Министерство образования и науки Российской Федерации

Ивановский государственный химико-технологический университет

Компьютерная графика. Алгоритмизация

Методические указания к лабораторным работам

Составитель Ю.И. Никифоров

Иваново
2011

Составитель Ю.И. Никифоров

УДК 613.19

Компьютерная графика. Алгоритмизация: методические указания к лабораторным работам/ сост. Ю. И. Никифоров; Иван. гос. хим.-технол. ун-т. – Иваново, 2011 - 48 с.

В методических указаниях приведены алгоритмы получения основных типов проекций, создания графических примитивов, закраски областей, отсечения линий, удаления невидимых линий.

Предназначены для студентов направления «Информационные технологии» при выполнении лабораторных работ по дисциплине «Компьютерная графика»

Рис. 30. Библиогр.: 3 назв.

Рецензент
доктор технических наук В.Ю. Волынский
(Ивановский государственный химико-технологический университет)

1. ПРОЕКЦИИ

1.1. Введение

Методические указания составлены на основе работ [1-3]. В разделе 1 рассматриваются параллельные, перспективные и стереопроекции; плоские преобразования растровых картин.

При визуализации двумерных изображений достаточно задать окно видимости в системе координат пользователя и порт отображения на экране дисплея, в котором будет выдаваться изображение из окна. В этом случае достаточно провести отсечение изображения по окну и выполнить двумерные преобразования окно-порт. На рис. 1.1 показан простой пример двумерных преобразований. Окном отсекается часть изображения домика и один улей (см. рис. 1.1 слева). Отсеченное изображение передается в порт отображения дисплея с выполнением преобразований окно-порт (см. рис. 1.1 справа). В данном (простом) случае выполняется только преобразование сдвига.

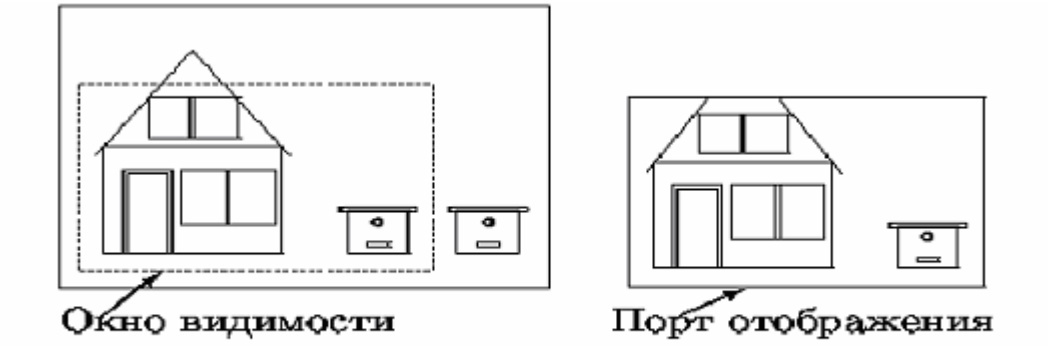


Рис. 1.1. Пример визуализации для двумерных изображений

В случае же трехмерных изображений отсечение выполняется уже не по окну, а по объему видимости и затем выполняется проецирование в порт отображения, который, в свою очередь, может быть проекцией объема видимости. Модель процесса визуализации трехмерных изображений приведена на рис. 1.2.

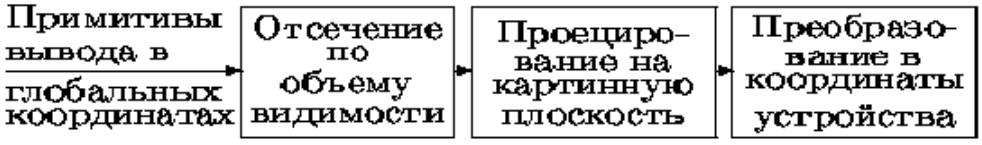


Рис. 1.2. Модель процесса визуализации трехмерных изображений

Проецирование в общем случае - отображение точек, заданных в системе координат размерностью N, в точки системы с меньшей размерностью. При отображении трехмерных изображений на дисплей три измерения отображаются в два.

Проецирование выполняется с помощью прямолинейных проекторов (проецирующих лучей), идущих из центра проекции через каждую точку объекта до пересечения с картинной поверхностью (поверхностью проекции). Далее рассматриваются только плоские проекции, при которых поверхность проекции - плоскость в трехмерном пространстве.

По расположению центра проекции относительно плоскости проекции различаются центральная и параллельные проекции.

При **параллельной проекции** центр проекции находится на бесконечном расстоянии от плоскости проекции. Проекторы представляют собой пучок параллельных лучей. В этом случае необходимо задавать направление проецирования и расположение плоскости проекции. По взаимному расположению проекторов, плоскости проекции и главных осей координат различаются *ортогональные, прямоугольные аксонометрические* и *косоугольные аксонометрические* проекции.

При *ортогональной проекции* проекторы перпендикулярны плоскости проекции, а плоскость проекции перпендикулярна главной оси, то есть проекторы параллельны главной оси.

При аксонометрической проекции имеется одна из двух перпендикулярностей:

- *при прямоугольной аксонометрической проекции* проекторы перпендикулярны плоскости проекции, которая расположена под углом к главной оси;
- *при косоугольной аксонометрической проекции* проекторы не перпендикулярны плоскости проекции, но плоскость проекции перпендикулярна главной оси.

Изображение, полученное при параллельном проецировании, не доста-

точно реалистично, но передаются точные форма и размеры, хотя и возможно различное укорачивание для различных осей.

При **центральной проекции** расстояние от центра проекции до плоскости проецирования конечно, поэтому проекторы представляют собой пучок лучей, исходящих из центра проекции. В этом случае надо задавать расположение и центра проекции и плоскости проекции. Изображения на плоскости проекции имеют так называемые перспективные искажения, когда размер видимого изображения зависит от взаимного расположения центра проекции, объекта и плоскости проекции. Из-за перспективных искажений изображения, полученные центральной проекцией, более реалистичны, но нельзя точно передать форму и размеры.

Рис. 1.3. Классификация плоских проекций

Различаются одно, двух и трехточечные центральные проекции в зависимости от того, по скольким осям выполняется перспективное искажение.

На рис. 1.3 приведена классификация описанных выше плоских проекций.

1.2. Параллельные проекции

Вначале мы рассмотрим ортогональные проекции, используемые в техническом черчении, в регламентированной для него правосторонней системе координат, когда ось Z изображается вертикальной. Затем будут проиллюстрированы аксонометрические проекции также в правосторонней системе координат, но уже более близкой к машинной графике (ось Y вертикальна, ось X на-

правлена горизонтально вправо, а ось Z - от экрана к наблюдателю). Наконец выведем матрицы преобразования в левосторонней системе координат, часто используемой в машинной графике, с вертикальной осью Y, осью X, направленной вправо, и осью Z, направленной от наблюдателя.

Использование проекций в техническом черчении регламентируется стандартом ГОСТ 2.317-69. Наиболее широко, особенно в САПР, используются ортогональные проекции (виды). Вид - ортогональная проекция обращенной к наблюдателю видимой части поверхности предмета, расположенного между наблюдателем и плоскостью чертежа.

В техническом черчении за основные плоскости проекций принимают шесть граней куба (рис. 1.4).

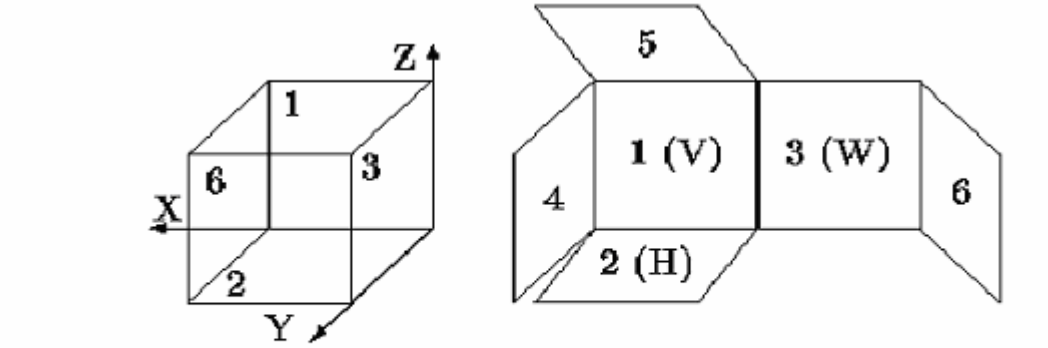


Рис. 1.4. Ортогональные проекции (основные виды) и их расположение на листе чертежа

На рисунке обозначены:

1) вид спереди, главный вид, фронтальная проекция, (на заднюю грань V);

2) вид сверху, план, горизонтальная проекция, (на нижнюю грань H);

3) вид слева, профильная проекция, (на правую грань W);

4) вид справа (на левую грань);

5) вид снизу (на верхнюю грань);

6) вид сзади (на переднюю грань).

Очевидно, что при ортогональной проекции не происходит изменения ни углов, ни масштабов.

При аксонометрическом проецировании (см. рис. 1.4) сохраняется параллельность прямых, а углы изменяются; измерение же расстояний вдоль каждой из координатных осей в общем случае должно выполняться со своим масштабным коэффициентом.

При изометрических проекциях укорачивания вдоль всех координатных осей одинаковы, поэтому можно производить измерения вдоль направлений осей с одним и тем же масштабом (отсюда и название изометрия). На рис. 1.4 приведена (аксонометрическая прямоугольная) изометрическая проекция куба со стороной А. При этой проекции плоскость проецирования наклонена ко всем главным координатным осям под одинаковым углом. Стандартом регламентируется коэффициент сжатия, равный 0,82, а также расположение и взаимные углы главных координатных осей, равные 120^0, как это показано в левом верхнем углу рис. 1.4. Обычно сжатие не делается.

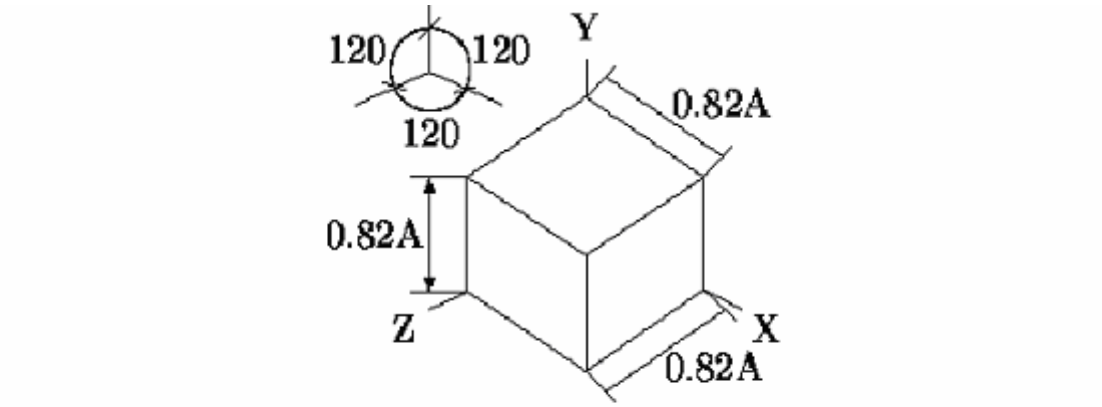


Рис. 1.4. Аксонометрическая прямоугольная изометрическая проекция куба со стороной А

При диметрической проекции две из трех осей сокращены одинаково, т.е. из трех углов между нормалью к плоскости проекции и главными координатными осями два угла одинаковы. На рис. 1.5 приведена (аксонометрическая прямоугольная) диметрическая проекция куба со стороной А. Там же показаны регламентируемое расположение осей и коэффициенты сжатия. Обычно вместо коэффициента сжатия 0,94 используется 1, а вместо 0,47 – 0,5.

В косоугольных проекциях плоскость проекции перпендикулярна главной координатной оси, а проекторы расположены под углом к ней. Таким об-

разом, аксонометрические косоугольные проекции сочетают в себе свойства ортогональных и аксонометрических прямоугольных проекций.

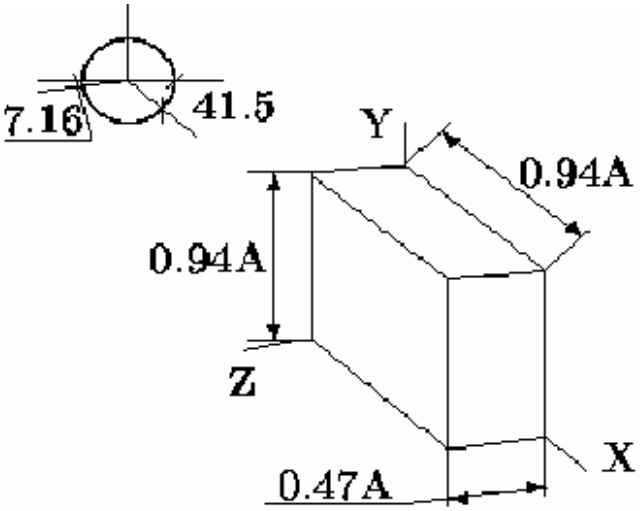


Рис. 1.5. Аксонометрическая прямоугольная диметрическая проекция куба со стороной А

Наиболее употребимы два вида косоугольной проекции - фронтальная (косоугольная) диметрия (проекция Kabinett - кабине) и горизонтальная (косоугольная) изометрия (проекция Kavalier - кавалье) или военная перспектива.

В случае фронтальной (косоугольной) диметрии при использовании правосторонней системы координат экрана плоскость проецирования перпендикулярна оси Z. Ось X направлена горизонтально вправо. Ось Z изображается по углом в 45^0 относительно горизонтального направления. Допускается угол наклона в 30 и 60^0. При этом отрезки, перпендикулярные плоскости проекции, при проецирования сокращаются до 1/2 их истинной длины. На рис.1.6 приве-

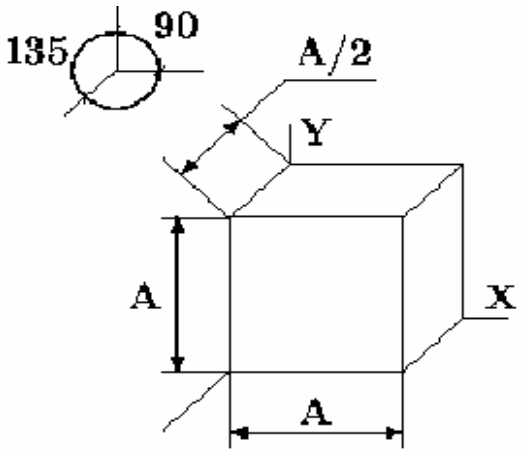


Рис. 1.6. Аксонометрическая косоугольная фронтальная диметрическая проекция куба со стороной А

дена (аксонометрическая косоугольная) фронтальная диметрическая проекция куба со стороной A, там же показаны регламентируемые коэффициент сжатия, равный 0,5, и расположение осей.

В случае же (аксонометрической косоугольной) горизонтальной изометрии, как следует из названия, плоскость проецирования перпендикулярна оси Y, а укорачивания по всем осям одинаковы и равны 1. Угол поворота изображения оси X относительно горизонтального направления составляет 30^0. Допускается 45 и 60^0 при сохранении угла 90^0 между изображениями осей X и Z. Иллюстрация этого приведена на рис. 1.7.

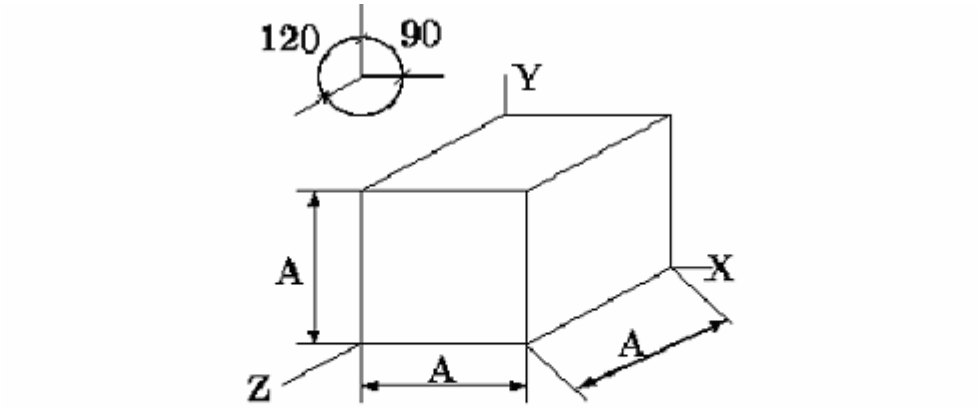


Рис. 1.7. Аксонометрическая косоугольная горизонтальная изометрическая проекция куба со стороной A

Выведем выражения для матриц преобразования, используя теперь левостороннюю систему координат более естественную для машинной графики.

Простейшее параллельное проецирование - ортогональное выполняется на плоскость, перпендикулярную какой-либо оси, т.е. при направлении проецирования вдоль этой оси. В частности, проецирование в XY-плоскость, заданную соотношением $Z = Z_0$, выполняется следующим образом:

$$[x \quad y \quad z \quad 1]\cdot\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & Z_0 & 1\end{bmatrix} = [xn \quad yn \quad zn \quad wn]$$

Рассмотрим теперь косоугольное проецирование, при котором плоскость проецирования перпендикулярна главной оси, а проекторы составляют с плоскостью проецирования угол не равный 90^0. Матрица для этого преобразования

может быть найдена исходя из значений угла проецирования и координат преобразованной точки. На рис. 1.8 показана косоугольная параллельная проекция единичного куба.

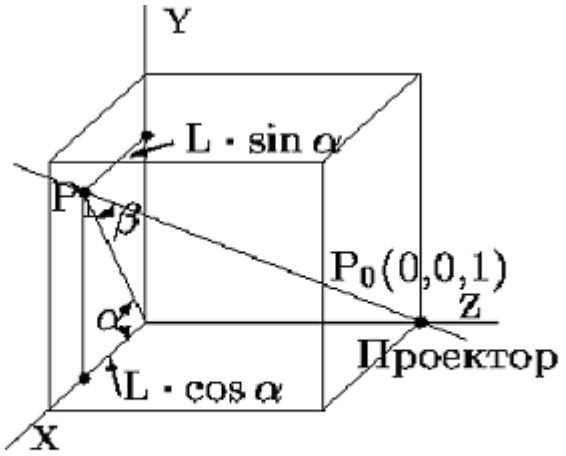


Рис. 1.8. Косоугольная параллельная проекция $P_1(L\cdot\cos\alpha, L\cdot\sin\alpha, 0)$ точки $P_0(0,0,1)$

Из рисунка видно, что проектором, идущим из точки P_0 в P_1, точка $P_0(0,0,1)$ проецируется в $P_1(L\cdot\cos\alpha, L\cdot\sin\alpha, 0)$.

Теперь проектором, параллельным рассмотренному (рис.1.8), спроецируем некоторую точку (X,Y,Z) в точку (X_p,Y_p,Z_p).

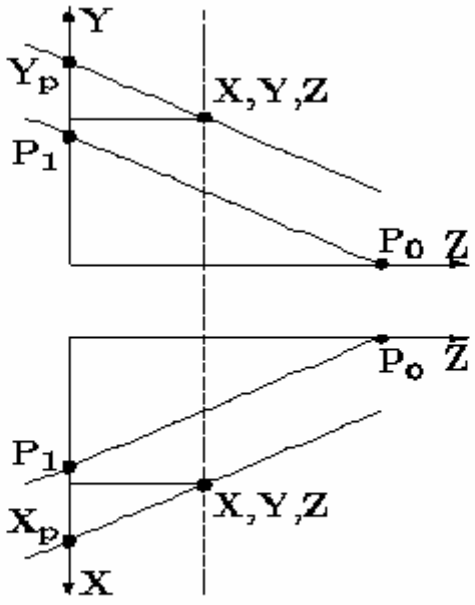


Рис. 1.9. Косоугольная параллельная проекция $(X_p,Y_p,0)$ точки (X,Y,Z)

Из подобия треугольников получаем:

$$(X_p - X)/Z = L\cdot\cos\alpha \quad \Rightarrow \quad X_p = X + Z\cdot L\cdot\cos\alpha$$

$$(Y_p - Y)/Z = L\cdot\sin\alpha \quad \Rightarrow \quad Y_p = Y + Z\cdot L\cdot\sin\alpha$$

Это соответствует следующему матричному выражению:

$$\begin{bmatrix} x & y & z & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ L \cdot \cos\alpha & L \cdot \sin\alpha & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x_p & y_p & z_p & 1 \end{bmatrix}$$

Таким образом, матрица аксонометрической косоугольной проекции для случая проецирования в плоскость Z = 0, выполняет следующее:

- вначале плоскости с заданной координатой Z_0 переносятся вдоль оси X на $Z_0 \cdot L \cdot \cos\alpha$ и вдоль оси Y на $Z_0 \cdot L \cdot \sin\alpha$;
- затем производится проецирование в плоскость Z = 0.

Различные варианты параллельных проекций формируются из полученной подстановкой значений L и углов α и β (см. рис. 1.8). В частности, для фронтальной косоугольной диметрии L = 1/2, следовательно, угол β между проекторами и плоскостью проецирования равен arctan 2 = $63{,}4^0$. Угол же α равен 45^0 и допускается 30 и 60^0, как это сказано выше. (Обратите внимание, что в этой системе координат плоскость фронтальной проекции - плоскость XY, в отличие от системы координат технического черчения, где фронтальная проекция, как это показано на рис. 1.4, формируется в плоскости XZ).

1.3. Центральная проекция

Наиболее реалистично трехмерные объекты выглядят в центральной проекции из-за перспективных искажений сцены. Центральные проекции параллельных прямых, не параллельных плоскости проекции, будут сходиться в *точке схода*. В зависимости от числа точек схода, т.е. от числа координатных осей, которые пересекает плоскость проекции, различаются одно, двух и трехточечные центральные проекции. Иллюстрация одно-, двух- и трехточечной центральных проекций куба приведена на рис. 1.10.

Наиболее широко используется двухточечная центральная проекция.

Выведем матрицу, определяющую центральное проецирование для простого случая одноточечной проекции (рис. 1.11), когда плоскость проекции перпендикулярна оси Z и расположена на расстоянии d от начала координат.

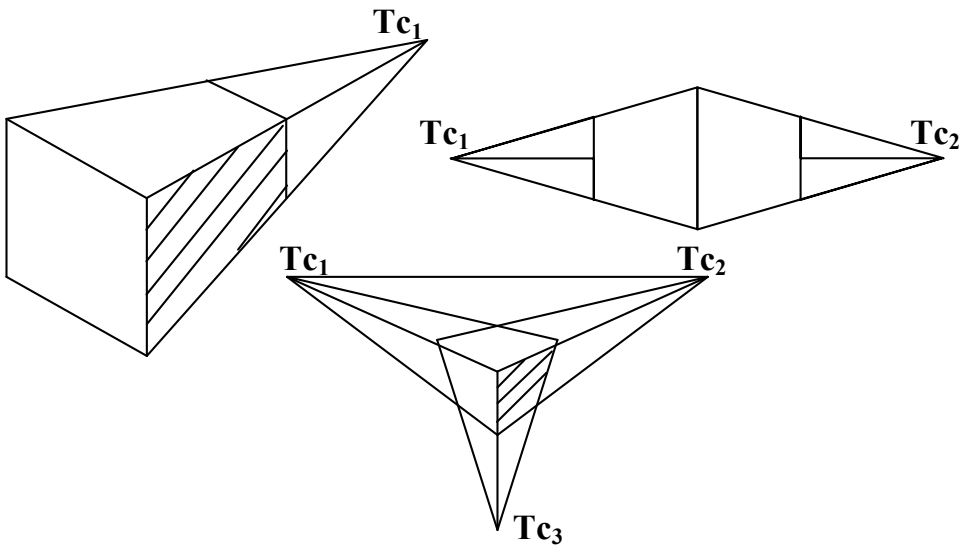


Рис. 1.10. Одно-, двух- и трехточечная центральные проекции

(Здесь используется удобная для машинной графики левосторонняя система координат).

Начало отсчета находится в точке просмотра. Ясно, что изображения объектов, находящихся между началом координат и плоскостью проекции, увеличиваются, а изображения объектов, расположенных дальше от начала координат, чем плоскость проекции, уменьшаются.

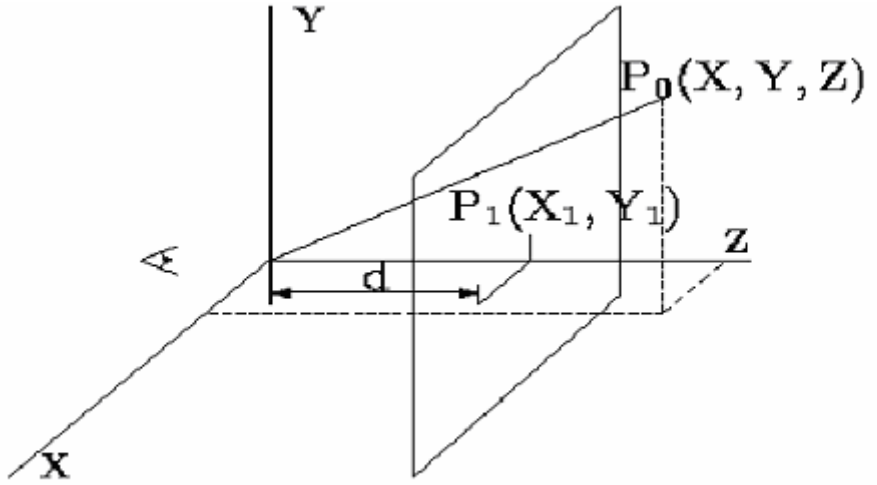


Рис. 1.11. Центральная проекция точки P_0 в плоскость $Z = d$

Из рис. 1.11 видно, что для координат (X1,Y1) точки P_1, полученной проецированием точки $P_0(X,Y,Z)$ в плоскость Z = d (плоскость экрана) выполняются следующие соотношения:

$$\frac{X_1}{d} = \frac{X}{Z}, \frac{Y_1}{d} = \frac{Y}{Z}, X_1 = \frac{X}{Z/d}, Y_1 = \frac{Y}{Z/d}.$$

Такое преобразование может быть представлено матрицей 4×4

$$[x_1 \quad y_1 \quad z_1 \quad w_1] = [x \quad y \quad z \quad 1] \cdot \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1/d \\ 0 & 0 & 0 & 0 \end{bmatrix} =$$

$$[x \quad y \quad z \quad 1] \cdot M_u = [x \quad y \quad z \quad z/d]$$

Для перехода к декартовым координатам делим все на z/d и получаем:

$$[X/(Z/d) \quad Y/(Z/d) \quad d \quad 1].$$

Если же точка просмотра расположена в плоскости проекции, тогда центр проекции расположен в точке (0,0,-d). Рассматривая подобные треугольники, аналогично вышеописанному, можем получить:

$$X_1 = \frac{X}{Z/d+1}; \quad Y_1 = \frac{Y}{Z/d+1}.$$

Матрица преобразования в этом случае имеет вид:

$$M_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Матрица M_0 может быть представлена в виде:

$$M_0 = T(0,0,d) \cdot M_u \cdot T(0,0,-d),$$

т.е. преобразование проецирования выполняется для этого случая путем переноса начала координат в центр проецирования, собственно проецирования и обратного сдвига начала координат.

2. ГЕНЕРАЦИЯ ВЕКТОРОВ

2.1. Введение

Назначение генератора векторов - соединение двух точек изображения отрезком прямой.

Далее будет рассмотрен алгоритм Брезенхема для генерации векторов.

Перед рассмотрением алгоритма сформулируем общие требования к изображению отрезка:

- концы отрезка должны находиться в заданных точках;
- отрезки должны выглядеть прямыми;
- яркость вдоль отрезка должна быть постоянной и не зависеть от длины

и наклона.

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

 концы отрезка в общем случае располагаются на пикселях, лишь наиболее близких к требуемым позициям и только в частных случаях координаты концов отрезка точно совпадают с координатами пикселей;

 отрезок аппроксимируется набором пикселей и лишь в частных случаях вертикальных, горизонтальных и отрезков под 45^0 они будут выглядеть прямыми, причем гладкими прямыми, без ступенек только для вертикальных и горизонтальных отрезков (рис. 2.1);

 яркость для различных отрезков и даже вдоль отрезка в общем случае различна, так как, например, расстояние между центрами пикселей для вертикального отрезка и отрезка под 45^0 различно (см. рис. 2.1).

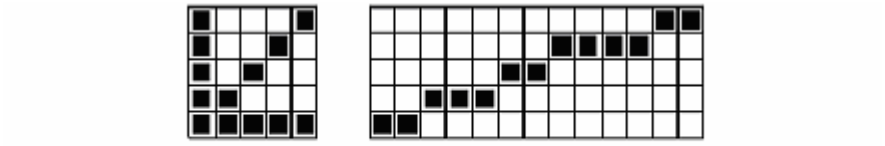


Рис. 2.1. Растровое представление различных векторов

Объективное улучшение аппроксимации достигается увеличением разрешения дисплея, но в силу существенных технологических проблем 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.

Далее в этом разделе рассмотрен один из алгоритмов генерации отрезка.

2.2. Алгоритм Брезенхема

Брезенхем предложил алгоритм, обеспечивающий минимизацию откло-

нения сгенерированного образа от истинного отрезка. Основная идея алгоритма состоит в том, что если угловой коэффициент прямой < 1/2, то естественно точку, следующую за точкой (0,0), поставить в позицию (1,0) (рис. 2.2,а), а если угловой коэффициент > 1/2, то - в позицию (1,1) (рис. 2.2,б). Для принятия решения, куда заносить очередной пиксель вводится величина отклонения E точной позиции от середины между двумя возможными растровыми точками в направлении наименьшей относительной координаты. Знак E используется как критерий для выбора ближайшей растровой точки.

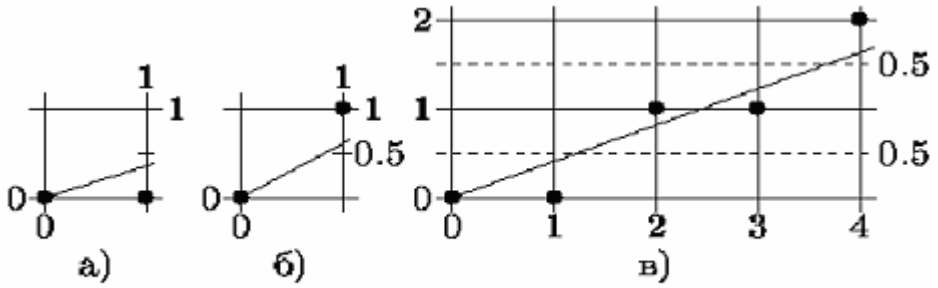


Рис. 2.2. Алгоритм Брезенхема генерации отрезков

Если E < 0, то точное Y-значение округляется до последнего меньшего целочисленного значения Y, т.е. Y-координата не меняется по сравнению с предыдущей точкой. В противном случае Y увеличивается на 1.

Для вычисления E без ограничения общности для упрощения положим, что рассматриваемый вектор начинается в точке (0,0) и проходит через точку (4;1,5) (см. рис. 2.2,в), т.е. имеет положительный наклон меньший 1. Из рис. 2.2,в видно отклонение для первого шага:

$$E_1 = P_y / P_x - 1/2 < 0,$$

где $P_y = Y_k - Y_n$ - приращение координат отрезка по оси Y, а $P_x = X_k - X_n$ - приращение координат отрезка по оси X, поэтому для занесения пикселя выбирается точка (1,0).

Отклонение для второго шага вычисляется добавлением приращения Y-координаты для следующей X-позиции (см. рис.2.2,в):

$$E_2 = E_1 + P_y / P_x > 0,$$

поэтому для занесения пикселя выбирается точка (2,1). Так как отклонение считается от Y-координаты, которая теперь увеличилась на 1, то из накоплен-

ного отклонения для вычисления последующих отклонений надо вычесть 1:

$$E_2 = E_2 - 1.$$

Отклонение для третьего шага:

$$E_3 = E_2 + P_y / P_x < 0,$$

поэтому для занесения пикселя выбирается точка (3,1).

Суммируя и обозначая большими буквами растровые точки, а маленькими - точки вектора, получаем:

$$E_1 = y_1 - 1/2 = dY / dX - 1/2.$$

Возможны случаи: $E_1 > 0$ или $E_1 \le 0$, ближайшая точка есть:

$$X_1 = X_0 + 1; \quad Y_1 = Y_0 + 1; \quad E_2 = E_1 + P_y / P_x - 1, \text{если } E_1 > 0,$$
$$X_1 = X_0 + 1; \quad Y_1 = Y_0 + 1; \quad E_2 = E_1 + P_y / P_x, \quad \text{если } E_1 \le 0.$$

Так как интересует только знак E, то можно избавиться от неудобных частных умножением E на $2 \cdot P_x$:

$$E_1 = 2 \cdot P_y - P_x$$
$$E_1 > 0 : E_2 = E_1 + 2 \cdot (P_y - P_x)$$
$$E_1 \le 0 : E_2 = E_1 + 2 \cdot P_y$$

3. ГЕНЕРАЦИЯ ОКРУЖНОСТИ

3.1. Введение

Во многих областях приложений, таких как, например, системы автоматизированного проектирования машиностроительного направления, естественными графическими примитивами, кроме отрезков прямых и строк текстов, являются и конические сечения, т.е. окружности, эллипсы, параболы и гиперболы. Наиболее употребительным примитивом, естественно, является окружность. Один из наиболее простых и эффективных алгоритмов генерации окружности разработан Брезенхемом.

3.2. Алгоритм Брезенхема

Для простоты и без ограничения общности рассмотрим генерацию 1/8 окружности, центр которой лежит в начале координат. Остальные части окружности могут быть получены последовательными отражениями (использо-

ванием симметрии точек на окружности относительно центра и осей координат).

Окружность с центром в начале координат описывается уравнением:

$$X^2 + Y^2 = R^2$$

Алгоритм Брезенхема пошагово генерирует очередные точки окружности, выбирая на каждом шаге для занесения пикселя точку растра $P_i(X_i,Y_i)$, ближайшую к истинной окружности, так чтобы ошибка

$$E_i(P_i) = (X_i^2 + Y_i^2) - R^2$$

была минимальной. Причем, как и в алгоритме Брезенхема для генерации отрезков, выбор ближайшей точки выполняется с помощью анализа значений управляющих переменных, для вычисления которых не требуется вещественной арифметики. Для выбора очередной точки достаточно проанализировать знаки.

Рассмотрим генерацию 1/8 окружности по часовой стрелке, начиная от точки X=0, Y=R.

Проанализируем возможные варианты занесения i+1-й точки, после занесения i-й.

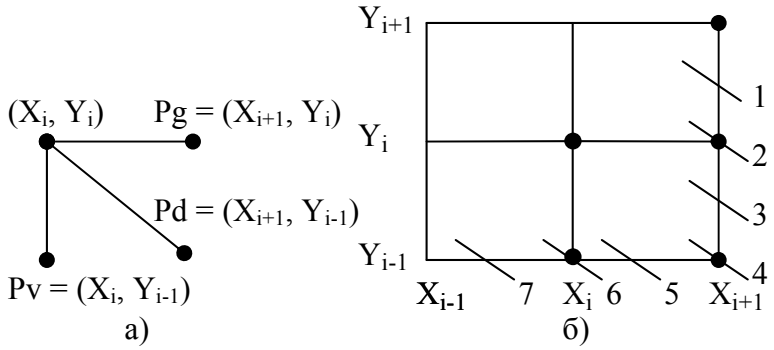


Рис. 3.1. Варианты расположения очередного пикселя окружности

При генерации окружности по часовой стрелке после занесения точки (X_i, Y_i) следующая точка может быть (см. рис.3.1,а) либо $P_g = (X_{i+1}, Y_i)$ - перемещение по горизонтали, либо $P_d = (X_{i+1}, Y_{i-1})$ - перемещение по диагонали, либо $P_v = (X_i, Y_{i-1})$ - перемещение по вертикали.

Для этих возможных точек вычислим и сравним абсолютные значения разностей квадратов расстояний от центра окружности до точки и окружности:

$$|D_g| = |(X+1)^2 + Y^2 - R^2|$$
$$|D_d| = |(X+1)^2 + (Y-1)^2 - R^2|$$
$$|D_v| = |X^2 + (Y-1)^2 - R^2|$$

Выбирается и заносится та точка, для которой это значение минимально.

Выбор способа расчета определяется по значению D_d. Если $D_d < 0$, то диагональная точка внутри окружности. Это варианты 1-3 (см. рис. 3.1,б). Если $D_d > 0$, то диагональная точка вне окружности. Это варианты 5-7. И, наконец, если $D_d = 0$, то диагональная точка лежит точно на окружности. Это вариант 4. Рассмотрим случаи различных значений D_d в только что приведенной последовательности.

Случай $D_d < 0$

Здесь в качестве следующего пикселя могут быть выбраны или горизонтальный - P_g или диагональный - P_d.

Для определения того, какой пиксель выбрать P_g или P_d, составим разность:

$$d_i = |D_g| - |D_d| = |(X+1)^2 + Y^2 - R^2| - |(X+1)^2 + (Y-1)^2 - R^2|.$$

И будем выбирать точку P_g при $d_i \le 0$, в противном случае выберем P_d.

Рассмотрим вычисление d_i для разных вариантов.

Для вариантов 2 и 3:

$D_g \ge 0$ и $D_d < 0$, так как горизонтальный пиксель либо вне, либо на окружности, а диагональный внутри.

$$d_i = (X+1)^2 + Y^2 - R^2 + (X+1)^2 + (Y-1)^2 - R^2.$$

Добавляя и вычитая $(Y-1)^2$, получим:

$$d_i = 2 \cdot [(X+1)^2 + (Y-1)^2 - R^2] + 2 \cdot Y - 1.$$

В квадратных скобках стоит D_d, так что

$$d_i = 2 \cdot (D_d + Y) - 1.$$

Для варианта 1:

Ясно, что должен быть выбран горизонтальный пиксель P_g. Проверка

компонент d_i показывает, что $D_g < 0$ и $D_d < 0$, причем $d_i < 0$, так как диагональная точка больше удалена от окружности, т.е. по критерию $d_i < 0$ как и в предыдущих случаях следует выбрать горизонтальный пиксель P_g, что верно.

Случай $D_d > 0$

Здесь в качестве следующего пикселя могут быть выбраны или диагональный - P_d или вертикальный P_v.

Для определения того, какой пиксель выбрать P_d или P_v составим разность:

$$s_i = \left|D_d\right| - \left|D_v\right| = \left|(X+1)^2 + (Y-1)^2 - R^2\right| - \left|X^2 + (Y-1)^2 - R^2\right|.$$

Если $s_i \le 0$, то расстояние до вертикальной точки больше и надо выбирать диагональный пиксель P_d, если же $s_i > 0$, то выбираем вертикальный пиксель P_v.

Рассмотрим вычисление s_i для разных вариантов.

Для вариантов 5 и 6:

$D_d > 0$ и $D_v \le 0$, так как диагональный пиксель вне, а вертикальный либо вне, либо на окружности.

$$s_i = (X+1)^2 + (Y-1)^2 - R^2 + X^2 + (Y-1)^2 - R^2.$$

Добавляя и вычитая $(X+1)^2$, получим:

$$s_i = 2 \cdot[(X+1)^2 + (Y-1)^2 - R^2] - 2 \cdot X - 1.$$

В квадратных скобках стоит D_d, так что

$$s_i = 2 \cdot(D_d - X) - 1.$$

Для варианта 7:

Ясно, что должен быть выбран вертикальный пиксель P_v. Проверка компонент s_i показывает, что $D_d > 0$ и $D_v > 0$, причем $s_i > 0$, так как диагональная точка больше удалена от окружности, т.е. по критерию $s_i > 0$ как и в предыдущих случаях следует выбрать вертикальный пиксель P_v, что соответствует выбору для вариантов 5 и 6.

Случай $D_d = 0$

Для компонент d_i имеем: $D_g > 0$ и $D_d = 0$, следовательно по критерию $d_i >$

0 выбираем диагональный пиксель.

С другой стороны, для компонент s_i имеем: $D_d = 0$ и $D_v < 0$, так что по критерию $s_i \leq 0$ также выбираем диагональный пиксель.

Итак:

$D_d < 0$

$d_i \leq 0$ - выбор горизонтального пикселя P_g

$d_i > 0$ - выбор диагонального пикселя P_d

$D_d > 0$

$s_i \leq 0$ - выбор диагонального пикселя P_d

$s_i > 0$ - выбор вертикального пикселя P_v

$D_d = 0$

выбор диагонального пикселя P_d.

Выведем рекуррентные соотношения для вычисления D_d для (i+1)-го шага, после выполнения i-го.

1. Для горизонтального шага к X_{i+1}, Y_i

$X_{i+1}=X_i+1$

$Y_{i+1}=Y_i$

$D_{di}=(X_{i+1}+1)^2+(Y_{i+1}-1)^2-R^2=X_{i+1}{}^2+2\cdot X_{i+1}+1+(Y_{i+1}-1)^2-R^2=$

$(X_i+1)^2+(Y_i-1)^2-R^2+2\cdot X_{i+1}+1=D_{d\,i} + 2\cdot X_{i+1} + 1$

2. Для диагонального шага к X_{i+1}, Y_{i-1}

$X_{i+1}=X_i+1$

$Y_{i+1}=Y_i-1$

$D_{d\ i+1} = D_{d\,i} + 2 \cdot X_{i+1} - 2 \cdot Y_{i+1} + 2$

3. Для вертикального шага к X_i, Y_{i-1}

$X_{i+1}=X_i$

$Y_{i+1}=Y_i-1$

$D_{d\ i+1} = D_{d\,i} - 2 \cdot Y_{i+1} + 1$

Начальная инициализация должна быть:

$X= 0$

$Y= R$

$D_d = (X+1)^2 + (Y-1)^2 - R^2 = 1 + (R-1)^2 - R^2 = 2\cdot(1 - R)$

Пиксели в остальных четвертях можно получить отражением. Кроме того, достаточно сформировать дугу только во втором октанте, а остальные пиксели сформировать из соображений симметрии.

Остальная часть окружности строится симметрично.

4. ЗАПОЛНЕНИЕ МНОГОУГОЛЬНИКА

4.1. Введение

В большинстве приложений используется одно из существенных достоинств растровых устройств - возможность заполнения областей экрана.

Существует две разновидности заполнения:

- первая, связанная как с интерактивной работой, так и с программным синтезом изображения, служит для заполнения внутренней части многоугольника, заданного координатами его вершин.
- вторая, связанная, в первую очередь, с интерактивной работой, служит для заливки области, которая либо очерчена границей с кодом пикселя, отличающимся от кодов любых пикселей внутри области, либо закрашена пикселями с заданным кодом.

4.2. Заполнение многоугольника

В данном разделе рассмотрим алгоритм заполнения многоугольника. В следующем разделе будут рассмотрены алгоритмы заливки области.

Простейший способ заполнения многоугольника, заданного координатами вершин, заключается в определении: принадлежит ли текущий пиксель внутренней части многоугольника. Если принадлежит, то пиксель заносится.

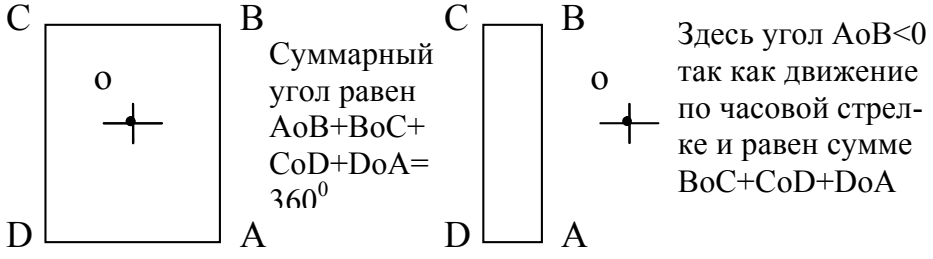


Рис. 4.1. Определение принадлежности пикселя многоугольнику

Определить принадлежность пикселя многоугольнику можно, например, подсчетом суммарного угла с вершиной на пикселе при обходе контура многоугольника. Если пиксель внутри, то угол будет равен 360^0, если вне – 0^0 (рис.4.1).

Вычисление принадлежности должно производиться для всех пикселей экрана и так как большинство пикселей скорее всего вне многоугольников, то данный способ слишком расточителен. Объем лишних вычислений в некоторых случаях можно сократить использованием прямоугольной оболочки - минимального прямоугольника, объемлющего интересующий объект, но все равно вычислений будет много. Другой метод определения принадлежности точки внутренней части многоугольника будет рассмотрен ниже при изучении отсечения отрезков по алгоритму Кируса-Бека.

4.3. Построчное заполнение

Реально используются алгоритмы построчного заполнения, основанные на том, что соседние пиксели в строке скорее всего одинаковы и меняются только там, где строка пересекается с ребром многоугольника. Это называется когерентностью растровых строк (строки сканирования Y_i, Y_{i+1}, Y_{i+2} на рис. 4.2). При этом достаточно определить X-координаты пересечений строк сканирования с ребрами. Пары отсортированных точек пересечения задают интервалы заливки.

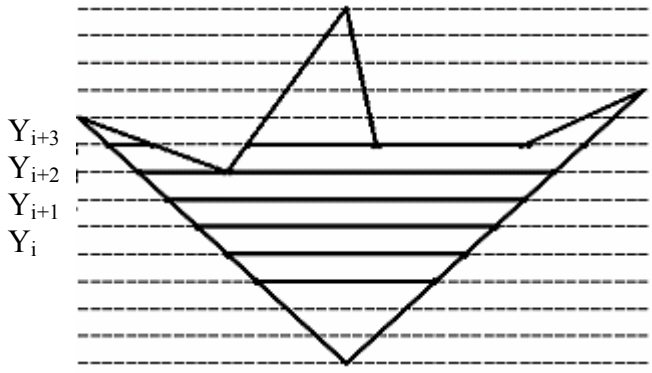


Рис. 4.2. Построчная закраска многоугольника

Кроме того, если какие-либо ребра пересекались i-й строкой, то они скорее всего будут пересекаться также и строкой i+1 (строки сканирования Y_i и

Y_{i+1} на рис. 4.2.). Это называется когерентностью ребер. При переходе к новой строке легко вычислить новую X-координату точки пересечения ребра, используя X-координату старой точки пересечения и тангенс угла наклона ребра:

$$X_{i+1} = X_i + 1/k$$

(тангенс угла наклона ребра - k = dy/dx, так как dy = 1, то 1/k = dx).

Смена же количества интервалов заливки происходит только тогда, когда в строке сканирования появляется вершина.

Учет когерентности строк и ребер позволяет построить для заполнения многоугольников различные высокоэффективные алгоритмы построчного сканирования. Для каждой строки сканирования рассматриваются только те ребра, которые пересекают строку. Они задаются списком активных ребер (САР). При переходе к следующей строке для пересекаемых ребер перевычисляются X-координаты пересечений. При появлении в строке сканирования вершин производится перестройка САР. Ребра, которые перестали пересекаться, удаляются из САР, а все новые ребра, пересекаемые строкой заносятся в него.

Общая схема алгоритма, динамически формирующего список активных ребер и заполняющего многоугольник снизу-вверх, следующая:

1. Подготовить служебные целочисленные массивы Y-координат вершин и номеров вершин.
2. Совместно отсортировать Y-координаты по возрастанию и массив номеров вершин для того, чтобы можно было определить исходный номер вершины.
3. Определить пределы заполнения по оси Y - Y_min и Y_max. Стартуя с текущим значением Y_tek = Y_min, исполнять пункты 4-9 до завершения раскраски.
4. Определить число вершин, расположенных на строке Y_tek - текущей строке сканирования.
5. Если вершины есть, то для каждой из вершин дополнить список активных ребер, используя информацию о соседних вершинах. Для каждого ребра в список активных ребер заносятся:

- максимальное значение Y-координаты ребра;
- приращение X-координаты при увеличении Y на 1;
- начальное значение X-координаты.

Если обнаруживаются горизонтальные ребра, то они просто закрашиваются и информация о них в список активных ребер не заносится. Если после этого обнаруживается, что список активных ребер пуст, то заполнение закончено.

6. По списку активных ребер определяется Y_след - Y-координата ближайшей вершины. (Вплоть до Y_след можно не заботиться о модификации САР, а только менять X-координаты пересечений строки сканирования с активными ребрами).

7. В цикле от Y_tek до Y_след:
   - выбрать из списка активных ребер и отсортировать X-координаты пересечений активных ребер со строкой сканирования;
   - определить интервалы и выполнить закраску;
   - перевычислить координаты пересечений для следующей строки сканирования.

8. Проверить не достигли ли максимальной Y-координаты. Если достигли, то заливка закончена, иначе выполнить пункт 9.

9. Очистить список активных ребер от ребер, закончившихся на строке Y_след, и перейти к пункту 4.

4.4. Заливка области с затравкой

Как уже отмечалось, для приложений, связанных в основном с интерактивной работой, используются алгоритмы заполнения области с затравкой.

При этом тем или иным образом задается заливаемая (перекрашиваемая) область, код пикселя, которым будет выполняться заливка, и начальная точка в области, начиная с которой начнется заливка.

По способу задания области делятся на два типа:

- гранично-определенные, задаваемые своей (замкнутой) границей такой, что коды пикселей границы отличны от кодов внутренней, перекраши-

ваемой части области. На коды пикселей внутренней части области налагаются два условия - они должны быть отличны от кода пикселей границы и кода пикселя перекраски. Если внутри гранично-определенной области имеется еще одна граница, нарисованная пикселями с тем же кодом, что и внешняя граница, то соответствующая часть области не должна перекрашиваться;

- внутренне-определенные, нарисованные одним определенным кодом пикселя. При заливке этот код заменяется на новый код закраски.

В этом состоит основное отличие заливки области с затравкой от заполнения многоугольника. В последнем случае мы сразу имеем всю информацию о предельных размерах части экрана, занятой многоугольником. Поэтому определение принадлежности пикселя многоугольнику базируется на быстро работающих алгоритмах, использующих когерентность строк и ребер (см. предыдущий раздел). В алгоритмах же заливки области с затравкой нам вначале надо прочитать пиксель, затем определить принадлежит ли он области и если принадлежит, то перекрасить.

Заливаемая область или ее граница - некоторое связное множество пикселей. По способам доступа к соседним пикселям области делятся на 4-х и 8-ми связные. В 4-х связных областях доступ к соседним пикселям осуществляется по четырем направлениям - горизонтально влево и вправо и вертикально вверх и вниз. В 8-ми связных областях к этим направлениям добавляются еще 4 диагональных. Используя связность, мы можем, двигаясь от точки затравки, достичь и закрасить все пиксели области.

Важно отметить, что для 4-х связной прямоугольной области граница 8-ми связна (рис. 4.3.а) и наоборот у 8-ми связной области граница 4-х связна (см. рис. 4.3,б). Поэтому заполнение 4-х связной области 8-ми связным алгоритмом может привести к "просачиванию" через границу и заливке пикселей в примыкающей области.

В общем, 4-х связную область мы можем заполнить как 4-х, так и 8-ми связным алгоритмом. Обратное же неверно. Так, область на рис. 4.3,а мы можем заполнить любым алгоритмом, а область на рис. 4.3.б, состоящую из двух

примыкающих 4-х связных областей, можно заполнить только 8-ми связным алгоритмом.

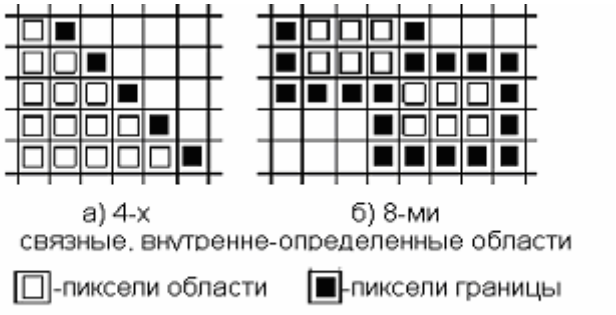


Рис. 4.3. Связность областей и их границ

С использованием связности областей и стека можно построить простые алгоритмы закраски как внутренней, так и гранично-определенной области.

4.5. Построчный алгоритм заливки с затравкой

Использует пространственную когерентность:

- пиксели в строке меняются только на границах;
- при перемещении к следующей строке размер заливаемой строки скорее всего или неизменен или меняется на 1 пиксель.

Таким образом, на каждый закрашиваемый фрагмент строки в стеке хранятся координаты только одного начального пикселя, что приводит к существенному уменьшению размера стека.

Последовательность работы алгоритма для гранично определенной области следующая:

1. Координата затравки помещается в стек, затем до исчерпания стека выполняются пункты 2-4.
2. Координата очередной затравки извлекается из стека и выполняется максимально возможное закрашивание вправо и влево по строке с затравкой, т.е. пока не попадется граничный пиксель. Пусть это $X_{лев}$ и $X_{прав}$, соответственно.
3. Анализируется строка ниже закрашиваемой в пределах от $X_{лев}$ до $X_{прав}$ и в ней находятся крайние правые пиксели всех не закрашенных фрагментов. Их координаты заносятся в стек.
4. То же самое проделывается для строки выше закрашиваемой.

5. ОТСЕЧЕНИЕ ОТРЕЗКОВ

5.1. Введение

Если изображение выходит за пределы экрана, то на части дисплеев увеличивается время построения за счет того, что изображение строится в "уме". В некоторых дисплеях выход за пределы экрана приводит к искажению картины, так как координаты просто ограничиваются при достижении ими граничных значений, а не выполняется точный расчет координат пересечения (эффект "стягивания" изображения). Некоторые, в основном, простые дисплеи просто не допускают выхода за пределы экрана. Все это, особенно в связи с широким использованием технологии просмотра окнами, требует выполнения отсечения сцены по границам окна видимости.

В простых графических системах достаточно двумерного отсечения, в трехмерных пакетах используется трех и четырехмерное отсечение.

Программное исполнение отсечения достаточно медленный процесс, поэтому, естественно, в мощные дисплеи встраивается соответствующая аппаратура. Здесь мы рассмотрим алгоритмы для программной реализации отсечения.

Отсекаемые отрезки могут быть трех классов - целиком видимые, целиком невидимые и пересекающие окно. Очевидно, что целесообразно возможно более рано, без выполнения большого объема вычислений принять решение об видимости целиком или отбрасывании. По способу выбора простого решения об отбрасывании невидимого отрезка целиком или принятия его существует два основных типа алгоритмов отсечения - алгоритмы, использующие кодирование концов отрезка или всего отрезка, и алгоритмы, использующие параметрическое представление отсекаемых отрезков и окна отсечения.

Алгоритмы с кодированием применимы для прямоугольного окна, стороны которого параллельны осям координат, в то время как алгоритмы с параметрическим представлением применимы для произвольного окна.

Вначале мы рассмотрим алгоритм с кодированием Коэна-Сазерленда, обладающий одним из лучших быстродействий при компактной реализации. Затем рассмотрим алгоритм Лианга-Барски для отсечения прямоугольным ок-

ном с использованием параметрического представления. Быстродействие этого алгоритма сравнимо с быстродействием алгоритма Коэна-Сазерленда при большей компактности и наличии 3D и 4D реализаций.

5.2. Двумерный алгоритм Коэна-Сазерленда

Этот алгоритм позволяет быстро выявить отрезки, которые могут быть или приняты или отброшены целиком. Вычисление пересечений требуется когда отрезок не попадает ни в один из этих классов. Этот алгоритм особенно эффективен в двух крайних случаях:

- большинство примитивов содержится целиком в большом окне;
- большинство примитивов лежит целиком вне относительно маленького окна.

Идея алгоритма состоит в следующем:

Окно отсечения и прилегающие к нему части плоскости вместе образуют 9 областей (рис. 5.1). Каждой из областей присвоен 4-х разрядный код.

Две конечные точки отрезка получают 4-х разрядные коды, соответствующие областям, в которые они попали. Смысл разрядов кода:

1 рр = 1 - точка над верхним краем окна;

2 рр = 1 - точка под нижним краем окна;

3 рр = 1 - точка справа от правого края окна;

4 рр = 1 - точка слева от левого края окна.

Определение того, лежит ли отрезок целиком внутри окна или целиком вне окна, выполняется следующим образом:

- если коды обоих концов отрезка равны 0, то отрезок целиком внутри окна, отсечение не нужно, отрезок принимается как тривиально видимый (отрезок AB на рис. 5.1);
- если логическое & кодов обоих концов отрезка не равно нулю, то отрезок целиком вне окна, отсечение не нужно, отрезок отбрасывается как тривиально невидимый (отрезок KL на рис. 5.1);
- если логическое & кодов обоих концов отрезка равно нулю, то отрезок подозрительный, он может быть частично видимым (отрезки CD, EF, GH) или

целиком невидимым (отрезок IJ); для него нужно определить координаты пересечений со сторонами окна и для каждой полученной части определить тривиальную видимость или невидимость. При этом для отрезков CD и IJ потребуется вычисление одного пересечения, для остальных (EF и GH) - двух.

При расчете пересечения используется горизонтальность либо вертикальность сторон окна, что позволяет определить координату X или Y точки пересечения без вычислений.

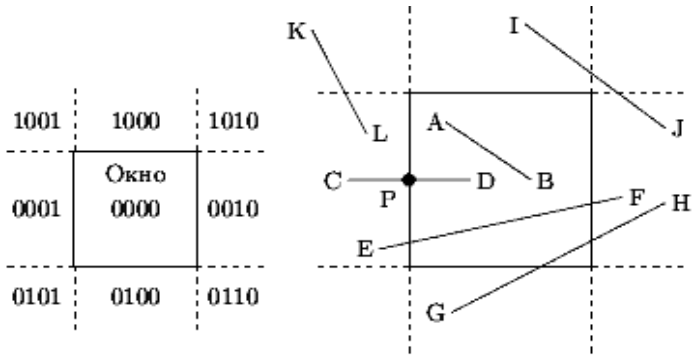


Рис. 5.1. Отсечение по методу Коэна-Сазерленда

При непосредственном использовании описанного выше способа отбора целиком видимого или целиком невидимого отрезка после расчета пересечения потребовалось бы вычисление кода расположения точки пересечения. Для примера рассмотрим отрезок CD. Точка пересечения обозначена как P. В силу того, что граница окна считается принадлежащей окну, то можно просто принять только часть отрезка PD, попавшую в окно. Часть же отрезка CP, на самом деле оказавшаяся вне окна, потребует дальнейшего рассмотрения, так как логическое & кодов точек C и P даст 0, т.е. отрезок CP нельзя просто отбросить. Для решения этой проблемы Коэн и Сазерленд предложили заменять конечную точку с ненулевым кодом конца на точку, лежащую на стороне окна, либо на ее продолжении.

В целом схема алгоритма Коэна-Сазерленда следующая:

1. Рассчитать коды конечных точек отсекаемого отрезка.

В цикле повторять пункты 2-6:

2. Если логическое & кодов конечных точек не равно 0, то отре-

зок целиком вне окна. Он отбрасывается и отсечение закончено.

3. Если оба кода равны 0, то отрезок целиком видим. Он принимается, и отсечение закончено.

4. Если начальная точка внутри окна, то она меняется местами с конечной точкой.

5. Анализируется код начальной точки для определения стороны окна с которой есть пересечение, и выполняется расчет пересечения. При этом вычисленная точка пересечения заменяет начальную точку.

6. Определение нового кода начальной точки.

5.3. Двумерный алгоритм Лианга-Барски

Алгоритм Лианг и Барски использует параметрическое представление для двух, трех и четырехмерного отсечения

При 2D отсечении прямые отсекаются по 2D области, называемой окном отсечения. В частности, внутренняя часть окна отсечения может быть выражена с помощью следующих неравенств (рис. 5.2).

$$X_{лев} \le x \le X_{прав}$$
$$Y_{низ} \le y \le Y_{верх}$$

Рис. 5.2. Внутренняя часть окна отсечения

Продолжим каждую из четырех границ окна до бесконечных прямых. Каждая из таких прямых делит плоскость на 2 области. Назовем "видимой частью" ту, в которой находится окно отсечения (рис.5.3). Видимой части соответствует внутренняя сторона линии границы. Невидимой части плоскости соответствует внешняя сторона линии границы.

Рис. 5.3. Видимая часть линии границы

Таким образом, окно отсечения может быть определено как область, которая находится на внутренней стороне всех линий границ.

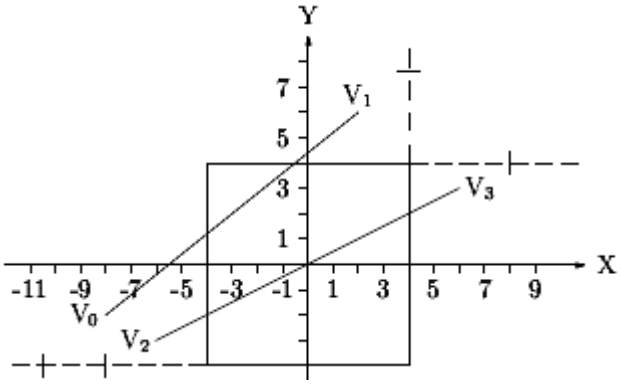


Рис. 5.4 . Пример расчета отсечения

Отсекаемый отрезок прямой может быть преобразован в параметрическое представление следующим образом. Пусть конечные точки отрезка есть V_0 и V_1 с координатами (x0,y0) и (x1,y1), соответственно (рис. 5.4). Тогда параметрическое представление линии может быть задано следующим образом:

$$x = x0 + dx \cdot t; \quad y = y0 + dy \cdot t, \qquad (5.3.1)$$

где $dx = x1 - x0; \; dy = y1 - y0.$ (5.3.2)

Или в общем виде для отрезка, заданного точками V_0 и V_1:

$$V(t) = V_0 + (V_1 - V_0) \cdot t \qquad (5.3.3)$$

Для точек V_0 и V_1 параметр t равен 0 и 1, соответственно. Меняя t от 0 до 1, перемещаемся по отрезку V_0V_1 от точки V_0 к точке V_1. Изменяя t в интервале от $-\infty$ до $+\infty$, получаем бесконечную (далее удлиненную) прямую, ориентация которой - от точки V_0 к точке V_1.

Подставляя параметрическое представление, заданное уравнениями (5.3.1) и (5.3.2), в неравенства (рис. 5.2), получим следующие соотношения для частей удлиненной линии, которая находится в окне отсечения:

$$\begin{aligned} -dx \cdot t &\le x0 - Xлев \text{ и } dx \cdot t \le Xправ - x0 \\ -dy \cdot t &\le y0 - Yниз \text{ и } dy \cdot t \le Yверх - y0 \end{aligned} \qquad (5.3.4)$$

Заметим, что соотношения (5.3.4) - неравенства, описывающие внутреннюю часть окна отсечения, в то время как равенства определяют его границы.

Рассматривая неравенства (5.3.4), видим, что они имеют одинаковую

форму вида:

$$P_i \cdot t \le Q_i \quad \text{для} \, i = 1,2,3,4. \tag{5.3.5}$$

Здесь использованы следующие обозначения:

$$\begin{aligned} P_1 &= -dx; \quad Q_1 = x0 - Xлев; \\ P_2 &= dx; \quad Q_2 = Xлев - x0; \\ P_3 &= -dy; \quad Q_3 = y0 - Yниз; \\ P_4 &= dy; \quad Q_4 = Yверх - y0; \end{aligned} \tag{5.3.6}$$

Вспоминая определения внутренней и внешней стороны линии границы (см. рис. 5.3), замечаем, что каждое из неравенств (5.3.5) соответствует одной из граничных линий (левой, правой, нижней и верхней, соответственно) и описывает ее видимую сторону. (Например, для i=1 имеем: $P_1 \cdot t \le Q_1 \Rightarrow -dx \cdot t \le x0 - X_{лев} \Rightarrow x0 + dx \cdot t \ge X_{лев}$). Удлиним V_0V_1 в бесконечную прямую. Тогда каждое неравенство задает диапазон значений параметра t, для которых эта удлиненная линия находится на видимой стороне соответствующей линии границы. Более того, конкретное значение параметра t для точки пересечения есть $t = Q_i/P_i$. Причем знак Q_i показывает, на какой стороне соответствующей линии границы находится точка V_0. А именно, если $Qi \ge 0$, тогда V_0 находится на видимой стороне линии границы, включая и ее. Если же $Q_i < 0$, тогда V_0 находится на невидимой стороне.

Рассмотрим P_i в соотношениях (5.3.6). Ясно, что любое P_i может быть меньше 0, больше 0 и равно 0.

$\underline{P_i < 0.}$

Если $P_i < 0$, тогда соответствующее неравенство становится:

$$t \ge Q_i / P_i. \tag{5.3.7}$$

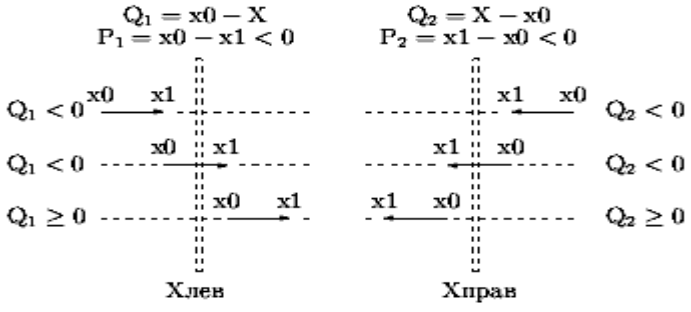


Рис. 5.5. Пересечение удлиненной линии, определяемой точками V_0V_1 и идущей с невидимой на видимую сторону, с левой и правой границами.

Для пояснения на рис.5.5 показано пересечение с левой и правой границами при $P_i < 0$.

Очевидно, что диапазон значений параметра t, для которых удлиненная линия находится на видимой стороне соответствующей граничной линии, имеет минимум в точке пересечения направленной удлиненной линии, заданной вектором V_0V_1 и идущей с невидимой на видимую сторону граничной линии (так как только на границе t равно Q_i / P_i, а в остальной части видимой стороны больше).

$$\underline{\mathbf{P_i} > 0.}$$

Аналогично, если $P_i > 0$, тогда соответствующее неравенство становится:

$$t \leq Q_i / P_i. \qquad (5.3.8)$$

Для пояснения на рис. 5.6 показано пересечение с левой и правой границами при $P_i > 0$.

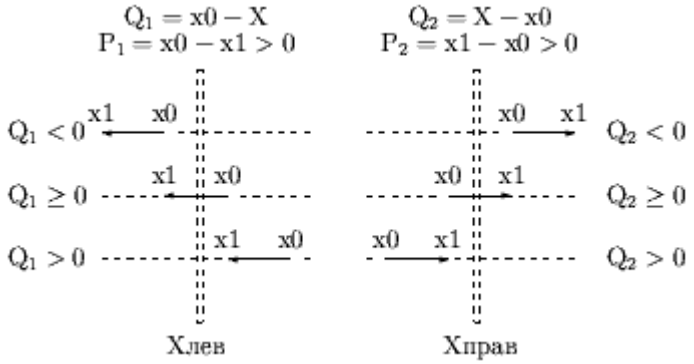


Рис. 5.6. Пересечение удлиненной линии, определяемой точками V_0V_1 и идущей с видимой на невидимую сторону, с левой и правой границами.

Так как значения параметра t только на границе равны Q_i/P_i, а в остальной видимой части меньше Q_i/P_i, то значение параметра t имеет максимум на границе.

$$\underline{\mathbf{P_i} = 0.}$$

Наконец, если $P_i = 0$, тогда соответствующее неравенство имеет вид:

$$0 \leq Q_i. \qquad (5.3.9.)$$

Заметим, что здесь нет зависимости от t, т.е. неравенство выполняется

для всех t, если $Q_i \geq 0$ и не имеет решения при $Q_i < 0$. Для пояснения на рис. 5.7 иллюстрируется случай $P_i = 0$.

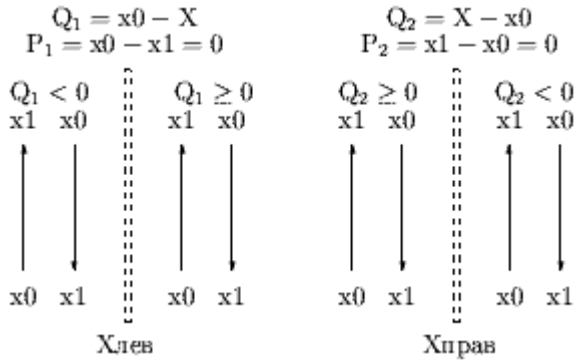


Рис. 5.7. Относительное расположение удлиненной линии, заданной точками V_0V_1 и идущей параллельно левой и правой границам

Геометрически, если $P_i = 0$, то нет точек пересечения удлиненной линии, определяемой точками V_0V_1, с линиями границы. Более того, если $Q_i < 0$, то удлиненная линия находится на внешней стороне линии границы, а при $Q_i \geq 0$ - на внутренней стороне (включая ее). В последнем случае отрезок V_0V_1 может быть видим или нет в зависимости от того, где находятся точки V_0V_1 на удлиненной линии. В предыдущем же случае нет видимого сегмента, так как удлиненная линия вне окна, т.е. это случай тривиального отбрасывания.

Все эти случаи суммированы на блок-схеме, представленной на рис. 5.8.

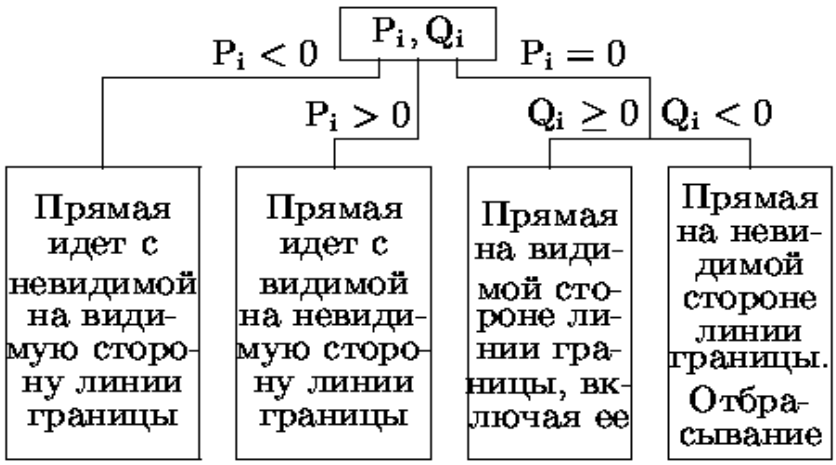


Рис. 5.8. Блок-схема алгоритма Лианга-Барски

Итак, рассмотрение четырех неравенств дает диапазон значений параметра t, для которого удлиненная линия находится внутри окна отсечения. Од-

нако отрезок V_0V_1 только часть удлиненной линии и он описывается значениями параметра t в диапазоне: $0 \le t \le 1$. Таким образом, решение задачи двумерного отсечения эквивалентно решению неравенств (5.3.5) при условии $0 \le t \le 1$. Решение этой задачи сводится к далее описанному отысканию максимумов и минимумов.

Вспомним, что для всех i таких, что $P_i < 0$, условие видимости имеет вид: $t \ge Q_i / P_i$. Из условия принадлежности точек удлиненной линии отрезку V_0V_1 имеем $t \ge 0$. Таким образом, нужно искать:

$$t \ge \max(\{Q_i / P_i \mid P_i < 0, i = 1,2,3,4\} \cup \{0\}). \qquad (5.3.10)$$

Аналогично для всех i таких что $P_i > 0$, условие видимости - $t \le Q_i / P_i$ и, следовательно, ≤ 1.

$$t \le \min(\{Q_i / P_i \mid P_i > 0, i = 1,2,3,4\} \cup \{1\}). \qquad (5.3.11)$$

Наконец, для всех i, таких что $P_i = 0$, следует проверить знак Q_i. Если $Q_i < 0$, то это случай тривиального отбрасывания, задача отсечения решена и дальнейшие вычисления не нужны. Если же $Q_i \ge 0$, то информации, даваемой неравенством, недостаточно и это неравенство игнорируется.

Правая часть неравенств (5.3.10) и (5.3.11) - значения параметра t, соответствующие началу и концу видимого сегмента, соответственно. Обозначим эти значения как t_0 и t_1:

$$\begin{aligned} t_0 &\ge \max(\{Q_i / P_i \mid P_i < 0, i = 1,2,3,4\} \cup \{0\}), \\ t_1 &\le \min(\{Q_i / P_i \mid P_i > 0, i = 1,2,3,4\} \cup \{1\}). \end{aligned} \qquad (5.3.12)$$

Если сегмент отрезка V_0V_1 видим, то ему соответствует интервал параметра:

$$t_0 \le t \le t_1. \qquad (5.3.13)$$

Следовательно, необходимое условие видимости сегмента:

$$t_0 \le t_1. \qquad (5.3.14)$$

Но это недостаточное условие, так как оно игнорирует случай тривиального отбрасывания при $P_i = 0$, если $Q_i < 0$. Тем не менее, это достаточное условие для отбрасывания, т.е. если $t_0 > t_1$, то отрезок должен быть отбро-

шен. Алгоритм проверяет, если $P_i = 0$ и $Q_i < 0$, или $t_0 > t_1$ и в этом случае отрезок немедленно отбрасывается без дальнейших вычислений.

В алгоритме t_0 и t_1 инициализируются в 0 и 1, соответственно. Затем последовательно рассматривается каждое отношение Q_i/P_i.

Если $P_i < 0$, то отношение вначале сравнивается с t_1 и если оно больше t_1, то это случай отбрасывания. В противном случае оно сравнивается с t_0 и если оно больше, то t_0 должно быть заменено на новое значение.

Если $P_i > 0$, то отношение вначале сравнивается с t_0 и если оно меньше t_0, то это случай отбрасывания. В противном случае оно сравнивается с t_1 и, если оно меньше, то t_1 должно быть заменено на новое значение.

Наконец, если Pi = 0 и $Q_i < 0$, то это случай отбрасывания.

На последнем этапе алгоритма, если отрезок еще не отброшен, то t_0 и t_1 используются для вычисления соответствующих точек. Однако если $t_0 = 0$, то конечная точка равна V_0 и не требуется вычислений. Аналогично, если $t_1 = 1$, то конечная точка - V_1 и вычисления также не нужны.

Геометрический смысл этого процесса состоит в том, что отрезок удлиняется для определения, где эта удлиненная линия пересекает каждую линию границы. Более детально, каждая конечная точка заданного отрезка V_0V_1 используется как начальное значение для конечных точек отсеченного отрезка C_0C_1. Затем вычисляются точки пересечения удлиненной линии с каждой линией границы. Если для данной линии границы направление, определяемое V_0V_1, идет с невидимой на видимую сторону линии границы, то эта точка пересечения вначале сравнивается с C_1. Если точка находится далее вдоль линии, тогда C_1 (и таким образом, C_0C_1) должна быть на невидимой стороне линии, поэтому отрезок должен быть отброшен. В противном случае точка пересечения сравнивается с C_0; если точка далее вдоль линии, тогда C_0 перемещается вперед к этой точке.

С другой стороны, если направление с видимой на невидимую сторону, тогда точка пересечения вначале сравнивается с C_0. Если C_0 далее вдоль линии, чем точка пересечения, тогда C_0 (и, следовательно, C_0C_1) находится на невиди-

мой стороне линии границы, т.е. отрезок должен быть отброшен. В противном случае точка пересечения сравнивается с C_1 и, если C_1 далее вдоль линии, тогда C_1 перемещается назад к точке пересечения.

Наконец, если удлиненная линия параллельна граничной линии и она на невидимой стороне, то отрезок отбрасывается. В конце алгоритма, если отрезок не отброшен, тогда C_0 и C_1 используются как конечные точки видимой части отрезка.

6. УДАЛЕНИЕ СКРЫТЫХ ЛИНИЙ И ПОВЕРХНОСТЕЙ

6.1. Введение

Методы удаления невидимых частей сцены можно классифицировать:

1. По выбору удаляемых частей: удаление невидимых линий, ребер, поверхностей, объемов.
2. По порядку обработки элементов сцены: удаление в произвольном порядке и в порядке, определяемом процессом визуализации.
3. По системе координат:

а) алгоритмы, работающие в пространстве объектов, когда каждая из N граней объекта сравнивается с остальными N-1 гранями (объем вычислений растет как N^2);

б) алгоритмы, работающие в пространстве изображения, когда для каждого пикселя изображения определяется какая из N граней объекта видна (при разрешении экрана M×M объем вычислений растет как $M^2 \times N$).

6.2. Алгоритмы удаления линий

Применение - векторные устройства. Могут применяться и в растровых устройствах для ускорения процесса визуализации, но при этом не используется основное ценное качество растрового дисплея - возможность закраски поверхностей.

Наиболее известный ранний алгоритм - алгоритм Робертса (1963 г.). Работает только с выпуклыми телами в пространстве объектов. Каждый объект сцены представляется многогранным телом, полученным в результате пересечения плоскостей, т.е. тело описывается списком граней, состоящих из ребер,

которые в свою очередь образованы вершинами.

Вначале из описания каждого тела удаляются нелицевые плоскости, экранированные самим телом. Затем каждое из ребер сравнивается с каждым телом для определения видимости или невидимости, т.е. объем вычислений растет как квадрат числа объектов в сцене. Наконец вычисляются новые ребра, полученные при “протыкании” телами друг друга.

6.3. Алгоритм удаления поверхностей с Z-буфером

Алгоритм предложен Эдом Кэтмулом и представляет собой обобщение буфера кадра. Обычный буфер кадра хранит коды цвета для каждого пикселя в пространстве изображения. Идея алгоритма состоит в том, чтобы для каждого пикселя дополнительно хранить еще и координату Z или глубину. При занесении очередного пикселя в буфер кадра значение его Z-координаты сравнивается с Z-координатой пикселя, который уже находится в буфере. Если Z-координата нового пикселя больше, чем координата старого, т.е. он ближе к наблюдателю, то атрибуты нового пикселя и его Z-координата заносятся в буфер, если нет, то ничего не делается.

Этот алгоритм наиболее простой из всех алгоритмов удаления невидимых поверхностей, но требует большого объема памяти. Данные о глубине для реалистичности изображения обычно достаточно иметь с разрядностью порядка 20 бит. В этом случае при изображении нормального телевизионного размера в 768×576 пикселей для хранения Z-координат необходим объем памяти порядка 1 Мбайта. Суммарный объем памяти при 3 байтах для значений RGB составит более 2,3 Мбайта.

Время работы алгоритма не зависит от сложности сцены. Многоугольники, составляющие сцену, могут обрабатываться в произвольном порядке. Для сокращения затрат времени нелицевые многоугольники могут быть удалены. По сути дела алгоритм с Z-буфером - некоторая модификация уже рассмотренного алгоритма заливки многоугольника. Если используется построчный алгоритм заливки, то легко сделать пошаговое вычисление Z-координаты очередного пикселя, дополнительно храня Z-координаты его вершин и вычисляя прира-

щение dz Z-координаты при перемещении вдоль X на dx, равное 1. Если известно уравнение плоскости, в которой лежит обрабатываемый многоугольник, то можно обойтись без хранения Z-координат вершин. Пусть уравнение плоскости имеет вид:

$$A \cdot x + B \cdot y + C \cdot z + D = 0.$$

Тогда при C не равном нулю

$$z = -(A \cdot x + B \cdot y + D)/C.$$

Найдем приращение Z-координаты пикселя при шаге по X на dx, помня, что Y очередной обрабатываемой строки - константа.

$$dz = -(A \cdot (x + dx) + D)/C + (A \cdot x + D)/C = -A \cdot dx/C,$$

но dx = 1, поэтому

$$dz = -A/C.$$

Основной недостаток алгоритма с Z-буфером - дополнительные затраты памяти. Для их уменьшения можно разбивать изображение на несколько прямоугольников или полос. В пределе можно использовать Z-буфер в виде одной строки. Понятно, что это приведет к увеличению времени, так как каждый прямоугольник будет обрабатываться столько раз, на сколько областей разбито пространство изображения. Уменьшение затрат времени в этом случае может быть обеспечено предварительной сортировкой многоугольников на плоскости.

Другие недостатки алгоритма с Z-буфером заключаются в том, что так как пиксели в буфер заносятся в произвольном порядке, то возникают трудности с реализацией эффектов прозрачности или просвечивания и устранением лестничного эффекта с использованием предфильтрации, когда каждый пиксель экрана трактуется как точка конечного размера и его атрибуты устанавливаются в зависимости от того, какая часть пикселя изображения попадает в пиксель экрана. Но другой подход к устранению лестничного эффекта, основанный на постфильтрации - усреднении значений пикселя с использованием изображения с большим разрешением реализуется сравнительно просто за счет увеличения расхода памяти (и времени). В этом случае используются два метода. Первый состоит в том, что увеличивается разрешение только кадрового буфера, хранящего атрибуты пикселей, а разрешение Z-буфера делается совпадающим с размерами пространства изображения. Глубина изображения вычис-

ляется только для центра группы усредняемых пикселей. Это метод неприменим, когда расстояние до наблюдателя имитируется изменением интенсивности пикселей. Во втором методе и кадровый и Z- буфера имеют увеличенное разрешение и усредняются атрибуты как пикселя, так и его глубина.

Общая схема алгоритма с Z-буфером:

- Инициализировать кадровый и Z-буфера. Кадровый буфер закрашивается фоном. Z-буфер закрашивается минимальным значением Z.
- Выполнить преобразование каждого многоугольника сцены в растровую форму. При этом для каждого пикселя вычисляется его глубина z. Если вычисленная глубина больше, чем глубина, уже имеющаяся в Z-буфере, то занести в буфера атрибуты пикселя и его глубину, иначе никаких занесений не выполнять.
- Выполнить, если это было предусмотрено, усреднение изображения с понижением разрешения.

6.4. Построчный алгоритм с Z-буфером

Рассмотрим теперь алгоритм с Z-буфером размером в одну строку, который представляет собой обобщение алгоритма построчной заливки многоугольника. Для каждой строки сканирования теперь может обрабатываться не один многоугольник.

Общая схема такого алгоритма следующая:

1. Подготовка данных:
   а) для каждого многоугольника определить максимальную Y-координату;
   б) занести многоугольник в группу многоугольников, соответствующую данной Y-координате.
2. Собственно заливка.

6.5. Алгоритм разбиения области Варнока

Алгоритм работает в пространстве изображения и анализирует область на экране дисплея (окно) на наличие в них видимых элементов. Если в окне нет изображения, то оно просто закрашивается фоном. Если же в окне имеется эле-

мент, то проверяется, достаточно ли он прост для визуализации. Если объект сложный, то окно разбивается на более мелкие, для каждого из которых выполняется тест на отсутствие и/или простоту изображения. Рекурсивный процесс разбиения может продолжаться до тех пор, пока не будет достигнут предел разрешения экрана.

Можно выделить 4 случая взаимного расположения окна и многоугольника (рис. 6.1):

- многоугольник целиком вне окна;
- многоугольник целиком внутри окна;

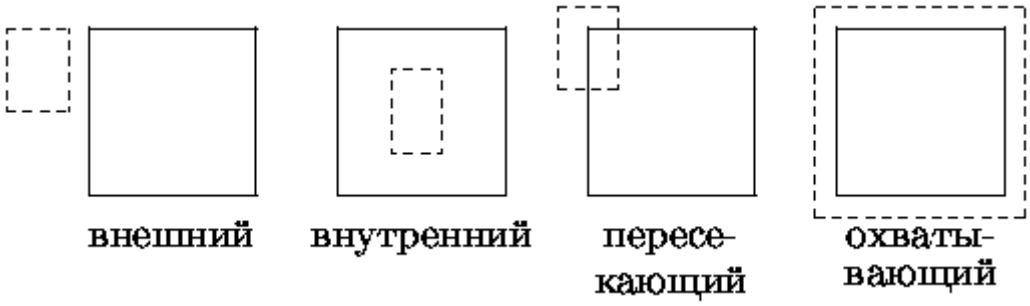


Рис. 6.1. Соотношения между окном экрана (сплошная рамка) и многоугольником (штриховая рамка)

- многоугольник пересекает окно;
- многоугольник охватывает окно.

В четырех случаях можно сразу принять решение о правилах закраски области экрана:

- все многоугольники сцены - внешние по отношению к окну. В этом случае окно закрашивается фоном;
- имеется всего один внутренний или пересекающий многоугольник. В этом случае все окно закрашивается фоном, и затем часть окна, соответствующая внутреннему или пересекающему окну, закрашивается цветом многоугольника;
- имеется единственный охватывающий многоугольник. В этом случае окно закрашивается его цветом;
- имеется несколько различных многоугольников и хотя бы один из них

охватывающий. Если при этом охватывающий многоугольник расположен ближе остальных к наблюдателю, то окно закрашивается его цветом.

В любых других случаях процесс разбиения окна продолжается. Легко видеть, что при растре 1024×1024 и делении стороны окна пополам требуется не более 10 разбиений. Если достигнуто максимальное разбиение, но не обнаружено ни одного из приведенных выше четырех случаев, то для точки с центром в полученном минимальном окне (размером в пиксель) вычисляются глубины оставшихся многоугольников и закраску определяет многоугольник, наиболее близкий к наблюдателю. При этом для устранения лестничного эффекта можно выполнить дополнительные разбиения и закрасить пиксель с учетом всех многоугольников, видимых в минимальном окне.

Первые три случая идентифицируются легко. Последний же случай фактически сводится к поиску охватывающего многоугольника, перекрывающего все остальные многоугольники, связанные с окном. Проверка на такой многоугольник может быть выполнена следующим образом: в угловых точках окна вычисляются Z-координаты для всех многоугольников, связанных с окном. Если все четыре такие Z-координаты охватывающего многоугольника ближе к наблюдателю, чем все остальные, то окно закрашивается цветом соответствующего охватывающего многоугольника. Если же нет, то мы имеем сложный случай, и разбиение следует продолжить.

Очевидно, что после разбиения окна охватывающие и внешние многоугольники наследуются от исходного окна. Поэтому необходимо проверять лишь внутренние и пересекающие многоугольники.

Из изложенного ясно, что важной частью алгоритма является определение расположения многоугольника относительно окна.

Проверка на то, что многоугольник внешний или внутренний относительно окна для случая прямоугольных окон легко реализуется использованием прямоугольной оболочки многоугольника и сравнением координат. Для внутреннего многоугольника должны одновременно выполняться условия:

$$Xmin \geq Wл \quad и \quad Xmax \leq Wп \quad и \quad Ymin \geq Wн \quad и \quad Ymax \leq Wв,$$

здесь Xmin,Xmax,Ymin,Ymax - ребра оболочки

Wл, Wп, Wн, Wв - ребра окна

Для внешнего многоугольника достаточно выполнение любого из следующих условий:

$$Xmin > Xп, \quad Xmax < Wл, \quad Ymin > Wв, \quad Ymax < Wн$$

Таким способом внешний многоугольник, охватывающий угол окна, не будет идентифицирован как внешний (см. рис. 6.2).

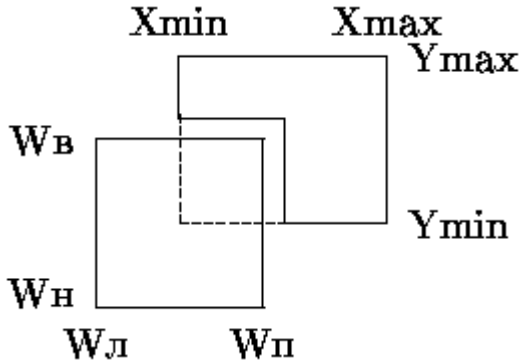


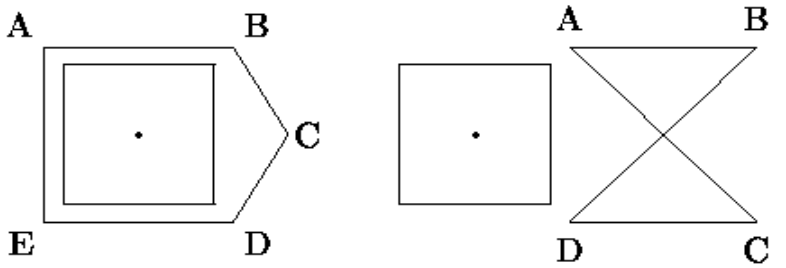
Рис. 6.2. Ошибочное определение внешнего многоугольника как пересекающего при использовании прямоугольной оболочки

Проверка на пересечение окна многоугольником может быть выполнена проверкой на расположение всех вершин окна по одну сторону от прямой, на которой расположено ребро многоугольника. Пусть ребро многоугольника задано точками P1(x1,y1,z1) и P2(x2,y2,z2), а очередная вершина окна задается точкой P3(x3,y3,z3). Векторное произведение вектора **P1P3** на вектор **P1P2**, равное (x3-x1)(y2-y1) - (y3-y1)(x2-x1), будет меньше 0, равно 0 или больше 0, если вершина лежит слева, на или справа от прямой P1P2. Если знаки различны, то окно и многоугольник пересекаются. Если же все знаки одинаковы, то окно лежит по одну сторону от ребра, т.е. многоугольник может быть либо внешним, либо охватывающим.

Вернемся к примеру рис. 6.2. Такой многоугольник рассмотренными тестами не был идентифицирован ни как внутренний, ни как пересекающий, т.е. он может быть либо внешним, либо охватывающим. Для завершающей классифи-

кации может использоваться тест с подсчетом угла, рассматривавшийся ранее для определения нахождения точки внутри/вне многоугольника. В этом тесте вычисляется суммарный угол, на который повернется луч, исходящий из некоторой точки окна (обычно центра), при последовательном обходе вершин многоугольника.

Если суммарный угол равен 0, то многоугольник - внешний. Если же угол равен N×360°, то многоугольник охватывает окно N раз. Простейшая иллюстрация этого теста приведена на рис. 6.3.



а) многоугольник ABCDE — охватывающий Суммарный угол равен 360°

б) многоугольник ABCD — внешний Суммарный угол равен 0°

Рис. 6.3. Тест на охватывающий/внешний многоугольник

6.6. Построчный алгоритм Уоткинса

В алгоритмах построчного сканирования результирующее изображение генерируется построчно, причем, подобно ранее рассмотренному алгоритму построчной заливки многоугольника, используется связность соседних растровых строк изображения. Отличие состоит в том, что учитываются все, а не один многоугольник.

Алгоритм работает в пространстве изображения с окном высотой в одну строку и шириной в экран, тем самым трехмерная задача сводится к двумерной.

Последовательность шагов алгоритма:

- построение списка ребер;
- построение списка многоугольников;
- построение списка активных ребер - создается таблица ребер, включающая все негоризонтальные ребра многоугольников, причем элементы таблицы по

значению Y-координаты отсортированы по группам.

6.7. Алгоритм трассировки лучей

При рассмотрении этого алгоритма предполагается, что наблюдатель находится на положительной полуоси Z, а экран дисплея перпендикулярен оси Z и располагается между объектом и наблюдателем.

Удаление невидимых (скрытых) поверхностей в алгоритме трассировки лучей выполняется следующим образом:

- сцена преобразуется в пространство изображения;
- из точки наблюдения в каждый пиксель экрана проводится луч и определяется, какие именно объекты сцены пересекаются с лучом;
- вычисляются и упорядочиваются по Z координаты точек пересечения объектов с лучом. В простейшем случае для непрозрачных поверхностей без отражений и преломлений видимой точкой будет точка с максимальным значением Z-координаты. Для более сложных случаев требуется сортировка точек пересечения вдоль луча.

Ясно, что наиболее важная часть алгоритма - процедура определения пересечения, которая в принципе выполняется Rx×Ry×N раз (здесь Rx,Ry - разрешение дисплея по X и Y, соответственно, а N - количество многоугольников в сцене).

Очевидно, что повышение эффективности может достигаться сокращением времени вычисления пересечений и избавлением от ненужных вычислений. Последнее обеспечивается использованием геометрически простой оболочки, объемлющей объект - если луч не пересекает оболочку, то не нужно вычислять пересечения с ним многоугольников, составляющих исследуемый объект.

При использовании прямоугольной оболочки определяется преобразование, совмещающее луч с осью Z. Оболочка подвергается этому преобразованию, а затем попарно сравниваются знаки Xmin с Xmax и Ymin с Ymax. Если они различны, то есть пересечение луча с оболочкой (см. рис. 6.4)

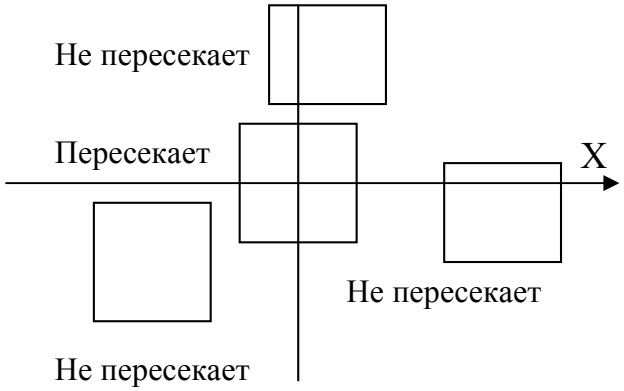


Рис. 6.4. Определение пересечения луча и оболочки

При использовании сферической оболочки для определения пересечения луча со сферой достаточно сосчитать расстояние от луча до центра сферы. Если оно больше радиуса, то пересечения нет. Параметрическое уравнение луча, проходящего через две точки P1(x1,y1,z1) и P2(x2,y2,z2), имеет вид:

$$P(t) = P1 + (P2 - P1) \cdot t.$$

Минимальное расстояние от точки центра сферы P0(x0,y0,z0) до луча равно:

$$d^2 = (x\text{-}x0)^2 + (y\text{-}y0)^2 + (z\text{-}z0)^2$$

Этому соответствует значение t:

$$t = \frac{(x2 - x1)\cdot(x1 - x0) + (y2 - y1)\cdot(y1 - y0) + (z2 - z1)\cdot(z1 - z0)}{(x2 - x1)^2 + (y2 - y1)^2 + (z2 - z1)^2}.$$

Если d2 > R2, то луч не пересекает объекты, заключенные в оболочку.

Дальнейшее сокращение расчетов пересечений основывается на использовании групп пространственно связанных объектов. Каждая такая группа окружается общей оболочкой. Получается иерархическая последовательность оболочек, вложенная в общую оболочку для всей сцены. Если луч не пересекает какую-либо оболочку, то из рассмотрения исключаются все оболочки, вложенные в нее и, следовательно, объекты. Если же луч пересекает некоторую оболочку, то рекурсивно анализируются все оболочки, вложенные в нее.

Наряду с вложенными оболочками для сокращения расчетов пересечений используется отложенное вычисление пересечений с объектами. Если обнаруживается, что объект пересекается лучом, то он заносится в специальный список пересеченных. После завершения обработки всех объектов сцены объекты, попавшие в список пересеченных, упорядочиваются по глубине. Заведомо не-

видимые объекты отбрасываются, а для оставшихся выполняется расчет пересечений, и отображается точка пересечения наиболее близкая к наблюдателю.

Дополнительное сокращение объема вычислений может достигаться отбрасыванием нелицевых граней, учетов связности строк растрового разложения и т.д.

Для сокращения времени вычислений собственно пересечений предложено достаточно много алгоритмов, упрощающих вычисления для определенной формы задания поверхностей.

СПИСОК ЛИТЕРАТУРЫ

1. Компьютерная графика. Математический аппарат: метод. указания к лабораторным работам / сост. Ю.И. Никифоров; Иван. гос. хим.-технол. ун-т.-Иваново, 2007.-48с.
2. Роджерс,Д. Математические основы машинной графики/ Д. Роджерс, Дж. Адамс; пер. с англ.-М.:Машиностроение,2000.-240с.
3. Вельтмандер,П.В. Основные алгоритмы компьютерной графики. В 3-х кн. Кн.2. Машинная графика: учеб. пособие / П.В.Вельтмандер; Новосибирский гос. ун-т. 1997 /http://ermak.cs.nstu.ru/kg_rivs/kg02.htm.

Содержание

Редактор В.Л. Родичева
Подписано в печать 26.12.2011. Формат 60x84 $^{1}/_{16}$. Бумага писчая.

Усл.печ.л.2,79. Уч.-изд.л. 3,10. Тираж 50 экз. Заказ

Ивановский государственный химико-технологический университет

Отпечатано на полиграфическом оборудовании

кафедры экономики и финансов ИГХТУ

153000, г.Иваново, пр. Ф. Энгельса, 7