# 1. Введение.

При обработке информации, связанной с изображением на мониторе, принято выделять три основных направления: распознавание образов, обработку изображений и машинную графику.

Основная задача распознавания образов состоит в преобразовании уже имеющегося изображения на формально понятный язык символов. ***Распознавание образов или система технического зрения (COMPUTER VISION)*** – совокупность методов, позволяющих получить описание изображения, поданного на вход, либо отнести заданное изображение к некоторому классу (так поступают, например, при сортировке почты). Одной из задач COMPUTER VISION является так называемая склетизация объектов, при которой восстанавливается некая основа объекта, его «скелет».

***Обработка изображений (IMAGE PROCESSING)*** рассматривает задачи в которых и входные и выходные данные являются изображениями. Например, передача изображения с устранением шумов и сжатием данных, переход от одного вида изображения к другому (от цветного к черно белому) и т.д.

***Компьютерная (машинная) графика (COMPUTER GRAPHICS)*** воспроизводит изображение в случае, когда исходной является информация неизобразительной природы. Например, визуализация экспериментальных данных в виде графиков, гистограмм или диаграмм, вывод информации на экран компьютерных играх, синтез сцен на тренажерах.

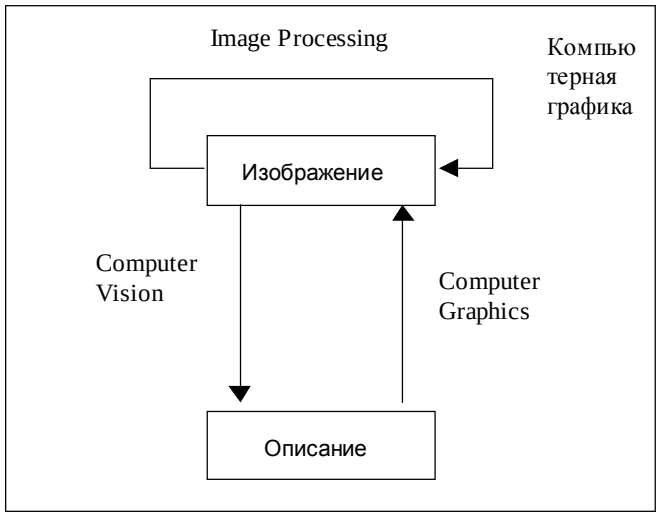



Рис. 1.1

***Компьютерная графика*** – это наука, предметом изучения которой является создание, хранение и обработка моделей и их изображений с помощью ЭВМ. В том случае, если пользователь может управлять характеристиками объектов, говорят об интерактивной компьютерной графике. Очевидно, что неинтерактивная компьютерная графика является частным случаем интерактивной.

Существуют изображения двух типов: аналоговые и цифровые. Возможны синтез изображений в реальном времени и высокореалистический синтез. В данном курсе рассматривается проблема синтеза цифровых изображений в реальном времени.

## 2. Алгоритмы растровой графики.

### 2.1 Растровые представления изображений.

Термин растровая графика достаточно очевиден, если усвоить понятия, относящиеся к растровым изображениям. Растровые изображения напоминают лист клетчатой бумаги, на котором любая клетка закрашена либо черным, либо белым цветом, образуя в совокупности рисунок. Пиксел - основной элемент растровых изображений. Именно из таких элементов состоит растровое изображение.

***Цифровое изображение*** – это совокупность пикселей. Каждый пиксел растрового изображения характеризуется координатами x и y и яркостью V(x,y) (для черно-белых изображений). Поскольку пикселы имеют дискретный характер, то их координаты- это дискретные величины, обычно целые или рациональные числа. В случае цветного изображения, каждый пиксел характеризуется координатами x и y, и тремя яркостями: яркостью красного, яркостью синего и яркостью зеленого цветов ($V_R$, $V_B$, $V_G$). Комбинируя данные три цвета можно получить большое количество различных оттенков.

Заметим, что в случае, если хотя бы одна из характеристик изображения не является числом, то изображение относится к виду ***аналоговых***. Примерами аналоговых изображений могут служить галограмы и фотографии. Для работы с такими изображениями существуют специальные методы, в частности, оптические преобразования. В ряде случаев аналоговые изображения переводят в цифровой вид. Эту задачу осуществляет Image Processing.

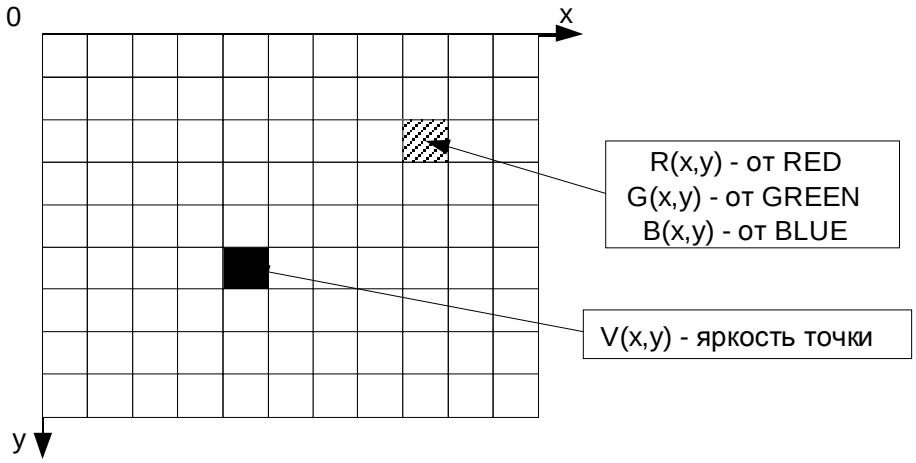



Рис. 2.1.1

Цвет любого пиксела растрового изображения запоминается с помощью комбинации битов. Чем больше битов для этого используется, тем больше оттенков цветов можно получить. Под градацию яркости обычно отводится 1 байт (256 градаций), причем 0 – черный цвет, а 255 – белый (максимальная интенсивность). В случае цветного изображения отводится по байту на градации яркостей всех трех цветов. Возможно кодирование градаций яркости другим количеством битов (4 или 12), но человеческий глаз способен различать только 8 бит градаций на каждый цвет, хотя специальная аппаратура может потребовать и более точную передачу цветов. Цвета, описываемые 24 битами, обеспечивают более 16 миллионов доступных цветов и их часто называют естественными цветами.

В цветовых палитрах каждый пиксел описан кодом. Поддерживается связь этого кода с таблицей цветов, состоящей из 256 ячеек. Разрядность каждой ячейки- 24 разряда. На выходе каждой ячейки- по 8 разрядов для красного, зеленого и синего цветов.

Цветовое пространство, образуемое интенсивностями красного, зеленого и синего, представляют в виде цветового куба.

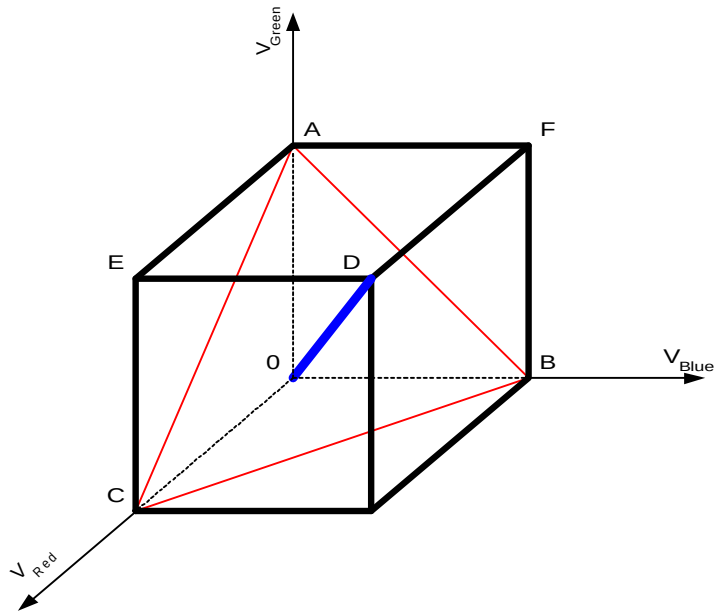



Рис. 2.1.2 «Цветовой Куб»

Вершины куба A, B, C являются максимальными интенсивностями зеленого, синего и красного соответственно, а треугольник которые они образуют называется ***треугольником Паскаля.*** Периметр этого треугольника соответствует максимально насыщенным цветам. Цвет максимальной насыщенности содержит всегда только две компоненты. На отрезке OD находятся оттенки серого, причем тока O соответствует черному, а точка D белому цвету.

***Растр*** – это порядок расположения точек (растровых элементов). На рис. 2.1.1 изображен растр элементами которого являются квадраты, такой растр называется ***прямоугольным***, именно такие растры наиболее часто используются. Хотя возможно использование в качестве растрового элемента фигуры другой формы: треугольника, шестиугольника; соответствующего следующим требованиям:

1. Все фигуры должны быть одинаковые;
2. Должны полностью покрывать плоскость без наезжания и дырок.

Так в качестве растрового элемента возможно использование равностороннего треугольника рис. 2.1.3, правильного шестиугольника (гексаэдра) рис. 2.1.4. Можно строить растры, используя неправильные многоугольники, но практический смысл в подобных растрах отсутствует.

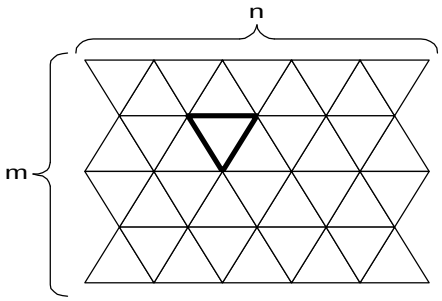



Рис. 2.1.3 «Треугольный растр»

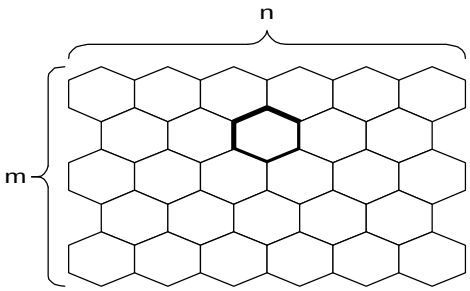



Рис. 2.1.4 «Гексагональный растр»

Рассмотрим способы построения линий в прямоугольном и гексагональном растре.

В прямоугольном растре построение линии осуществляется двумя способами:

1) Результат – восьмисвязная линия. Соседние пиксели линии могут находится в одном из восьми возможных (см. рис. 2.1.5а) положениях. Недостаток – слишком тонкая линия при угле 45°.
2) Результат – четырехсвязная линия. Соседние пиксели линии могут находится в одном из четырех возможных (см. рис. 2.1.5б) положениях. Недостаток – избыточно толстая линия при угле 45°.

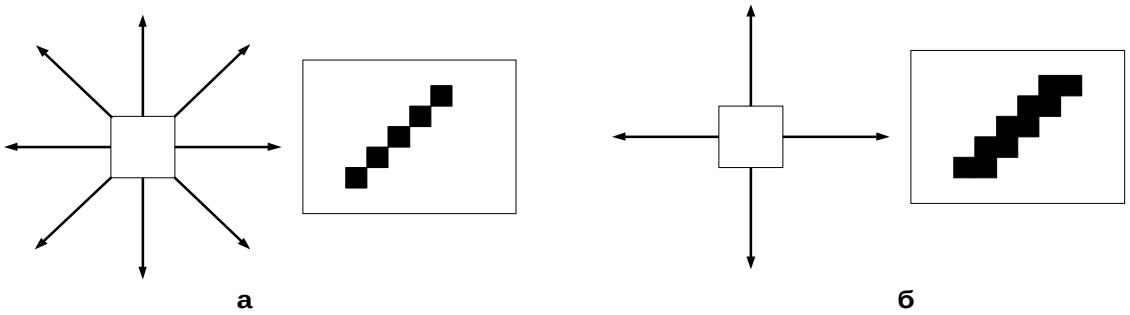



Рис. 2.1.5 «Построение линии в прямоугольном растре»

В гексагональном растре линии шестисвязные (см. рис. 2.1.6) такие линии более стабильны по ширине, т.е. дисперсия ширины линии меньше, чем в квадратном растре.

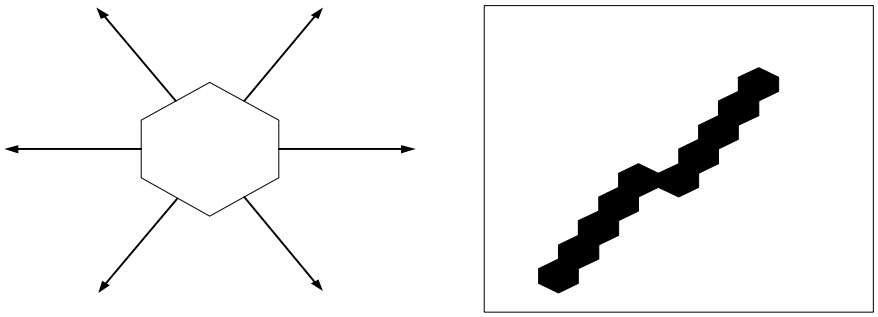


Рис. 2.1.6 «Построение линии в гексагональном растре»

Каким образом можно оценить, какой растр лучше?

Одним из способов оценки является передача по каналу связи кодированного, с учетом используемого растра, изображения с последующим восстановлением и визуальным анализом достигнутого качества. Экспериментально и математически доказано, что гексагональный растр лучше, т.к. обеспечивает наименьшее отклонение от оригинала. Но разница не велика.

***Моделирование гексагонального растра.*** Возможно построение гексагонального растра на основе квадратного. Для этого гексаугольник представляют в виде прямоугольника (см. рис. 2.1.7).

Определим, какие пропорции должно иметь гексагональное изображение?

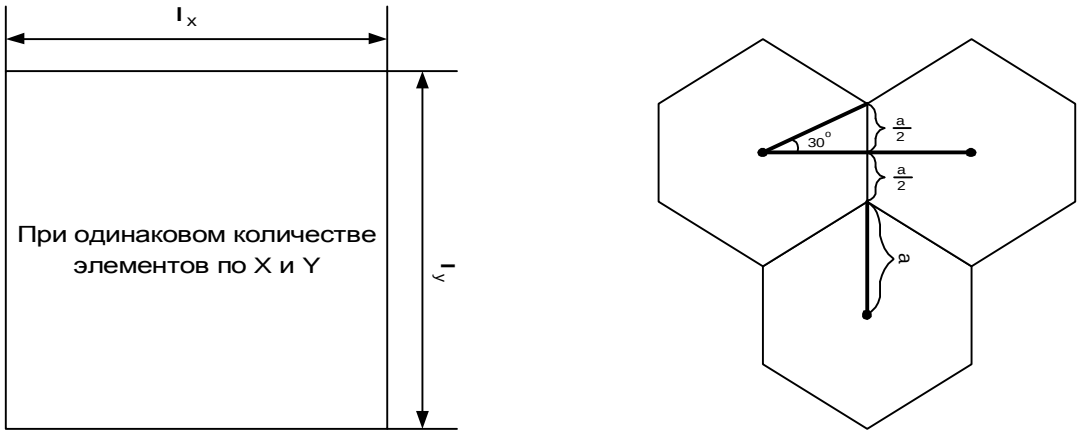



Рис. 2.1.8

$$\frac{l_X}{l_Y} = \frac{\frac{a}{2} \cdot \tan(30^\circ) \cdot 2}{\frac{3}{2} \cdot a} = \frac{2}{\sqrt{3}};$$

Техническая реализация гексагонального растра не сложна. Модель гексагонального растра получают из прямоугольного, задержав на 1 пиксель каждую нечетную строчку изображения, и растянув изображение на экране таким образом, чтобы $\frac{l_x}{l_y} = \frac{2}{\sqrt{3}}$.

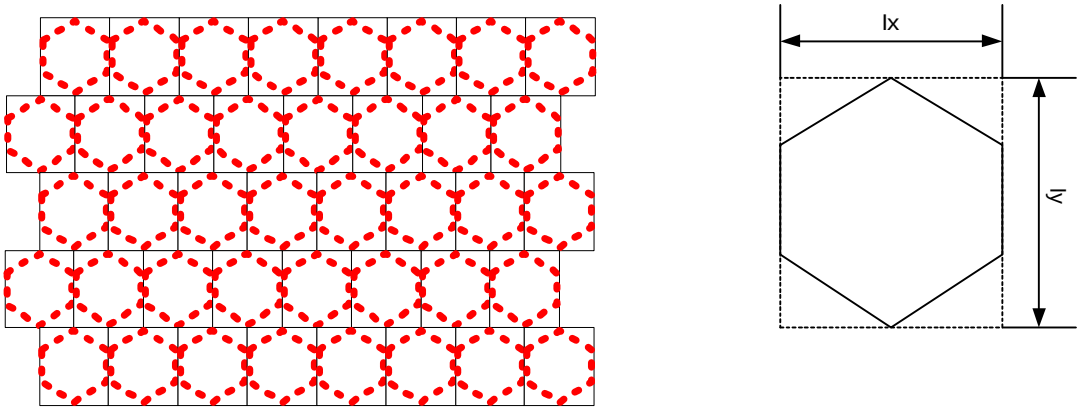



Рис. 2.1.7 «Построение гексагонального растра на квадратном»

Тот факт, что гексагональный растр не используется, объясняется следующими причинами:

1. некоторое усложнение алгоритмов;
2. преимущество гексагонального растра не очень велико;
3. историческая ориентация на прямоугольный растр.

Для программного построения гексагонального растра в квадратном можно использовать модель представленную на рис. 2.1.8.

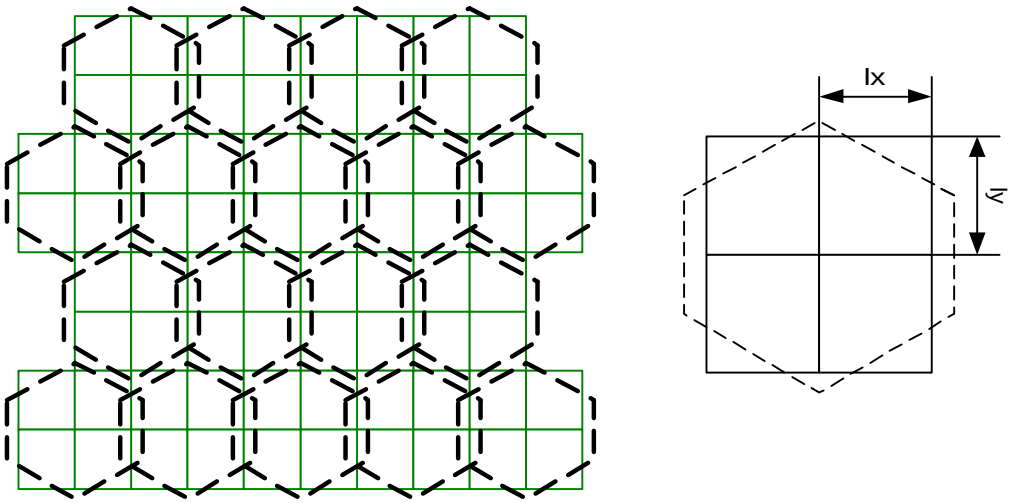



Рис. 2.1.8 «Построение гексагонального растра в квадратном»

## 2.2 Построение линии в квадратном растре.

Поскольку экран растрового дисплея с электронно-лучевой трубкой (ЭЛТ) можно рассматривать как матрицу дискретных элементов (пикселей), каждый из которых может быть подсвечен, нельзя непосредственно провести отрезок из одной точки в другую. Процесс определения пикселей, наилучшим образом аппроксимирующих заданный отрезок, называется разложением в растр. В сочетании с процессом построчной визуализации изображения он известен как преобразование растровой развертки. Для горизонтальных, вертикальных и наклоненных под углом 45° отрезков выбор растровых элементов очевиден. При любой другой ориентации выбрать нужные пиксели труднее, что показано на рис. 2.2.1.

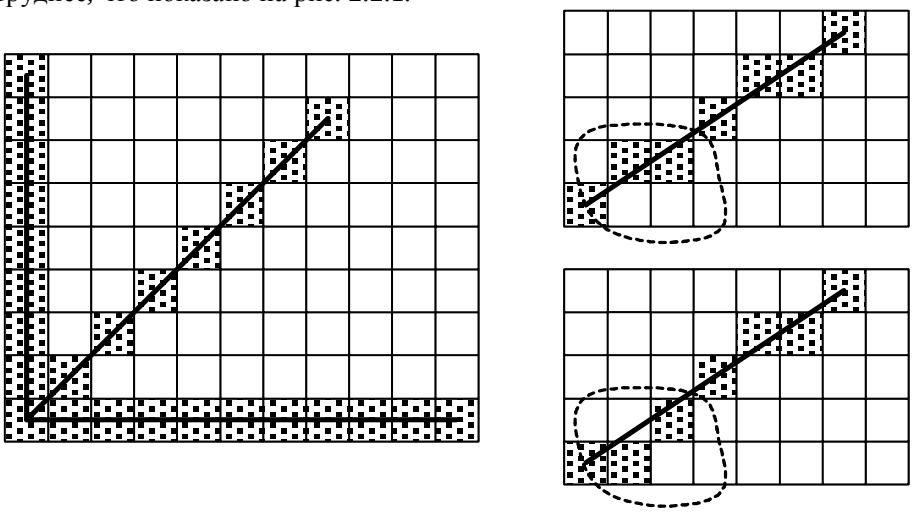


Рис. 2.2.1 «Разложение в растр отрезков»

***Общие требования к изображению отрезка.***

- концы отрезка должны находиться в заданных точках;
- хороший зрительный эффект;
- яркость вдоль отрезка должна быть постоянной и не зависеть от длины и наклона;
- и, наконец, алгоритм рисования должен работать быстро.

Ни одно из этих условий не может быть точно выполнено на растровом дисплее в силу того, что изображение строится из пикселов конечных размеров, а именно:

- концы отрезка в общем случае располагаются на пикселах, лишь наиболее близких к требуемым позициям и только в частных случаях координаты концов отрезка точно совпадают с координатами пикселов;
- отрезок аппроксимируется набором пикселов и лишь в частных случаях вертикальных, горизонтальных и отрезков под $45^{\circ}$ они будут выглядеть прямыми, причем гладкими прямыми без ступенек, только для вертикальных и горизонтальных отрезков (рис. 2.2.1);
- яркость для различных отрезков и даже вдоль отрезка в общем случае различна, так как, например, расстояние между центрами пикселов для вертикального отрезка и отрезка под $45^{\circ}$ различно (см. рис. 2.2.1).

Обеспечение одинаковой яркости вдоль отрезков разных длин и ориентации требует извлечения квадратного корня, а это замедляет вычисления. Ширина линий, проведенных под различным углом, колеблется:

для восьмисвязных от $\frac{1}{\sqrt{2}}$ до 1;

для четырехсвязных от $\sqrt{2}$ до 1.

Объективное улучшение аппроксимации достигается увеличением разрешения дисплея, но в силу существенных технологических проблем разрешение для растровых систем приемлемой скорости разрешение составляет порядка 1280×1024.

Субъективное улучшение аппроксимации основано на психофизиологических особенностях зрения и, в частности, может достигаться просто уменьшением размеров экрана. Другие способы субъективного улучшения качества аппроксимации основаны на различных программных ухищрениях по "размыванию" резких границ изображения.

## 2.3. Параметрический алгоритм рисования линии.

Является самым простым способом рисования линии. Здесь уместно оговориться, что у этого метода есть существенные недостатки, а именно: необходимость выполнения операций над вещественными числами и использования операции деления, что значительно усложняет аппаратную организацию и увеличивает время работы алгоритма. Необходимо провести линию из точки ($x_1$, $y_1$) в точку ($x_2$, $y_2$) с учетом того, что заданы яркости в начальной и конечной точках линии, и по яркости производится линейная интерполяция. рис. 2.3.1

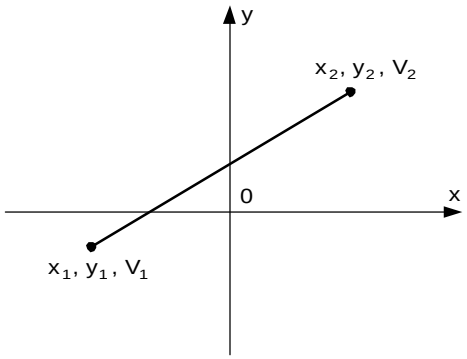



Рис. 2.3.1

Любую точку на этой линии можно представить в виде

$$x = ]x_1 + t \cdot (x_2 - x_1)[$$
$$y = ]y_1 + t \cdot (y_2 - y_1)[ \text{; где } t \in [0;1], \text{ ][ – знак округления до целого.}$$
$$V = ]V_1 + t \cdot (V_2 - V_1)[$$

Необходимо выбрать оптимальный шаг приращения $t$. Он выбирается как

$$t = \frac{1}{\max\{|x_2 - x_1|, |y_2 - y_1|\}} = \frac{1}{N-1}, \text{ где N – длина линии в пикселях.}$$

Можно проводить вычисления через приращение координат $x, y, V$ (с начальным присвоением $x = x1, y = y1, V = V1$) на

$$\Delta x = \frac{x_2 - x_1}{N-1}; \Delta y = \frac{y_2 - y_1}{N-1}; \Delta V = \frac{V_2 - V_1}{N-1} \text{ соответственно.}$$

Значения приращений считаются в начале функции и не входят в цикл построения линии на экране, за счет чего повышается быстродействие.

К достоинствам алгоритма следует отнести: простоту его программной, в частности, реализации линейной интерполяции по яркости. Альтернативой приведенному алгоритму является алгоритм Брезенхема.

## 2.4 Алгоритм Брезенхема рисования линии.

В 1965 году Брезенхемом был предложен простой целочисленный алгоритм для растрового построения отрезка. Рассмотрим алгоритм для частного случая – рисования линии из начала координат под углом к оси q менее $45^{\circ}$, т.е. $r > 0, q > 0, r \geq q$. Поскольку $r \geq q$, то на i-ой итерации значение $r_i$ рассчитывается так: $r_i = r_{i-1} + \Delta r$, где $\Delta r = 1$. По ходу движения по оси r алгоритм смотрит, насколько пиксел отклонился от реальной линии по оси q. Если ошибка настолько велика, что соседний по оси q пиксел оказывается ближе к реальной линии, чем тот, что мы собираемся поставить, то алгоритм переходит на соседний по этой оси пиксел. Таким образом $q_i = q_{i-1} + 1 - \Delta q_i$, где $\Delta q_i = 0$, при $h_i \leq 0.5$ и $q_i = 1$, при $h_i > 0$. В алгоритме используется управляющая переменная $h_i$, которая называется ошибкой отклонения на каждом i-ом шаге и рассчитывается как $h_i = \frac{q}{r} \times (r_{i-1} + 1) - q_{i-1}$.

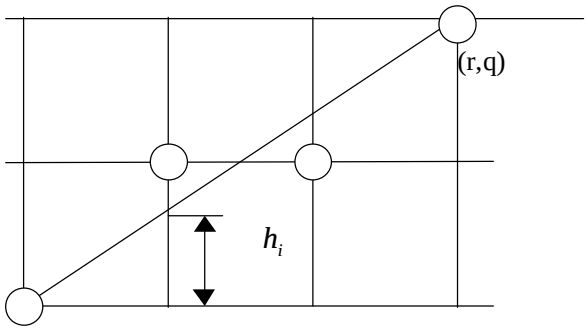



Рис. 2.4.1. Текущая ошибка в алгоритме Брезенхема

Согласно рисунку 2.4.1 линия начинается в точке (0,0), и в этой точке ошибка отклонения равна 0. Следующий пиксел по оси r будет иметь координату 1. Вертикальная координата следующего пиксела будет либо 0, либо 1. Текущая ошибка

по рисунку больше, чем полпиксела, поэтому вертикальная координата переходит в 1. Следующая точка сдвигается по оси r на один пиксел, а по оси q остается в 1. Наконец, последняя точка в линии имеет ошибку меньше полпиксела и, следовательно, сдвигается по вертикали на единицу.

Введем вспомогательную функцию $d$.

$d_i = 2(h_i - 0.5) \times r = 2q(r_{i-1} + 1) - 2q_{i-1}r - r$, тогда

$\Delta q_i = 0$ при $d_i \le 0$

$\Delta q_i = 1$ при $d_i > 0$

$d_{i+1} = d_i + \Delta d_{i+1}$,

$\Delta d_{i+1} = d_{i+1} - d_i = 2(h_{i+1} - 0{,}5) \times r - d_i = 2q(r_i + 1) - 2q_i r - r - d_i = 2q - 2\Delta q_i r$

Тогда возможные приращения: $\Delta d' = 2q$

$\Delta d'' = 2q - 2r$ и зависимость $\Delta d_{i+1}$ такова:

$\Delta d_{i+1} = \Delta d'$ при $d_i \le 0$

$\Delta d_{i+1} = \Delta d''$ при $d_i > 0$

Такой алгоритм не использует операций с плавающей точкой, не имеет в основном цикле ни делений, ни умножений, и его реализация занимает очень мало места.

Существует модификация данного алгоритма, которая позволяет получить еще более простую программу.

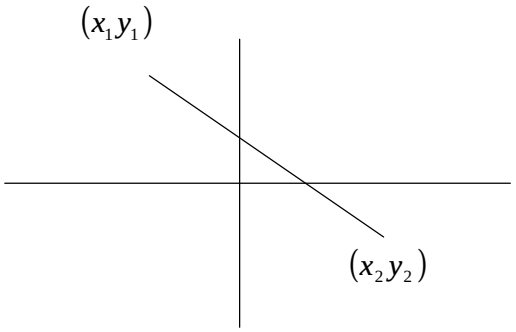



Рис 2.4.2

$D_x = x_2 - x_1$ $\quad MD_x = |D_x|$

$D_y = y_2 - y_1$ $\quad MD_y = |D_y|$

$\Delta x = 1$ при $D_x > 0$ $\quad \Delta y = 1$ при $D_y > 0$

$\Delta x = 0$ при $D_x = 0$ $\quad \Delta y = 0$ при $D_y = 0$

$\Delta x = -1$ при $D_x < 0$ $\quad \Delta y = -1$ при $D_y < 0$

Введем присвоения: $x = x_1, y = y_1, d_x = MD_x, d_y = MD_y$.

$D = \max\left(MD_x, MD_y\right)$

```
for (i=0;i<=D+1;i++)
{ PutPixel (x,y);
  d_x+=MD_x;
  d_y+=MD_y;
  if (d_x > D)  {d_x-=D; x+=Δx}
  if  (d_y > D)  {d_y-=D; y+=Δy;}
}
```

Диапазоны изменения $d_x и d_y$: $0 \le d_x \le 2MD_x$

$$0 \le d_y \le 2MD_y$$

Рассмотренный алгоритм выводит восьмисвязную линию.

Линейная интерполяция по яркости при построении отрезка по алгоритму Брезенхема получается параметрическим способом, т.е. $V = V_{i-1} + \Delta V; \Delta V = \frac{V_2 - V_1}{N-1}$, где N – длина отрезка в пикселях.

## 2.5 Алгоритмы построения окружности.

Рассмотрим окружность с центром в начале координат, для которой $x^2+y^2=R^2$, или в параметрической форме:

1. x=R· cos(a);
2. y=R· sin(a).

То есть легко написать программу рисования окружности:

```
void Circle (int x, int y, int R, int color)
{
    int a;
    int x1;
    int x2;
    int y1;
    int y2;
    x2=x+R;
    y2=y;
    for ( int a=1; a<=360; a++)
    {
        x1=x2; y1=y2;
        x2=round(R*cos(a))+x;
        y2=round(R*sin(a))+x;
        Line (x1, y1, x2, y2, color);
    }
 }
```

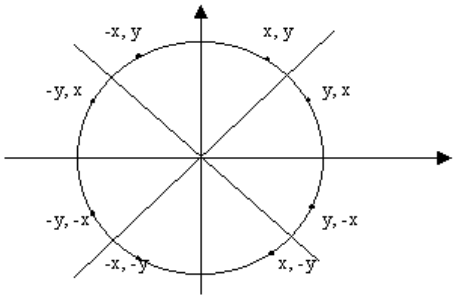


Рис. 2.5.1.

Если воспользоваться симметрией окружности, то можно построить более эффективный алгоритм. Если точка (x, y) лежит на окружности, то легко вычислить семь точек, принадлежащих окружности, симметричных этой. То есть, имея функцию вычисления значения y по x=0..R/SQRT(2) для построения дуги от $0^0$ до $45^0$. Построим процедуру, которая будет по одной координате ставить восемь точек, симметричных центру окружности.

```
void Circle_Pixel(int x0, int y0, int x, int y, int color);
{
    putpixel(x0 + x, y0 + y, color);
    putpixel(x0 + y, y0 + x, color);
    putpixel(x0 + y, y0 - x, color);
    putpixel(x0 + x, y0 - y, color);
    putpixel(x0 - x, y0 - y, color);
    putpixel(x0 - y, y0 - x, color);
    putpixel(x0 - y, y0 + x, color);
    putpixel(x0 - x, y0 + y, color);
}
```

Таким образом можно написать программу рисование окружности по точкам:

```
void Circle (int x0, int y0, int R, int color)
{
    for ( int x=0; x<=R/sqrt(2); x++)
    {
        int y = (int)(sqrt(sqr(R)-sqr(x)));
        Circle_Pixel (x0, y0, x, y, color);
    }
}
```

***Алгоритм Брезенхема генерации окружности.*** Брезенхем разработал алгоритм более эффективный, чем каждый рассмотренный выше. Здесь используется тот же принцип, что и для рисования линии. Будем рассматривать сегмент окружности, соответствующий $x=x_0..\ x_0+R/sqrt(2)$. На каждом шаге выбираем точку, ближайшую к реальной окружности. В качестве ошибки возьмем величину $D(P_i)=(x^2_i+ y^2_i) – R^2$.

Рассмотрим рис. 2.5.2 , на котором показаны различные возможные способы прохождения истинной окружности через сетку пикселов. Пусть пиксел $P_{i-1}$ уже найден как ближайший к реальной изображенной окружности, и теперь требуется определить, какой из пикселов должен быть установлен следующим: $T_i$ или $S_i$. Для этого определим точку, которой соответствует минимальная ошибка: $D(S_i)=((q+1)^2+ p^2 ) – R^2$, $D(T_i)=((q+1)^2+ (p+1)^2 ) – R^2$.

Если $|D(S_i)| < |D(T_i)|$, то выбираем $S_i$, иначе - $T_i$. Введем величину $d_i = |D(S_i)| - |D(T_i)|$, тогда $S_i$ выбирается при $d_i < 0$, иначе выбирается $T_i$. Если рассматривать только часть окружности, дугу от $0^0$ до $45^0$, то $D(S_i) > 0$ так как точка$S_i$ лежит за пределами окружности, а $D(T_i) < 0$, так как $T_i$ находится внутри окружности, поэтому $d_i = D(S_i) + D(T_i)$.

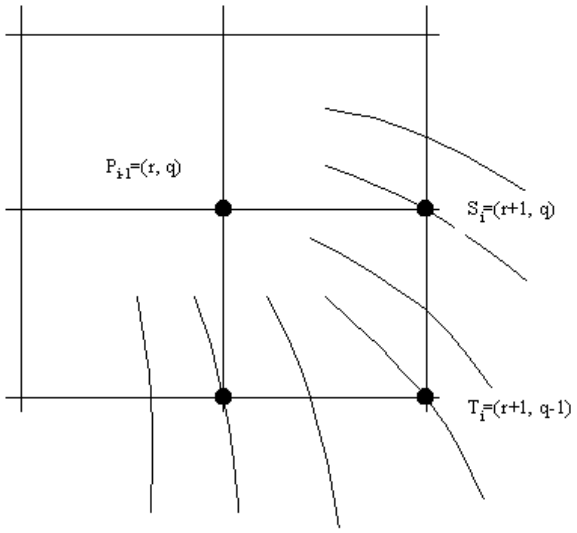



Рис. 2.5.2

Алгебраические вычисления, аналогичные тем, которые проводились для линии, приводят к результату:

$d_1=3-2R$.

Если выбираем $S_i$ (когда $d_i < 0$ ),

$\Delta_i=4x_{i-1}+6$;

если выбираем $T_i$ (когда $d_i >=0$ ),

$\Delta_i=4(x_{i-1} * y_{i-1}) +10$.

Блок-схема этого алгоритма (рис. 2.5.3).

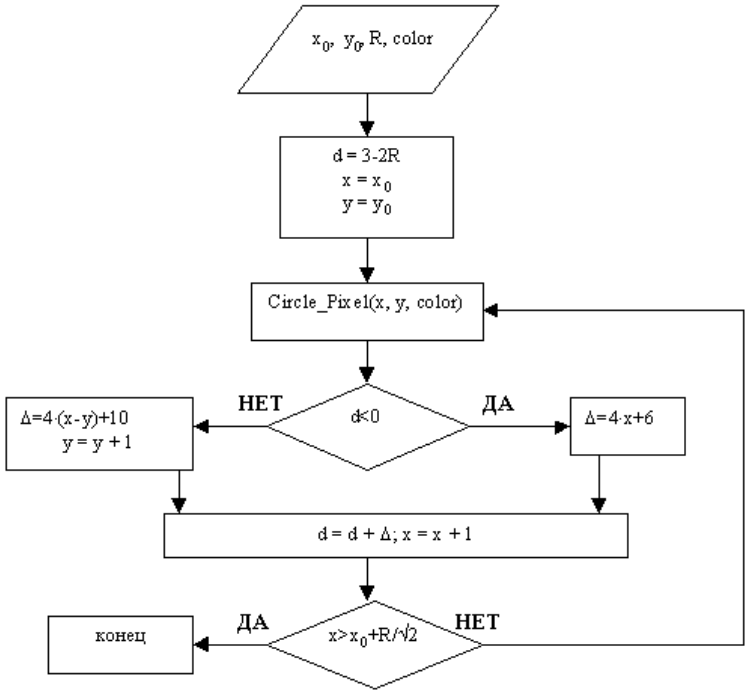



Рис. 2.5.3

Алгоритм хорош тем что отсутствуют операции с плавающей точкой, а также операции деления и извлечения корня.

## 2.6 Отсечение по полю вывода

На рис.2.6.1. показана плоская сцена и отсекающее окно регулярной формы. Окно задается левым (Л), правым (П), верхним (В) и нижним (Н) двумерными ребрами. Регулярным отсекающим окном является прямоугольник, стороны которого параллельны осям координат объектного пространства или осям координат экрана. Целью алгоритма отсечения является определение тех точек, отрезков или их частей, которые лежат внутри отсекающего окна. Эти точки, отрезки или их части остаются для визуализации. А все остальное отбрасывается. Поскольку в обычных сценах или картинках необходимо отсекать большое число отрезков или точек, то эффективность алгоритмов отсечения представляет особый интерес. Во многих случаях подавляющее большинство точек или отрезков лежит целиком внутри или вне отсекающего окна. Поэтому важно уметь быстро отбирать отрезки, подобные *ab*, или точки, подобные *p*, и отбрасывать отрезки, подобные *ij*, или точки, подобные *q* на рис.2.6.1.

Точки, лежащие внутри отсекающего окна, удовлетворяют условию: $x_л <= x <= x_п$ и $y_н <= y <= y_в$. Знак равенства здесь показывает, что точки, лежащие на границе окна, считаются находящимися внутри него.

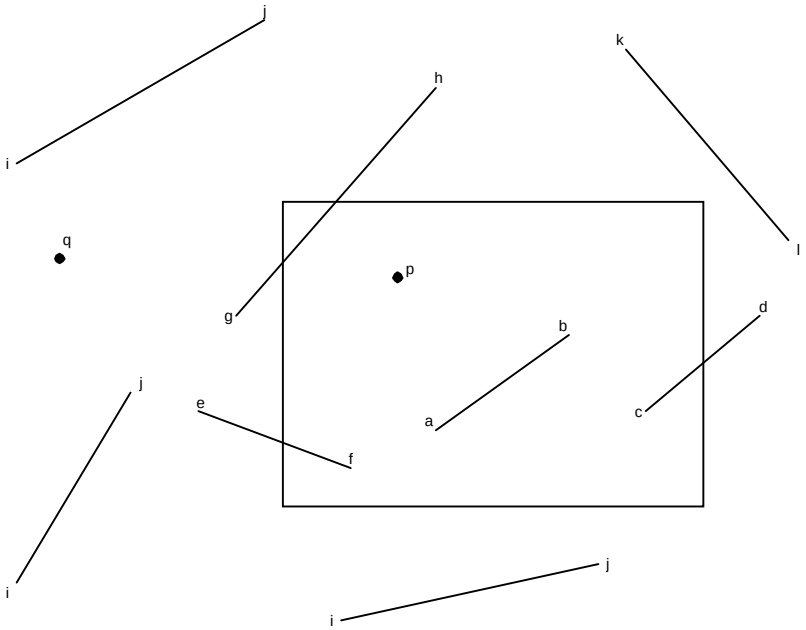



Рис. 2.6.1.

Отрезок лежит внутри окна и, следовательно, является видимым, если обе его концевые точки лежат внутри окна, например отрезок *ab* на рис. 2.6.1. Однако если оба конца отрезка лежат вне окна, то этот отрезок не обязательно лежит целиком вне окна, например отрезок *gh* на рис. 2.6.1. Если же оба конца отрезка лежат справа, слева, выше или ниже окна, то этот отрезок целиком лежит вне окна, а значит, невидим. Проверка последнего условия устранит все отрезки, помеченные *ij* на рис. 2.6.1. Но она не устранит ни отрезка *gh*, который видим частично, ни отрезка *kl*, который целиком невидим.

***Алгоритм Коэна-Сазерленда.*** Этот алгоритм нацелен на то, чтобы максимально эффективно отбрасывать линии, заведомо не попадающие в поле вывода. Для решения задачи определения принадлежности отрезка полю вывода используется следующий метод. Пространство разбивается на 9 областей, каждая из областей кодируется бинарным 4-х битным кодом рис. 2.6.2. Само поле вывода на рисунке помечено нулевым кодом (0000).

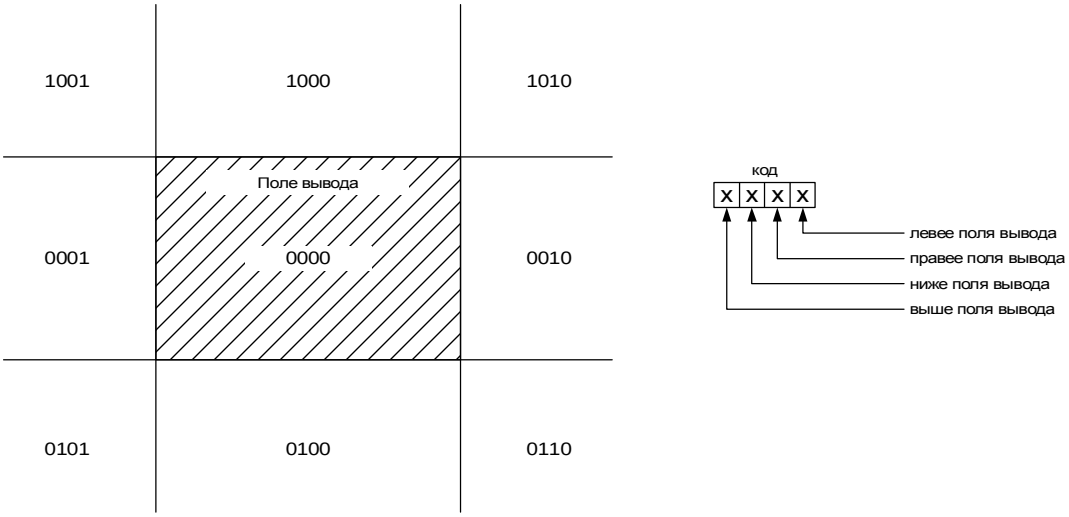



Рис. 2.6.2. «Кодирование пространства»

Для каждого отрезка рассчитываются коды концов ($К_1$, $К_2$) затем производится экспресс анализ:

- Если $K_1 \wedge K_2 \neq 0$, тогда отрезок лежит вне поля вывода – отрезок отбрасывается;
- Если $K_1 = K_2 = 0$, тогда отрезок полностью лежит внутри поля вывода – отсечение не нужно, отрезок полностью прорисовывается;
- Если $K_1 \wedge K_2 = 0$, отрезок может частично лежать внутри поля вывода – необходимо отсечение по полю вывода.

***Алгоритм отсечения по прямоугольной области.*** Отметим, что отсечение удобно проводить по каждой из границ поля по порядку. Например, сначала осуществить отсечение по верхней горизонтальной, затем по правой вертикальной и т.д. рис. 2.6.3. Таким образом многократно повторяется процесс отсечения по полуплоскости. На рис.2.6.3. жирным выделено ребро по которому происходит отсечение.

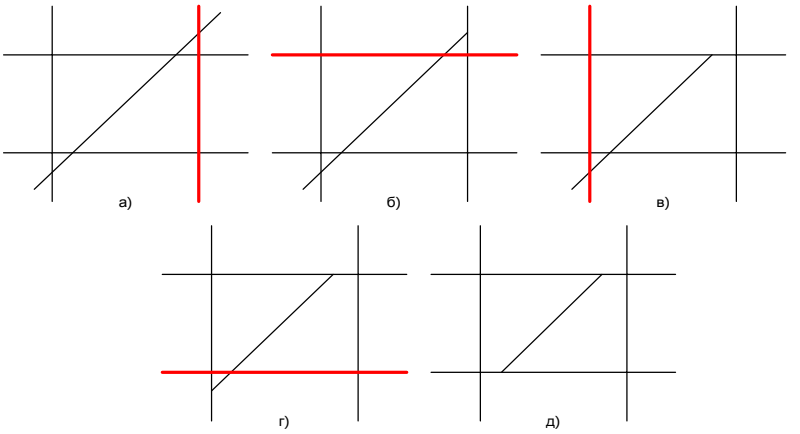



Рис.2.6.3. «Отсечение отрезка по прямоугольной области»

Чтобы решить задачу отсечения по отношению к одной из границ надо рассмотреть все возможные варианты расположения линии относительно полуплоскости: расположение «внутри» полуплоскости, расположение «вне», расположение «внутри-вне», расположение «вне-внутри». Также надо отметить, что точки лежащие на границе поля вывода принадлежат полю вывода.

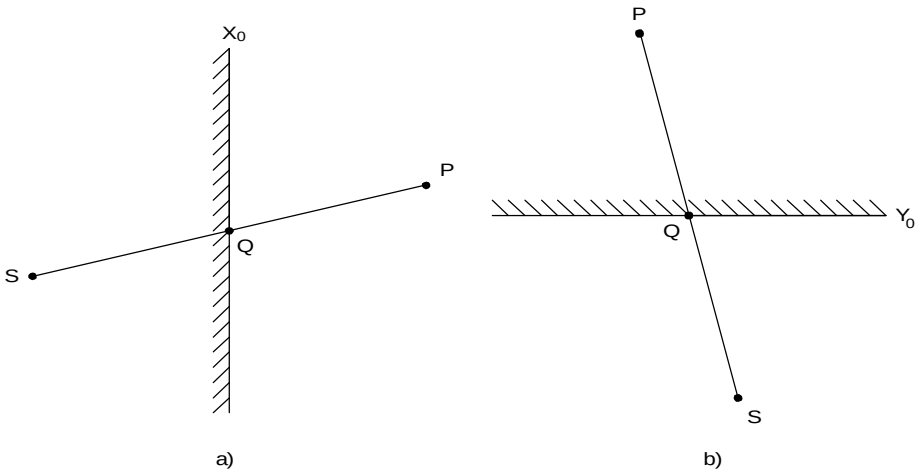



Рис.2.6.4.

На каждом шаге отсечения вычисляются новые координаты одной точек, найдем формулы для вычисления новых координат.

Формулы для расчета новых координат

для вертикальной границы: для горизонтальной границы:

$Q(x_0, y_Q, V_Q)$ $Q(x_Q, y_0, V_Q)$

а) $y_Q = y_S + \frac{x_0 - x_S}{x_p - x_S} \cdot (y_P - y_S)$; б) $x_Q = x_S + \frac{y_0 - y_S}{y_p - y_S} \cdot (x_P - x_S)$

$V_q = V_S + \frac{x_0 - x_S}{x_p - x_S} \cdot (V_P - V_S)$ $V_q = V_S + \frac{y_0 - y_S}{y_p - y_S} \cdot (V_P - V_S)$

## 2.7 Методы устранения ступенчатости.

Пусть имеются только две цветовые градации: черная и белая и стоит задача построения полутонового изображения рис. 2.7.1., т.е. необходимо получить оттенки серого. В условиях ограниченной цветовой палитры (8 цветов на точку) реализация цветового клина оборачивается получением восьми четко разделенных полос разных оттенков и потерей плавного перехода цвета. Для устранения такого эффекта существуют методы устранения ступенчатости.

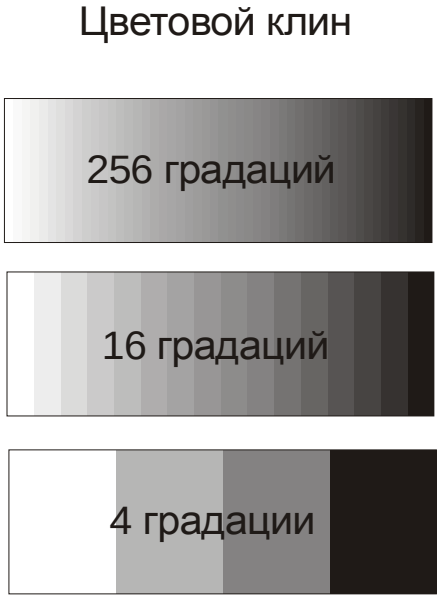



Рис. 2.7.1. «Цветовой клин»

***Метод полутонов.*** Сущность: каждый пиксель исходного изображения заменяется группой пикселей рис.2.7.2.

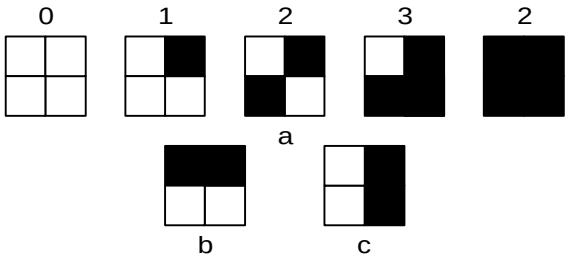



Рис. 2.7.2.

При такой организации получается пять возможных уровней или тонов серого (0-4). В общем случае для двухуровнего дисплея число возможных интенсивностей на единицу больше числа пикселов в клетке. При выборе конфигураций следует проявлять осторожность, так как иначе могут возникнуть нежелательные мелкомасштабные структуры. Например, не следует применять ни одну из конфигураций, изображенных на рис.2.7.2. b или c, иначе это приведет к тому, что для большой области с постоянной интенсивностью на изображении появятся нежелательные горизонтальные или вертикальные линии. Число доступных уровней интенсивности можно увеличить с помощью увеличения размера клетки.

Использование конфигураций ведет к потере пространственного разрешения, что приемлемо в случае, когда разрешение изображения меньше разрешения дисплея.

***Метод переноса.*** Метод дает полутоновые изображения с неплохим качеством. В данном методе разрешение изображения не изменяется. Сущность: предварительно задается число градаций серого, вычисляется середина этого диапазона и, проходя изображение в направлении справа - налево и сверху – вниз, рассматриваются точки изображения : если яркость точки ближе к белому, то ставится белая точка, а если к черному, то ставится черная точка, половина полученной ошибки прибавляется к яркости точки справа, а вторая половина к яркости точки снизу (пропорции можно изменять).

Пример реализации данного алгоритма для прямой приведен на рис.2.7.3.

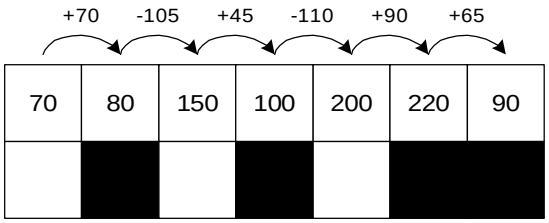



Рис. 2.7.3.

Метод переноса можно использовать и в случае большего числа градаций.

С помощью подобного метода можно рисовать цветную картинку. Если исходных градаций 16, то текущую яркость сравнивают с ближайшей, после чего выбирают соответствующую яркость, вычисляют ошибку и переносят на следующую точку. Алгоритм используется отдельно для каждого цвета.

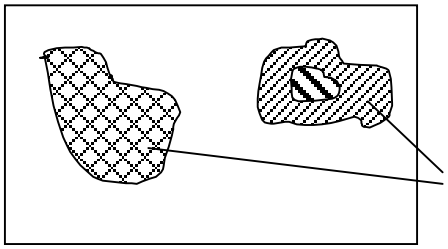
Хорошие результаты получаются в том случае если начальное значение погрешности для каждой строчки разыгрывается случайным образом, таким образом устраняется регулярность картинки.

## 2.8 Закраска областей

Существует несколько вариантов задания областей.

1) Внутренне определенные области.

Предполагается, что существуют группы точек, имеющих определенное значение кода, отличающееся для точек, находящихся за пределами границы области.

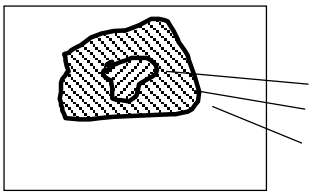


Эти области могут быть:
а) 4 – х связными;
б) 8 – и связными;

$V_1$ = const

Рис. 2.8.1.

2) Гранично-определенные области.



$V_{ВНУТР} \neq V_{ГРАН}$
$V_{ГРАН} = const$
$V_{ВНЕШ} \neq V_{ГРАН}$

Область, граница:
а) 4 – связная;
б) 8 – связная;

Рис. 2.8.2.

Существуют некоторые “затравочные” точки, соответственно которым необходимо закрашивать область. Для удержания 8-связной области нужно использовать 4-х связные границы, для удержания 4-связной области достаточно 8-и связных границ.

3) Область, заданная многоугольником, т.е. совокупностью заданных отрезков-, представляющая собой некоторую непересекающуюся ломаную.

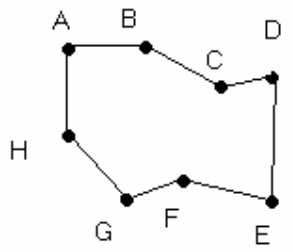



Рис 2.8.3

***Закраска гранично-связной области.*** Существует несколько алгоритмов закраски такой области. Рассмотрим два из них.

1) Рекурсивный метод. Метод прост для программирования, однако крайне неэффективен. По отношению к затравочной точке просматривают все точки. Если они не граничные, закрашивают и посылают в стек. Далее вытаскивают из стека, и если точка не является граничной и закрашенной, делают то же самое. См. блок-схему рис.2.8.4.

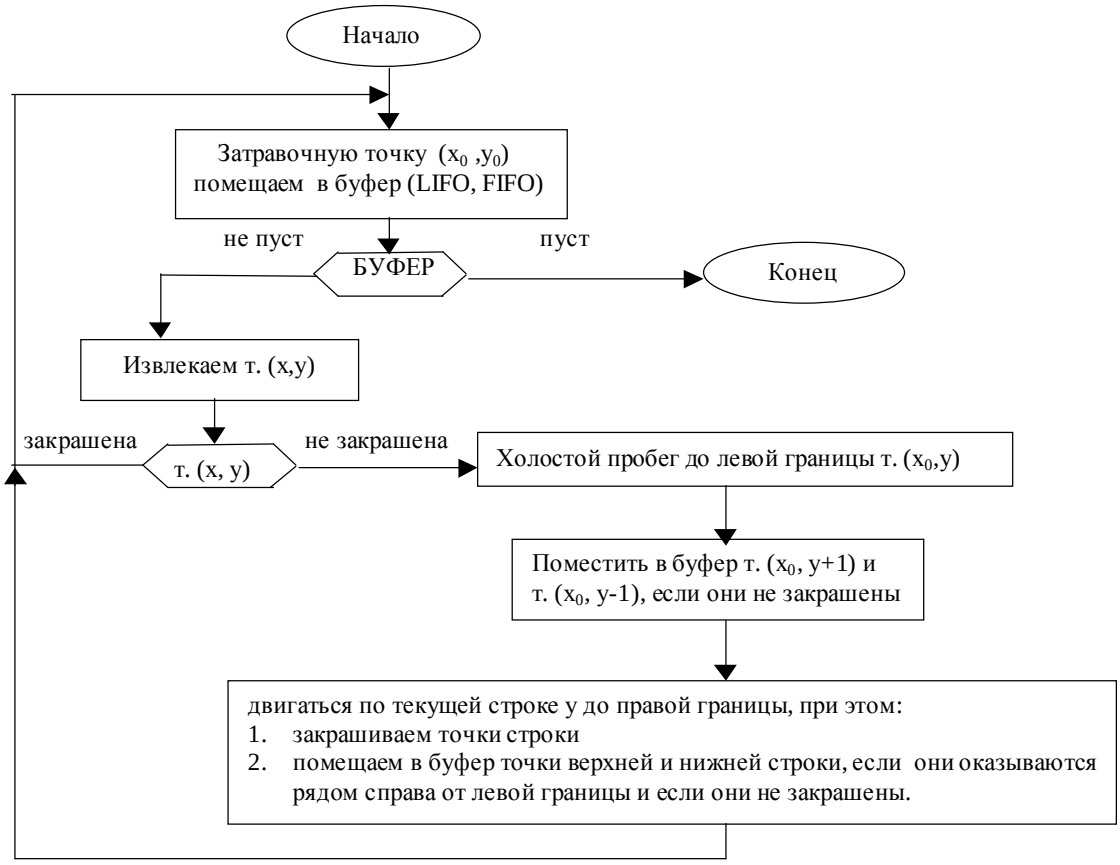



Рис. 2.8.4.

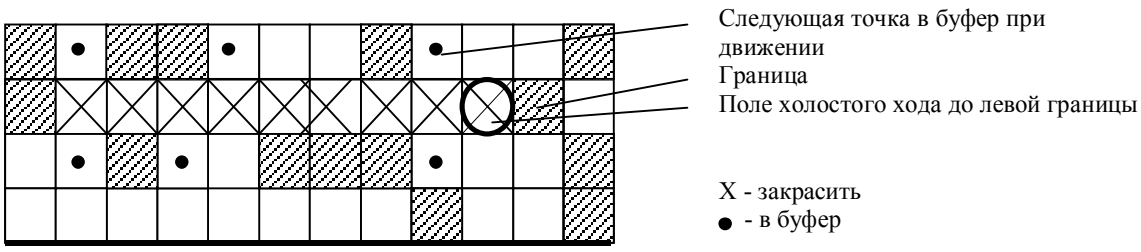



Рис. 2.8.5.

2). Алгоритм с применением метода построчного сканирования.

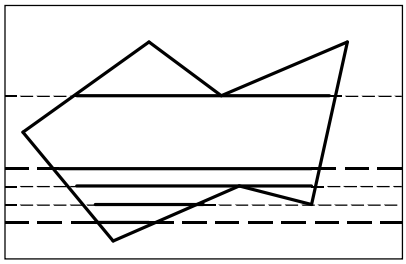
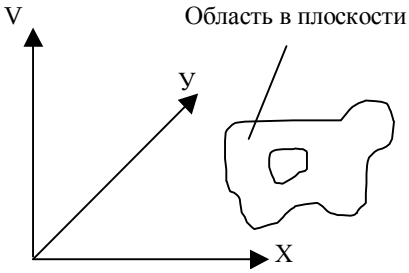


рис 2.8.6

Метод применяется при условии предварительно произведенного отсечения многоугольника по полю вывода. Общая идея метода заключается в заполнении области путем рисования горизонтальных (вертикальных) строк. Линии проводятся построчно от одного ребра поля вывода к противоположному в двух режимах: с прорисовкой (точки закрашиваются) и без. Причем смена режима происходит при достижении линией границы ребра закрашиваемой области (рис 2.8.6). Отследить это событие нетрудно при известных уравнениях всех ребер. Необходимо определить точки глобальных и локальных максимумов и(или) минимумов фигуры. При этом следует иметь ввиду, что в точках локальных экстремумов смены режима не происходит. Если граница фигуры содержит горизонтальные (вертикальные) ребра, то такие ребра не учитываются.

***Интерполяция яркости при закраске областей.*** О линейной интерполяции яркости при закраски области можно говорить, если закрашиваемая фигура плоская т.е. лежит в одной плоскости, например (XУ).




Тогда можно написать уравнение плоскости:

$A \cdot x + B \cdot y + C \cdot V + D = 0, \quad C \neq 0;$

Рис. 2.8.7.

Плоскость определяется по трём точкам:

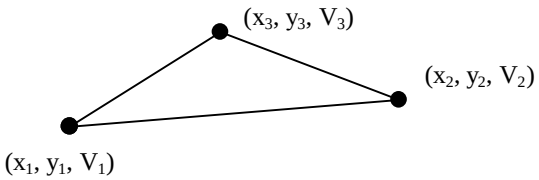



Рис. 2.8.8.

$$A = \begin{Vmatrix} y_1 & V_1 & 1 \\ y_2 & V_2 & 1 \\ y_3 & V_3 & 1 \end{Vmatrix}; \quad B = \begin{Vmatrix} V_1 & x_1 & 1 \\ V_2 & x_2 & 1 \\ V_3 & x_3 & 1 \end{Vmatrix}; \quad C = \begin{Vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{Vmatrix}; \quad D = -\begin{Vmatrix} x_1 & y_1 & V_1 \\ x_2 & y_2 & V_2 \\ x_3 & y_3 & V_3 \end{Vmatrix};$$

|| || - определитель матрицы;

$$V = -\frac{A \bullet x + B \bullet y + D}{C} = \alpha \bullet x + \beta \bullet y + \gamma, \text{ где } \alpha = -\frac{A}{C},\ \beta = -\frac{B}{C},\ \gamma = -\frac{D}{C};$$

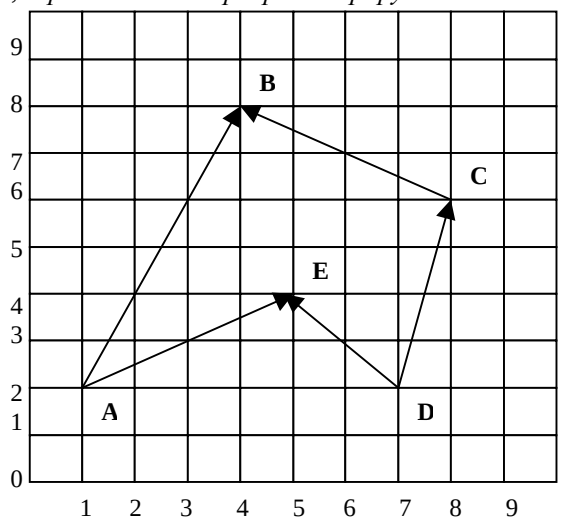
$V = V_1 + \alpha (x - x_1) + \beta (y - y_1)$, где
V – яркость в произвольной точке, $V_1$ – яркость известная.

Алгоритм работает с помощью 2-х таблиц:
1. таблица ребер (ТР);
2. таблица активных ребер (ТАР).

В ТР заносятся все ребра, а в ТАР лишь те ребра, которые пересекаются при сканировании.

Составление ТР:

Все ребра делятся на группы по нарастанию у-координаты, а внутри группы рёбра упорядочиваются в соответствии с нарастанием $x_{нач.}$ Ребро в таблицу заносится только 1 раз, *горизонтальные ребра игнорируются.*




A (1, 2)
B (4, 8)
C (8, 6)
D (7, 2)
E (5, 4)

Таблица рёбер.

| Группа, № | Ребро | $y_{min}$ | $y_{max}$ | $x_{НАЧ}$ | $x_{КОН}$ | $\Delta x$ | $V_{НАЧ}$ | $V_{КОНЕЧ}$ | $\Delta V$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | AB | 2 | 8 | 1 | 4 | 0.5 | | | |
| | AE | | 4 | 1 | 5 | 2 | | | |
| | DC | | 6 | 7 | 8 | 0.25 | | | |
| | DE | | 4 | 7 | 5 | -1 | | | |
| | | | | | | | | | |
| 2 | CB | 6 | 8 | 8 | 4 | -2 | | | |

$$\Delta x = \frac{x_{КОН} - x_{НАЧ}}{y_{max} - y_{min}}; \quad \Delta V = \frac{V_{КОНЕЧ} - V_{НАЧ}}{y_{max} - y_{min}};$$

Алгоритм:

1. Формируется ТР и инициализируется ТАР;
2. Выбирается первая координата сканируемой строки: $y = \min\{y_{min}\}$;
3. Если $y = y_{min}$, то осуществляется перенос группы из ТР в ТАР (горизонтальные ребра в таблицу не записываются).

Таблица активных ребер.

| Ребро | $У_{min}$ | $У_{max}$ | $x_{НАЧ}$ | $\Delta x$ | $V_{НАЧ}$ | $\Delta V$ |
|---|---|---|---|---|---|---|
| AB | 2 | 8 | 1 | 0.5 | | |
| AE | 2 | 4 | 1 | 2 | | |
| DC | 2 | 6 | 7 | 0.25 | | |
| DE | 2 | 4 | 7 | -1 | | |

4. Упорядочивание ребер в ТАР по возрастанию $x_{НАЧ;}$
5. Сканирование (проводят сканирующую строку);

а) переключение режимов по $x_{НАЧ.}$

б) учёт локальных экстремумов. Если вершина – локальный экстремум, то режим следует сохранить. Для отслеживания данной ситуации можно включить в ТР $y_{max}$, при совпадении необходим дополнительный анализ.

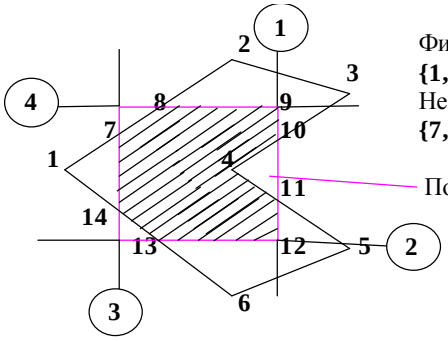
в) проводить линейную интерполяцию яркости при закраске от $(x_{НАЧ}, V_{НАЧ})_i$ до $(x_{НАЧ}, V_{НАЧ})_{i+1}$.

6. Удалить из ТАР те ребра, для которых справедливо $y = y_{max;}$
7. Для всех элементов в ТАР произвести: $x_{НАЧ} = x_{НАЧ} + \Delta x$; $V_{НАЧ} = V_{НАЧ} + \Delta V$;
8. Переход к следующей сканирующей строке: y=y+1;
9. Проверка на окончание: если $y>\max\{y_{max}\}$, то конец, иначе осуществить переход к п.3).

***Отсечение плоских фигур (алгоритм Коэна-Сазерленда).***

Рассмотрим задачу об отсечении произвольного многоугольника по границе заданного выпуклого многоугольника. Одним из наиболее простых алгоритмов для решения этой задачи является алгоритм Коэна –Сазерленда.

Алгоритм сводит основную задачу к серии более простых задач об отсечении многоугольника вдоль прямой, проходящей через одно из ребер отсекающего многоугольника.




Фигура задана списком вершин:
**{1, 2, 3, 4, 5, 6}**
Необходимо получить:
**{7, 8, 9, 10, 4, 11, 12, 13, 14}**

Поле вывода.

Рис. 2.8.9

На каждом шаге выбираем очередное (рис 2.8.10) ребро отсекающего многоугольника. В нашем случае это ребра помеченные цифрами. Предварительно необходимо договориться о направлении обхода вершин отсекаемого многоугольника (по часовой стрелке или против неё). После этого поочередно проверяется положение всех вершин данного многоугольника относительно прямой, проходящей через выбранное текущее ребро. При этом в результирующий многоугольник добавляется 0,1 или 2 вершины.

.

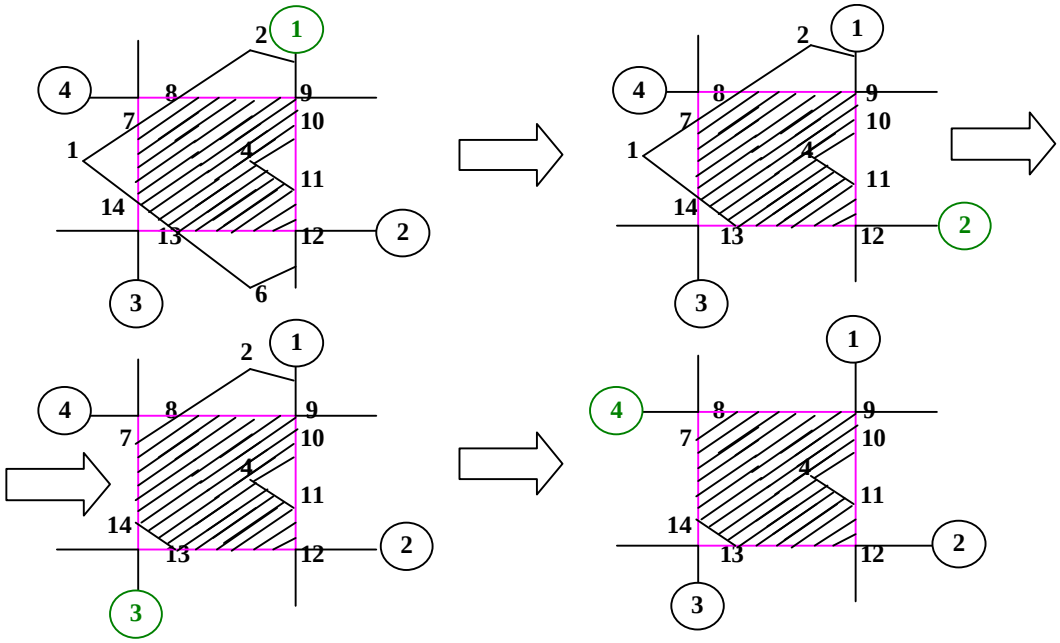


Рис 2.8.10

При последовательном отсечении многоугольника по четырем границам возможны 4 различных ситуации (рис 2.8.10). Может также возникнуть вариант, когда ребро совпадает с границей.

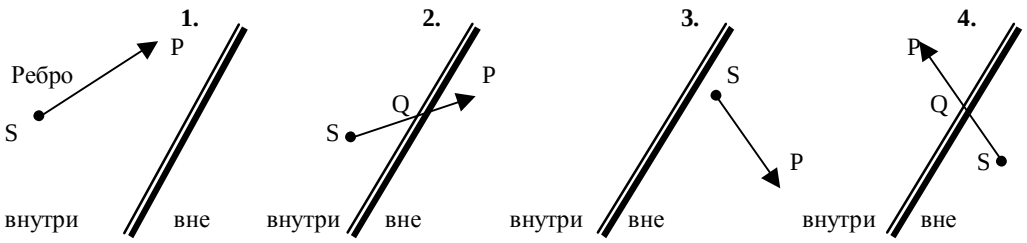



Рис 2.8.11

Предположим, что точка S уже обработана.

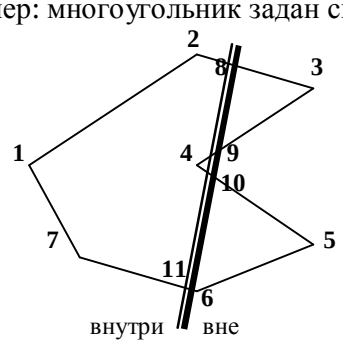
В случае 1: ребро целиком лежит во внутренней области и точка P добавляется в результирующий многоугольник.

В случае 2: в результирующий многоугольник добавляется точка пересечения Q.

В случае 3: обе вершины лежат во внешней области и поэтому в результирующий многоугольник не добавляется ни одной точки.

В случае 4: в результирующий многоугольник добавляются точка пересечения Q и точка P.

Пример: многоугольник задан списком вершин {1,2,3,4,5,6,7,1}



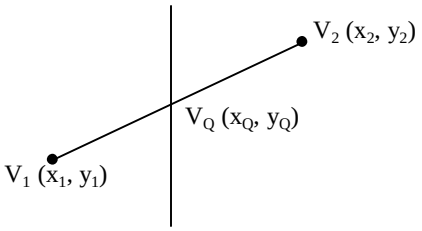

Отсчет необходимо начинать с точки, находящейся внутри границы.

Последовательно переберём все рёбра. Начнем процесс с точки 1 и ребра **1-2**. Начнём формировать новый список вершин. В соответствии с ориентацией ребра занесём в выходной список вершину 2 => {2}. Далее рассмотрим ребро **2-3**: добавляется точка 8 => {2, 8}

**3-4**: {2, 8, 9, 4}.
**4-5**: {2, 8, 9, 4, 10}.
**5-6**: {2, 8, 9, 4, 10}. Ребро полностью оказалось вне, поэтому ничего не добавляется.
**6-7**: {2, 8, 9, 4, 10, 11, 7}.
**7-1**: {2, 8, 9, 4, 10, 11, 7, 1}.

Таким образом, новый список вершин: {2, 8, 9, 4, 10, 11, 7, 1}.

Определение яркости в точке пересечения с областью вывода.



Вычисления проводятся по одной из двух формул:

(1) $V_Q = V_1 + \frac{y_Q - y_1}{y_2 - y_1} \cdot (V_2 - V_1);$

(2) $V_Q = V_1 + \frac{x_Q - x_1}{x_2 - x_1} \cdot (V_2 - V_1);$

Примечание:
По формуле (1) считают, если $|y_2 - y_1| \geq |x_2 - x_1|$, по формуле (2) - если $|y_2 - y_1| < |x_2 - x_1|$;

## 3. Геометрические преобразования

### 3.1 Координаты и преобразования

Описание, конструирование, манипулирование и представление геометрических объектов являются центральными работами в графических системах. Их поддержка в требуемом объеме за счет соответствующих математических методов, алгоритмов и программ оказывают существенное влияние на возможности и эффективность графической системы. В данном разделе будут рассмотрены математические методы для описания геометрических преобразований координат в двух и трехмерном случае, будут обсуждены некоторые вопросы эффективности, рассмотрены геометрические преобразования растровых картин.

Далее большими буквами x, y, z будут обозначаться обычные декартовые координаты, а маленькие буквы X, Y, Z будут использоваться для обозначения т.н. однородных координат.

### 3.2 Аффинные преобразования на плоскости

В компьтерной графике все что относится к двумерному случаю, принято обозначать символом 2D (2-dimension). Здесь поговорим о трех типах преобразований: параллельном переносе, масштабировании, повороте.

***Параллельный перенос***

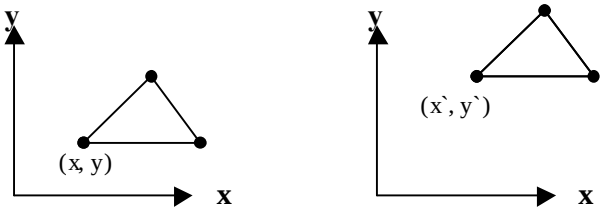



Рис 3.2.1

Допустим, что на плоскости введена прямолинейная координатная система. Тогда каждой точке ставится в соответствие упорядоченная пара чисел (x, y) ее координат (рис 3.1.1). Вводя на плоскости еще одну прямолинейную систему координат, мы ставим в соответствие той же точке другую пару чисел – (x’, y’). Параллельный перенос в плоском случае имеет вид:

$$\begin{cases} x` = x + Dx \\ y` = y + Dy \end{cases}$$

$$\underbrace{[x', y']}_{P'} = \underbrace{[x, y]}_{P} + \underbrace{[Dx, Dy]}_{T}$$

или в векторной форме:

$$\boxed{\mathbf{P`} = \mathbf{P} + \mathbf{T},}$$

где

**P`** = [x` y`] - вектор-строка преобразованных координат,
**P** = [x y] - вектор-строка исходных координат,
**T** = [Tx Ty] - вектор-строка сдвига,
x`, y` - преобразованные координаты,
x, y - исходные координаты точки,
Tx, Ty - величина сдвига по осям.

***Масштабирование***

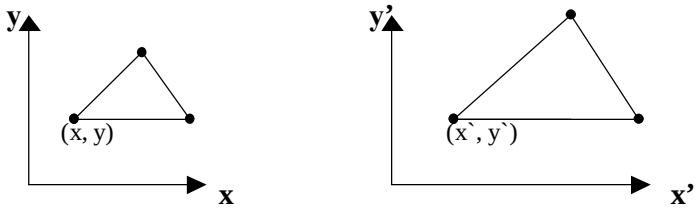



Рис 3.2.2

Преобразование масштабирования представляется как произведение исходных координат и матрицы преобразования и относительно начала координат имеет вид:

$$\begin{cases} x` = x \bullet Sx \\ y` = y \bullet Sy \end{cases}$$

или

$$\underbrace{[x`, y`]}_{P`} = \underbrace{[x, y]}_{P} \bullet \underbrace{\begin{pmatrix} Sx & 0 \\ 0 & Sy \end{pmatrix}}_{S}$$

или в матричной форме:

$$\boxed{P` = P \cdot S,}$$

где

Sx, Sy - коэффициенты масштабирования по осям, а
S - матрица преобразования

***Поворот***

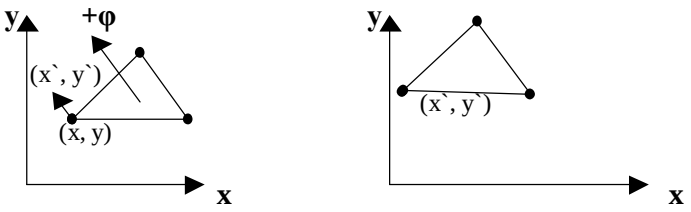



Рис 3.2.3

Преобразование поворота относительно начала координат имеет вид:

$$\begin{cases} x` = x \bullet \cos(\varphi) - y \bullet \sin(\varphi) \\ y` = x \bullet \sin(\varphi) + y \bullet \cos(\varphi) \end{cases}$$

или

$$\underbrace{[x`, y`]}_{P`} = \underbrace{[x, y]}_{P} \bullet \underbrace{\begin{pmatrix} \cos(\varphi) & \sin(\varphi) \\ -\sin(\varphi) & \cos(\varphi) \end{pmatrix}}_{R}, \text{ где}$$

R – матрица поворота
φ – положительный угол поворота (направление против часовой стрелки).

или в матричной форме:

$$P` = P \cdot R,$$

***Преобразование в однородную систему координат***

Как видно двумерные преобразования имеют различный вид. Сдвиг реализуется сложением, а масштабирование и поворот - умножением. Это различие затрудняет формирование суммарного преобразования. Это можно устранить, например, так: перейти к описанию произвольной точки плоскости не упорядоченной парой чисел, как в декартовой системе координат, а упорядоченной тройкой чисел:

[ X Y W ].

Однородными координатами произвольной точки плоскости, заданной координатами x и y, называется любая тройка одновременно неравных нулю чисел X, Y, W, связанных с заданными координатами x и y следующими соотношениями:

$$x = \frac{X}{Y}; y = \frac{Y}{W}; W \neq 0$$

Однородные координаты можно представить как промасштабированные с коэффициентом W значения двумерных координат, расположенные в плоскости с Z = W.

В силу произвольности значения W в однородных координатах не существует единственного представления точки, заданной в декартовых координатах. Преобразования параллельного переноса, масштабирования и поворота в однородных координатах относительно центра координат все имеют одинаковую форму произведения вектора исходных координат на матрицу преобразования.

При помощи троек однородных координат и матриц третьего порядка можно описать любое аффинное преобразование плоскости. В самом деле, считая W=1, три выше рассмотренные операции можно записать в виде произведения на соответствующие матрицы:

***Параллельный перенос:***

$$[X`, Y`, 1]=[X, Y, 1]\bullet \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Dx & Dy & 1 \end{pmatrix};$$

Перемножив, получим: [X + Dx, Y + Dy, 1]. Переход в декартову систему координат даст [x+Dx, y+Dy].

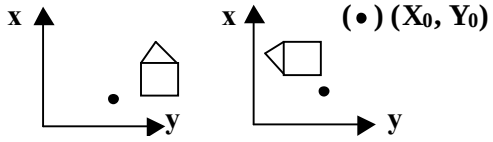
***Масштабирование:***

$$P` = P\bullet S; \quad \text{где } S = \begin{bmatrix} S_X & 0 & 0 \\ 0 & S_Y & 0 \\ 0 & 0 & 1 \end{bmatrix};$$

***Поворот:***

$$P' = P\bullet R; \quad \text{где } R = \begin{bmatrix} \cos j & \sin j & 0 \\ -\sin j & \cos j & 0 \\ 0 & 0 & 1 \end{bmatrix};$$

Рассмотрим примеры аффинных преобразований плоскости. Задачу поворота вокруг фиксированной точки проще решать, разделив ее на три более простых подзадачи: преобразование смещения, преобразование поворота, преобразование смещения.




P` = P•M,

***Преобразование смещения***

представляет собой перенос на вектор $T(-X_0, -Y_0)$ для совмещения центра поворота с началом координат;

$$T = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -X_0 & -Y_0 & 1 \end{pmatrix}$$

***Преобразование поворота***

Поворот на угол φ:

$$R(\varphi)=\begin{pmatrix} \cos(\varphi) & \sin(\varphi) & 0 \\ -\sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

***Преобразование смещения***

представляет собой перенос на вектор $T(X_0, Y_0)$ для возвращения центра поворота в прежнее положение.

Перемножим матрицы в том же порядке, как они выписаны:

$T(-X_0, -Y_0) \cdot R(\varphi) \cdot T(X_0, Y_0)$.

Смещаем точку в начало координат

Возвращаем точку в исходное состояние

В результате произведений матриц получаем матрицу преобразования M:

$$M = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -X_0 & -Y_0 & 1 \end{pmatrix} \bullet \begin{pmatrix} \cos(\varphi) & \sin(\varphi) & 0 \\ -\sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \bullet \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ X_0 & Y_0 & 1 \end{pmatrix} =$$

$$= \begin{pmatrix} \cos(\varphi) & \sin(\varphi) & 0 \\ -\sin(\varphi) & \cos(\varphi) & 0 \\ X_0\bullet(1-\cos(\varphi))+Y_0\bullet\sin(\varphi) & Y_0\bullet(1-\cos(\varphi)) - X_0\bullet\sin(\varphi) & 1 \end{pmatrix}$$

В общем случае матрицу преобразований можно записать следующим образом:

$$M = \begin{pmatrix} m_{11} & m_{12} & 0 \\ m_{21} & m_{22} & 0 \\ m_{31} & m_{32} & 1 \end{pmatrix}; \quad P` = P\bullet M$$

Перейдём к алгебраическому выражению:

$$\begin{cases} x' = x \bullet m_{11} + y \bullet m_{21} + m_{31} \\ y' = x \bullet m_{12} + y \bullet m_{22} + m_{32} \end{cases}$$

## 3.3 Трехмерные геометрические преобразования

Обратимся теперь к трехмерному случаю (3D) – (3-dimension). Далее при рассмотрении трехмерных преобразований, в основном, используется общепринятая в векторной алгебре правая система координат (рис. 3.3.1а). При этом, если смотреть со стороны положительной полуоси в центр координат, то поворот на $+90^{\circ}$ (против часовой стрелки) переводит одну положительную ось в другую (направление движения расположенного вдоль оси и поворачивающегося против часовой стрелки правого винта и положительной полуоси совпадают). В некоторых, специально оговариваемых случаях,

используется левая система координат (см. рис. 3.3.1б). В трехмерной машинной графике более удобной является левая система координат. Тогда если, например, поверхность экрана совмещена с плоскостью XY, то большим удалениям от наблюдателя соответствуют точки с большим значением Z (см. рис. 3.3.1б).

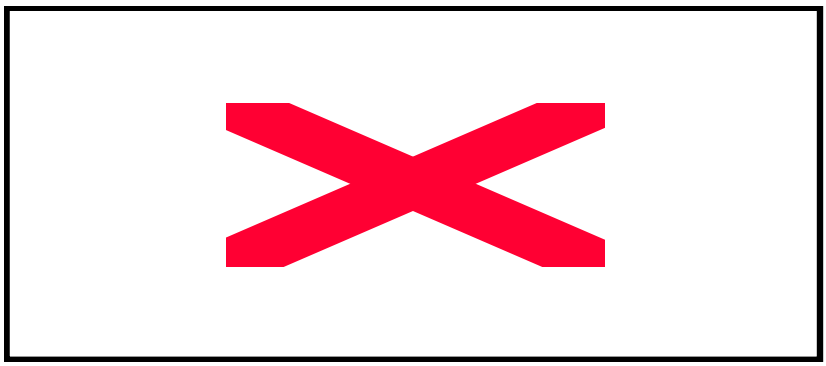


Рис 3.3.1

Работа с однородными трехмерными координатами и матрицами преобразования (формирование и композиция) подобна таковой для двумерного случая, поэтому здесь будут рассмотрены только матрицы преобразований сдвига, масштабирования и поворота и пример конструирования матрицы преобразования по известному его результату.

Подобно тому как в двумерном случае точка в однородных координатах представляется трехмерным вектором [ X Y W ], а матрицы преобразований имеют размер 3×3, для трехмерного случая точка представляется четырехмерным вектором [ X Y Z W ], где W не равно нулю, а матрицы преобразований имеют размер 4×4.

Формулы для преобразования: $x = \frac{X}{W}$; $y = \frac{Y}{W}$; $z = \frac{Z}{W}$; $W \neq 0$;

***Параллельный перенос***

$$P` = P \cdot T;$$

Где T = $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ Dx & Dy & Dz & 1 \end{pmatrix}$ - матрица переноса

***Масштабирование***

$$P` = P \cdot S;$$

Где S = $\begin{pmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

***Поворот***

Первой особенностью поворота в 3-х мерном пространстве является то , что один поворот в пространстве следует раскладывать на 3 поворота: вокруг оси абсцисс, вокруг оси ординат, вокруг оси аппликат. Вторая особенность- это направление угла поворота. Существует договоренность, что положительным

направлением угла поворота для обеих систем координат считается то направление, при котором осуществляется кратчайший переход:

Для X: Y→Z
Y: Z→X
Z: X→Y

В левой системе координат положительными будут повороты по часовой стрелке, если смотреть с положительного конца полуоси; для правой – против часовой стрелки(рис 3.3.2).

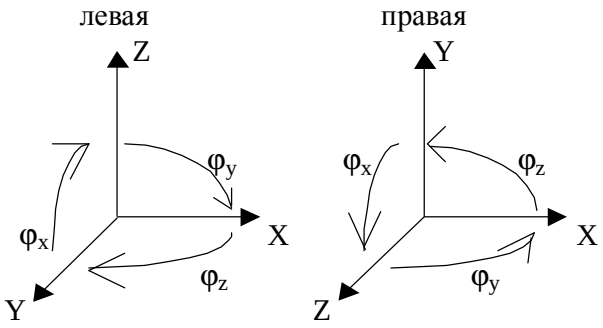



Рис 3.3.2

Ранее рассмотренная для двумерного случая матрица поворота является в то же время трехмерным поворотом вокруг оси Z. Так как при трехмерном повороте вокруг оси Z (поворот в плоскости XY) размеры вдоль оси Z неизменны, то все элементы третьей строки и третьего столбца равны 0, кроме диагонального, равного 1:

$$Rz(\varphi_z) = \begin{pmatrix} \cos(\varphi_z) & \sin(\varphi_z) & 0 & 0 \\ -\sin(\varphi_z) & \cos(\varphi_z) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

При повороте вокруг оси X (в плоскости YZ) размеры вдоль оси X не меняются, поэтому все элементы первой строки и первого столбца равны 0, за исключением диагонального, равного 1:

$$Rx(\varphi_x) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\varphi_x) & \sin(\varphi_x) & 0 \\ 0 & -\sin(\varphi_x) & \cos(\varphi_x) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

При повороте вокруг оси Y (в плоскости XZ) размеры вдоль оси Y не меняются, поэтому все элементы второй строки и второго столбца равны 0, за исключением диагонального, равного 1:

$$Ry(\varphi_y) = \begin{pmatrix} \cos(\varphi_y) & 0 & -\sin(\varphi_y) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\varphi_y) & 0 & \cos(\varphi_y) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

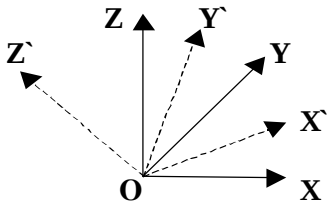
В общем случае любой поворот в пространстве может быть описан с помощью некоторых комбинаций этих трех поворотов. Причём повороты не обладают свойством коммутативности: $R_X \cdot R_Y \cdot R_Z \neq R_Z \cdot R_X \cdot R_Y$. Любой произвольный поворот может быть представлен 6 сочетаниями элементарных поворотов. Причем в каждом случае будут свои углы (за исключением вырожденных ситуаций).

Договоримся в общем случае матрицу поворота записывать в следующем виде:

$$R = \begin{pmatrix} a_1 & a_2 & a_3 & 0 \\ b_1 & b_2 & b_3 & 0 \\ c_1 & c_2 & c_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Для нее действует свойство ортогональности: $R^{-1} = R^T$.

Рассмотрим ситуацию, где координатная система 2 повернута относительно системы 1.




система координат1: OXYZ
система координат2: OX`Y`Z`

$P = P' \bullet R^{-1} = P' \bullet R^T$; где R – матрица поворота.

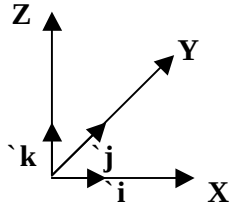
P` = [X`, Y`, Z`, 1];

P = [X, Y, Z, 1];

Тогда все элементы R могут быть интерпретированы как косинусы соответствующих углов, а матрица называется матрицей “направляющих косинусов”.

$$R = \begin{pmatrix} a_1 & a_2 & a_3 & 0 \\ b_1 & b_2 & b_3 & 0 \\ c_1 & c_2 & c_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \cos(XOX`) & \cos(XOY`) & \cos(XOZ`) & 0 \\ \cos(YOX') & \cos(YOY') & \cos(YOZ') & 0 \\ \cos(ZOX') & \cos(ZOY') & \cos(ZOZ') & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Элементы матрицы R можно также рассматривать в виде векторов:




**`i, `j, `k –** это вектора единичной длины.

Введем вспомогательные вектора:

$\bar{N}_1 = \bar{i} \bullet a_1 + \bar{j} \bullet b_1 + \bar{k} \bullet c_1$ $\quad$ $\bar{M}_1 = \bar{i} \bullet a_1 + \bar{j} \bullet a_2 + \bar{k} \bullet a_3$

$\bar{N}_2 = \bar{i} \bullet a_2 + \bar{j} \bullet b_2 + \bar{k} \bullet c_2$ $\quad$ $\bar{M}_2 = \bar{i} \bullet b_1 + \bar{j} \bullet b_2 + \bar{k} \bullet b_3$

$\bar{N}_3 = \bar{i} \bullet a_3 + \bar{j} \bullet b_3 + \bar{k} \bullet c_3$ $\quad$ $\bar{M}_3 = \bar{i} \bullet c_1 + \bar{j} \bullet c_2 + \bar{k} \bullet c_3$

Модули всех векторов равны единице: $|\bar{N}_2| = |\bar{N}_3| = |\bar{N}_1| = |\bar{M}_1| = |\bar{M}_2| = |\bar{M}_3|$.

Все скалярные произведения равны нулю:

$\bar{N}_1 \bullet \bar{N}_2 = \bar{N}_1 \bullet \bar{N}_3 = \bar{N}_2 \bullet \bar{N}_3 = 0.$

Запишем векторное произведение:

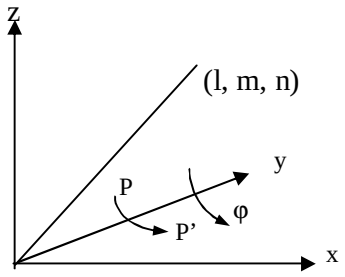
$\bar{N}_1 \times \bar{N}_2 = \bar{N}_3 \qquad \bar{M}_1 \times \bar{M}_2 = \bar{M}_3$

$\bar{N}_2 \times \bar{N}_3 = \bar{N}_1 \qquad \bar{M}_2 \times \bar{M}_3 = \bar{M}_1$

$\bar{N}_3 \times \bar{N}_1 = \bar{N}_2 \qquad \bar{M}_3 \times \bar{M}_1 = \bar{M}_2$

Приведем важный пример построения матрицы сложного преобразования по его геометрическому описанию. Построим матрицу вращения на угол φ вокруг прямой L, проходящей через начало координат и имеющую направляющий вектор (l,m,n).

$l^2+m^2+n^2=1$,т.е. координата нормирована




Решение сформулированной задачи разбивается на несколько шагов:

1. Совмещение оси аппликат с прямой L двумя поворотами вокруг оси абсцисс и оси ординат,
2. Вращение вокруг прямой L на заданный угол φ,
3. Поворот вокруг оси ординат и абсцисс на углы противоположные по направлению углам поворота на шаге 1.

На каждом шаге строится матрица поворота. При этом крайне важно заметить, что вращение в пространсве некоммутативно, поэтому порядок, в котором проводятся вращения, является существенным. Перемножив найденные матрицы в порядке их построения, получим следующую матрицу:

$$M = \begin{pmatrix} l^2+\cos(\varphi)\bullet(1-l^2) & l\bullet(1-\cos(\varphi))\bullet m+n\bullet\sin(\varphi) & l\bullet(1-\cos(\varphi))\bullet n-m\bullet\sin(\varphi) & 0 \\ l\bullet(1-\cos(\varphi))\bullet m & m^2+\cos(\varphi)\bullet(1-m^2) & m\bullet(1-\cos(\varphi))\bullet n+l\bullet\sin(\varphi) & 0 \\ l\bullet(1-\cos(\varphi))\bullet n+m\bullet\sin(\varphi) & m\bullet(1-\cos(\varphi))\bullet n-l\bullet\sin(\varphi) & n^2+\cos(\varphi)\bullet(1-n^2) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

M – в общем случае не ортогональная матрица, т.е. $M^{-1} \neq M^T$,

а R–ортогональная ($R^{-1}=R^T$).

В общем виде матрица преобразований имеет вид:

$$M = \begin{pmatrix} m_{11} & m_{12} & m_{13} & 0 \\ m_{21} & m_{22} & m_{23} & 0 \\ m_{31} & m_{32} & m_{33} & 0 \\ m_{41} & m_{42} & m_{43} & 1 \end{pmatrix}$$

Координаты точки вычисляются по следующим формулам:

$X` = X*m_{11}+Y* m_{21}+ Z* m_{31}+ m_{41}$

$Y` = X*m_{12}+Y* m_{22}+ Z* m_{32}+ m_{42}$

$Z` = X*m_{13}+Y* m_{23}+ Z* m_{33}+ m_{43}$

## 3.4 Проецирование

Изображение объектов на плоскости связано еще с одной операцией – проецированием при помощи пучка прямых. Для получения проекции объекта на картинную плоскость необходимо провести через каждую его точку прямую из заданного проектирующего пучка и затем найти координаты точки пересечения этой прямой с плоскостью изображения.

***Центральное проецирование.***

В случае центрального проецирования все прямые исходят из одной точки – центра собственного пучка.

Чтобы перейти к обсуждению процесса получения проекции математически введем еще несколько понятий:

1. Пиксель, точка – центр проецирующего пучка;
2. Картинная плоскость - плоскость, на которую проецируется объект;
3. Объект, который необходимо проецировать;
4. Проектор – линии через центр и точки объекта (в общем случае проекторы – это не прямые линии).

Если картина поверхности (КП) – плоская, то говорят о плоских проекциях. Если проекторы являются прямыми линиями, то говорят о геометрических проекциях. При наличии обоих этих факторов говорят о плоских геометрических проекциях (ПГП). Именно ПГП отображают процессы фотографирования и зрения.

Введем **мировую** систему координат и рассмотрим наблюдателя. Центр проецирования связан с наблюдателем.

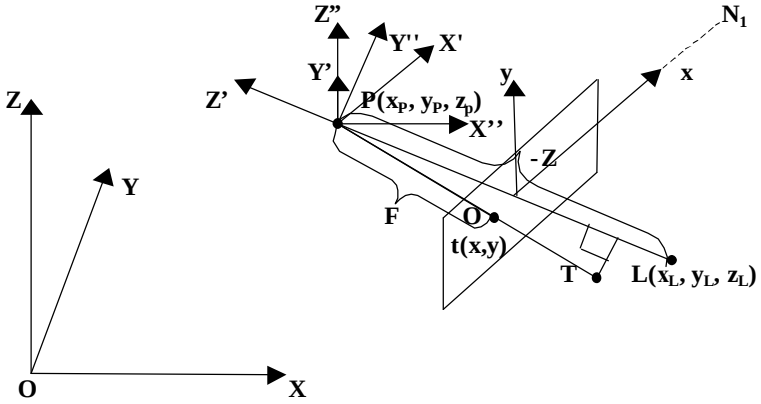



Рис 3.4.1

Введем главный луч проецирования перпендикулярно плоскости изображения - его задают точки P и L (рис 3.4.1). Начало координат обычно связывают с точкой пересечения главного луча и плоскости. (x)L – точка фиксации взгляда (точка, в которую смотрит наблюдатель). Мы нанизали плоскость на луч проецирования, теперь ее надо зафиксировать, для чего

указывают фокусное расстояние. F – это расстояние от наблюдателя до плоскости (по лучу)(рис 3.4.1).

Можно задать вектора $N_1$ (определяет ось x), $N_2$ (определяет ось y) и $N_3$ (определяет ось z). Оси **x** и **y** – оси **видовой** системы координат (связанной с плоскостью).

Для ориентации плоскости в пространстве задаются : главное положение – это положение, при котором отсутствует крен, т.е. ось x параллельна плоскости XOY мировой системы координат или угол поворота ее относительно главной оси.

Необходимо найти зависимость:

$$\begin{cases} x = x(X, Y, Z, параметры) \\ y = y(X, Y, Z, параметры) \end{cases}$$

Параметры проецирования: (ϰ) P($x_P$, $y_P$, $z_P$), F, ориентация плоскости (L, $N_1$, $N_2$).

Введем две вспомогательных системы координат: с одним штрихом

X'Y'Z': PX'||ox
PY'||oy
PZ' – продолжение LP

и двумя штрихами:

X"Y"Z": PX"||OX
PY"||OY
PZ"||OZ

В системе PX'Y'Z' рассмотрим координату пространственной точки T(X',Y',Z'), имеющей также проекцию в видовой системе координат (.)t(x,y, -F).

Составим уравнение прямой PT:

$$\frac{X'}{x} = \frac{Y'}{y} = \frac{Z'}{-F};$$

$$\begin{cases} x = -F \cdot \dfrac{X'}{Z'} \\ y = -F \cdot \dfrac{Y'}{Z'} \end{cases} \quad \textbf{(1)}$$

Запишем данные уравнения в матричном виде:

$$(2)\ \left[ -\frac{x}{F} \quad -\frac{y}{F} \quad A \quad 1 \right] = [X' \ Y' \ Z' \ 1] \cdot M; \quad \textbf{(2)}$$

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & a & 0 \end{bmatrix};$$

Убедимся, что **(2)** абсолютно идентично **(1)**.

$$[X' \ Y' \ Z' \ 1] \cdot M = [X' \ Y' \ a \ Z'];$$

Перейдем к декартовой системе координат.

$$\left[ \frac{X'}{Z'} \frac{Y'}{Z'} \frac{a}{Z'} 1 \right];$$

Остается сравнить полученное выражение с **(2)**. Имеем

$$-\frac{x}{F}=\frac{X^{|}}{Z^{|}};\quad -\frac{y}{F}=\frac{Y^{|}}{Z^{|}};\quad A=\frac{a}{Z};\ \alpha\neq 0.$$

$-S^{-1} = 1/Z'$, следовательно $S= -Z'$. Величина S носит название глубины пространственной точки. Оценив знак S, можно сказать, находится точка перед наблюдателем (знак +) или сзади него (знак–).

Тогда матричное выражение для видовой системы координат согласно

$\left[-\frac{x}{F},-\frac{y}{F},-\frac{1}{S},1\right]$ будет $x=-F\frac{X'}{Z'}, y=-F\frac{Y'}{Z'}, S=-Z$

Запишем изложенное ранее с учетом глубины:

$$\left[-\frac{x}{F}\quad -\frac{y}{F}\quad -\frac{1}{S}\quad 1\right]\cdot M^{-1}=[-\frac{x}{F}\quad -\frac{y}{F}\quad -\frac{1}{S}\quad 1]\cdot M^{T};$$

Для перехода в мировую систему координат:

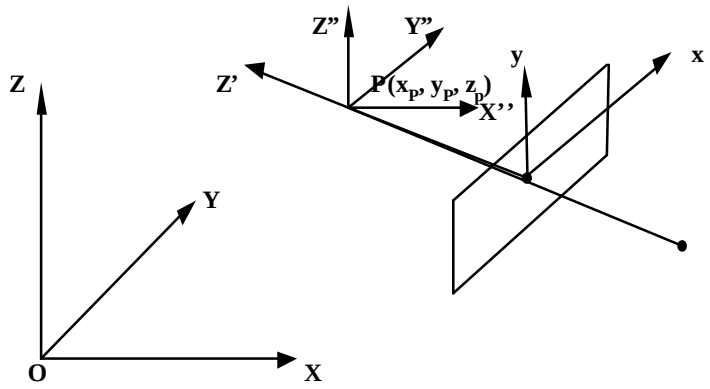



Рис 3.4.2

[X’ Y’ Z’ 1]=[X” Y” Z” 1]·R, где R – матрица поворота.

$$R=\begin{pmatrix}\cos(X''PX') & \cos(X''PY') & \cos(X''PZ') & 0\\ \cos(Y''PX') & \cos(Y''PY') & \cos(Y''PZ') & 0\\ \cos(Z''PX') & \cos(Z''PY') & \cos(Z''PZ') & 0\\ 0 & 0 & 0 & 1\end{pmatrix}=\begin{pmatrix}a_1 & a_2 & a_3 & 0\\ b_1 & b_2 & b_3 & 0\\ c_1 & c_2 & c_3 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}$$

[X’ Y’ Z’ 1]=[X” Y” Z” 1]·RM; ориентация наблюдателя

$RM=Q=\begin{bmatrix}a_1 & a_2 & 0 & a_3\\ b_1 & b_2 & 0 & b_3\\ c_1 & c_2 & 0 & c_3\\ 0 & 0 & 1 & 0\end{bmatrix}$, Q – матрица проецирования в мировой системе координат.

Матрица Q обладает свойствами ортогональности: $Q^{-1}=Q^{T}$.

Решение задачи в общем виде:

$$\left[-\frac{x}{F} \quad -\frac{y}{F} \quad -S^{-1} \quad 1\right] = [X - Xp \;\; Y - Yp \;\; Z - Zp \;\; 1] \cdot Q = [X\ Y\ Z\ 1] \bullet T_p\left(-X_p - Y_p - Z_p\right);$$

$$\left[-\frac{x}{F} \; -\frac{y}{F} \; -S^{-1} \; 1\right] = [X \;\; Y \;\; Z \;\; 1] \bullet T_p R \bullet M,$$

$T_p R$ - некоторая матрица описывающая наблюдателя иногда ее называют матрицей камеры $M_{ка}$.

Преобразуем первую скобку таким образом, чтобы избавиться от F и знака «-».

$$\left[-\frac{x}{F} \quad -\frac{y}{F} \quad -S^{-1} \quad 1\right] = [x \quad y \quad FS^{-1} \quad -F] = [x \quad y \quad S^{-1} \quad 1] = [x \quad y \quad S^{-1} \quad 1] \bullet \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & F & 0 \\ 0 & 0 & 0 & -F \end{bmatrix}}_{M_F}$$

т.е. $[x \;\; y \;\; S^{-1} \;\; 1] = [X \;\; Y \;\; Z \;\; 1] \bullet T_p R \bullet M \bullet M^{-1}{}_F$;

$$[X\ Y\ Z\ 1] = [X\ Y\ Z\ 1] \bullet M_{об};$$

$$[x\ y\ S^{-1}\ 1] = [X\ Y\ Z\ 1] \bullet \underbrace{M_{об} \bullet M_{пр}}_{\text{матрицы аффинного преобразования}} \bullet \underbrace{M_{пр}}_{\text{матрица проецирования}}$$

$$M_{пр} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{F} & 0 \\ 0 & 0 & 0 & -\frac{1}{F} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -F^{-1} \\ 0 & 0 & F^{-1} & 0 \end{bmatrix}$$

Для проецирования необходимо задать следующие параметры:

1. Месторасположение наблюдателя - его координаты: P(Xp, Yp, Zp).
2. Координаты (x)L – той точки, в которую наблюдатель смотрит: L($X_L$, $Y_L$, $Z_L$).
3. F – фокусное расстояние.
4. Коэффициенты $a_1$, $a_2$, $a_3$ задают наблюдателя.

Алгебраическая форма записи:

$$\begin{cases} x = -F \bullet \dfrac{a_1 \bullet (X\text{-}Xp) + b_1 \bullet (Y\text{-}Yp) + c_1 \bullet (Z\text{-}Zp)}{a_3 \bullet (X\text{-}Xp) + b_3 \bullet (Y\text{-}Yp) + c_3 \bullet (Z\text{-}Zp)} \\ y = -F \bullet \dfrac{a_2 \bullet (X\text{-}Xp) + b_2 \bullet (Y\text{-}Yp) + c_2 \bullet (Z\text{-}Zp)}{a_3 \bullet (X\text{-}Xp) + b_3 \bullet (Y\text{-}Yp) + c_3 \bullet (Z\text{-}Zp)} \\ s = -\left(a_3 \bullet (X\text{-}Xp) + b_3 \bullet (Y\text{-}Yp) + c_3 \bullet (Z\text{-}Zp)\right) \end{cases}$$

Векторная форма записи:

$\overline{N}_1 = \overline{i} \bullet a_1 + \overline{j} \bullet b_1 + \overline{k} \bullet c_1$

$\overline{N}_2 = \overline{i} \bullet a_2 + \overline{j} \bullet b_2 + \overline{k} \bullet c_2$
$\overline{N}_3 = \overline{i} \bullet a_3 + \overline{j} \bullet b_3 + \overline{k} \bullet c_3$

$[X–Xp, Y–Yp, Z–Zp] = \overline{PT}$;

Тогда получим следующую систему уравнений:

$$x = -F\frac{\overline{N_1}\,\overline{P_T}}{\overline{N_3}\,\overline{P_T}}$$

$$y = -F\frac{\overline{N_2}\,\overline{P_T}}{\overline{N_3}\,\overline{P_T}}$$

$$S = -\overline{N_3}\,\overline{P_T}$$

Величины X, Y, Z, Xp, Yp, Zp, x, y, S выражаются в метрах, но a, b, c – безразмерные. Измерив x, y в метрах величину F получим так же в метрах. Теперь легко провести аналогию с процессом фотографирования.

***Параллельное проецирование***

При параллельном проецировании центр (несобственного) пучка считается лежащим в бесконечности. Параллельное проецирование является частным случаем центрального.

Формулы для проецирования могут быть получены из общего случая.

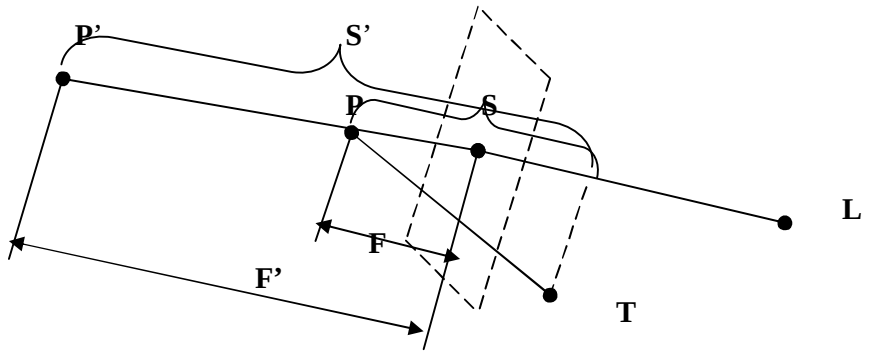



Рис 3.4.3

Согласно рис. 3.4.3 T – точка, которую мы проецируем, а L – точка фиксации взгляда, P- точка, в которой находится наблюдатель.

Параллельную проекцию получаем сместив центр проекции в (.)P’.

$$\begin{cases} x = \dfrac{F'}{S'}\overrightarrow{N_1}\,\overrightarrow{P'T} \\ y = \dfrac{F'}{S'}\overrightarrow{N_2}\,\overrightarrow{P'T} \\ S' = -\overrightarrow{N_3}\,\overrightarrow{P'T} \end{cases}$$

$$\overrightarrow{P'T} = \overrightarrow{P'P} + \overrightarrow{PT}$$

Запшем те же соотношения, но теперь для точки P:

$$\begin{cases} x = \dfrac{F'}{S'}\overrightarrow{N_1}\,\overrightarrow{P'T} = \overrightarrow{N_1}\,\overrightarrow{PT} \\ y = \dfrac{F'}{S'}\overrightarrow{N_2}\,\overrightarrow{P'T} = \overrightarrow{N_2}\,\overrightarrow{PT} \\ S' = -\overrightarrow{N_3}\,\overrightarrow{P'P} - \underset{S}{\overrightarrow{N_3}\,\overrightarrow{PT}} \end{cases}$$

Итак, устремив F к ∞, а S’ - к F’ записываем векторную форму:

$$\begin{cases} x = \overline{N_1} \times \overline{PT} \\ y = \overline{N_2} \times \overline{PT} \\ s = -\overline{N_3} \times \overline{PT} \end{cases}$$

В матричном виде векторное выражение эквивалентно следующему:

$$[x\ y\ S\ 1] = [X - Xp\ \ Y - Yp\ \ Z - Zp\ \ 1] \cdot P, \text{ где}$$

$$P = \begin{bmatrix} a_1 & a_2 & a_3 & 0 \\ b_1 & b_2 & b_3 & 0 \\ c_1 & c_2 & c_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P = R \bullet M = R, \text{т.к. } M = E \equiv \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

матрица параллельного проецирования в видовой системе координат

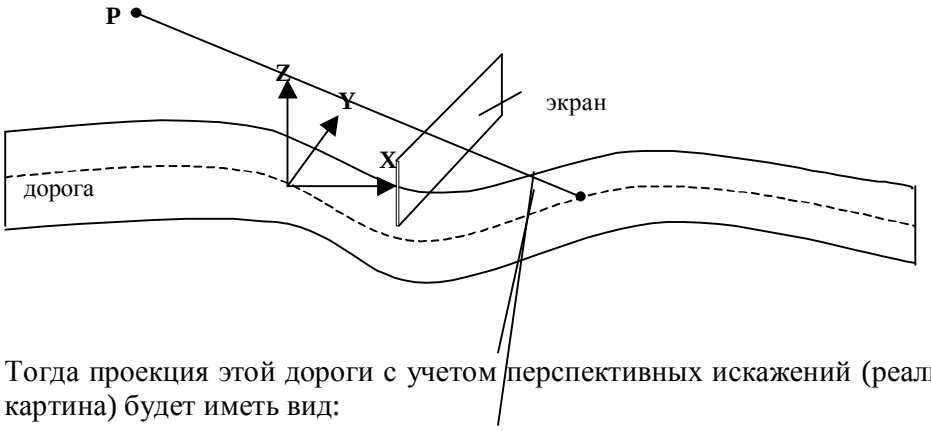
Перейдем к алгебраической форме:

$$\begin{cases} x = (X - Xp) \cdot a_1 + (Y - Yp) \cdot b_1 + (Z - Zp) \cdot c_1; \\ y = (X - Xp) \cdot a_2 + (Y - Yp) \cdot b_2 + (Z - Zp) \cdot c_2; \\ S = -[(X - Xp) \cdot a_3 + (Y - Yp) \cdot b_3 + (Z - Zp) \cdot c_3]; \end{cases}$$

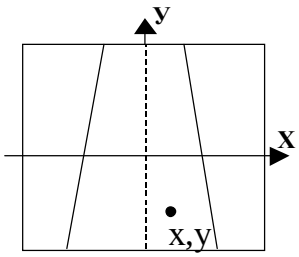
Параллельное проецирование широко используется в черчении для параллельных и косоугольных проекций, для высокоточных измерений по чертежу.

***<u>Пример задачи, связанный с проецированием</u>***

Есть наблюдатель, который смотрит на дорогу:

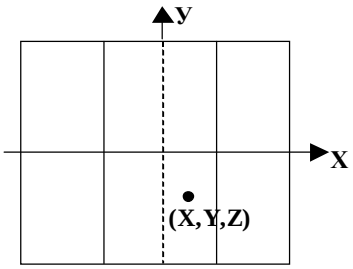



Тогда проекция этой дороги с учетом перспективных искажений (реалистичная картина) будет иметь вид:




Надо получить по этой картинке план дороги, т.е. изображение без искажений (|| проецирование)

Обратный переход осуществляется при Z=0




Будем считать, что Z – уровень дороги – известен. Тогда найдем координаты (X, У), через выражение:

$$\left[-\frac{x}{F} \quad -\frac{y}{F} \quad -S^{-1} \quad 1\right] = [X - Xp \;\; Y - Yp \;\; Z - Zp \;\; 1] \cdot Q;$$

Отсюда:

$$[X - Xp \;\; Y - Yp \;\; Z - Zp \;\; 1] = \left[ -\frac{x}{F} \quad -\frac{y}{F} \quad -S^{-1} \quad 1 \right] \cdot Q^{-1};$$

где

$$Q^{-1} = Q^{T} = \begin{bmatrix} a_1 & b_1 & c_1 & 0 \\ a_2 & b_2 & c_2 & 0 \\ 0 & 0 & 0 & 1 \\ a_3 & b_3 & c_3 & 0 \end{bmatrix};$$

Перемножим матрицы:

$$\left[ \left( -\frac{x}{F} \cdot a_1 - \frac{y}{F} \cdot a_2 + a_3 \right), \left( -\frac{x}{F} \cdot b_1 - \frac{y}{F} \cdot b_2 + b_3 \right), \left( -\frac{x}{F} \cdot c_1 - \frac{y}{F} \cdot c_2 + c_3 \right), -S^{-1} \right] =$$
$$= [X - Xp, Y - Yp, Z - Zp, 1];$$

Разделим на $S^{-1}$, тогда

$$X - Xp = -\left( -\frac{x}{F} \cdot a_1 - \frac{y}{F} \cdot a_2 + a_3 \right) \cdot S;$$
$$Y - Yp = -\left( -\frac{x}{F} \cdot b_1 - \frac{y}{F} \cdot b_2 + b_3 \right) \cdot S;$$
$$Z - Zp = -\left( -\frac{x}{F} \cdot c_1 - \frac{y}{F} \cdot c_2 + c_3 \right) \cdot S;$$

т.к. Z известно, найдем S и подставим в 2 верхних уравнения.

$$X = Xp + (Z - Zp) \cdot \frac{x \cdot a_1 + y \cdot a_2 - F \cdot a_3}{x \cdot c_1 + y \cdot c_2 - F \cdot c_3};$$
$$Y = Yp + (Z - Zp) \cdot \frac{x \cdot b_1 + y \cdot b_2 - F \cdot b_3}{x \cdot c_1 + y \cdot c_2 - F \cdot c_3};$$

Теперь можно пересчитать в плане пространственные координаты.

***Частные случаи расчета элементов матрицы проецирования***

1. Полет без крена при условии фиксации взгляда в одной точке. Расчет осуществляется по координатам этой точки.

$$\begin{bmatrix} a & a & a \\ b & b & b \\ c & c & c \end{bmatrix} = \begin{bmatrix} -\frac{Y_0}{L_{xy}} & -\frac{X_u Z_u}{L L_{xy}} & \frac{X_l}{L} \\ \frac{X_u}{L_{xy}} & -\frac{Y_u Z_u}{L L_{xy}} & \frac{Y_u}{L} \\ 0 & \frac{L_{xy}}{L} & \frac{Z_u}{L} \end{bmatrix}$$

$$X_u = X_p - X_L$$
$$Y_u = Y_p - Y_L \qquad L_{xy} \equiv \sqrt{X_u^{\,2} + +Y_u^{\,2}}\;;\;\; L \equiv \sqrt{X_u^{\,2} + Y_u^{\,2} + Z^2{}_u}$$
$$Z_u = Z_p - Z_L$$

Отдельного рассмотрения заслуживает ситуация, когда полет осуществляется над точкой фиксации взгляда.

2. Полет с креном.

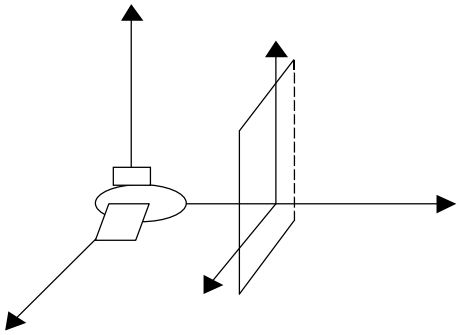


Рис 3.4.4

$$\begin{bmatrix} a & a & a \\ b & b & b \\ c & c & c \end{bmatrix} = \begin{bmatrix} \sin\varphi_{ку}\cos\varphi_к - \cos\varphi_{ку}\sin\varphi_т\sin\varphi_к & \sin\varphi_{ку}\sin\varphi_к + \cos\varphi_{ку}\sin\varphi_т\cos\varphi_к \\ -\cos\varphi_{ку}\cos\varphi_к - \sin\varphi_{ку}\sin\varphi_т\sin\varphi_{кр} & -\cos\varphi_{ку}\sin\varphi_к + \sin\varphi_к\sin\varphi_т\cos\varphi_к \\ -\cos\varphi_т\sin\varphi_к & \cos\varphi_т\cos\varphi_к \end{bmatrix}$$

$$\begin{matrix} -\cos\varphi_{ку}\cos\varphi_т \\ -\sin\varphi_{ку}\cos\varphi_т \\ \sin\varphi_т \end{matrix}\Bigg], \text{ где}$$

$\varphi_{ку}$

- курсовой угол;

$\varphi_т$

- угол тангажа;

$\varphi_к$

- угол крена.

$$\sin\varphi_{ку} = \frac{-b_3}{\sqrt{a_3^{\,2} + b_3^{\,2}}};\cos\varphi_{ку} = \frac{-a_3}{\sqrt{a_3^{\,2} + b_3^{\,2}}};\sin\varphi_т = c_3$$

$$\sin\varphi_к = \frac{-c_1}{\sqrt{a_3^{\,2} + b_3^{\,2}}};\cos\varphi_к = \frac{c_2}{\sqrt{a_3^{\,2} + b_3^{\,2}}};\cos\varphi_т = \sqrt{1 - c_3^{\,2}}$$

3 Движение по рельефу

Нормаль к поверхности определяет $N_2$ => $a_2b_2c_2$
$V_z$ как правило не известен

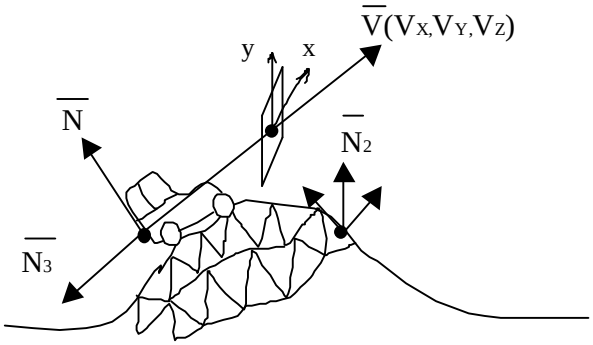
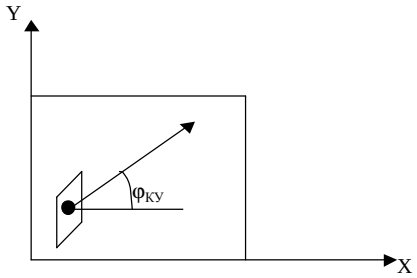



Рис 3.4.5

Задается курсовой угол $\varphi_{ку}$ $\{V_X V_Y\}$ – компоненты вектора




Нормаль к поверхности рельефа совпадает с вектором $\overline{N_2}$.
Спроецировав вектор скорости на плоскость xy получаем курсовой угол.
При этом рельеф задается с помощью плоских ($a_2$ $b_2$ $c_2$)

$$\frac{\vec{V}}{|\vec{V}|} = -\vec{N}_3$$

Воспользуемся формулой:

Рис 3.4.6

$$\vec{N}_2 * \vec{N}_3 = 0$$

Находим значение $V_2$

$$V_2 = \frac{V_X * a_2 + V_Y * b_2}{c_2}$$

$$\vec{N}_3 = \frac{\vec{V}}{|\vec{V}|} \qquad \vec{N}_1 = \vec{N}_3 * \vec{N}_2$$

Поверхность рельефа описывается регулярной сеткой высот.

$$x \to \vec{N}_1 \qquad z \to \vec{N}_3$$

Смотрим сверху

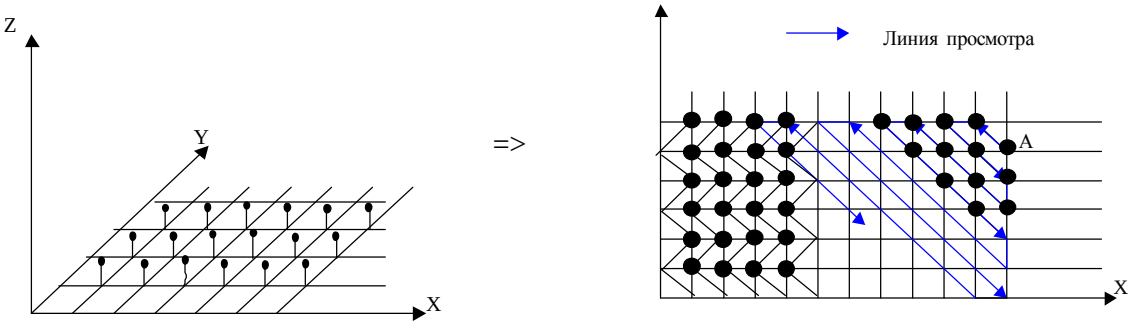
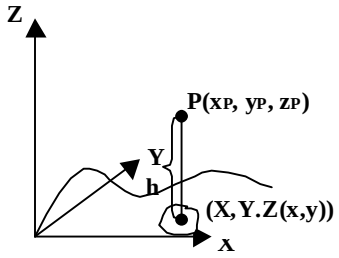


Рис 3.4.7 Рис 3.4.8

Просмотр начинаем с точки А.(рис 3.4.7)

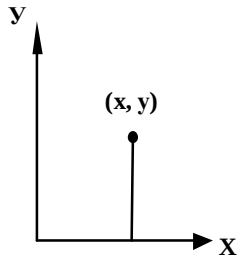
4. Движение над рельефом

Надо учитывать, что при движении над рельефом необходимо учитывать высоту наблюдателя h.




Z(x,y) – функция рельефа
$P(X_P, Y_P, Z_P)$ – положение наблюдателя

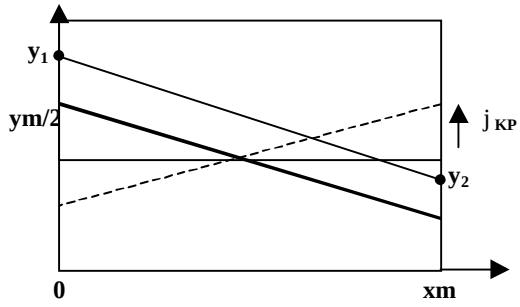
Тогда в плане получим:




Таким образом, получаем следующие координаты наблюдателя с учетом того, что он приподнят над рельефом.

$$\begin{cases} Xp = X + h \cdot a_2 \\ Yp = Y + h \cdot b_2 \\ Zp = Z(X,Y) + h \cdot c_2 \end{cases}$$

Реальность картинке придаёт линия горизонта. В случае отсутствия тангажа (наклон отрезка PL(наблюдателя)) и крена линия горизонта является горизонтальной. Расположить ее можно либо строго по середине, либо искусственно приподнять или опустить.




------ - положительный крен

—— - отрицательный крен

В случае наличия крена линия горизонта будет повернута на некоторый угол (в зависимости от угла крена). При наличии тангажа линия будет смещаться либо вниз, либо вверх. Таким образом, линия горизонта, нарисованная с учетом крена при наличии тангажа, смещается параллельно самой себе. Необходимо рассчитать точки $y_1$ и $y_2$ (их координаты).

$$\begin{cases} y_1 = \frac{y_{\max}}{2} - \frac{x_{\max}}{2} \cdot tg\, j_{RP} - F \cdot tg\, j_{TAH} \\ y_2 = \frac{y_{\max}}{2} - \frac{x_{\max}}{2} \cdot tg\, j_{RP} - F \cdot tg\, j_{TAH} \end{cases}$$

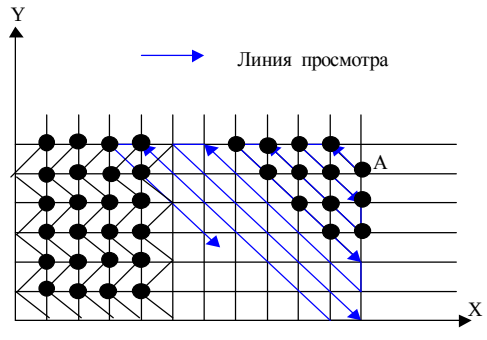
Можно также интерполировать яркость закраски «неба» и «земли», прорисовывать отдельные элементы рельефа.

## 3.5 Синтез трехмерных сцен.

Одной из важнейших задач трехмерной графики является следующая: определить, какие части объектов (ребра, грани), находящиеся в пространстве будут видны при заданном способе проецирования, а какие будут закрыты от наблюдателя другими объектами. В качестве возможных видов проецирования традиционно рассматриваются парралельное и центральное . Задача удаления невидимых линий и поверхностей является достаточно сложной и часто требует больших объемов вычислений. Поэтому существует целый ряд методов решения этой задачи.

***Алгоритм художника.***

Прорисовка начинается с заднего плана по мере приближения к наблюдателю. Суть: если для двух граней А и В самая дальняя точка грани А ближе к наблюдателю, чем самая ближняя точка грани В, то грань В никак не может закрыть грань А от наблюдателя. Реализация: множество лицевых граней сортируется по ближайшему расстоянию до картинной плоскости и потом эти грани выводятся в порядке прибли



жения к наблюдателю. (рис 3.5.1)

Рис 3.5.1

***Разбиение.***
Делим поле рельефа на сектора. В каждом секторе применяем первый алгоритм. (рис. 3.5.2)

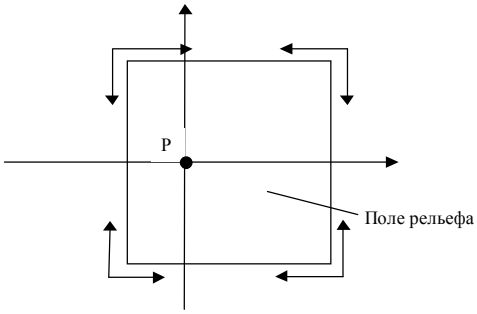
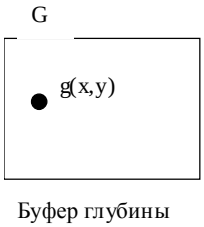



Рис 3.5.2

***Алгоритм буфера глубины.*** Синтез изображения в общем случае: алгоритм с использованием буфера глубины.



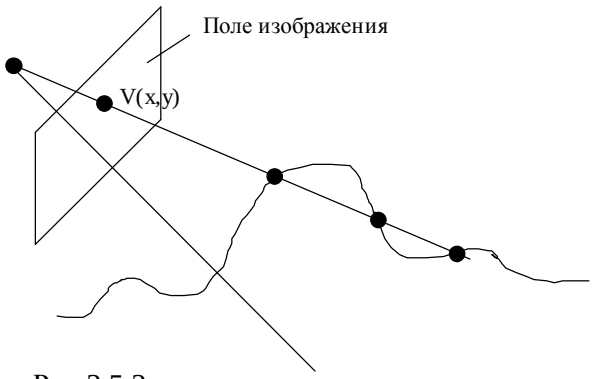
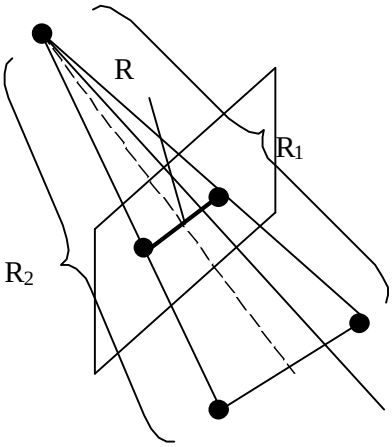




Рис 3.5.3

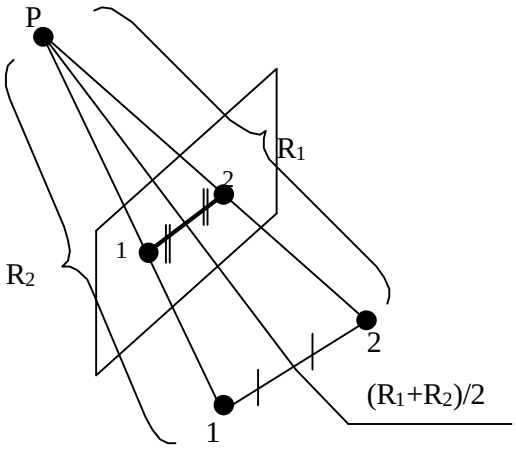
Суть метода: кроме плоскости изображения резервируем память, точно соответствующую объему изображения, а в буфер глубины записываем величину (g) равную удаленности точки объекта от наблюдателя.(рис 3.5.3). При обработки любой точки сравниваем удаленность точки с величиной, хранимой в буфере. В буфер глубины можно записывать Евклидово расстояние.




Можно восстановить пространственную координату и посчитать R для любой точки (нужно для прорисовки отрезка на поле изображения).

Рис 3.5.4

Ситуация, когда в буфер глубины записываем Евклидово расстояние представлена на рис 3.5.5.




В результате интерполяции Евклидово расстояние это середина отрезка. В буфер G записывать величину R при линейной интерполяции нельзя. Глубину (S) тоже нельзя записывать в буфер глубины, т. к. ее нельзя будет линейно интерполировать. Но в буфер глубины можно записать параметр глубины (h),т. к. h может быть линейно интерполирован в плоскости изображения.

Рис 3.5.5

$h = \frac{A}{S} - B$, где A и B – любые числа

Для величины h вводим буфер глубины H. H-буфер – это некоторая область памяти. Она имеет разрядность. Размер этой области равен размеру изображения.

$h = 0...2^r - 1$, где r- разрядность буфера глубины.

$h_{min} = 0$ при $S = S_{max}$; $S_{max}$ – задний план

$h_{max} = 2^r - 1$ при $S = S_{min}$; $S_{min}$ – передний план

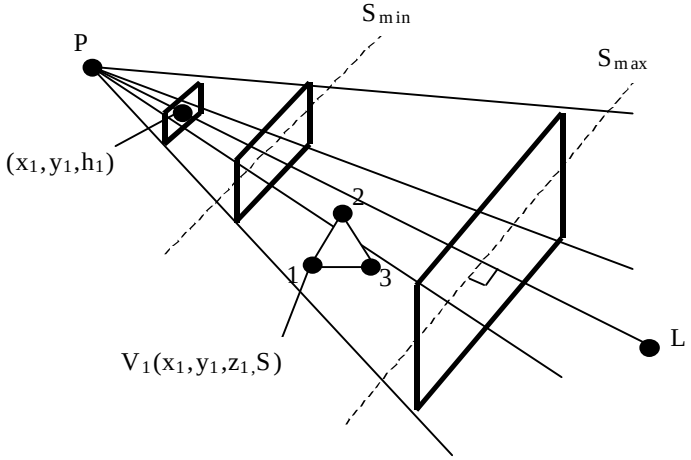



Рис 3.5.6

Значения $h_{max}$ и $h_{min}$ определяют так называемое отсечение по глубине. Все что находится за $S_{max}$ и перед $S_{min}$ отсекается. Пирамида видимости – это пространственная пирамида (все что находится в ней попадает в поле видимости). Если объект находится за пределами пирамиды видимости, то он не изображается.

h обладает рядом достоинств:

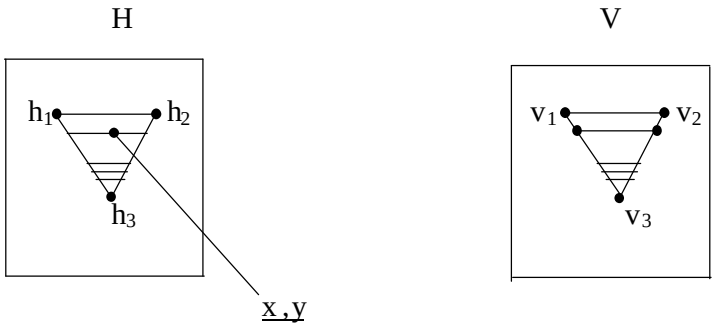
1)соответствие, т. е. $S_{min}$ соответствует $h_{max}$, $S_{max}$ соответствует $h_{min}$

удаленные элементы имеют меньшее значение h, приближенные элементы – большее значение h.
2)яркость можно связать с h. Что ближе более яркое, что дальше, то более темное.

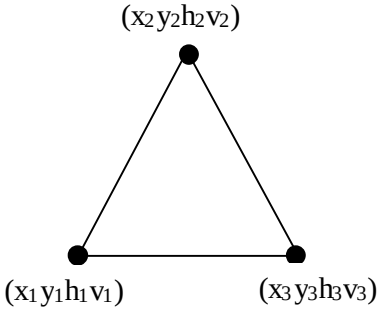
При выборе размерности буфера глубины нужно учитывать проблему плохого разрешения по глубине. Способы решения этой проблемы: увеличение разрядности буфера глубины; подбор оптимальных значений $S_{min}$, $S_{max}$, добиваясь наиболее оптимального соотношения этих величин.

**Обработка h при непосредственном синтезе изображения.**

Порядок в котором мы проецируем грани не играет роли. При этом работа идет с 2-мя полями : V и H. При синтезе изображения в поле H записываем величину, характеризующую удалённость точки от наблюдателя ($R_1$), сравнивая текущую точку с последующей обрабатываемой.

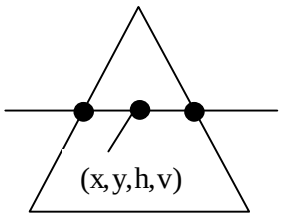



1. Инициализируем поля V, H = 0. Очищаем поле V (например, делаем его черным). В H записываем минимальное значение, т. е. заполняем его нулями. Минимальные значения соответствуют максимальной удаленности.



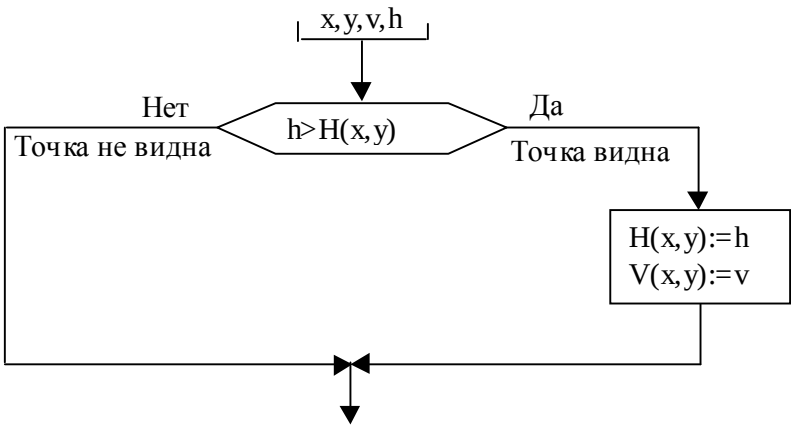

2. Для пространственного многоугольника определяем точки с координатами (X,Y,Z,S). Для плоскости изображения используются координаты (x,y,h,v), где v – это яркость в точке.

3. Используем алгоритм построчного сканирования.




h и v – линейно интерполируем

Обработка текущей точки:




а) точка вдали

$$\begin{cases} A = \dfrac{h\max \cdot S\min \cdot S\max}{S\max - S\min} \\ B = \dfrac{h\max \cdot S\min}{S\max - S\min} \end{cases}$$

$$\boxed{S = \frac{A}{h+B}} \qquad ; \begin{matrix} h_1 = 0 \\ h_2 = 1 \end{matrix}$$

$$\Delta S = S(0) - S(1) \cong \frac{A}{B} - \frac{A}{B-1}$$

$$\Delta S(0) = \frac{S\max \cdot (S\max - S\min)}{h\max \cdot S\min}$$

б)точка вблизи

$$h_2 = h_{\max} \qquad h_1 = h_{\max} - 1$$

$$\Delta S(h\max - 1) \approx \frac{A}{h_{\max} - 1 + B} - \frac{A}{h_{\max} + B}$$

$$\Delta S(h_{\max} - 1) = \frac{S_{\min} \cdot (S_{\max} - S_{\min})}{h_{\max} \cdot S_{\max}}$$

Пример:

Пусть, $S\max = 1000м$

$h\max = 255$

$\underline{h\min = 1}$

а) точка вдали

$$\Delta S(0) = \frac{1000 \cdot 1000}{255 \cdot 10} \cong 400м$$ ошибка по глубине

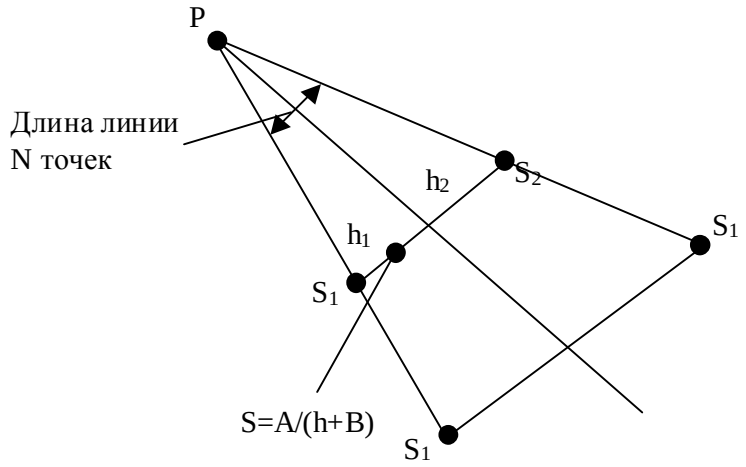
б) точка вблизи

$$\Delta S(h_{\max} - 1) = \frac{10 \cdot 1000}{255 \cdot 1000} \cong 0.04м$$

При малоразрядном буфере отношение Smin и Smax уменьшается.
Значение h – нелинейно зависимое разрешение по глубине от дальности.

Если записывать в G значение S, то нельзя линейно интерполировать.

$$h = h_1 + \frac{i}{N-1} \cdot (h_2 - h_1)$$




$$h_1 = \frac{A}{S_1} - B_1 \qquad h_2 = \frac{A}{S_2} - B_2$$

Формула нелинейной интерполяции величины S:

$$S = \frac{1}{\frac{1}{S_1} + (\frac{1}{S_2} - \frac{1}{S_1}) \cdot \frac{i}{N-1}}$$

Недостатки:
большие вычислительные затраты на каждую текущую точку;
S имеет равномерную (т. е. постоянную) разрешающую способность по глубине;
вместо линейной - нелинейная интерполяция.

$S \min = 0$

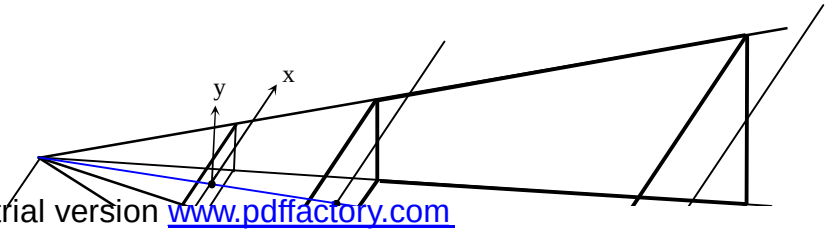
$S \max > S \min$

$$S = \frac{S_1 \cdot S_2 (N-1)}{S_1 \cdot i + S_2 (N-i-1)}$$

***Алгоритм отсечения по пирамиде видимости.***

Необходимость в этой процедуре возникает, когда, в конце концов, оказывается, что надо нарисовать грань, у которой часть вершин лежит перед камерой, а часть – за камерой. То есть грань, пересекающуюся с экраном. Сама по себе она правильно не нарисуется.

Поскольку камера видит только то, что перед ней находится, все те точки, для которых Smin > z > Smax, рисовать не надо. То есть, каждую грань надо обрезать плоскостями z = Smin и z = Smax.



В плоскость изображения попадают только те точки, которые находятся внутри ПВ.

Алгоритм:

В качестве первого шага следует провести экспресс анализ на предмет попадания объекта в пирамиду, либо полного непопадания в нее. Пусть имеется пространственный многоугольник с вершинами i = 1…n. Для каждой вершины вычисляем значения $S_i$, $X'_i$ и $Y'_i$, которые будут исходными данными

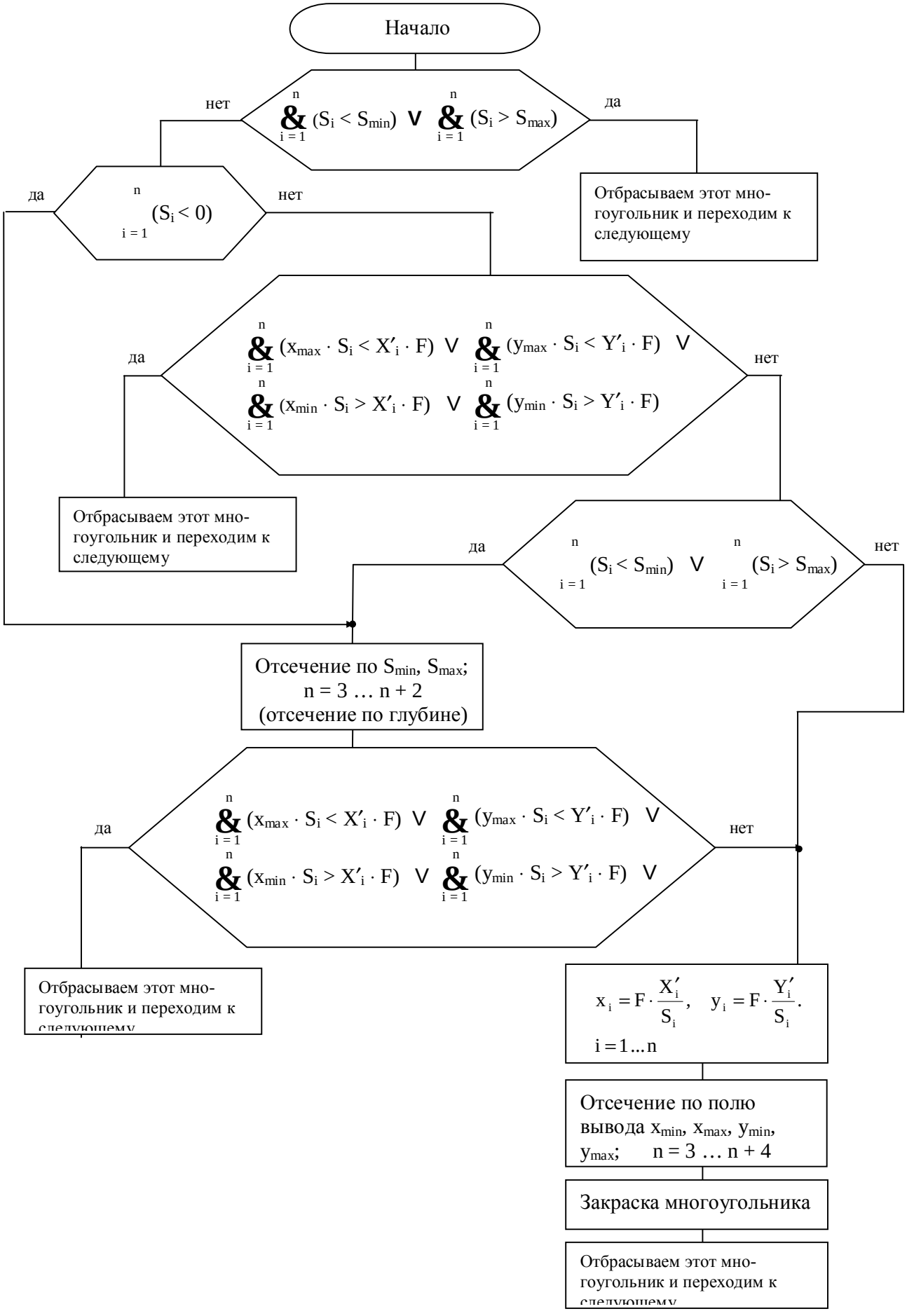
$$x_i = F \cdot \frac{X'_i}{S_i}; \quad y_i = F \cdot \frac{Y'_i}{S_i};$$

$$X'_i = a_1(x_i - x_p) + b_1(y_i - y_p) + c_1(z_i - z_p);$$

$$Y'_i = a_2(x_i - x_p) + b_2(y_i - y_p) + c_2(z_i - z_p);$$

$$S_i = -(a_3(x_i - x_p) + b_3(y_i - y_p) + c_3(z_i - z_p)).$$

Итак для случая центрального проецирования имеем схему:



В случае параллельного проецирования допускается работа по той же схеме. Для вычисления величины, которая пишется в буфер используется выражение:

$$h = -A \bullet S + B$$

Аналогично вычисляются:

$$S_{min} \leftrightarrow h_{max}$$

$$S_{max} \leftrightarrow h_{min} = 0$$

$$A = \frac{h_{max}}{S_{max} - S_{min}}$$

$$B = \frac{h_{max} \bullet S_{max}}{S_{max} - S_{min}}$$