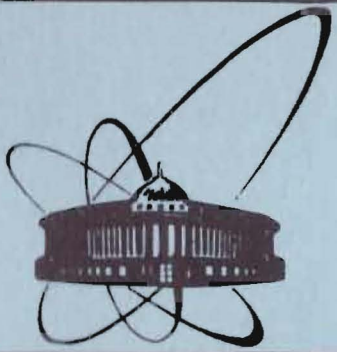


объединенный
институт
ядерных
исследований
дубна

Р11-85-314

А.Баер, В.И.Приходько, К.Фогт

АЛГОРИТМЫ ЗАПОЛНЕНИЯ ДЛЯ ДИСПЛЕЕВ РАСТРОВОГО ТИПА

Направлено в журнал "Rechentechnik/Datenverarbeitung"
/ГДР/

1985

ВВЕДЕНИЕ

Машинная графика является важным средством организации диалога человек-ЭВМ и представления графической информации. Прогресс микроэлектроники и связанное с этим снижение стоимости электронных компонентов в последние годы привели к более широкому применению растровых дисплеев в системах машинной графики.

Одним из основных преимуществ цветных растровых дисплеев является возможность наглядного представления поверхностей графических объектов путем заполнения /раскрашивания/ поверхностей в пределах ограничивающих их контуров, что особенно важно при решении задач проектирования с помощью ЭВМ (CAD-Computer Aided Design) в различных областях науки и техники.

При создании в Лаборатории вычислительной техники и автоматизации ОИЯИ цветного растрового дисплея с высокой разрешающей способностью потребовалось выбрать и реализовать подходящий алгоритм заполнения. С этой целью были проведены анализ и сравнение основных характеристик известных к настоящему времени алгоритмов, а также разработаны конкретные версии алгоритмов для заполнения многоугольников и конических сечений.

АЛГОРИТМЫ ЗАПОЛНЕНИЯ

По применяемому принципу алгоритмы заполнения могут быть разделены на три большие группы: алгоритмы "связности" (Connectivity), "проверки четности" (Parity Check) и "контурные" (Edge). В основе алгоритмов первой группы лежит принцип опроса соседних элементов изображения (pixel) с целью выявления их принадлежности к контуру. Алгоритмы второй группы определяют число точек пересечения растровой строки от данного элемента до края экрана с контуром подлежащего заполнению объекта. В зависимости от того, является это число нечетным или четным, элемент объявляется внутренним или нет. Алгоритмы, как правило, работают быстрее, чем алгоритмы "связности", но при их использовании необходимо вводить дополнительные тесты для распознавания касательных и правильного заполнения объектов.

"Контурные" алгоритмы исходят скорее из контура объекта, чем из отдельных элементов изображения. Элементы, лежащие между двумя краями, маркируются и выдаются как внутренние. Алгоритмы этой группы отличаются малыми временами обработки, но требуют

специальных мер при работе в локальных экстремумах У /проблема касательных/.

1.1. Алгоритмы связности

Seed Fill (SF)

Исходя из одной начальной точки внутри заполняемого объекта, SF-алгоритм проверяет все соседние точки на принадлежность их элементам контура [1]. В зависимости от этого исходная точка объявляется внутренней или нет. Затем одна из проверяемых точек выбирается как новая исходная точка и проверка продолжается. Чтобы обеспечить заполнение сложных фигур без пробелов, остальные проверяемые точки запоминаются в стеке и используются при необходимости как новые исходные точки. Процесс заполнения заканчивается, когда все точки проверены, т.е. когда вокруг исходной точки расположены только уже отмеченные элементы или элементы контура, а стек пустой. Данный алгоритм позволяет полностью раскрашивать любой контур, находящийся в памяти изображения.

Так как SF-алгоритм проверяет отдельные элементы изображения, то он имеет малое быстродействие. Кроме того, для данного алгоритма требуется задавать первую исходную точку, что в неинтерактивном режиме работы может вызывать определенные проблемы. Такая исходная точка должна задаваться для каждой области заполнения отдельно. Особенно трудоемкой такая операция является при заполнении самопересекающихся объектов, для которых возникают внешне разные области заполнения /см.рис.1/.

Class Fill (CF)

CF-алгоритм последовательно проверяет элементы каждой растровой строки и в зависимости от левого и верхнего элементов-соседей относит их к определенному классу [2]. Распознаются существующие эквивалентные соотношения, а классы разделяются на группы, которые характеризуют принадлежность к отдельным областям заполнения.

Этот алгоритм без пробелов заполняет все вогнутые и выпуклые объекты. Так как количество классов ограничивается лишь глубиной памяти изображения, то имеется возможность определять несколько областей заполнения. Так же как и при использовании SF-алгоритма, в данном случае проверяется каждый элемент в отдельности, поэтому количество обращений к памяти изображения очень велико, что отрицательно сказывается на времени обработки.

1.2. Алгоритмы проверки четности

Эти алгоритмы объявляют данный элемент внутренним, если линия от него до края экрана пересекает границу подлежащего заполнению объекта нечетное количество раз. В работах [1,3] представлено несколько алгоритмов проверки четности. Самые простые из них работают безошибочно лишь для некоторых типов объектов, а самый сложный позволяет с помощью ряда дополнительных тестов заполнять без пробелов любые объекты.

Применение алгоритмов, использующих метод проверки четности, всегда влечет за собой две принципиальные проблемы. Во-первых, алгоритм в самой простой своей форме не отличает два следующих друг за другом контурных элемента от элементов, ограничивающих "внутреннюю растровую линию". Во-вторых, наличие локальных экстремумов приводит к ошибочному заполнению.

1.3. Контурные алгоритмы

Ordered Edge List (OEL)

Классическим методом заполнения многоугольников является определение сегментов растровых линий из таблицы, содержащей все ребра многоугольника [1]. Алгоритм формирует таблицу тех ребер, которые пересекают текущую растровую линию, и с помощью разностного метода вычисляет точку пересечения "активного" ребра с этой растровой линией. Эти точки упорядочиваются и затем используются для нахождения сегментов линий. Для упрощения процедуры упорядочивания ребер и уменьшения времени обработки многоугольники предварительно могут быть разложены на треугольники и/или трапеции.

Существенным преимуществом OEL является тот факт, что каждый активный элемент обрабатывается только один раз. Это уменьшает количество обращений к памяти изображения. Кроме того, OEL может эффективно применяться для алгоритмов образования оттенков и Antialiasing-алгоритмов, т.к. начальная и конечная точки сегмента линии известны до выдачи на экран, а интенсивность соседних точек интерполируется.

Edge Fill (EF)

EF-алгоритм исходит из математического контура объекта, подлежащего заполнению. При обходе этого контура осуществляется инвертирование всех элементов, лежащих справа от него [4]. После завершения обхода контура все его внутренние элементы будут маркированными. В [5] под названием "Fence Fill" приведена модификация этого алгоритма, в которой инвертирование ограничивается X-координатой крайней правой точки заполняемого объекта.

EF-алгоритм позволяет безошибочно заполнять любые замкнутые фигуры, в том числе фигуры с несколькими областями заполнения и с "дырками". Время работы этого алгоритма, однако, сильно зависит от количества вертикалей объекта, так как с ростом числа вертикалей растет и число операций инвертирования.

Edge Flag (EFL)

EFL-алгоритм аналогичен EF-алгоритму, но в нем устранен существенный недостаток последнего - многократное инвертирование. Это достигается тем, что в рабочей памяти инвертируется лишь первый элемент, лежащий справа от математического контура[4]. Рабочая память с помощью простого механизма считывания, работающего по принципу проверки четности, копируется в память изображения и тем самым завершается процесс заполнения.

Этот алгоритм безошибочно заполняет любые объекты. Поскольку для определения внутренней области используется только контур, то генерирование рабочей памяти /т.е. определение объекта/ осуществляется за короткое время и с помощью простых целочисленных операций. Некоторые проблемы могут возникнуть при заполнении очень острых углов,т.к. из-за двойного инвертирования одного и того же элемента появляются "закругления".

Pairwise Fill (PF)

PF-алгоритм генерирует контур заполняемого объекта и запоминает в рабочей памяти на каждой строке растра первую X-координату, принадлежащую контуру[5]. Когда появляется вторая X-координата, то заполняется растровая линия между обеими X-координатами, а X-координата в рабочей памяти заменяется признаком, который разрешает формирование нового сегмента линии для данной строки.

Данный алгоритм без пробелов заполняет все выпуклые объекты, однако при заполнении некоторых вогнутых объектов /см.рис.2/ время его работы увеличивается в несколько раз. Как и OEL, этот алгоритм хорошо подходит для нанесения оттенков и реализации Antialiasing.

2. СРАВНИТЕЛЬНАЯ ОЦЕНКА АЛГОРИТМОВ ЗАПОЛНЕНИЯ

Сравнение проводится по следующим критериям:

а/ корректность заполнения,
б/ время обработки,
в/ требуемый объем памяти,
г/ возможность аппаратной реализации.

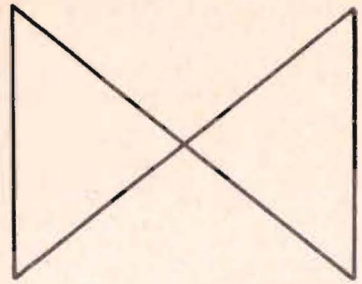


Рис.1. Самопересекающийся многоугольник.

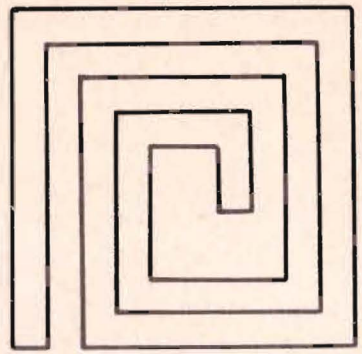


Рис.2. "Патологический" многоугольник.

а/. Все рассмотренные алгоритмы обеспечивают правильное заполнение объектов при решении соответствующих проблем, к числу которых, в первую очередь, относятся:
- определение исходной точки для SF;
- распознавание связанных контурных элементов в Parity Check(PC);
- учет локальных экстремумов во всех алгоритмах связности и проверки четности.

Алгоритм OEL подходит только для многоугольников, а PF не справляется с определенными типами фигур /рис.2/.

б/. Исследование быстродействия основных алгоритмов заполнения, проведенное в работе[4], дало следующие результаты:
- алгоритм OEL из-за незначительного количества операций ввода/вывода имеет, как правило, максимальное быстродействие. SF и PC-алгоритмы в 6-8 раз медленнее, так как они проверяют отдельные элементы заполняемого объекта;
- быстродействие EF сильно зависит от количества вертикалей в изображении объекта и лишь при небольшом их количестве скорости EF и OEL сравнимы между собой;
- при заполнении относительно простых фигур алгоритмы EFL и OEL имеют примерно одинаковое быстродействие, но EFL явно превосходит по скорости OEL при обработке сложных изображений. Это же справедливо и для PF, который в этом отношении аналогичен EFL-алгоритму;
- алгоритм CF по времени обработки сравним с SF- и PC-алгоритмами.

в/. Важным параметром в оценке алгоритмов заполнения является объем памяти, необходимой для их микропрограммной реализации. При использовании SF-алгоритма требуется очень большой стек для безошибочного заполнения сложных объектов. CF- и OEL-алгоритмы

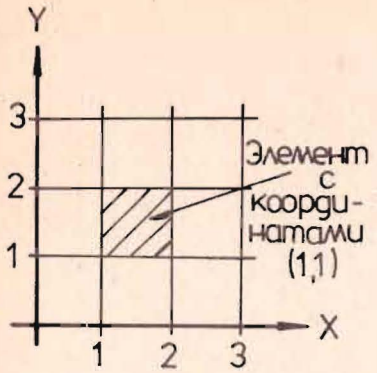


Рис.3. Система граничных координат.

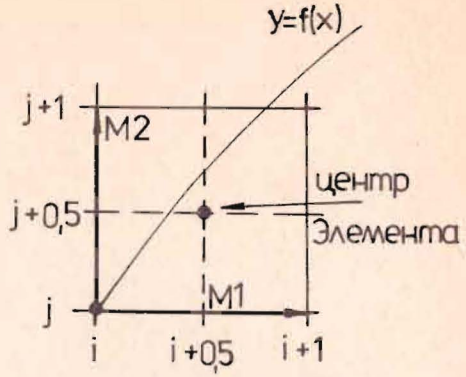


Рис.4. Условие проверки для кривой в первом квадранте.

работают с обширными списками данных,для хранения и обработки которых необходима большая память. Из рассмотренных выше алгоритмов наибольший объем памяти требуется для PC. Это обусловлено тем, что с целью исключения операции сортировки PC-алгоритм использует сложные программы проверки элементов.

В EF- и EFL-алгоритмах для самой программы требуется минимальная память, а для данных она вообще не нужна.

PF-алгоритм также обходится малым объемом памяти для программ, однако память для хранения X-координат в определенных условиях может иметь большие размеры.

г/. С целью повышения скорости работы алгоритмов было исследовано, насколько возможна их аппаратурная поддержка. Анализ показал, что такая возможность имеется только для контурных алгоритмов.

Другие алгоритмы содержат настолько сложные тесты и объемную обработку данных, что их аппаратная поддержка вряд ли целесообразна.

В контурных алгоритмах возможно применение генераторов сегмента линии(OEL, EF, PF), генераторов контура (EF, EFL) и аппаратных средств для реализации считывания по принципу проверки четности (EFL).

3. ГЕНЕРАЦИЯ ФЛАГОВ КОНТУРА В EFL-АЛГОРИТМЕ

Анализ и сравнение алгоритмов заполнения выявил однозначные преимущества EFL-алгоритма. Главной проблемой при его реализации является эффективная генерация маркировок /флагов/ в рабо-

чей памяти. Эта память реализована в виде области памяти с глубиной в 1 бит и получила название "рабочая плата".

3.1. Многоугольники

Заполнение многоугольника означает окрашивание всех элементов, центры которых находятся внутри контура. Однако такой контур имеет нулевые размеры и плохо определяется с помощью элементов, имеющих конечные размеры. По этой причине было принято, что каждый элемент имеет координаты своего левого нижнего угла /см.рис.3/. Полученная таким образом новая система координат - система граничных координат - используется как основа для генерации флагов на рабочей плате. Генерация осуществляется с помощью модифицированного алгоритма Брезенхама [6].

Прямая в первом квадранте в системе граничных координат описывается следующим уравнением:

$$2u \cdot Y = 2v \cdot X. \qquad /1/$$

Фактор 2 вводится для упрощения вычислений с целыми числами. Поскольку маркироваться /инвертироваться/ должен каждый элемент, центр которого лежит правее этой прямой, то из рис.4 получается условие для проверки данной точки:

$$r = 2u(j + 0{,}5) - 2v(i + 0{,}5). \qquad /2/$$

Для $r \geq 0$ /проверяемая точка лежит левее прямой/ требуется произвести M1-шаг /X+1,Y/ и новую проверку. При $r < 0$ /проверяемая точка лежит справа от прямой/ маркируется соответствующий элемент и выполняется один M2-шаг /X,Y+1/. С учетом этих условий из /2/ получается следующее рекуррентное соотношение:

$$r_{n+1} = r_n + 2u(j_{n+1} - j_n) - 2v(i_{n+1} - i_n), \qquad /3/$$

где $j_{n+1} - j_n = 1$ для M2, иначе 0,
и $i_{n+1} - i_n = 1$ для M1, иначе 0.

Если исходная точка лежит на прямой, а это можно предполагать, то ее координаты X_H и Y_H также удовлетворяют уравнению прямой:

$$Y_H = \frac{v}{u} \cdot X_H. \qquad /4/$$

С учетом /4/ из уравнения /1/ получаем начальную величину для r:

$$r_H = u - v. \qquad /5/$$

На рис.5 представлен основанный на соотношениях /3/ и /5/ алгоритм генерирования флагов контура для прямой в первом квадранте. Для второго квадранта M1-шаг модифицируется таким образом, что текущая X-координата не увеличивается, а уменьшается на 1. Кроме того, поскольку u принимает отрицательные значения, то вычисления проводятся с абсолютной величиной u. Хотя

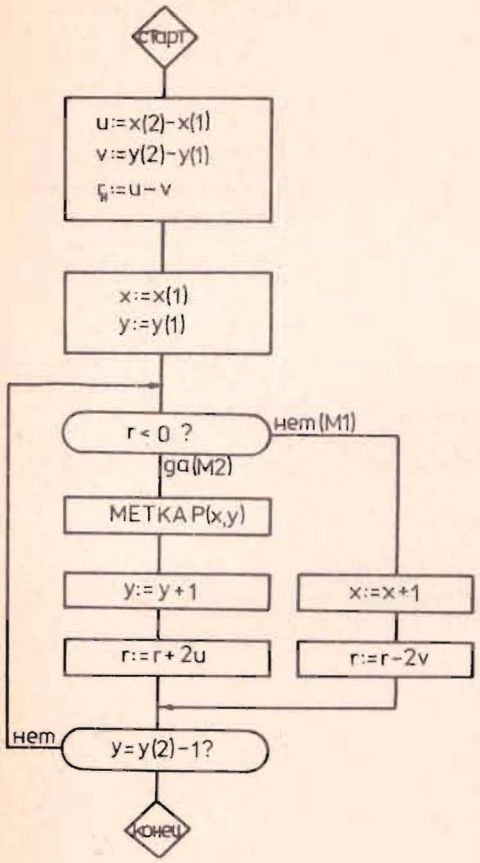


Рис.5. Алгоритм генерации флагов в первом квадранте.

в этом случае проверка проводится на прямой, отраженной осью У, но из-за уменьшения X-координаты маркируется нужная прямая. Для третьего и четвертого квадрантов необходимо или отражать прямую на оси X, или же поменять местами начальную и конечную точки, чтобы опять получить прямую во втором или первом квадрантах.

Авторами разработана микропрограмма, реализующая этот алгоритм с помощью микропроцессорных секций серии КР1804. Как показывает опыт [7], для значительного ускорения работы алгоритма некоторые участки микропрограммы, требующие больших затрат времени, целесообразно реализовать схемным путем. На рис.6 приведена схема, которая вычисляет координаты точек контура в пределах определенного квадранта. Время вычисления координат одной точки для этой схемы составляет примерно 100 нс.

3.2. Конические сечения

Конические сечения являются часто встречающимися примитивами вывода графической информации. Исходя из этого был разработан алгоритм заполнения, который на примере эллипса показывает применение EFL-техники.

Питтвей в [8] показал, что эллипсы достаточно хорошо аппроксимируются своими касательными в отдельных точках.

Общее уравнение конического сечения имеет вид:

$$\alpha Y^2 + \beta X^2 + 2\gamma XY + 2uY - 2vX = k . \qquad /6/$$

Фактор 2 вводится для облегчения целочисленных вычислений. Знак минус перед последним членом выбирается для того, чтобы уравнение в частном случае α, β, γ, k = 0 соответствовало уравнению /1/.

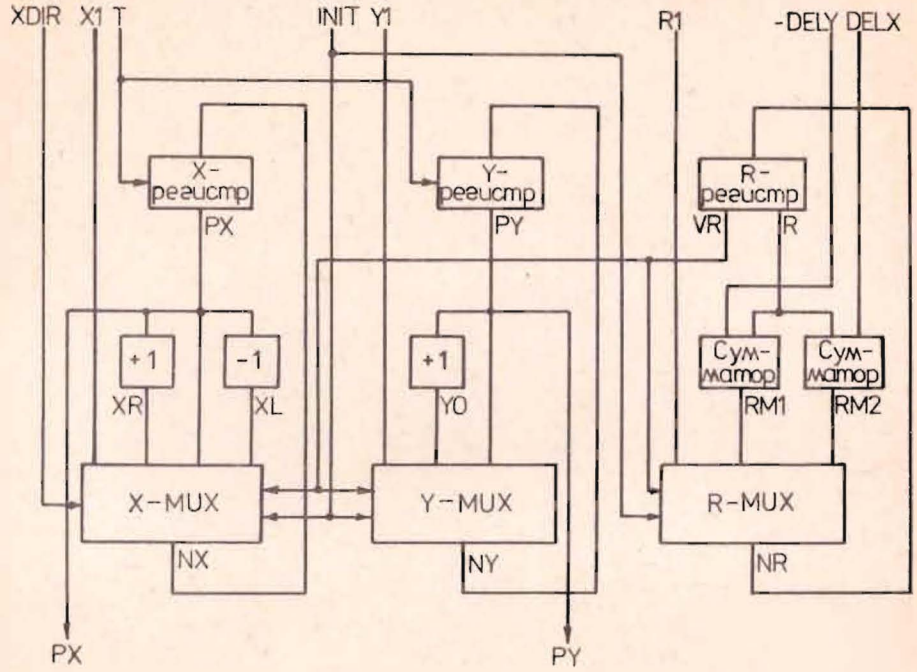


Рис.6. Схема генератора флагов. X1, Y1 - начальные координаты; PX, PY - текущие координаты; XR, XL - X-координата точки, расположенной справа или слева от текущей точки; YO - Y-координата точки, расположенной над текущей точкой; R - результат проверки для управления циклом; RM1, RM2 - результаты проверки; VR - знак результата проверки; DELX, DELY - приращения результата проверки; NX, NY, NR - следующие координаты и новый результат проверки; XDIR - знак; T - такт цикла; INIT - инициализация; таблица управления мультиплексором:

XDIR	INIT		NX	NY	NR
0	0	0	XR	PY	RMI
0	0	1	PX	YO	RM2
0	1	0	X1	Y1	R1
0	1	1	X1	Y1	R1
1	0	0	XL	YO	RM1
1	0	1	PX	YO	RM2
1	1	0	X1	Y1	R1
1	1	1	X1	Y1	R1

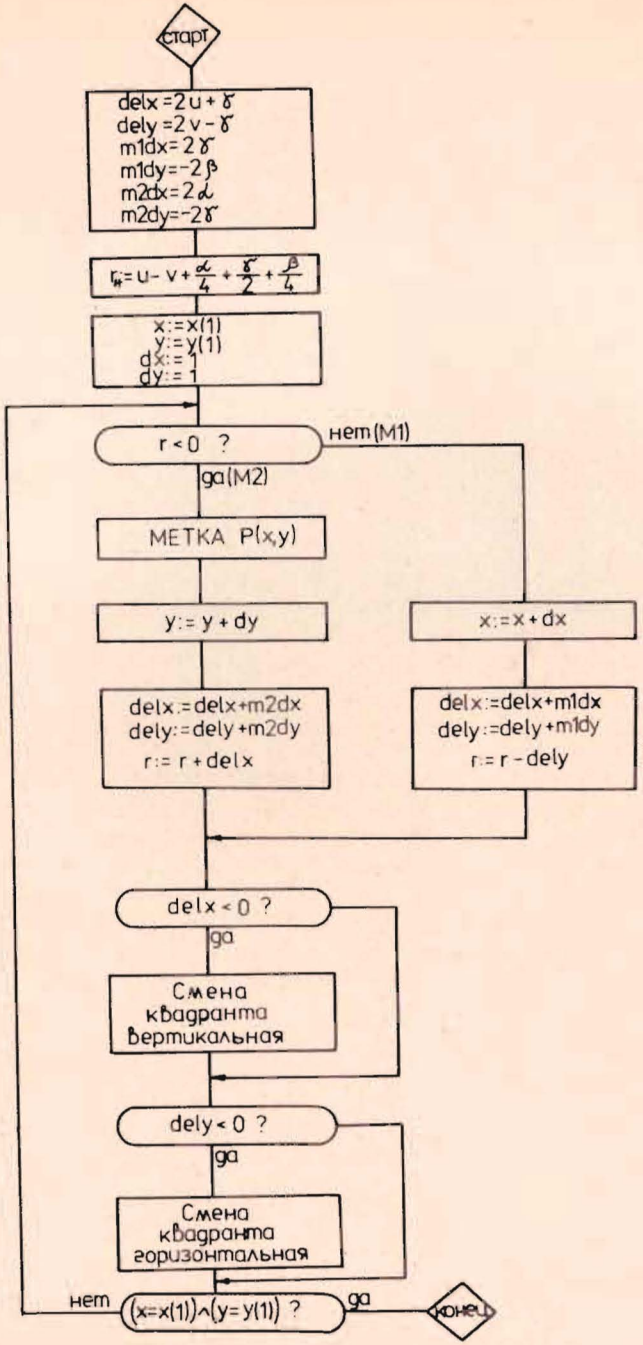


Рис.7. Алгоритм генерации флагов для конических сечений.

В вычислениях предполагается, что $k = 0$ /начальная точка лежит на сечении/. Этого всегда можно добиться соответствующими преобразованиями. Кроме того, предполагается, что касательная к начальной точке лежит в первом квадранте $(u, v \geq 0)$. Из рис.4 видно, что в критической точке /центр элемента/ для кривой в первом квадранте можно записать:

$$X(Y = j + 0{,}5) = i + 0{,}5. \qquad /7/$$

Отсюда получаем условие проверки:

$$r = X(Y = j + 0{,}5) - i - 0{,}5. \qquad /8/$$

Для $r \geq 0$ проверяемая точка /$i + 0{,}5$; $j + 0{,}5$/ лежит левее кривой; в этом случае производится M1-шаг. При $r < 0$ проверяемая точка лежит правее кривой; элемент (X, Y) маркируется и выполняется M2-шаг.

Выделение X из уравнения конусного сечения /6/ для $k = 0$ дает

$$X = \frac{1}{\beta}\left[v - \gamma \cdot Y + \sqrt{(\gamma \cdot Y - v)^2 - \beta(\alpha Y^2 + 2uY)}\right]. \qquad /9/$$

Положительный корень используется для того, чтобы $X = 0$ при $Y = 0$ и $\lim_{\beta \to 0} X$ соответствовал решению уравнения /6/ для частного случая $\beta = 0$. Подстановка /9/ в формулу /8/ дает следующее условие проверки:

$$r = [\alpha(j + 0{,}5) + 2u + 2\gamma \cdot i + \gamma](j + 0{,}5) - v(2i + 1) + \beta(i + 0{,}5)^2. \qquad /10/$$

Отсюда находится начальная величина для $r(i, j = 0)$:

$$r_н = \frac{\alpha}{4} + u + \frac{\gamma}{2} - v + \frac{\beta}{4}. \qquad /11/$$

Из /10/ получается рекуррентное соотношение

$$r_{n+1} = r_n + (j_{n+1} - j_n)[\alpha(j_{n+1} + j_n + 1) + 2u + \gamma] - (i_{n+1} - i_n)[2v - \gamma - \beta(i_{n+1} + i_n + 1)] + 2\gamma(i_{n+1} \cdot j_{n+1} - i_n j_n). \qquad /12/$$

В первом квадранте действует условие

$$j_{n+1} = j_n + 1 \qquad /13/$$

$$i_{n+1} = i_n + 1. \qquad /14/$$

Учет этих соотношений приводит к окончательной рекуррентной формуле

$$r_{n+1} = r_n + (j_{n+1} - j_n)[2\alpha j_n + 2u + \gamma(2i_n + 1)] - (i_{n+1} - i_n)[2v - \gamma(2j_n + 1) - 2\beta i_n], \qquad /15/$$

где $j_{n+1} - j_n = 1$ для M2, иначе 0,
$i_{n+1} - i_n = 1$ для M1, иначе 0.

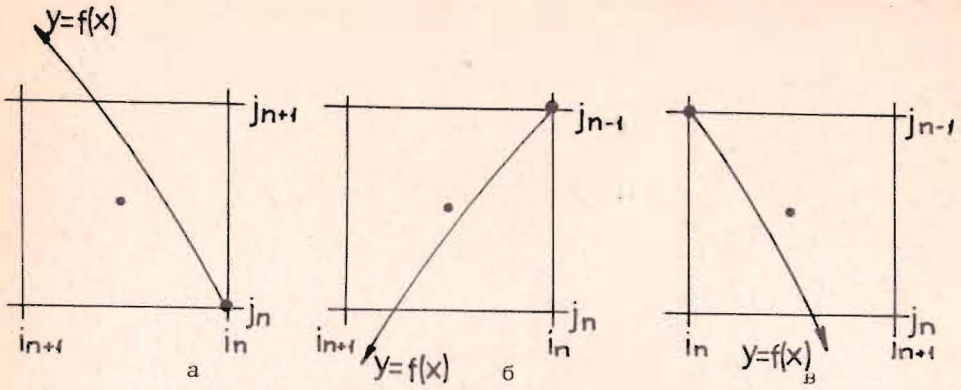


Рис.8. Представление кривых во втором /а/, третьем /б/ и четвертом /в/ квадрантах.

Структура уравнения /15/ явно подсказывает, что можно применять алгоритм, похожий на алгоритм генерации флагов для многоугольников. Такой алгоритм показан на рис.7. При этом для упрощения введены следующие вспомогательные переменные:

$delX = 2\alpha j + 2u + \gamma(2i + 1)$, $delY = 2v - 2\beta i - \gamma(2j + 1)$,

$m_1dX = 2\gamma$, $m_1dY = -2\beta$, $m_2dX = 2\alpha$, $m_2dY = -2\gamma$.

В ходе генерации может возникнуть такой случай, когда кривая из первого квадранта переходит во второй или четвертый. В этом случае требуется горизонтальная ($delY < 0$) или вертикальная ($delX < 0$) смена квадрантов. Также возможно, что касательная в начальной точке лежит не в первом квадранте. Вывод рекуррентных формул для остальных квадрантов осуществляется аналогичным образом. При этом, однако, следует учитывать соотношения, показанные на рис.8 /исходная точка - последующая точка/, и то обстоятельство, что знак r теряется при возведении в квадрат /при этом требуется проверка с помощью r_H, а иногда и отрицание r/. При вычислениях в третьем и четвертом квадрантах надо исходить из точки, находящейся над текущей точкой, чтобы EFL-алгоритм мог корректно работать /см.рис.3 б,в/. Полученные таким образом величины для переменных, встречающихся в алгоритме, приведены в таблице. По этой таблице производятся необходимые начальные установки для отдельных квадрантов. При смене квадрантов производятся следующие изменения:

1. Горизонтальная схема квадрантов:

$dX = -dX$, $m_1dY = -m_1dY$, $m_2dX = -m_2dX$, $delX = -delX + m_1dX$, $r = -r - delY$.

2. Вертикальная схема квадрантов:

$dY = -dY$, $m_1dY = -m_1dY$, $m_2dX = -m_2dX$, $delY = -delY + m_2dY$, $r = -r + delX$.

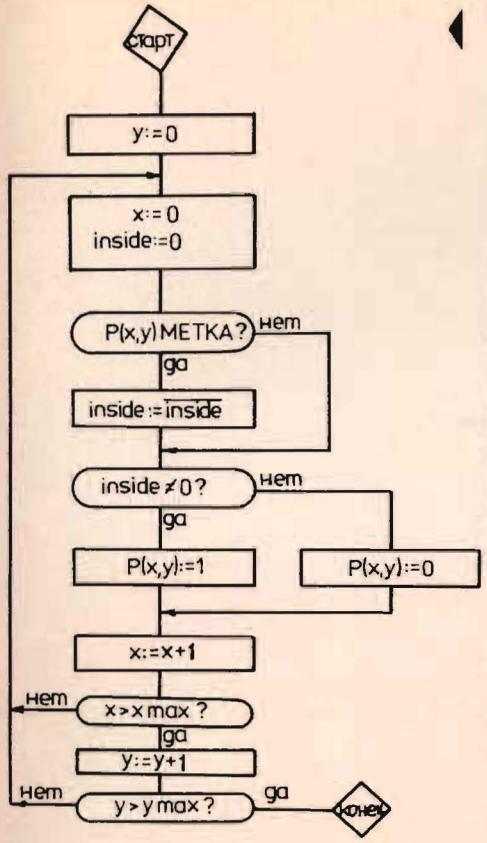


Рис.9. Алгоритм считывания флагов.

3. Коррекция сдвинутой начальной точки в третьем и четвертом квадрантах: $Y = Y + dY$.

Таким образом, с помощью данного алгоритма при соответствующей инициализации могут быть заполнены любые конусные сечения.

Следует подчеркнуть некоторые отличия рассмотренного выше алгоритма от алгоритма Питтвея, которые заключаются в выборе системы координат элемента /см.п.3.1./, а также в процедуре инициализации и генерации флагов /у Питтвея при соответствующей инициализации флаги генерируются по октантам, а в данном алгоритме - по квадрантам/, что, естественно, приводит к другим рекуррентным соотношениям для вычисления r.

4. СЧИТЫВАНИЕ ФЛАГОВ ИЗ РАБОЧЕЙ ПАМЯТИ

После подготовки флагов в в рабочей памяти запускается второй этап EFL-алгоритма - считывание в память изображения. Считывание флагов осуществляется одинаковым образом при заполнении как многоугольников, так и конических сечений. Для этого применяется алгоритм, работающий по принципу проверки четности /рис.9/.

Данный алгоритм считывания реализуется простой электронной схемой /рис.10/, использование которой существенно ускоряет работу EFL-алгоритма и позволяет за разумное время заполнять даже очень сложные объекты.

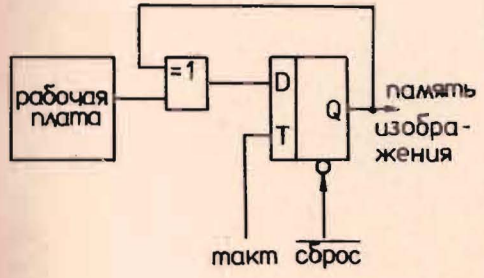
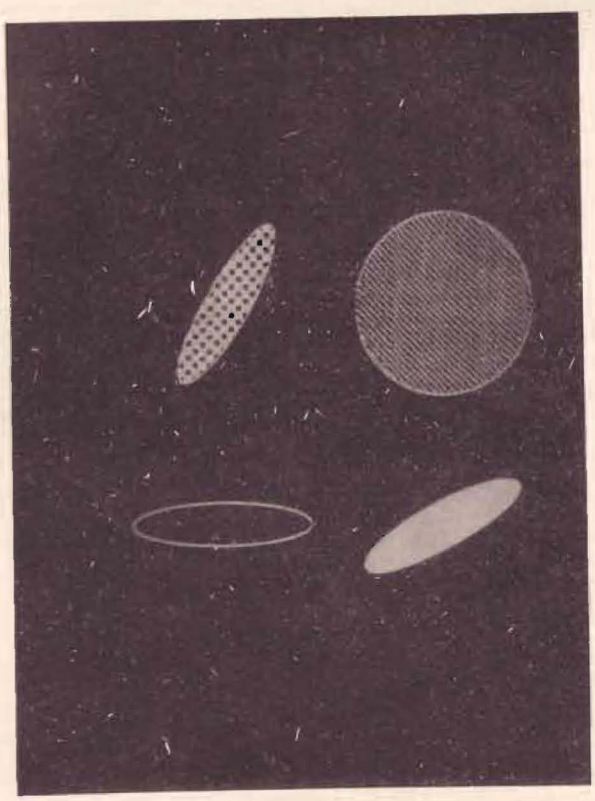


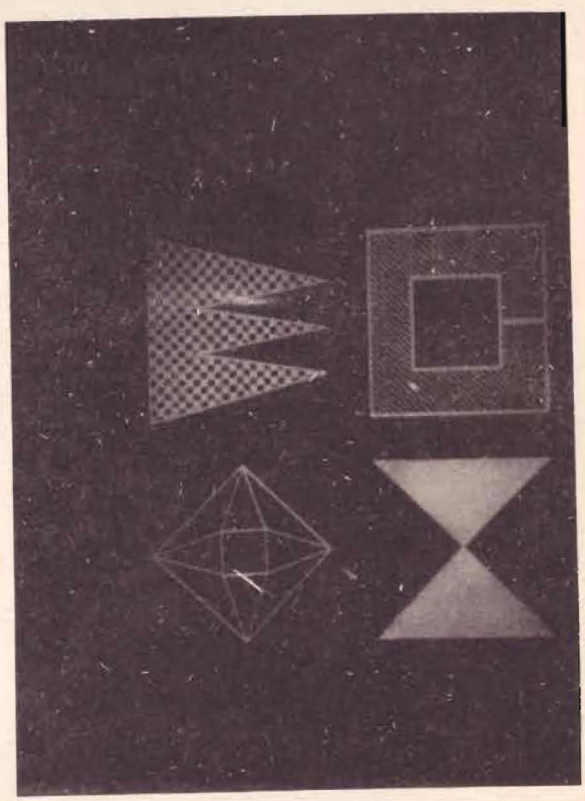
Рис.10. Схема считывания флагов для EFL-алгоритма.

Таблица

квадрант	dx	dy	m1dx	m1dy	m2dx	m2dy	del x	dely	r
I	+1	+1	2γ	-2β	2α	-2γ	$2\alpha j+2u+\gamma(2i+1)$	$2v-2\beta i-\gamma(2j+1)$	$[\alpha(j+0{,}5)+2u+\gamma(2i+1)](j+0{,}5)$ $-v(2i+1)$ $+\beta(i+\frac{1}{2})^2$
II	-1	+1	2γ	2β	-2α	-2γ	$-2\alpha j-2u-\gamma(2i-1)$	$2v-2\beta i-\gamma(2j+1)$	$[-\alpha(j+0{,}5)-2u-\gamma(2i-1)](j+0{,}5)$ $+v(2i-1)$ $-\beta(i-\frac{1}{2})^2$
III	-1	-1	2γ	-2β	2α	-2γ	$-2\alpha j-2u-\gamma(2i-1)$	$-2v+2\beta i+\gamma(2j-1)$	$[\alpha(j-0{,}5)+2u+\gamma(2i-1)](j-0{,}5)$ $-v(2i-1)$ $+\beta(i-\frac{1}{2})^2$
IV	+1	-1	2γ	2β	-2α	-2γ	$2\alpha j+2u+\gamma(2i+1)$	$-2v+2\beta i+\gamma(2j-1)$	$[-\alpha(j-0{,}5)-2u-\gamma(2i+1)](j-0{,}5)$ $+v(2i+1)$ $-\beta(i+\frac{1}{2})^2$



б



а

Рис.11. Примеры заполнения многоугольников /а/ и конических сечений /б/.

ЗАКЛЮЧЕНИЕ

В результате исследований для разрабатываемого в ЛВТА ОИЯИ цветного растрового дисплея был выбран алгоритм заполнения Edge-Flag, который превосходит другие алгоритмы по совокупности основных характеристик/правильность заполнения, время обработки, требуемый объем памяти/ и, что особенно важно, допускает сравнительно простую программно-аппаратную реализацию.

Для этого алгоритма разработана микропрограмма генерации флагов при заполнении многоугольников /рис.11а/ и предложена схемная реализация отдельных ее участков.

Были разработаны также алгоритм и соответствующая микропрограмма для заполнения конических сечений с использованием EFL-техники /рис.11б/.

С помощью указанных алгоритмов можно заполнять фигуры произвольной формы путем их разложения на многоугольники и конические сечения.

ЛИТЕРАТУРА

1. Pavlidis T., Algorithms for Graphics and Image Processing, Springer-Verlag W.Berlin, 1982.
2. Distante A., Veneziani N., A Two-Pass Filling Algorithm for Raster Graphics, Computer Graphics and Image Processing, 1982, 20, 3.
3. Pavlidis T., Contour Fillings in Raster-Graphics, Computer-Graphics - SIGGRAPH 81 Conference Proceedings. New York, 1981, 15, 3.
4. Ackland B.D., Weste N.H. IEEE Transaction on Computers, 1981, C-30, 1.
5. Dunlavey M.R. ACY Transaction on Graphics 4, 1983, 2.
6. Bresenham J.E. IBM System Journal, 1975, 4.1, p.25.
7. Лайх Х., Приходько В.И., Фогт К. ОИЯИ, Р11-83-909, Дубна, 1983.
8. Pitteway M.L.V. The Computer Journal, 1967, 10, p.282.

Рукопись поступила в издательский отдел
30 апреля 1985 года

Баер А., Приходько В.И., Фогт К.

Р11-85-314

Алгоритмы заполнения для дисплеев растрового типа

На основе анализа различных алгоритмов заполнения контуров был выбран алгоритм Edge Flag как наиболее эффективный с точки зрения микропрограммной и аппаратной реализации в графическом процессоре цветного растрового дисплея, разрабатываемого в ЛВТА ОИЯИ. Описаны разработанные авторами конкретные версии алгоритмов заполнения многоугольников и конических сечений, а также микропрограммы и электронные схемы для их реализации.

Работа выполнена в Лаборатории вычислительной техники и автоматизации ОИЯИ.

Препринт Объединенного института ядерных исследований. Дубна 1985

Перевод О.С.Виноградовой

Baer A., Prikhodko V.I., Voigt K.

P11-85-314

The Fill Algorithms for Raster Scan Displays

Based on the analysis of various algorithms for the countour filling the Edge Flag algorithm has been selected as the most effective one for a microprogram and hardware implementation in the graphics processor of a high performance colour raster scan display designed at the JINR. The concrete versions of algorithms for filling polygons and conics as well their microprogram and hardware realization offered by authors are described.

The investigation has been performed at the Laboratory of Computing Techniques and Automation, JINR.

Preprint of the Joint Institute for Nuclear Research. Dubna 1985