пикселя $x_l \leq x \leq x_n$ вычислить $Z(x, y=\text{const})$:

$$z_1 = z_n, \dots, z_k = z_{k-1} + \Delta z_x;$$

- 2) Если $Z(x, y=\text{const}) > Z_{\text{буф}}(x, y=\text{const})$, то $Z_{\text{буф}} = Z$ и ещё занести атрибуты ми-ка в буфер кадра;

2.5) Записать буфер кадра скан. строки в буфер кадра дисплея.

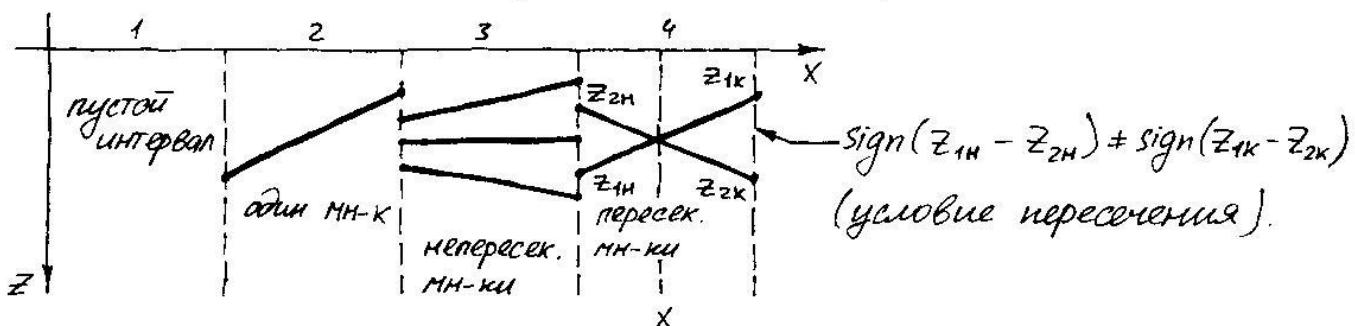
2.6) Скорректировать САР:

- $\Delta Y_n = \Delta Y_1 - 1$ и $\Delta Y_0 = \Delta Y_n - 1$;
 - $X_n = X_1 + \Delta X_1$ и $X_0 = X_n + \Delta X_n$;
 - $Z_n = Z_1 + \Delta Z_x \cdot \Delta X + \Delta Z_y$;
- 2) Если $\Delta Y_1 < 0$ или $\Delta Y_n < 0$, то удалить соотв. ребро из списка, пометив положение обоих рёбер в списке и породивший их МН-К.
- 2.7) Скорректировать САМ:
- $\Delta Y = \Delta Y - 1$;
 - Если $\Delta Y < 0$, то удалить МН-К из САМ.

Интервальные методы построчного сканирования

В алгоритме построчного сканирования с использованием Z-буфера ширина МН-Ка вычисляется для каждого пикселя на скан. строке. Кол-во вычислений можно сократить, если использовать понятие интервалов.

Решение задачи удаления невидимых поверхностей сводится к выбору видимых отрезков в каждом из интервалов, полученных путём деления скан. строки проекциями точек пересечения рёбер.



- 1) Изобразить фон.
- 2) Изобразить атрибуты МН-Ка, соответствующие этому отрезку.
- 3) Изобразить атрибуты МН-Ка, соответствующие отрезку с MAX Z.
- 4) $sign(z_{1H} - z_{2H}) + sign(z_{1K} - z_{2K}) \Rightarrow$ разбить интервал точкой пересечения .. и выбрать тот, для которого

#3S) Основные этапы алгоритма Робертса
и их содержание

① Подготовка исходных данных:

- а) сформировать списки граней и рёбер для каждого тела исходя из списка вершин;
- б) вычислить коэф-ты ур-ний пл-тей, проходящих через грани;
- в) сформировать матрицу тела;
- г) проверить матрицу на корректность её составления (взять произвольную точку внутри тела);
- д) умножить вектор координат внутр. точки на матрицу тела, в полученном векторе найти отриц. компоненты, умножить соответствующ. столбцы матрицы на -1;
- е) если после формирования матр. происходит преобразование тела, то нужно эту матрицу слева умножить на обратную матрицу преобразований;
- ж) вычислить $X_{min}, X_{max}, Y_{min}, Y_{max}, Z_{min}, Z_{max}$.

② Удаление невидимых плоскостей:

- а) определить положение наблюдателя (обычно он находится в $+\infty$ и смотрит в сторону нач. коорд.);
- б) вектор взгляда наблюдателя умножить на матр. тела, в полученнем векторе отриц. компоненты соответствуют невидимым граням;
- в) удалить невидимые грани;
- г) в случае одного тела работа всего алгоритма заканчивается.

(3) Удаление невидимых рёбер или их частей, экранируемых другими телами сцены:

а) упорядочить тела по возрастанию $Z_{\text{мат}}$;

тело, ~~относительное~~ которое подвергается проверке наловом пробным ~~телом~~ объектом, а тело, относительно которого проверяется пробный объект, наловом пробным телом.

б) $(x_{\max T} < x_{\min ob}) \vee (x_{\min T} > x_{\max ob}) \vee (y_{\max T} < y_{\min ob}) \vee (y_{\min T} > y_{\max ob}) \Rightarrow$ экранирование отсутствует; протыкание невозможно, если $Z_{\max ob} < Z_{\min T}$, а иначе оно может быть;

в) $(x_{\max ob} > x_{\min T}) \vee (x_{\min ob} < x_{\max T}) \Rightarrow$ протыкание слева возможно; $(y_{\max ob} > y_{\min T}) \vee (y_{\min ob} < y_{\max T}) \Rightarrow$ протыкание сверху возможно;

г) сформировать $[g], [S], [d]$; вычислить $[P], [q], [w]$; проверить полную видимость, составив систему:

$$h_j = p_j + t q_j + \alpha w_j > 0, \quad 0 \leq t \leq 1, \quad 0 \leq \alpha;$$

д) составить все возможные пары, найти решения всей системы, выбрать $t_{\max min}$ и $t_{\min max}$; найти точки протыкания;

е) определить видимые участки рёбер и сохранить для последующего анализа;

(4) Определение невидимых участков новых рёбер, получаемых при протыкании тел:

а) сформировать все возможные рёбра, связывающие все точки протыкания для пар тел, связанных отношением протыкания;

б) проверить экранирование рёбер телами, кот. их породили;

в) оставшиеся видимые участки проверить на экранирование другими телами;

г) визуализировать оставшиеся рёбра.

#36) Алгоритм определения видимых поверхностей путём трассировки лучей.

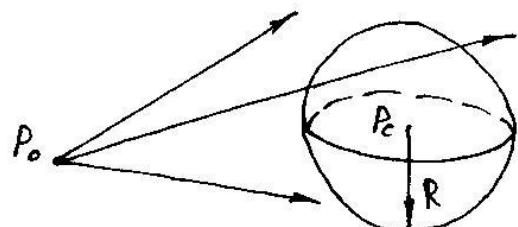
Наблюдатель видит объект посредством испускаемого неким источником света, который падает на поверхность объекта и затем как-то доходит до наблюдателя. В алгоритме отслеживаются лучи в обратном направлении: от наблюдателя к объекту.

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 :

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

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

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

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

Тест с прямоугольной оболочкой (труднее):

Применить преобразование, которое совместит луч с осью Z :

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

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 :

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

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

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 (САТ).

2.2) САТ пуст \Rightarrow изобразить данный пикセル цветом фона.

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

2.3) Для каждого объекта из САТ 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 пикセル 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.

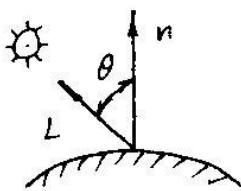
#37 Физические и психологические факторы, учитываемые при создании реалистических изображений.
Простая модель освещения.

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

Оражённый 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 отражённый свет рассеивается равномерно во всех направлениях, значит, положение наблюдателя не имеет значения.

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

I - интенсивность отражённого света; I_i - 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.

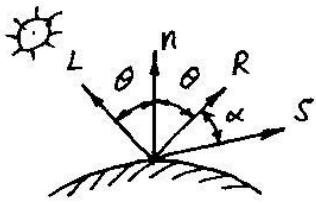
На объекты реальных сцен падает ещё и рассеянный свет:

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

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

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

d - расстояние от центра проекции до объекта; K - в 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 сформуированное, у шероховатых - более широкое.

В простых моделях освещения пользуются формулой:

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

I_s - интенсивность отраженного зеркального света;

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

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

Ф-ция закраски, примененная для расчета интенсивности
или тона точек объекта или пикселов изображения:

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

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

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

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

**#38 Метод Гуро закраски поверхностей
(получение слаженного изображения).**

Методы закраски:

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

Однотонной закраской можно пользоваться при выполнении трёх условий:

- 1) источник света находится в бесконечности $\Rightarrow \bar{n} \cdot \bar{I} = \text{const}$;
- 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.

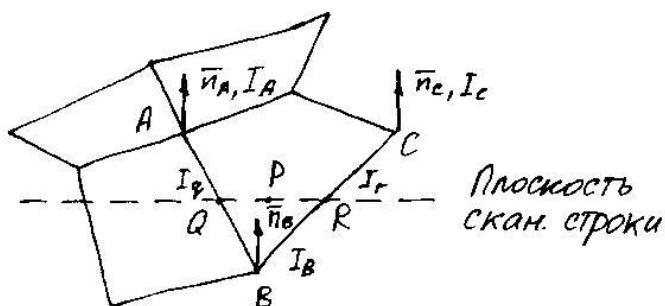
Билинейная интерполяция интенсивностей — это определение значений ф-ции в точке, лежащей внутри какого-то интервала.

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.

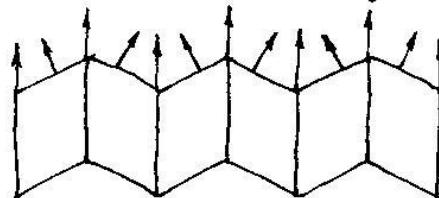
Методом Гуро можно получить слаженное изображение.



1. Вычисление векторов нормалей к каждой грани.
2. Вычисление векторов нормалей к каждой грани ^{вершины} (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. строки:
 $I_p = t I_q + (1-t) I_r, 0 \leq t \leq 1, t = \frac{QP}{QR}.$

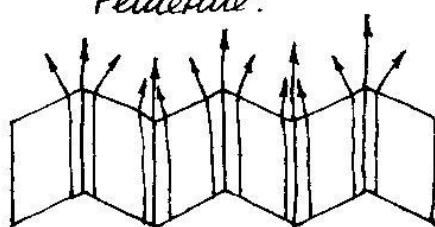
Закраска по Гуро хорошо сочетается с диффузными отражениями. Данный метод интерполяции обеспечивает лишь непрерывность значений интенсивности вдоль границ мн-ков, но не обеспечивает непрерывности изменения интенсивности. Значит, возможны проявления полос Маха.

Недостаток: усреднение нормалей.



Поверхность закрашивается с одной интенсивностью. будет выглядеть плоской.

Решение:



Если нужно сохранить острый переход, то не делается усреднение.

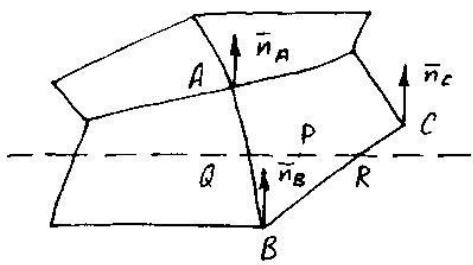
Если нужно, чтобы ребро сгладилось, вводят дополнительные мн-ки.

Инкрементальное вычисление вдоль скан. строки:

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

(#39) Закраска Ронга (улучшение аппроксимации кривизны поверхностей).

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



Трудность метода: (при создании кимкадра) 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.

Итерационное вычисление вдоль скан. строки:

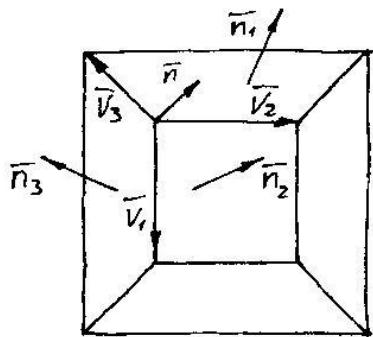
$$\bar{n}_{P_2} = \bar{n}_{P_1} + (\bar{n}_q - \bar{n}_r)(t_2 - t_1) = \bar{n}_{P_1} + \Delta \bar{n} \cdot \Delta t.$$

$$t_2 - t_1 = \frac{QP_2}{QR} - \frac{QP_1}{QR} = \frac{\Delta QR}{QR} = \frac{1}{QR} \Rightarrow \bar{n}_{P_2} = \bar{n}_{P_1} + \frac{\Delta \bar{n}}{RQ}$$

#40 Определение нормали к поверхности и вектора отражения в алгоритмах построения реалистических изображений.

Определение нормали

Нормаль к поверхности представляет её локальную кривизну, а следовательно, и направление зеркального отражения.



$$1) \bar{n} = \bar{n}_1 + \bar{n}_2 + \bar{n}_3$$

$$\bar{n} = (a_1 + a_2 + a_3)\bar{i} + (b_1 + b_2 + b_3)\bar{j} + (c_1 + c_2 + c_3)\bar{k}$$

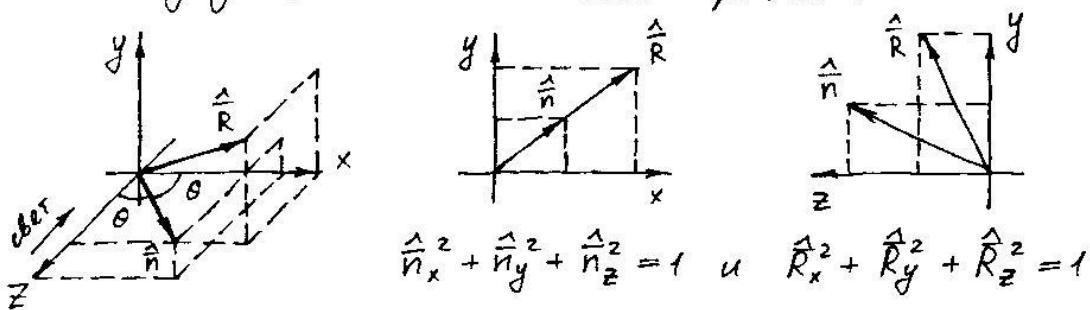
$$2) \bar{n} = \bar{V}_1 \times \bar{V}_2 + \bar{V}_2 \times \bar{V}_3 + \bar{V}_3 \times \bar{V}_1$$

Определение вектора отражения

1 способ (модель освещения с одним источником света)

Это случай, когда свет падает вдоль оси Z.

Перенесём начало координат в точку поверхности. Тогда проекции нормали и вектора отражения на плоскость XY будут лежать на одной прямой.



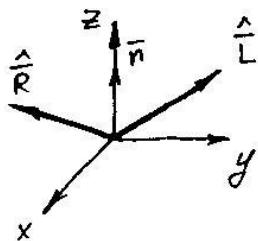
$$\frac{\hat{R}_x}{\hat{R}_y} = \frac{\hat{n}_x}{\hat{n}_y} \text{ и } \hat{n}_z = \cos \theta, 0 \leq \theta \leq \frac{\pi}{2} \text{ и } \hat{R}_z = \cos 2\theta = \cos^2 \theta - \sin^2 \theta = 2\hat{n}_z^2 - 1$$

$$\frac{\hat{R}_x^2}{\hat{n}_z^2} = \frac{\hat{n}_x^2}{\hat{n}_y^2} \text{ и } \hat{R}_x^2 + \hat{R}_y^2 = 1 - \hat{R}_z^2 \Rightarrow \frac{\hat{n}_x^2}{\hat{n}_y^2} \left(\frac{\hat{R}_x^2}{\hat{n}_z^2} + 1 \right) = 1 - (2\hat{n}_z^2 - 1)^2 \Rightarrow$$

$$\Rightarrow \hat{R}_x^2 \left(1 + \frac{\hat{R}_y^2}{\hat{n}_z^2} \right) = 1 - 4\hat{n}_z^4 + 4\hat{n}_z^2 - 1 \Rightarrow \hat{R}_x^2 \left(1 + \frac{\hat{n}_y^2}{\hat{n}_x^2} \right) = 4\hat{n}_z^2 \left(1 - \hat{n}_z^2 \right) = \frac{\hat{R}_x^2}{\hat{n}_x^2} \left(1 - \hat{n}_z^2 \right) \Rightarrow$$

$$\Rightarrow \hat{R}_x = 2\hat{n}_x \cdot \hat{n}_z \Rightarrow \hat{R}_y = 2\hat{n}_y \cdot \hat{n}_z$$

2 способ (совмещение вектора нормали с осью Z)



В изменённой СК:
 $\hat{L}_z = \hat{R}_z$ и $\hat{L}_x = -\hat{R}_x$ и $\hat{L}_y = -\hat{R}_y$

3 способ

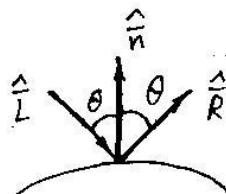
Условие того, что единичная нормаль, единичный вектор излучения и единичный вектор отражения лежат в одной плоскости, записывается так:

$$\hat{n} \times \hat{L} = \hat{R} \times \hat{n}$$

$$\begin{vmatrix} \bar{i} & \bar{j} & \bar{k} \\ \hat{n}_x & \hat{n}_y & \hat{n}_z \\ \hat{L}_x & \hat{L}_y & \hat{L}_z \end{vmatrix} = \bar{i} (\hat{n}_y \hat{L}_z - \hat{n}_z \hat{L}_y) + \bar{j} (\hat{n}_z \hat{L}_x - \hat{L}_z \hat{n}_x) + \bar{k} (\hat{n}_x \hat{L}_y - \hat{L}_x \hat{n}_y)$$

$$\begin{vmatrix} \bar{i} & \bar{j} & \bar{k} \\ \hat{R}_x & \hat{R}_y & \hat{R}_z \\ \hat{R}_x & \hat{R}_y & \hat{R}_z \end{vmatrix} = \bar{i} (\bar{R}_y \bar{n}_z - \bar{R}_z \bar{n}_y) + \bar{j} (\bar{R}_z \bar{n}_x - \bar{R}_x \bar{n}_z) + \bar{k} (\bar{R}_x \bar{n}_y - \bar{R}_y \bar{n}_x)$$

$$\begin{cases} -n_z R_y + n_y R_z = n_z L_y - n_y L_z \\ n_z R_x - n_x R_z = n_x L_z - n_z L_x \\ -n_y R_x + n_x R_y = n_y L_x - n_x L_y \end{cases}$$



$$\cos \theta = \langle \hat{L}, \hat{n} \rangle = \langle \hat{n}, \hat{R} \rangle \quad \text{и} \quad \hat{R} = \alpha \hat{L} + \beta \hat{n}$$

$$(1) \quad \hat{n} \cdot (\alpha \hat{L} + \beta \hat{n}) = \cos \theta \Rightarrow \alpha \hat{n} \cdot \hat{L} + \beta = \cos \theta \Rightarrow \beta = \cos \theta + \alpha \cos \theta = (1+\alpha) \cos \theta$$

$$(2) \quad (\alpha \hat{L} + \beta \hat{n})^2 = 1 \Rightarrow \alpha^2 + 2\alpha\beta \hat{L} \cdot \hat{n} + \beta^2 = 1 \Rightarrow \alpha^2 - 2\alpha\beta \cos \theta + \beta^2 = 1$$

$$\alpha^2 - 2\alpha\beta \cos \theta + \beta^2 = 1 \Rightarrow \alpha^2 - 2\alpha(1+\alpha) \cos^2 \theta + (1+\alpha)^2 \cos^2 \theta = 1$$

$$\alpha^2 - 2\alpha \cos^2 \theta - 2\alpha^2 \cos^2 \theta + \cos^2 \theta + 2\alpha \cos^2 \theta + \alpha^2 \cos^2 \theta = 1 \Rightarrow$$

$$\Rightarrow \alpha^2(1 - \cos^2 \theta) = (1 - \cos^2 \theta) \Rightarrow \alpha = \pm 1$$

$$\alpha = 1 \Rightarrow \beta = 2 \cos \theta = -2 \hat{L} \cdot \hat{n}$$

(#41) Построение теней. Учит тени в алгоритмах удаления невидимых поверхностей.

Если положение наблюдателя и источника света совпадают, то тени не видно, но они появляются, когда наблюдатель перемещается в любую другую точку.

Тень состоит из 2 частей:

- 1) Полная тень - это центральная, тёмная, резко очерченная часть.
- 2) Полутень - окружающая полную тень 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 нелицевых граней, если точку наблюдения совместить с источником света (алгоритм Робертса).

Построение проекционной тени:

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

Интервальный алгоритм построчного сканирования:

1. Для каждого источника и мн-ка определить собств. тени и проекц. тени. Они записываются в виде двойной матрицы: строки - мн-ки, отбрасывающие тень, столбцы - затеняющие мн-ки.
2. Обработка сцены относительно положения наблюдателя:
 - 2.1) Определить видимые отрезки на интервале.
 - 2.2) С помощью списка теневых мн-ков определить, падает ли тень на мн-к, который создает видим. отрезок на данном интервале:
 - а) нет теневых мн-ков \Rightarrow изобразить видим. отрезок;
 - б) есть теневые мн-ки, но они не пересекают и не покрывают данный интервал \Rightarrow изобразить видим. отрезок;
 - в) интервал полностью покрывается хотя бы одним мн-ком \Rightarrow \Rightarrow интенсивность изображаемого отрезка определяется с учётом интенсивностей этих мн-ков и самого отрезка;
 - г) хотя бы один теневой мн-к частично покрывает интервал \Rightarrow \Rightarrow рекурсивно разделять интервал и работать с подинтервалами.

Алгоритм, использующий Z-буфер:

1. Строится сцена, где точка наблюдения совпадает с источником. Запоминать значения Z в отдельном теневом Z-буфере.
2. Строится сцена из точки наблюдателя. При обработке каждой поверхности ширина каждого пикселя сравнивается с шириной Z-буф. Если $Z(x,y) > Z_{\text{буф}}(x,y) \Rightarrow x, y, z$ из виду наблюдателя линейно преобразуются в x', y', z' на виду из источника:
$$z' \geq Z_{\text{теп.буф.}}(x',y') \Rightarrow x, y - в буфер кадра;$$
$$z' < Z_{\text{теп.буф.}}(x',y') \Rightarrow (x, y, z) в тени и изображается согласно соответствующему правилу расчёта интенсивности с учётом затенения; ~~Аудиомикшер для звука в игре~~.$$
$$Z_{\text{буф.}} = Z(x,y).$$

(#42) Учтёт прозрачности в модели освещения.

Учтёт прозрачности в алгоритмах удаления невидимых поверхностей.

Чтобы включить преломление в модель освещения, нужно при построении видимых поверхностей учитывать не только падающий, но и отраженный и пропущенный свет. Эффективнее всего это выполняется с помощью шаблонной 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 кол-во беспорядочно ориентированных лучей.

Формула расчёта интенсивности для прозрачных тел:

$$I = k_a \cdot I_a + k_d \cdot I_d + k_s \cdot I_s + k_t \cdot I_t,$$

где индексы a, d, s, t обозначают рассеянный, диффузный, зеркальный и пропущенный свет.

Пропускание:

1) Зеркальное (направленное) – свойственно прозрачным телам (стекло). Если смотреть на объект сквозь такое стекло, то за исключением контурных линий привычных поверхностей искажения происходить не будет.

Наицнайная аппроксимация коэф-та прозрачности:

$$t = t_{\min} + (t_{\max} - t_{\min}) \left(1 - \left(1 - \frac{1}{|\hat{n}_z|} \right)^p \right)$$

t_{\min} и t_{\max} – миним. и макс. прозрачность объекта;

\hat{n}_z – z -составляющая единичной нормали к поверхности;

p – коэф-т степени прозрачности;

t – прозрачность пикселя или точки объекта.

2) Диффузное (рассеянное) – вещества кажутся полупрозрачными или матовыми.

Простое пропускание света можно встроить в любой алгоритм удаления невидимых ~~пластин~~ поверхностей кроме алгоритма с Z-буфером:

Прозрачные поверхности помечаются и если видимая часть программы, то в буфер кадра записывается линейная комбинация двух ближайших поверхностей:

$$I = t I_1 + (1-t) I_2, \quad 0 \leq t \leq 1$$

I_1 – видимая поверхность; I_2 – поверхность за ней.

Алгоритм, использующий Z-буфер:

1. Для каждого мн-ка:

1.1) Если мн-к прозрачен \Rightarrow внести его в список прозрач. мн-ков.

1.2) Если мн-к непрозрачен и $Z > Z_{\text{буф}}$, то записать его в буфер кадра для непрозрачных мн-ков и скорректировать $Z_{\text{буф}}$.

2. Для каждого мн-ка из списка прозрачных мн-ков:

2.1) Если $Z \geq Z_{\text{буф}}$, то прибавить его коэф-т прозрачности к значению, содержащемуся в буфере весовых коэф-тов прозрачности; прибавить его интенсивность к значению, содержащемуся в буфере интенсивности прозрачности по правилу:

$$I_{\text{вн}} = I_{\text{вн}} \cdot t_{\text{вн}} + I_{\text{с}} \cdot t_{\text{с}} \quad (\text{вн} - \text{новое значение}, \text{вн} - \text{старое}, \text{с} - \text{для мн-ка}).$$

Таким образом, получается взвешенная сумма интенсивностей всех прозрачных мн-ков, находящихся перед непрозрачным).

Объединим буфера интенсивности для прозрач. и непрозрач. мн-ков:

$$I_{\text{fb}} = t_{\text{вн}} \cdot I_{\text{вн}} + (1-t_{\text{вн}}) \cdot I_{\text{фвн}}$$

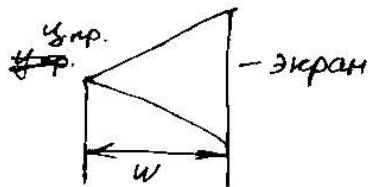
I_{fb} – окончательная интенсивность в буфере кадра для непрозрачных мн-ков;

$I_{\text{фвн}}$ – старое значение интенсивности в этом буфере.

#43 Преобразования на плоскости. Вывод
расчетных соотношений. Матрицы преобразований.

Однородные координаты: $(x \ y \ w)$

$$X = \frac{x}{w}, \quad Y = \frac{y}{w}$$



Преобразования на пл-ти:

$$(x, y) \rightarrow (x_1, y_1)$$

$$x_1 = Ax + By + C$$

$$y_1 = Dx + Ey + F$$

$$(x_1 \ y_1 \ 1) = (x \ y \ 1) M_{\text{пр}}$$

$$M_{\text{пр}} = \begin{pmatrix} A & D & 0 \\ B & E & 0 \\ C & F & 1 \end{pmatrix}; \quad (x_2 \ y_2 \ 1) = (x_1 \ y_1 \ 1) \cdot M_2 = (x \ y \ 1) \cdot M_1 \cdot M_2 = (x \ y \ 1) \cdot M_{\Sigma}$$

Аффинные преобразования: пл-ть не вырожд. в прямую, а прямая не вырождается в точку; сохраняется параллельность плоскостей и существует обратн. преобразования ($\det M \neq 0$).

Любое преобразование может быть представлено в виде трёх:

1) Перенос

dx, dy - 2 параметра.

$$x_1 = x + dx$$

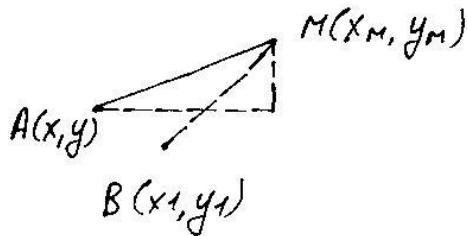
$$y_1 = y + dy$$

$$M_{\text{пер}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ dx & dy & 1 \end{pmatrix}$$

Масштабирование

x_m, y_m, k_x, k_y — 4 параметра.

$k_x \neq k_y$ — неоднородное масштабир.



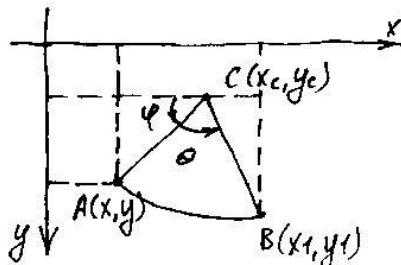
$$x_1 - x_m = k_x (x - x_m)$$

$$\begin{cases} x_1 = k_x \cdot x + (1 - k_x) \cdot x_m \\ y_1 = k_y \cdot y + (1 - k_y) \cdot y_m \end{cases}$$

$M_{\text{масшт}} = \begin{pmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$ — в матр. форме можно описать масштабир. только относительно центра координат.

Поворот

x_c, y_c, θ



$$M_{\text{повор}} = \begin{pmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\det M_{\text{повор}} = 1$$

$$\begin{aligned} x_1 &= x_c + R \cos(180^\circ - (\varphi + \theta)) = x_c - R \cos(\varphi + \theta) = \\ &= x_c - R \cdot \cos \varphi \cdot \cos \theta + R \cdot \sin \varphi \cdot \sin \theta = \\ &= x_c - (x_c - x) \cos \theta + (y - y_c) \sin \theta = \\ &= x_c + (x - x_c) \cos \theta + (y - y_c) \sin \theta \\ y_1 &= y_c + R \sin(180^\circ - (\varphi + \theta)) = y_c + R \sin(\varphi + \theta) = \\ &= y_c + R \sin \varphi \cos \theta + R \cos \varphi \sin \theta = \\ &= y_c + (y - y_c) \cos \theta + (x_c - x) \sin \theta = \\ &= y_c + (y - y_c) \cos \theta - (x - x_c) \sin \theta \end{aligned}$$

Коммутативно:

M_1	M_2
перемос	перемос
Масшт.	Масшт.
Поворот	Поворот
Масшт (аддн. операц.)	Поворот

Перемос и поворот коммутативны:

$$M_{\text{переп}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ dx_1 + dx_2 & dy_1 + dy_2 & 1 \end{pmatrix}$$

$$M_{\text{повор}} = \begin{pmatrix} \cos(\theta_1 + \theta_2) & -\sin(\theta_1 + \theta_2) & 0 \\ \sin(\theta_1 + \theta_2) & \cos(\theta_1 + \theta_2) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Масштабирование коммутативно:

$$M_{\text{масшт}} = \begin{pmatrix} k_{x1} \cdot k_{x2} & 0 & 0 \\ 0 & k_{y1} \cdot k_{y2} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$