2. ЛИНЕЙНЫЕ ПРЕОБРАЗОВАНИЯ, СЛЕД ПЕРА, ЭКРАНИРОВАНИЕ, ШТРИХОВКА

В главе 1 мы рассматривали построение графических объектов, которые чаще бывают элементами изображений, чем законченными изображениями. Средства системы Графор, рассматриваемые в этой главе, достаточно разнородны. Их объединяет то, что с их помощью упрощается компоновка сложных многоэлементных изображений, хотя они могут оказаться полезными и в некоторых других случаях.

На практике мы постоянно сталкиваемся с изображениями, которые содержат много компонент (подрисунков), отличающихся друг от друга только местоположением, ориентацией, масштабом, т. е. обладающих некоторым геометрическим сходством. В таких случаях бывает выгодно иметь одно описание рисунка, т. е. одну программу, рисующую эталонную компоненту, и получать из нее все остальные путем соответствующего преобразования плоскости рисунка. Чаще других для этой цели используется аппарат *линейных преобразований*, хорошо разработанный математически и легко реализуемый программно.

Однако в некоторых случаях желаемое преобразование изображения не является линейным. Поэтому в системе Графор предусмотрены средства для запоминания информации о траектории пера в виде так называемого *следа пера*. След пера представляет собой последовательность координат точек, принадлежащих траектории пера. Занесенные в память координаты можно переработать и получить тем самым произвольное преобразование изображения. След пера может оказаться полезным еще и для того, чтобы составить границы участков, которые должны быть заэкранированы или заштрихованы, а также для того, чтобы сохранить в памяти картинку и рисовать ее в дальнейшем, не повторяя вычислений.

Еще один предмет, рассматриваемый в этой главе - *экранирование*, которое можно считать особым видом нелинейного преобразования. При задании одного или нескольких экранов - участков с кусочно-линейной границей - части генерируемого изображения, покрытые экраном, не рисуются.

Программы преобразования являются установочными, т. е. они лишь задают соответствующие преобразования, а выполняет их программа MOVE. Если задано линейное преобразование или указан режим регистрации следа пера, или, наконец, установлены экраны, то соответствующие действия будут применяться ко всем генерируемым в дальнейшем элементам изображения вплоть до отмены соответствующих установок.

Материал о *штриховке* отнесен к этой главе, поскольку границы заштрихованных участков задаются так же, как границы экранов.

2.1. Линейные преобразования

Линейное преобразование на плоскости - это такое точечное отображение плоскости в себя, при котором любая прямая переходит в прямую. Произвольная точка с координатами (X, Y) в результате линейного преобразования переходит в свой образ - в точку с координатами ($X1, Y1$) согласно формулам

$$X1=A*X+B*Y+C, \quad Y1=D*X+E*Y+F,$$

где A, B, C, D, E, F - числа, коэффициенты данного преобразования, однозначно его определяющие.

Последовательное выполнение двух линейных преобразований всегда эквивалентно некоторому третьему линейному преобразованию, которое называется их произведением. Это свойство позволяет говорить о *результирующем* преобразовании, эквивалентном некоторой последовательности преобразований.

Если перейти к *однородным координатам* точки (см., например, [15, 16]), то формулы линейного преобразования можно записать в матричном виде:

$$(X1, Y1, 1) = (X, Y, 1) \cdot \begin{bmatrix} A & D & 0 \\ B & E & 0 \\ C & F & 1 \end{bmatrix} = (X, Y, 1) \cdot M .$$

Тогда последовательное применение двух преобразований выглядит следующим образом:

$$(X2, Y2, 1) = (X1, Y1, 1) \cdot M2 = (X, Y, 1) \cdot M1 \cdot M2 = (X, Y, 1) \cdot M ,$$

где $M = M1 \cdot M2$ - матрица результирующего преобразования. В общем случае операция умножения матриц некоммутативна. А значит и два последовательно выполняемых линейных преобразования также, вообще говоря, некоммутативны.

Если значение определителя матрицы *M* отлично от нуля то преобразование называется *аффинным*. В отличие от общего линейного преобразования при аффинном преобразовании плоскость не может вырождаться в линию или точку. Аффинное преобразование переводит параллельные прямые в параллельные и всегда имеет обратное преобразование.
В подавляющем большинстве случаев на практике мы имеем дело именно с аффинными преобразованиями. Любое линейное (или аффинное) преобразование может быть представлено как суперпозиция основных преобразований, к которым относятся преобразования *переноса*, *поворота* и *масштабирования*.

2.1.1. Программы установки линейных преобразований. Можно считать, что текущее преобразование задается матрицей коэффициентов. Вначале, пока преобразования не задавались, текущее преобразование является тождественным, а матрица единичной. То есть, коэффициенты $A = E = 1$, а $B = C = D = F = 0$. Тогда любая операция установки преобразования сводится к умножению матрицы текущего преобразования на матрицу заданного преобразования. Таким образом, определяется результирующее преобразование. При выводе рисунка координаты всех точек умножаются на матрицу результирующего преобразования.

Для установки линейных преобразований в Графоре предусмотрены пять программ, с помощью которых можно задать любое из названных выше основных преобразований, а также определить произвольное линейное преобразование, указав его коэффициенты.

Программа TRANSL(DX,DY) задает параллельный перенос с вектором переноса (DX, DY).

Программа ROTATE(X,Y,PSI) задает поворот вокруг центра с координатами (X, Y) на угол PSI, измеряемый в градусах.

Программа PSCALE(X,Y,R) задает растяжение в |R| раз относительно центра (X, Y) и, кроме того, если R < 0, задает центральную симметрию.

Программа LSCALE(X1,Y1,X2,Y2,R) задает растяжение в |R| раз относительно оси, проходящей через точки (X1, Y1) и (X2, Y2), и, кроме того, если R < 0, задает осевую симметрию. Если точки (X1, Y1) и (X2, Y2) совпадают, то осью преобразования считается прямая, параллельная оси *X*.

Примечание. Очевидно, что при R = -1 две последние программы будут задавать преобразования симметрии без растяжений, а при |R| < 1 получим преобразование сжатия в 1/|R| раз.

Программа ATRAN2(A,B,C,D,E,F) задает произвольное линейное преобразование с коэффициентами A, B, C, D, E, F (см. приведенные выше формулы). Для умножения матриц используется служебная программа MTMPL.

Программа RESET отменяет накопленное результирующее преобразование, делая матрицу преобразования единичной. Эта программа параметров не имеет. В программе RESET имеется внутренний вход ATRST, выполняющий начальные установки в общие блоки аффинных преобразований.

2.1.2. Уровни вложенности линейных преобразований. Во многих случаях удобно строить рисунок как совокупность подрисунков, состоящих, в свою очередь, из подподрисунков и т. д. При этом подрисунок может оформляться как подпрограмма или фрагмент программы и описываться с использованием своих внутренних (локальных) преобразований. Поскольку

подрисунок является частью некоторого объемлющего рисунка на него должны также распространяться внешние (глобальные по отношению к нему) преобразования.

Рассмотрим простой пример. Рисунок А, кроме некоторых собственных графических элементов, включает подрисунки В и С. Тогда структура программы может быть такой:

```
_______________Па_____________
|                              |
|        Пв              Пс    |
(- - - (- - - - -) - - - - -(- - - - -)- - - -)

  Ма  МвМа    Ма    МсМа   Ма
```

Пусть в подпрограммах (или фрагментах) Пв и Пс устанавливаются локальные преобразования, соответственно, Мв и Мс, а программа Па устанавливает свое (глобальное) преобразование Ма. Тогда элементы подрисунка В должны подвергаться преобразованию МвМа, элементы подрисунка С - преобразованию МсМа, а в программе Па повсюду (за исключением фрагментов Пв и Пс) должно действовать только преобразование Ма.

В Графоре начало каждого подрисунка отмечается вызовом подпрограммы BEGLEV, которая открывает новый уровень вложенности и сохраняет матрицу текущего преобразования глобальную по отношению к этому подрисунку. Подрисунок заканчивается вызовом подпрограммы ENDLEV, которая отменяет все локальные преобразования и восстанавливает матрицу преобразования предыдущего (более высокого) уровня. В Графоре допускается 16 уровней вложенности преобразований. При попытке открыть уровень сверх этого числа на печатающее устройство выдается диагностический текст СЛИШКОМ МНОГО СКОБОК, страница принудительно закрывается и вдоль линии разреза страниц пишется GFFALS - название программы Графора, выполняющей диагностические печати. Аналогичные действия производятся, если при обращении к программе ENDLEV оказывается, что не было соответствующего обращения к BEGLEV (нарушено соответствие скобок). Только диагностический текст будет другим: НЕВЕРНОЕ ЧИСЛО СКОБОК.

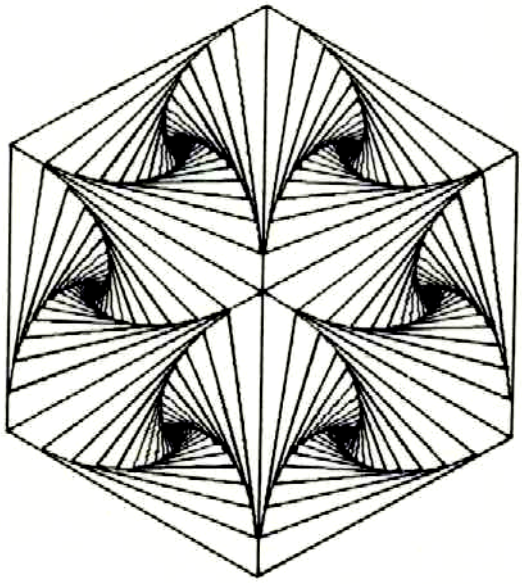

Рис. 2.1. Пример использования программ аффинных преобразований

Заметим, что подпрограмма RESET действует только на текущем уровне и отменяет только локальные преобразования. После аннулирования программой RESET накопленного преобразования на текущем уровне может накапливаться новое, и этот процесс может повторяться любое число раз. Закрытие страницы (обращение к программе ENDPG) закрывает

все еще не закрытые уровни подрисунков и, стало быть, аннулирует все накопленные преобразования.

Примером использования вложенных преобразований может служить рис. 2.1. Он построен с помощью следующей программы:

```
//MISH JOB
//    EXEC GRAFGCLG
//FORT.SYSIN DD *
      CALL GRINIT
      CALL BANTIK
      CALL GRFIN
      END
      SUBROUTINE BANTIK
      R=3.6
      XC=6.
      YC=6.
      YL=-R/2
      XL=YL/SQRT(3.)
      CALL PAGE(12.,12.,0,0,1)
      CALL TRANSL(XC,YC)
      CALL BEGLEV
      DO 1 I=1,3
      CALL BEGLEV
      DO 2 J=1,2
      CALL BEGLEV
      DO 3 K=1,20
      CALL MOVE(0.,0.,0)
      CALL MOVE(0.,-R,1)
      CALL MOVE(-SQRT(3.)/2*R,-R/2,1)
      CALL MOVE(0.,0.,1)
      CALL PSCALE(XL,YL,.8)
      CALL ROTATE(XL,YL,-8.)
    3 CONTINUE
      CALL ENDLEV
      CALL LSCALE(0.,0.,0.,1.,-1.)
    2 CONTINUE
      CALL ENDLEV
      CALL ROTATE(0.,0.,120.)
    1 CONTINUE
      CALL ENDLEV
      CALL ENDPG('2.1')
      RETURN
      END
//GO.SYSGRF DD UNIT=007,DCB=BLKSIZE=80
//
```

На самом нижнем уровне вложенности (в программе это цикл DO 3) строится фигура, образованная 20 треугольниками. В цикле каждый следующий треугольник уменьшается (коэффициент масштабирования 0.8) и поворачивается (на угол 8°) по отношению к предыдущему.

На следующем уровне (цикл DO 2) строится зеркальное отражение фигуры относительно одной из сторон исходного треугольника.

И, наконец, на самом верхнем уровне (цикл DO 1) фигура и ее зеркальное отражение трижды повторяются с поворотом на 120°.

2.2. След пера

Формирование следа пера сводится к тому, что с некоторой дискретностью отбираются точки, принадлежащие траектории пера, и координаты этих точек запоминаются. Отобранные точки мы будем называть *зарубками*, а массивы в памяти, где запоминаются их координаты, - *банком зарубок*. Введение последнего термина позволяет различать информацию (сам след) и место его хранения (банк).

Довольно легко перечислить параметры для управления процессом накопления следа, которые было бы желательно предоставить пользователю. Прежде всего, нужно уметь управлять выбором точек траектории так, чтобы при относительно небольшом числе точек траектория была задана достаточно хорошо. Далее очевидно, что интерес представляет не только *результирующая траектория* пера, т.е. та, которая получится после применения всех преобразований (линейных, экранирований, отсечений по странице), но и *предварительная траектория*, т.е. та, по которой перо двигалось бы, если бы никаких преобразований не выполнялось.

Без рассмотрения предварительной траектории не обойтись, если мы хотим, например, построить подрисунок, используя след пера, а затем подвергнуть ее аффинному преобразованию. В случае экранирования элементы, попавшие на заэкранированные участки, вообще не будут присутствовать в результирующей траектории, тогда как предварительная будет их содержать.

Далее ясно, что нам нужно уметь отключать рисование во время формирования следа, поскольку возможно, что рисовать мы будем после преобразования занесенных в банк координат.

И, наконец, определенные удобства предоставляла бы возможность работать одновременно с несколькими банками зарубок, индивидуально управляя накоплением следов в каждом из них.

Средства работы со следом пера представлены в Графоре, в основном, двумя программами: NOTCH - открытие банка зарубок и включение режима формирования следа и RENTCH - закрытие банка и отмена режима формирования следа.

Подробнее об этих программах будет сказано ниже.

2.2.1. Два метода выбора зарубок на траектории. Траектория пера представляет собой ломаную линию, каждое из прямолинейных звеньев которой соответствует перемещению пера, осуществляемому за одно обращение к программе MOVE. При этом, естественно, одни звенья проходятся с опущенным, а другие - с поднятым пером.

В Графоре реализовано два метода отбора зарубок вдоль ломаной, причем в каждом из них задается шаг *S*. При решении вопроса о включении текущей точки траектории в число зарубок рассматривается длина *L* участка траектории от предыдущей зарубки до текущей точки по отношению к шагу *S*.

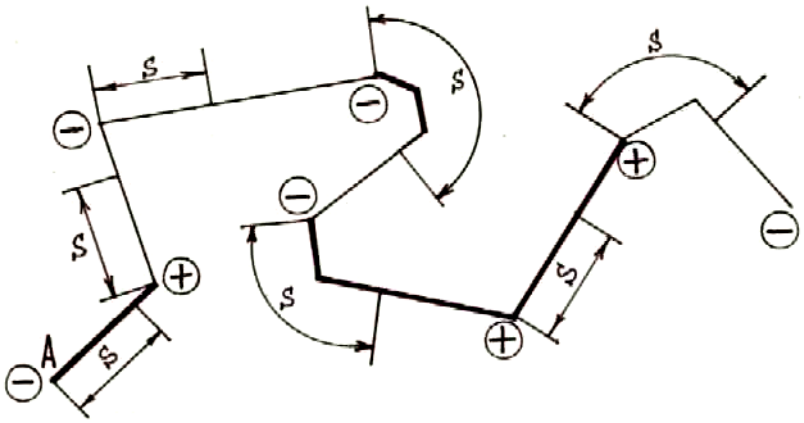

Рис. 2.2. Первый метод отбора зарубок. Обозначения: темная линия - перо опущено, светлая линия - перо поднято, (+)- зарубка с положительным *X*, (-) - зарубка с отрицательным *X*, *A* - начальная точка траектории (в нее перо пришло поднятым), *S* - шаг.

В качестве первой зарубки всегда берется начальная точка траектории, т. е. точка текущего положения пера в момент соответствующего вызова программы NOTCH. По первому методу в качестве зарубок берутся только вершины ломаной. Если для очередной вершины выполняется соотношение $L < S$, то вершина не становится зарубкой и рассматривается следующая вершина. Если же $L \geq S$, то вершина становится зарубкой. Таким образом, когда все отрезки траектории больше шага S, в качестве зарубок берутся все вершины и только они. Если же ломаная содержит короткие отрезки, часть вершин не попадет в число зарубок. На рис. 2.2 иллюстрируется описанный метод отбора зарубок.

При заданном на рисунке шаге S в число зарубок попадут вершины, отмеченные кружочками, при этом внутри кружочков будут стоять знак + , если координаты положительны (см. ниже), и знак - , если отрицательны.

Рассмотренный метод выбора зарубок удобен, например, когда требуется записать рисунок с точностью до шага S, а впоследствии воспроизвести его без преобразований. Им можно также воспользоваться, если след необходимо подвергнуть линейным преобразованиям (прямая при этом переходит в прямую). Для траекторий с относительно большой длиной звеньев получается достаточно экономная запись следа. Если же след формируется для выполнения впоследствии нелинейных преобразований рисунка, а средняя длина звеньев заметно превышает S, то такой метод отбора зарубок становится непригодным.

Второй метод ничем не отличается от первого для тех участков траектории, которые проходятся с поднятым пером. Когда же перо опущено, то зарубки расставляются на траектории равномерно с шагом S, невзирая на концы отрезков ломаной (рис. 2.3).

Информация о том, какие зарубки на траектории проходились с опущенным пером, а какие с поднятым, запоминается в виде знака координаты каждой зарубки. Мы знаем, что результирующие страничные координаты всегда положительны, и, следовательно, для результирующего следа пера проблем не возникает. Что же касается предварительных страничных координат, то они, в принципе, хотя и очень редко, могут принимать отрицательные значения. Поэтому, когда формируется предварительный след пера, значения всех отрицательных координат X заменяются малым по абсолютной величине числом, не равным 0, и знак этого числа используется для указания состояния пера.

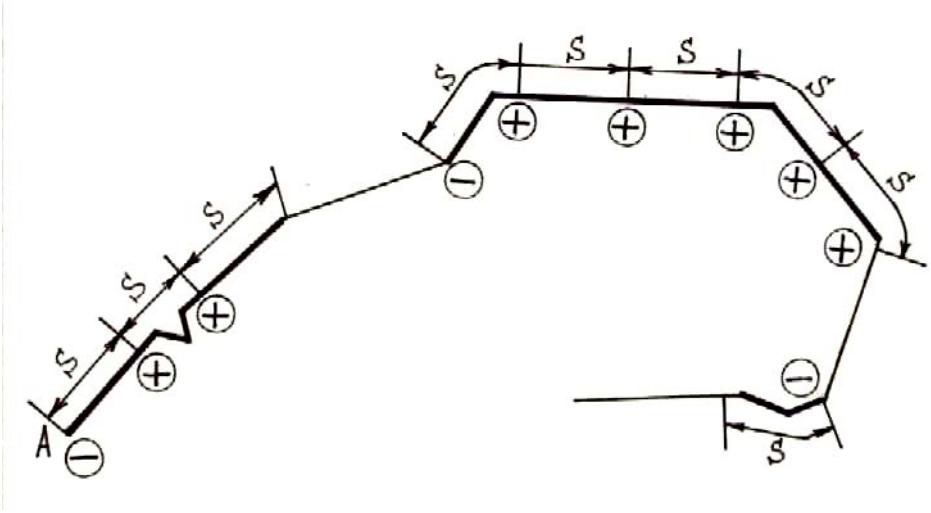

Рис. 2.3 Второй метод отбора зарубок. Обозначения см. на рис. 2.2

Фактически это соответствует отсечению по левому краю страницы в предварительных координатах.

Итак, координата X положительна, если перо пришло в зарубку опущенным, и отрицательна в противном случае. То есть согласно этому правилу, знак координаты X в первой зарубке определяется состоянием пера при вычерчивании звена, которое предшествует траектории.

2.2.2. Банки зарубок. Для того чтобы открыть банк зарубок, следует обратиться к программе NOTCH. Эта программа запоминает адреса переданных ей массивов *X*, *Y*, их длину и адрес переменной *N*, которые будут использоваться при накоплении зарубок. Программа NOTCH устанавливает значение *N* равным 1.

Программа NOTCH(X,Y,NMAX,N,S,J,ITR) открывает банк зарубок и включает заданный режим формирования следа пера. Параметры программы следующие:

X, Y - массивы для накопления координат зарубок;

NMAX - размер каждого из этих массивов;

N - переменная для хранения текущего индекса заполнения банка;

S - величина шага следа пера (|S|) в выбранных единицах измерения и метод его формирования: S > 0 - первый метод, S < 0 - второй метод;

J - признак рисования: J = 0 - рисование отключено, J = 1 - рисование включено;

ITR - тип траектории: ITR = 0 - предварительная, ITR = 1 - результирующая.

Важно помнить, что для формирования следа пера необходимо не только установить соответствующий режим, но и правильно выбрать вариант программы MOVE (MOVE3, если след пера используется в совокупности с экранированием, и MOVE2, если без экранирования). В функциональную часть своего пакета пользователь должен подложить подпрограмму:

```
SUBROUTINE MOVE(X,Y,J)    или    SUBROUTINE MOVE(X,Y,J)
CALL MOVE2(X,Y,J)                CALL MOVE3(X,Y,J)
RETURN                           RETURN
END                              END
```

В программе MOVE2 (MOVE3) есть обращение к служебной программе NOTCH1, которая осуществляет накопление следа пера, т. е. формирует массивы координат точек с признаком перемещения пера в поднятом или опущенном состоянии.

В каждый момент времени могут существовать до 16 открытых банков зарубок. При этом все параметры, кроме параметра J, действуют лишь на свой банк зарубок и не зависят от значений аналогичных параметров в других банках. Параметр J управляет рисованием одной единственной результирующей траектории, поэтому для того, чтобы отключить рисование, достаточно в одном банке задать J = 0.

Закрыть банк можно программой RENTCH. Программа не имеет параметров и закрывает тот из существующих банков, который был открыт последним. Можно продолжить заполнение уже закрытого банка, если предварительно сохранить значение параметра N, а сразу после повторного открытия банка восстановить это значение.

При попытке открыть банк сверх 16 уже существующих на печатающее устройство выдается диагностический текст: ЧИСЛО БАНКОВ БОЛЬШЕ ДОПУСТИМОГО, а сама попытка игнорируется.

Если в каком-то банке емкость выделенных массивов для записи зарубок исчерпана, то банк закрывается и печатается диагностический текст: БАНК НОМЕР I ПЕРЕПОЛНЕН. (Считается, что перенумерованы только существующие в данный момент банки в том порядке, как они открывались.)

При закрытии страницы (при обращении к программе ENDPG) закрываются все существующие банки зарубок, так что режим формирования следа пера, установленный для одной страницы, на другую страницу не распространяется.

Для получения адресов переменных в программе NOTCH используется служебная функция IADR(A). Она написана на автокоде.

В качестве примера использования следа пера рассмотрим рис. 2.4. Изображение построено из однотипных элементов, полученных аффинными преобразованиями одного и того же эталонного элемента. Эталонный элемент представляет собой пучок касательных к дуге эллипса и строится с помощью приведенной ниже программы UNIT. Основная идея состоит в том, что при малом шаге накопления следа прямая, проходящая через соседние зарубки, достаточно хорошо приближает касательную.

Рис. 2.4. Пример, иллюстрирующий действие программ формирования следа пера

Заметим, что след пера формируется в страничных координатах. Кроме того, при использовании следа пера для задания границ областей штриховки экранов следует учитывать, что координатам точек, в которые перо приходит в поднятом состоянии, приписывается знак -.

```
      SUBROUTINE UNIT
      DIMENSION X1(2),Y1(2),X(400),Y(400)
      CALL MOVE(220.,176.,0)
      CALL NOTCH(X,Y,400,N,3.,0,0)
C...N - ЧИСЛО ЗАРУБОК, ТОЧЕК НА ЭЛЛИПСЕ,
C...УДАЛЕННЫХ ОДНА ОТ ДРУГОЙ НА 3 ММ
      CALL ELIPS(220.,176.,54.,99.,270.,140.,273.)
      CALL RENTCH
      XN=212./FLOAT(N-1)
      N=N-2
      DO 1 J=1,N
      X(J)=ABS(X(J))
    1 Y(J)=ABS(Y(J))
      T=212.+3.*XN
      DO 2 J=1,N
      X1(1)=X(J)
      Y1(1)=Y(J)
      S=SQRT((X(J+1)-X(J))**2+(Y(J+1)-Y(J))**2)
      T=T-XN
      X1(2)=X(J)+T* (X(J+1)-X(J))/S
      Y1(2)=Y(J)+T* (Y(J+1)-Y(J))/S
    2 CALL LINEO(X1,Y1,2)
      RETURN
      END
      SUBROUTINE MOVE(X,Y,J)
      CALL MOVE2(X,Y,J)
      RETURN
      END
```

Еще один пример применения следа пера, демонстрирующий возможность выполнения произвольных (в том числе нелинейных) преобразований рисунка, будет рассмотрен в п. 2.5.

2.3. Экранирование

Экранирование можно считать особым видом преобразования изображения, заключающемся в "стирании" части рисунка, принадлежащей некоторому участку - экрану. Мы рассчитываем на то, что употребляемое тут слово "экран" читатель не спутает с экраном дисплея.

Экран представляет собой область с кусочно-линейной границей. Эта область может быть односвязной, несвязной или многосвязной, как показано на рис. 2.5; во всех случаях она сводится к односвязной с помощью фиктивных соединений и разрезов. Задать экран - значит, задать координаты вершин его границы, обойдя всю границу от точки к соседней точке. Начальную точку и направление обхода можно выбирать произвольно. При этом последняя вершина может не совпадать с первой, поскольку всегда строится дополнительное ребро, соединяющее последнюю и первую вершины. Некоторые из возможных порядков обхода показаны на рис. 2.5.

Включение режима экранирования и задания экрана выполняется программой BLANC(X,Y,N,IN) с параметрами:

X, Y - массивы координат *X* и *Y* вершин границы экрана;

N - количество вершин границы;

IN - признак "внутри - вне": IN = 1 - экранируется область внутри границы, IN = 0 - экранируется область вне границы.

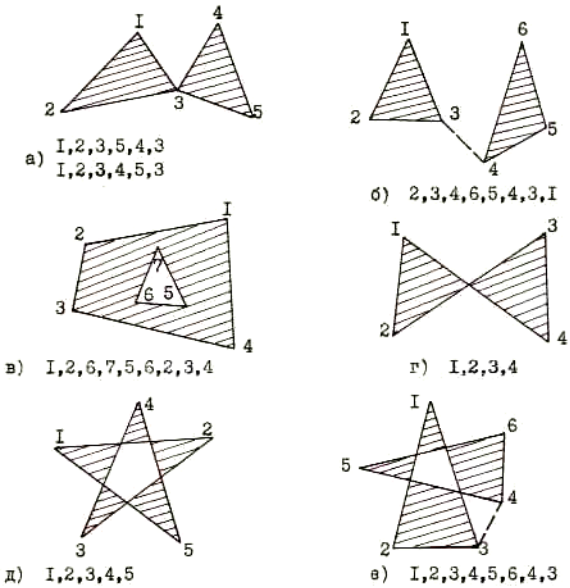

Рис. 2.5. Задание границ экранов. Штриховкой показан экранируемый участок при IN = 1

Кроме обращения к программе BLANC, необходимо выбрать соответствующий вариант программы MOVE (MOVE1 или MOVE3). Если в программе пользователя есть обращения к программам формирования следа пера и программам экранирования, необходимо использовать программу MOVE3. Если же программы экранирования применяются сами по себе, то должна быть выбрана программа MOVE1. В функциональную часть своего пакета пользователю следует вставить подпрограмму:

```
SUBROUTINE MOVE(X,Y,J)    или   SUBROUTINE MOVE(X,Y,J)
CALL MOVE1(X,Y,J)               CALL MOVE3(X,Y,J)
RETURN                          RETURN
END                             END
```

В программе MOVE1 (MOVE3) есть обращение к служебной программе BLANCH, выполняющей экранирование.

Программа BLANCH выделяет и отсекает экранируемые части заданного отрезка. Программа без параметров. В этой программе имеется дополнительный вход BLAN.

Если к моменту обращения к программе BLANC на странице действует некоторое линейное преобразование, то этому преобразованию подвергается и граница экрана. Задание границы не предусматривает ее очерчивания. Если такая необходимость возникает, то об этом следует специально позаботиться, например, воспользовавшись программами LINEO или LINEC.

При работе с экранированием реальная траектория пера существенно отличается от соответствующей траектории при отсутствии экранов. Перо, попадая на границу экранируемого участка, останавливается и остается там до тех пор, пока траектория снова не выйдет за границу экрана. Тогда перо перемещается в поднятом состоянии в соответствующую точку границы и рисование продолжается. Это следует иметь в виду, например, если формируется след пера.

До сих пор речь шла об одном экране, устанавливаемом одним обращением к программе BLANC. Однако, можно несколько раз обратиться к программе BLANC, и тогда на изображение будет действовать результирующий экран, который представляет собой объединение отдельных экранов. Любой из установленных экранов можно отменить, если он больше не нужен. При этом будет соответствующим образом скорректирован и результирующий экран.

Отмена экранов осуществляется программой REBLAN. Она не имеет параметров. Эта программа отменяет экран, установленный последним. При закрытии страницы по программе ENDPG отменяются все экраны.

Максимальное число одновременно установленных экранов не может превышать 16. Если пользователь обращается к программе BLANC, когда уже действуют 16 экранов, то отменяется самый старый из экранов и устанавливается новый. При этом печатается диагностический текст: ЧИСЛО ЭКРАНОВ БОЛЬШЕ ДОПУСТИМОГО.

На рис. 2.6 объектом изображения является ломаная линия, которая от раза к разу меняет координаты вершин и смещается на фиксированную величину по вертикали и по некоторому закону по горизонтали. Каждый раз после изображения ломаной устанавливается новый экран, в качестве точек границы которого указываются вершины этой ломаной. Таким образом, в каждый данный момент на странице действует результирующий экран, представляющий собой объединение ранее установленных экранов, ограниченных уже начерченными ломаными.

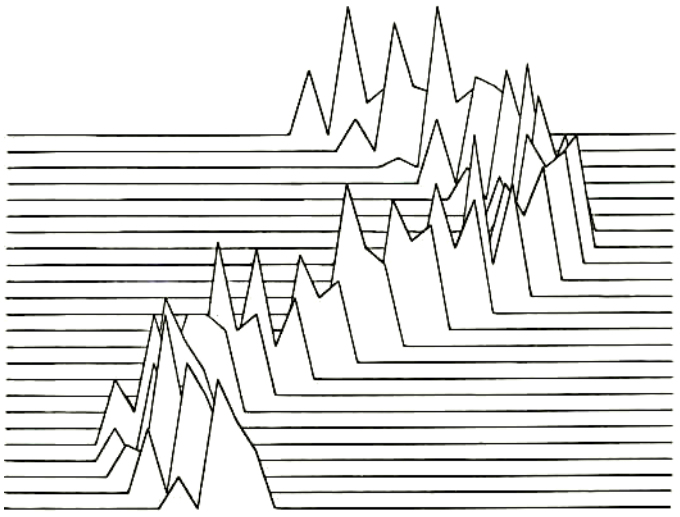

Рис. 2.6. Пример, иллюстрирующий объединение экранов.

В данном примере ограничение числа установленных экранов не играет роли: соотношение высоты "гор" и шага по вертикали таково, что автоматически отменяемый экран каждый раз оказывается вне области рисования очередной ломаной.

```
DIMENSION Y(9),XX(288),YY(288)
DATA Y/2., 2., 3., 2., 6., 4., 4.5, 2., 2./
CALL PAGE(20.,26.,0,0,0)
CALL LSCALE(0.,26.,1.,26.,0.85)
```

```
      DO 1 J=1, 24
      XX(1+(J-1)*9)=0
      YY(1+(J-1)*9)=0.5*  (J-1)+2.
      XX(2+(J-1)*9)=10.-5.*SIN(J-1)*3.14/13.)
      YY(2+(J-1)*9)=0.5*(J-1)+2.
      XX(8+(J-1)*9)=10.-5.*SIN(J-1)*3.14/13.)+3.
      YY(8+(J-1)*9)=0.5*(J-1)+2.
      XX(9+(J-1)*9)=20.
      YY(9+(J-1)*9)=0.5*(J-1)+2.
      DO 2 I=3, 7
      XX(I+(J-1)*9)=10.-5.*SIN(J-1)*3.14/13.)+0.5*(I-2)
    2 YY(I+(J-1)*9)=AMAX1(Y(I)+0.5*(J-1)+SIN(I-1)*(J-1),0.5*(J-1)+2.)
      CALL LINEO(XX(1+(J-1)*9),YY(1+(J-1)*9),9)
      CALL BLANC(XX(1+(J-1)*9),YY(1+(J-1)*9),9,1)
    1 CONTINUE
      CALL ENDPG('2.6')
      END
      SUBROUTINE MOVE(X,Y,J)
      CALL MOVE1(X,Y,J)
      RETURN
      END
```

2.4. Штриховка

Штриховка заданных полей является неотъемлемой частью подготовки различного вида чертежей и рисунков. Для выполнения штриховки служат две программы: SDPG и SHADE. Первая из них позволяет заштриховать все поле страницы. Совместное ее использование с программами экранирования дает возможность заштриховать требуемые участки на странице. Вторая программа штрихует требуемые участки, но для нее необходимо задавать границы участков. Оба способа выполнения штриховки совершенно независимы - программы SDPG и SHADE не используют друг друга.

Программа SDPG(STEP,EPS,BETA) позволяет заштриховать все поле страницы. Ее параметры:

STEP - расстояние между линиями штриховки в выбранных единицах измерения;

EPS - сдвиг линий штриховки относительно основной штриховки по нормали к ней;

BETA - угол наклона линий штриховки к оси *X* (в градусах).

Основной штриховкой в данном случае считается та, для которой одна из линий проходит через точку (0, 0), т. е. через левый нижний угол страницы.

Программа SHADE(X,Y,N,STEP,EPS,BETA) позволяет заштриховать участок страницы, ограниченный замкнутой кусочно-линейной ориентированной кривой. Параметры программы:

X, Y - массивы длины N абсцисс и ординат соответственно, задающие вершины границы;

N - число вершин границы;

STEP - расстояние между линиями штриховки в выбранных единицах измерения;

EPS - сдвиг линий штриховки относительно основной штриховки по нормали к ней;

BETA - угол наклона линий штриховки к оси *X* (в градусах).

Эта программа позволяет штриховать также и самопересекающиеся многоугольники. При задании многоугольника (т. е. массивов X и Y) последняя вершина может не совпадать с первой, поскольку всегда строится дополнительное ребро, которое их соединяет. Значение N (число вершин) не должно превышать 250. Никакая линия штриховки не должна пересекать более 40 ребер.

В тех случаях, когда обращением к программам LIMITS и REGION (см. гл. 4) заданы математические координаты и на странице определена область, программа SHADE будет проводить штриховку заданного многоугольника внутри этой области, не выходя за ее границы.

2.5. Примеры

Рассмотрим еще два примера, иллюстрирующих описанные в этой главе средства.

1. Изображение на рис. 2.7 построено с помощью приведенной ниже программы ASTRA. В качестве основной компоненты взят квадрат, который повернут на 45°, затем растянут в 1.75 раза и снова повернут. Все повороты и растяжения делаются относительно центра квадратов - точки (5., 5.). Все четыре квадрата используются как границы экранов для штриховки.

```
DIMENSION X(4),Y(4)
DATA X/3.,7.,7.,3./,Y/3.,3.,7.,7./
CALL PAGE(10.,10.,'ASTRA',5,0)
CALL BLANC(X,Y,4,1)
CALL ROTATE(5.,5.,45.)
CALL BLANC(X,Y,4,1
CALL PSCALE(5.,5.,1.75)
CALL BLANC(X,Y,4,0)
CALL ROTATE(5.,5.,45.)
CALL BLANC(X,Y,4,0)
CALL RESET
CALL SDPG(0.1,0.,45.)
CALL SDPG(0.1,0.,135.) )
CALL ENDPG('2.7')
END
SUBROUTINE MOVE(X,Y,J)
CALL MOVE1(X,Y,J)
RETURN
END
```

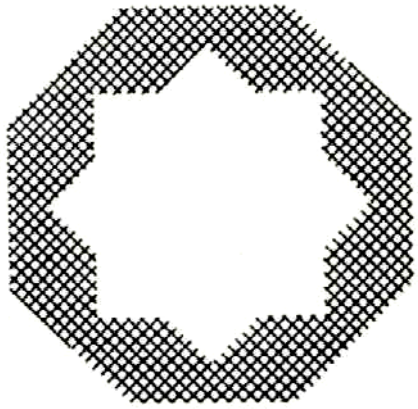

Рис. 2.7. Пример, иллюстрирующий применение программ аффинных преобразований, экранирования и штриховки

2. Обратимся к рис 2.8. Справа внизу изображен "эталонный" крокодил с закрытой пастью, нарисованный программой CROCKY, которая здесь не приводится. Программа JAWS рисует

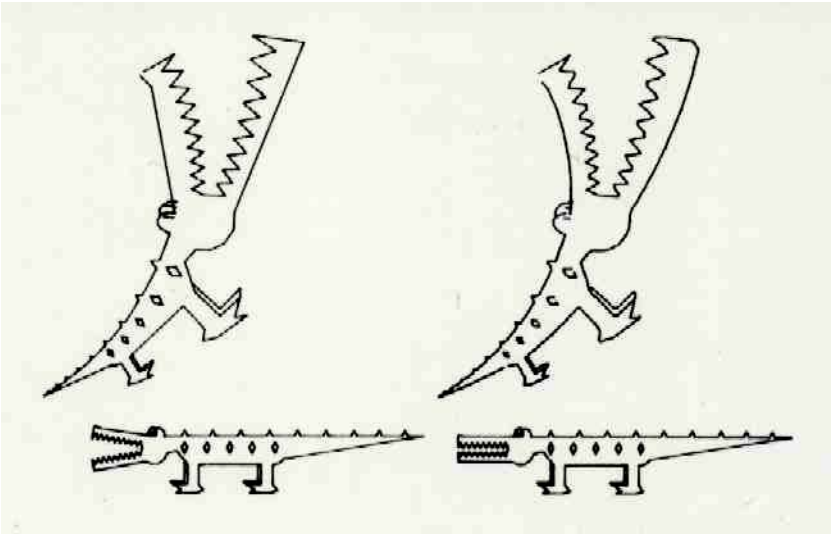

Рис. 2.8. Иллюстрация возможностей применения нелинейного преобразования к следу пера

крокодила с раскрытой пастью слева внизу. В ней до обращения к программе CROCKY устанавливается формирование следа пера и при последующем анализе точек следа выделяются челюсти и поворачиваются на заданный угол.

```
SUBROUTINE JAWS(XJ,YJ,NJ,R,TH)
DIMENSION XJ(NJ),YJ(NJ)
```

```
      CALL WHERE(DX,DY,F)
      CALL NOTCH(XJ,YJ,NJ,0.5,0,0)
      CALL CROCKY(DX,DY,R)
      CALL RENTCH
      DO 8 J=1, NT
      IF(ABS(XJ(J))-DX-26.*R.GT.0.) GOTO 4
      IF(ABS(YJ(J))-DY-15.*R.GT.0.) GOTO 3
      CALL ROTATE(DX+26.*R,DY+15.*R,TH)
      GO TO 4
    3 CALL ROTATE(DX+26.*R,DY+15.*R,-TH)
    4 IF(XJ(J).GT.0.) GOTO 6
      CALL MOVE(-XJ(J),-YJ(J),0)
      GO TO 7
    6 CALL MOVE(XJ(J),YJ(J),1)
    7 CALL RESET
    8 CONTINUE
      RETURN
      END
```

Оба крокодила наверху - результат применения нелинейного преобразования к крокодилу с раскрытой пастью. Это преобразование выполнялось над следом пера по формулам

$$x' = \frac{kx}{x^2 + y^2}, \qquad y' = \frac{ky}{x^2 + y^2},$$

причем для левого верхнего крокодила зарубки отбирались по первому методу, а для правого - по второму. На длинных отрезках хорошо заметно отличие.