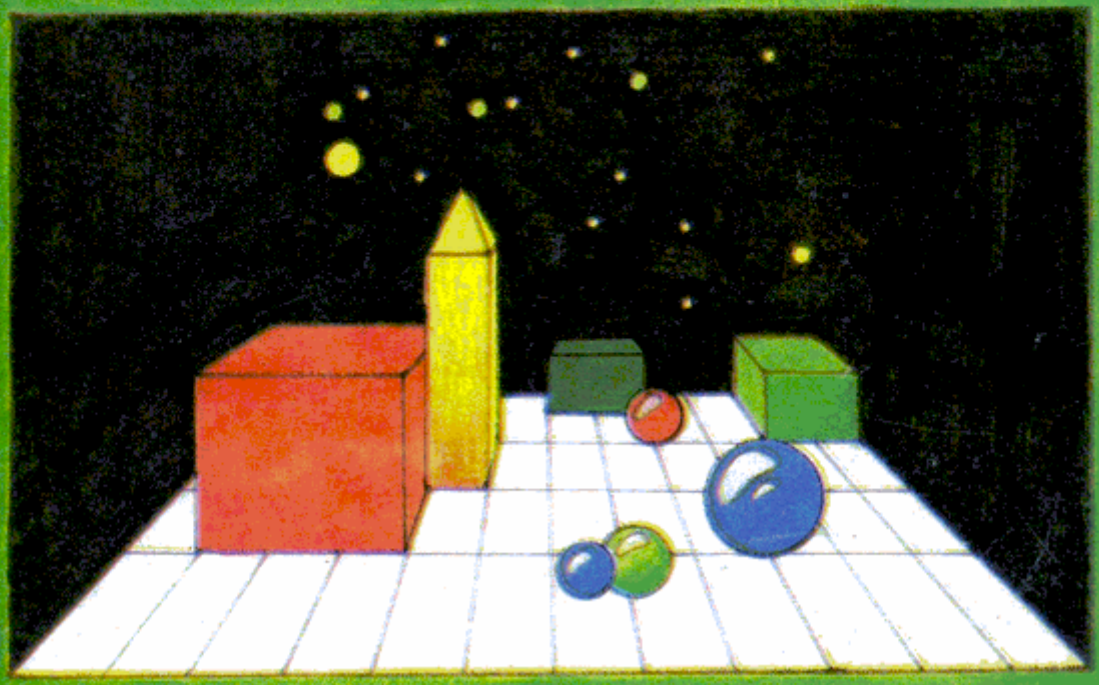
ZX-SPECTRUM

Графика ZX-SPECTRUM

PDF version by Deny (Денисенко Д.А.)
e-mail: DenyDA@mail.ru
2007

ZX-SPECTRUM



ГРАФИКА

ПРЕДИСЛОВИЕ ОТ ОБРАБОТЧИКА

В процессе обработки текста я для контроля правильности листингов набирал программы (не в ручную, а с помощью утилиты bas2tap), и как выяснилось позже, не все программы корректны:

Listing 9.9 – нет двух строчек в 6510 DATA и трёх строчек в 6520 DATA;

Listing 12.3 – нет строчки в 6520 DATA.

Так же неправильные входные параметры для Listing 12.1 и Listing 12.2 (Пример 12.1)

Некорректная программа перенумерации и уничтожения бейсик-строк Listing 13.9. Я замучался находить и исправлять ошибки в этой программе. Например, строки:

```
9948 LET C=C+2: LET C=D+2
9995 LET H=INT(NN;HI): POKE 23626,H: POKE 23625,NN-H*HI
```

исправил на

```
9948 LET C=C+2: LET D=D+2
9995 LET H=INT(NN/HI): POKE 23626,H: POKE 23625,NN-H*HI
```

Были ещё, но я уже не помню. В итоге я бросил это «гиблое» дело. Если кто сможет навести порядок, вышлите исправления к этой программе – обязательно исправлю и отошлю на trd.speccy.cz для распространения. Ну и обязательно укажу специалиста, восстановившего программу.

Самый больший беспорядок с Listing 14.1 – кроме того, что нет двух подпрограмм, так ещё куча ошибок. Я даже не пытался восстановить программу.

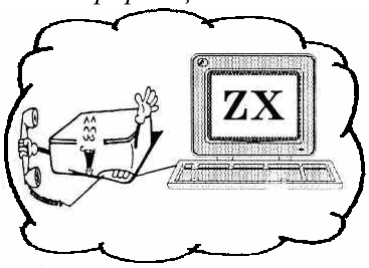
Поэтому не судите строго, если эти программы пойдут у Вас неправильно или с ошибкой.

С уважением – Deny
DenyDA@mail.ru

ОМЕГА ПЛЮС ПРЕДЛАГАЕТ:

Небольшое устройство (модем - по доступной цене), позволяющее владельцам ZX-Spectrum значительно расширить возможности своего компьютера, т.е. обмениваться информацией по телефонным каналам связи с нашей "Базой Данных" (BBS).

База Данных содержит широкий выбор игровых и системных программ, объявления, рекламу, текстовые файлы и много другой полезной информации.



ОМЕГА ПЛЮС РЕАЛИЗУЕТ:

① *Игровые и системные программы на кассетах (более 70 наименований кассет, постоянно дополняющихся).*

② *Литературу по персональным компьютерам.*

③ *Чистые аудиокассеты С-60 (в неограниченном количестве)*

④ *Немецкие телефонные аппараты (дисковые и кнопочные) с различными функциональными возможностями.*

Телефон для справок: (095) 490-14-26
Почтовый адрес: 123363, г. Москва, а/я 48.

ПРЕДИСЛОВИЕ

Быстрое развитие компьютерной технологии привело к существенному понижению цен на компьютеры. В ближайшие годы ожидается резкое подешевление периферийных устройств, что позволит пользователям с ограниченным бюджетом, которые ранее могли позволить себе лишь простейшие периферийные устройства, сменить их на самые совершенные. Это также позволит им выйти за рамки табличной формы выдачи, приобрести микропроцессорные приставки для телемониторов и недорогие устройства цветной графики специального назначения Фирма "Синклер" всегда занимала передовые позиции в этих областях. Отметим, что подешевление компьютеров не сопровождается подешевлением программного обеспечения.

Так как в прошлом машинная графика требовала значительных ресурсов, как программных, так и машинных, то в этой области работали в основном большие исследовательские группы, что привело к мифам о её исключительной сложности.

В данной книге мы собираемся, эти мифы развеять и показать, что рядовой пользователь может обойтись без дорогостоящих пакетов программ, которые столь ценны для исследовательских организаций Данная книга, являющаяся введением в предмет компьютерной графики, может рассматриваться как дешевый пакет программ: она значительно дешевле пакетов программ, имеющихся в продаже. В силу принятого в книге подхода, перед тем, как рисовать картинки кроме тех, которые приведены в книге, следует ознакомиться с принципами, лежащими в основе работы графического обеспечения. Это не является недостатком книги, так как эти сведения можно получить довольно быстро. В результате работы с пакетом, пользователь параллельно изучает принципы, лежащие в основе его построения, что снижает вероятность ошибки при использовании графических подпрограмм.

Предполагается, что читатель имеет элементарные сведения о декартовой геометрии. В книге содержится много полезных упражнений, которые помогут повысить ваш уровень культуры программирования на языке бейсик. Бейсик является широко распространенным языком, доступным (в различных реализациях) на всех типах микро-компьютеров, так что программы легко модифицировать для использования на компьютерах, отличных от SPECTRUMa; бейсик является хорошим средством для записи алгоритмов, используемых в компьютерной графике, и позволяет читателям перевести программы на любой другой язык программирования.

Необходимая для изучения компьютерной графики теория развивается по мере изложения. Материал иллюстрируется примерами, в том числе и текстами программ, которые следует рассматривать не только как описания алгоритмов решения той или иной задачи компьютерной графики, но и как программное обеспечение, написанное на языке бейсик, или, проще говоря, как программы для составления определенных рисунков. Наряду с примерами книга содержит ряд упражнений, в которых теоретические положения книги получают дальнейшее развитие. Во многих случаях оказывается, что учащемуся написать программу, реализующую тот или иной метод решения графической задачи, сложнее, чем понять идеи, лежащие в основе этого метода. Поэтому важно, чтобы читатель испытал в действии как можно больше программ, листинги которых приведены в тексте, и прорешал большую часть упражнений. Если читателя смущает используемый в книге математический аппарат, то перед чтением теории мы рекомендуем ему познакомиться с работой программ.

Для облегчения программирования математических формул все они в процессе объяснений приводятся в готовом виде для бейсика.

Возможность получать рисунки на цветном интерактивном графическом дисплее производит сильное впечатление, а метод обучения, принятый в данной книге, позволяет быстро подготовить читателя к выполнению довольно сложных графических работ, которые в данной книге не обсуждаются.

Эта книга предназначена для людей, имеющих опыт программирования на бейсике, но не знакомых с предметом машинной графики. В ней содержатся основные сведения о двумерной графике, которыми следует овладеть перед тем, как приступать к символьной и трехмерной графике. За этим материалом следует глава, посвященная символьной графике и представлению информации (в графиках и цвете) – пожалуй самое важное неспециализированное коммерческое приложение машинной графики.

Последующие главы вводят читателя в геометрию пространства трех измерений и многообразие проекций этого пространства на двумерную плоскость графических устройств. В деталях разобраны связанные с этими вопросами проблемы скрытых линий и поверхностей и построения сложных трехмерных объектов. В последней главе мы возвращаемся к программированию на бейсике и приводим пример действующей программы видеоигры.

Графика - одна из наиболее быстро развивающихся областей компьютерной науки. Она все больше используется в таких областях как автоматизированное программирование, АСУ, автоматизированные системы обучения. Одно время эта техника использовалась только в автомобилестроении и самолетостроении, но сейчас большинство компаний осознают возможности и экономичность такого подхода. Более того, компьютерная графика не только доходное дело, но это еще и развлечение! Компьютер SPECTRUM идеально подходит для изучения основ машинной графики и подготовки к работе с наиболее сложными (и дорогостоящими) графическими устройствами.

Мы надеемся, что эта книга поможет вам пережить то приятное волнение от соприкосновения с машинной графикой, которое испытали мы и наши коллеги. Для демонстрации важности компьютерной графики или иллюстрации книг все рисунки в этой книге нарисованы компьютером.

ВВЕДЕНИЕ

Эту книгу можно читать по-разному. Во-первых, ее можно рассматривать как книгу рецептов графических программ, позволяющих рисовать сложные картинки с помощью SPECTRUMa. Мы надеемся, что после того, как читатель нарисует эти картинки, у него возникнет желание узнать побольше о самих программах. Во-вторых, из некоторых программ можно составить пакет графических программ для генерации различного вида диаграмм (круговых диаграмм, гистограмм и графиков для коммерческих и лабораторных целей). И, наконец, главной целью данной книги является введение в предмет машинной графики, который начинается с ознакомления читателя с элементарными понятиями, а заканчивается такими сложными темами, как символьная графика, построение трехмерных объектов и алгоритмы удаления скрытых линий (и поверхностей).

Сложные программы, которые нам встретятся в этой книге, слишком связаны между собой, чтобы можно было ограничиться для их описания одними листингами. Более того, мы увидим, что некоторые элементарные алгоритмы повторяются чрезвычайно часто. По этой причине мы используем метод «сверху вниз» или модульный метод при составлении и изложении наших программ. Решение каждой задачи сводится к решению ряда более простых задач. Последние, в свою очередь, разлагаются на несколько еще меньших решаемых задач модулей. Каждый из этих модулей оформляется в виде подпрограммы бейсика, каждой из которых дается имя (состоящее из строчных букв). После чего эти модули объединяются для решения поставленной графической задачи.

Так как листинги программ используются для представления алгоритма решения той или иной подзадачи, то мы решили избегать использования утверждений вроде **GO SUB 6000**. Вместо этого мы предпочитаем присвоить идентификатору подпрограммы значение адреса в начале программы (например, **LET SCENE A/3=6000**), что позволяет нам использовать утверждения типа **GO SUB SCENE A/3**.

Спектрум-версия языка бейсик не позволяет передавать параметры в подпрограммы. Значения входных параметров должны задаваться операторами присваивания вне подпрограммы. А, кроме того, если выходные параметры используются вне подпрограммы, то их имена должны быть известны. Это обстоятельство может затруднить использование чужих пакетов подпрограмм. Пользователь должен знать имена входных и выходных параметров; в наших подпрограммах мы используем операторы-комментарии REM IN (для идентификации входных параметров). Мы нумеруем строки программы так, что все номера строк всех программных операторов оканчиваются нулями, а комментарии располагаются на строчках с номерами от 1 до 9 (кроме названий программ). Сразу за комментарием, объявляющим имя программы, на строчках с номерами, оканчивающимися на 1 и 2, следуют комментарии **REM IN** и **REM OUT**, описывающие входные и выходные параметры.

Для тех, кто желает только использовать программы, их полный список приводится в конце каждого параграфа вместе с подходящими значениями входных параметров.

Рекомендуем всем, включая серьезных читателей, перед чтением каждого параграфа, записать эти подпрограммы вместе с соответствующими входными данными на ленту или дискету. После этого программы можно загружать и выполнять (**LOAD**, **MERGE**, **RUN**) по мере их появления в тексте.

Наши программы написаны для SPECTRUMa с объемом памяти 48К: если у вашего компьютера память 16К, то вам следует прочитать приложение "А", где указаны необходимые изменения, которые надо внести в программы

В качестве показательного примера мы приводим программу, рисующую тело вращения (см. рис. 1).

Рис. 1

Эта программа требует выполнения операции **MERGE** с листингами программ 2.1 ("START"), 2.2 (две функции FNX и FNY), 2.3 ("SETORIGIN"), 2.4 ("MOVETO"), 3.3 ("CLIP" и "LINETO"). Эта комбинация программ называется "LIB1". Она предназначена для рисования на экране фигур, состоящих из линий.

К "LIB1" следует добавить листинги 3.4 ("ANGLE"), 8.1 ("MULT3", "IDR3"), 8.2 ("TRAN3"), 8.3 ("SCALE3"), 9.1 ("LOOK3"), 9.2 ("MAIN PROGRAM"). Комбинацию этих подпрограмм мы называем "LIB3", они служат для трансформации и осмотра тел в трехмерном пространстве.

Нам также потребуется подпрограмма, приведенная на листинге 10.3 ("REVBOD") и подпрограмма "FLYSAU" на листинге А.

LISTING A

```
6000 REM SCENE3/FLYING SAUSER
6010 DIM X(12): DIM Y(12)
6020 DIM S(6): DIM T(6)
6030 DIM A(4,4): DIM B(4,4): DIM R(4,4)
6040 DATA 0,3,3,2,5,1,5,0,4,-1,0,-3
6050 RESTORE SCENE3
```

```
6060 LET REVBOD=6500
6069 REM CREATE OBJECT.
6070 LET NUMV=5
6080 INPUT "NUMBER OF ";"HORIZONTAL LINES=";NUMH
6090 INPUT "ANGLE PHI=";PHI
6100 FOR I=1 TO NUMV+1: READ S(I),T(I):NEXT I
6109 REM POSITION THE OBSERVER.
6110 GO SUB IDR3: GO SUB LOOK3
6119 REM DRAW OBJECT.
6120 GO SUB REVBOD
6130 RETURN
```

Для того чтобы получить фигуру, на рис. 1, значения входных параметров должны быть:

HORIZ=12, VERT=8, NUMH=16, PHI=0, EX=1, EY=2, EZ=3, DX=0, DY=0, DZ=0.

Каждое значение задается после соответствующего запроса программы. Наберитесь терпения, так как получение этой картинки займет минут пять. Пропустите программу с различным значением входных параметров. Что произойдет, если HORIZ=6 и VERT=4, а значения остальных параметров те же? Положите HORIZ=15, VERT=10, EX=1, EY=2, EZ=3, DX=1, DY=0, DZ=0. Попробуйте NUMH=20, PHI=0,1. Чтобы понять, что при этом происходит, вам придется прочитать первые десять глав.

Этот пример иллюстрирует метод изложения, принятый в книге. Если вы хорошо печатаете, или приобрели кассету с программами, то вы получите рисунок относительно сложного объекта за минимальное время. Даже если вы печатаете одним пальцем (как это делают авторы), то тогда у вас не возникнет особых сложностей, чтобы пропустить ту или иную программу перед тем, как начать изучать эту книгу.

Мы надеемся, что этот пример вдохновит вас на то, чтобы опробовать все программы, приведенные в книге, проверить большую часть примеров, а затем приступить к составлению собственных рисунков.

Теперь вы можете приступить к чтению оставшейся части книги. Желаем вам многих приятных часов с вашим компьютером.

ГЛАВА 1
Графические команды ZX SPECTRUM

Мы полагаем, что читатель хорошо знает язык бейсик-спектрум. В этой главе, однако, мы приводим обзор некоторых команд бейсика - тех, которые связаны с графикой. Серия примеров и упражнений демонстрирует графические возможности SPECTRUMa. Сведения, приведенные в этой главе, используются в последующем для построения теории и практических приложений машинной графики.

Вначале мы рассмотрим графические возможности оборудования и программного обеспечения. Все микрокомпьютеры, которые могут производить телевизионное изображение, используют технологию RASTER SCAN. Это относится к большинству современных коммерческих микрокомпьютеров и большим машинам. В памяти отводится специальная область, в которой хранится информация об изображении на экране, которая просматривается бит за битом по мере того, как электронный луч скользит по поверхности экрана. Изображение на экране состоит из точек, каждая из которых представлена одним битом (бинарное да/нет) памяти. В простейшем случае точка экрана высвечивается на короткое время каждый раз, когда встречается "да".

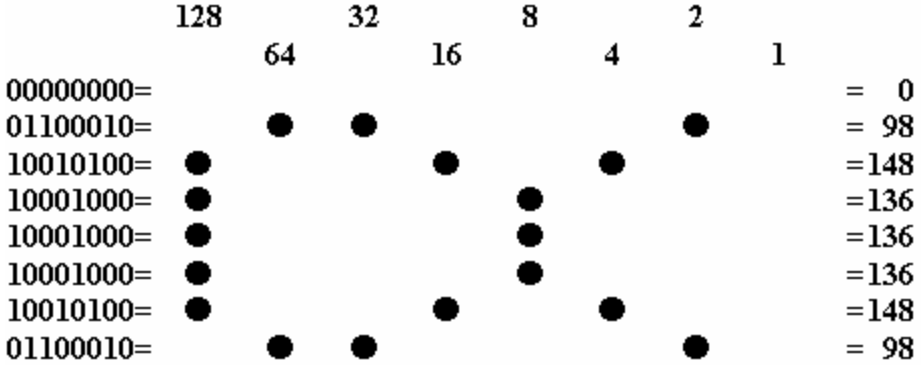
Команды PAPER и INK ("бумага" и "чернила", фон и тон)

Изображением на экране, состоящим из рисующих INK-точек (бинарное "да") и фоновых PAPER-точек (бинарное "нет"), можно по точечно управлять с помощью двух команд. Эти команды PAPER и INK вызываются указанием имени команды и следующего за ним номера N (0<=N<=9).

Команда PAPER N устанавливает N-ый цвет фона; изображение будет иметь фон цвета N, пока в программе не встретится другая команда PAPER.

Аналогичным образом команда INK N устанавливает цвет тона, если 0<=N<=7, тона соответствующей клавише клавиатуры дисплея написан цвет N. Если N=8, то используется цвет, который использовался ранее для области. Если N=9, то цвет PAPER/INK становится белым или черным и будет контрастным для текущего цвета INK/PAPER.

Сочетание черного INK цвета и белого PAPER цвета дают самые четкие картинки и являются нормальными значениями этих цветов для SPECTRUM.



Дисплейный файл (отобразить на экране содержимое файла)

Такой тип изображения мы называем "отображением участка памяти" (MEMORY-MAPPED), так как оно однозначно определяется содержимым участка памяти. В SPECTRUME этот участок называется дисплей-файлом и начинается с ячейки 16384.Программа на листинге 1.1 демонстрирует, как меняется изображение на экране при изменении содержимого памяти.

LISTING 1.1

```
10 LET CORNER=16384
20 LET VALUE=137
30 POKE CORNER,VALUE
40 STOP
```

Эта программа использует программу POKE для занесения значения (десятичного)переменной VALUE в первую позицию дисплей-файла. В этой позиции содержится информация, управляющая изображением в левом верхнем углу экрана. Так как каждая ячейка, или байт, содержит восемь бит, то программа затрагивает первые восемь точек экрана (двоичное значение - 10001001), так что изображение в первых восьми точках экрана соответствует этому числу.

Упражнение 1.1

1. Попробуйте различные значения VALUE и измените программу так чтобы либо
   а). использовать BIT (двоичное) представление VALUE, либо
   б). использовать для изменения VALUE цикл FOR..NEXT.
2. Использовать команды PAPER и INK и посмотреть, какой это производит эффект.

Команда BORDER (граница, рамка экрана)

Если изменить PAPER-цвет, то можно увидеть, что мы не можем рисовать на всем экране. По краям поля PAPER идет поле BORDER; это делается для того, чтобы избежать искажений, характерных для всех телевизионных дисплеев. Цвет этого поля может быть изменен аналогично цветам PAPER и INK с помощью команды BORDER N, где 0<=N<=7

Символьные блоки

Присваивая ячейкам памяти различные значения VALUE, можно получить любую картинку. Например, присваивая значения VALUE 0, 98, 148, 136, 136, 136, 148, 98 ячейкам, соответствующим первым восьми точкам восьми последовательных линий экрана (листинг 1.2), мы увидим картинку из INK-точек (см рис. 2).

Таким образом, определяются (и переопределяются) символы в SPECTRUMe, но оставим эту тему до главы 5. Мы видим, что для составления даже такой простой программы требуется сравнительно сложная подпрограмма.

LISTING 1.2

```
10 LET CORNER=16384
20 LET LINE=256
30 DATA 0,98,148,136,136,136,148,98
40 FOR I=0 TO 7
50 LET MEMORY=CORNER+I*LINE
60 READ VALUE
70 POKE MEMORY,VALUE
80 NEXT I
90 STOP
```

Команды PLOT и DRAW ("нанести точку" и "начертить линию")

Мы видим, что изображение на экране можно изменять, занося различные значения в дисплейный файл. Но было бы весьма утомительно менять содержимое каждой из6000 ячеек дисплейного файла; нам, очевидно, необходим более эффективный способ менять изображение.

Простейшими командами бейсика, служащими этой цели, являются PLOT и DRAW. Во всех графических команда дисплей считается сеткой, состоящей из 256горизонтальных строк и 176 вертикальных столбцов (всего 45056 точек). Точки, из которых состоит экран, называются пикселями (PIXELS), и каждой из них соответствует пара целых чисел. Графические команды позволяют нам управлять так называемым графическим пером, которое в начале работы расположено над точкой (0,0).Перейдем к самим командам.

По команде PLOT X,Y перо перемещается в точку X,Y и ставит там INK-точку.

По команде DRAW X,Y перо проводит прямую из точки, в которой оно находилось, в точку, которая удалена от нее на X столбцов по горизонтали и Y строк по вертикали. Если X отрицательно, то перо сдвинется влево, а если X положительно, то перо сдвинется вправо, а если Y положительно, то перо сдвинется вверх.

После выполнения этих команд перо останется в последней точке, в которую оно пришло. Перед тем, как познакомиться с более сложными командами, посмотрим, что можно произвести с помощью этих двух команд.

Мы готовы к созданию более крупных рисунков. Например, мы можем заключить графическое поле дисплея в рамку (листинг 1.3).

LISTING 1.3

```
10 PLOT 0,175
20 PLOT 255,175
30 PLOT 255,0
40 PLOT 0,0
50 IF INKEY$ <>"" THEN GO TO 50
60 IF INKEY$ ="" THEN GO TO 60
70 DRAW 0,175
80 DRAW 255,0
90 DRAW 0,-175
100 DRAW -255,0
110 STOP
```

Эта программа сначала ставит точки в углах экрана; затем она ожидает нажатия любой клавиши для того, чтобы соединить их прямыми. Сравнивая команды PLOT и DRAW, мы видим одно принципиальное различие между ними: PLOT использует абсолютные координаты, а DRAW - относительные координаты, соответствующие взаимному расположению точек на экране. Это означает, что перед тем, как провести линию между двумя точками экрана, сначала следует использовать команду PLOT для установки пера в первой точке, а затем вычислить расположение второй относительно первой. Обратите внимание, что в программе на листинге 1.3 позиции всех точек являются константами. Обычно они являются входными параметрами или вычисляются в процессе выполнения программы.

Упражнение 1.2

Напишите программу, которая бы вычисляла позиции линий в сетке. Начертите эти линии, используя два цикла FOR...NEXT (один для горизонтальных прямых, другой для вертикальных).

Упражнение 1.3

Напишите программу, которая содержит в качестве входной информации координаты N точек экрана, вводимые с клавиатуры, а рисует неправильный многоугольник с вершинами в этих точках (следует внимательно обдумать эту программу, так как первая точка должна быть соединена с последней).

Команды PRINT и LIST

Мы пока не обсуждали самого простого способа изменения изображения на экране, а именно, использования команд PRINT и LIST. Это произошло потому, что эти команды оперируют с символьными блоками и пригодны лишь для графических работ с низкой разрешающей способностью. Эти вопросы будут обсуждаться в главе 5, так как SPECTRUM позволяет сочетать технику графики с высокой разрешающей способностью. Рассмотрим один такой пример здесь. Предположим, что мы добавили строчку

```
5 LIST
```

к началу программы упражнения 1.2, а затем нарисуем с помощью программы сетку из 32 вертикальных и 22 горизонтальных линии. Мы получим картинку, на которой видны размеры и позиции символов.

Мы можем использовать команду PLOT для того, чтобы продемонстрировать графические возможности SPECTRUMa для получения изображения фракталей.

Для того чтобы нарисовать простейшую фрактальную линию, мы используем следующую процедуру. Рассмотрим квадрат со стороной 4 в степени N, это 16 квадратов, которые мы нумеруем согласно рисунку 3.

13	**14**	**15**	**16**
9	**10**	**11**	**12**
5	**6**	**7**	**8**
1	**2**	**3**	**4**

Рис. 3.

Четыре из этих квадратов, с номерами 2, 8, 9, 15 мы переставляем, как это показано на рис. 4.

Рис. 4.

Каждый из этих квадратов разбивается на 16 меньших квадратов, с каждым из которых производится та же процедура. Мы продолжаем этот процесс, пока длина квадрата не становится равной 1. В результате мы получаем фракталь, состоящую исключительно из единичных квадратов, которые мы можем интерпретировать как точки и нанести на экран командой PLOT. Программа, текст которой приведен на листинге 1.4, начинает с квадрата со стороной $64=4^3$ и содержит три вложенных цикла FOR...NEXT. В результате выполнения этой программы мы получим рисунок, приведенный на рис. 5.

Рис. 5.

LISTING 1.4

```
10 DIM X(16):DIM Y(16)
20 FOR I=1 TO 4
30 FOR J=1 TO 4
40 LET K=4*I+J-4
50 LET X(K)=J-3: LET Y(K)=I-3
60 NEXT J: NEXT I
70 LET X(2)=0: LET Y(2)=-3
80 LET X(8)=2: LET Y(8)=0
90 LET X(9)=-3: LET Y(9)=-1
100 LET X(15)=-1: LET Y(15)=2
110 FOR I=1 TO 16
120 FOR J=1 TO 16
130 FOR K=1 TO 16
140 LET XX=16*X(I)+4*X(J)+X(K)
150 LET YY=16*Y(I)+4*Y(J)+Y(K)
160 PLOT 126+XX,88+YY
170 NEXT K:NEXT J:NEXT I
180 STOP
```

Команды INVERSE и OVER

Сейчас мы рассмотрим возможности изменения способа изображения точек и прямых на экране. Имеются две такие команды, и при использовании их следует указывать имя этой команды, за которой должно следовать число: 1 - включает эффект, а 0- выключает..

При включенной команде OVER точка экрана, которую затрагивает любая из графических операций, меняет свое состояние на противоположное. Каждая точка цвета INK становится цвета PAPER и наоборот. Это означает перекидывание двоичного ключа из одной позиции в противоположную.

Используя эти команды, мы можем создавать программы, которые создают сложные на вид рисунки с быстроменяющимся изображением. На листинге 1.5 приведена программа, которая содержит два способа создания сложных узоров с помощью простых алгоритмов.

LISTING 1.5

```
10 OVER 1
20 LET LINES=400
30 LET A=0: LET ANGLE=2*PI /LINES
40 FOR I=1 TO LINES
50 LET X=85*COS A
60 LET Y=85*SIN A
70 PLOT 128,88
80 DRAW X,Y
90 LET A=A+ANGLE
100 NEXT I
110 OVER 0
120 STOP
```

На дисплее, состоящем из дискретных точек, линии, не являющиеся горизонталями или вертикалями, состоят из последовательно коротких горизонтальных и вертикальных участков. Когда две такие линии пересекаются под небольшим углом, то множество этих участков накладываются друг на друга. Рассмотрим рисунок 6, на котором изображена фигура, полученная с помощью программы на листинге 1.5. В центральной части линии многократно пересекаются, и если бы команда OVER не была включена, то центральный участок был бы черным от этих пересечений. Когда включена команда OVER, то включены те точки, которые лежат на пересечении нечетного числа линий, а остальные выключены. С другой стороны дыры по краям рисунка образованы точками, не лежащими ни на одной прямой.

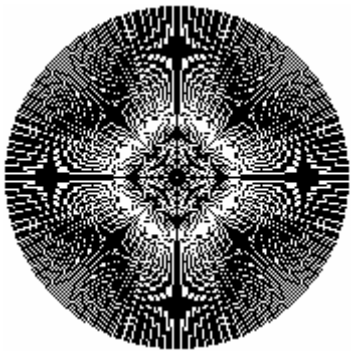


Рис. 6.

Упражнение 1.4

Модифицируйте текст программы на листинге 1.5 так, чтобы можно было вводить значение LINES и значение новой STRING переменной с помощью программы. Выясните, какую роль играет команда OVER и элементы, из которых состоят соседние линии.

Использование последовательно генерируемых случайных чисел для создания быстроменяющихся картин на экране - излюбленный прием для привлечения внимания к компьютерам. Этот прием применяется в программе на листинге 1.6, где используются команды RND и OVER.

LISTING 1.6

```
10 OVER 1
20 LET X=INT (RND *256)
30 LET Y=INT (RND *176)
40 PLOT X,Y
50 BEEP 0.05,(X-Y)/10
60 GO TO 20
```

Упражнение 1.5

Измените программу на листинге 1.6 с тем, чтобы она рисовала прямые, либо соединяющие случайные точки между собой, либо соединяющие случайные точки с центром (128,88).

В одном из предыдущих упражнений мы видим, что команда OVER позволяет изменять изображение на экране после выполнения каждой команды, даже если эта команда повторяется, например: повторное вычерчивание линии или точки. Команда OVER может использоваться для вычерчивания объекта на короткий промежуток времени - сначала рисуя его, а затем стирая. Программа на листинге 1.7 передвигает точку по экрану, сначала нанося ее программой PLOT в новой позиции, и тут же стирая ее со старой той же командой.

LISTING 1.7

```
10 OVER 1: PAPER 0: INK 7: BORDER 4: CLS
20 LET SPEED=2
30 LET X=0: LET Y=0
40 LET XADD=SPEED: LET YADD=SPEED
50 PLOT X,Y
60 LET OLDX=X: LET OLDY=Y
```

```
70 LET X=X+XADD
80 IF X>255-SPEED OR X<SPEED THEN LET XADD=-XADD
90 LET Y=Y+YADD
100 IF Y>174-SPEED OR Y<SPEED THEN LET YADD=-YADD
110 PLOT X,Y
120 PLOT OLDX,OLDY
130 GO TO 60
```

Эту программу можно дополнить с тем, чтобы можно было управлять с клавиатуры движением точки (листинг 1.8.). Расположенные вокруг "F" буквы нижнего регистра позволяют нам посылать точку в восьми различных направлениях. Если нажать клавишу "P", то точка при движении будет оставлять след; если нажать "Q", то следа не остается.

Этот способ приведения объекта в движение является важной и часто используемой техникой. Мы будем часто пользоваться этим приемом, как в программах игрового типа, так и в программах типа "курсор" главы 6.

LISTING 1.8

```
10 OVER 1
20 LET X=0: LET Y=0
30 PLOT X,Y
40 LET OLDX=X: LET OLDY=Y
50 LET XADD=0: LET YADD=0
60 LET A$=INKEY$ : IF A$="" THEN GO TO 60
70 IF A$="P" THEN OVER 0
80 IF A$="Q" THEN OVER 1
90 IF (A$="E" OR A$="D" OR A$="C") AND X>0 THEN LET XADD=-1
100 IF (A$="C" OR A$="V" OR A$="B") AND Y>0 THEN LET YADD=-1
110 IF (A$="T" OR A$="G" OR A$="B") AND X<255 THEN LET XADD=1
120 IF (A$="E" OR A$="R" OR A$="T") AND Y<175 THEN LET YADD=1
130 IF XADD=0 AND YADD=0 THEN GO TO 60
140 LET X=X+XADD: LET Y=Y+YADD
150 PLOT X,Y
160 PLOT OLDX,OLDY
170 GO TO 40
```

Хороших эффектов можно добиться, используя линии для увеличения или уменьшения областей, имеющих форму многоугольников. Программа на листинге 1.9 использует этот метод в комбинации с командой INVERSE для имитации эффекта быстрой трансфокации.

LISTING 1.9

```
10 LET I=0: INK 9
20 LET UP=175: LET ACROSS=255
30 LET X=0:LET Y=0
40 LET DIF=1
50 INVERSE 0
60 PLOT X,Y
70 DRAW 0,UP: DRAW ACROSS,0
80 PLOT X,Y
90 DRAW ACROSS,0: DRAW 0,UP
100 LET X=X+DIF: LET Y=Y+DIF
110 LET UP=UP-2*DIF
120 LET ACROSS=ACROSS-2*DIF
130 IF UP<0 OR UP=175 THEN LET DIF=-DIF: LET I=1-I
140 IF UP=175 THEN PAPER RND *7: CLS
150 GO TO 50
```

Упражнение 1.6

Нарисуйте квадрат 40 точек на 40 точек. Управляйте движением этой фигуры с помощью клавиатуры. Обратите внимание, что для этого достаточно управлять движением вершин квадрата.

Команды FLASH и BRIGHT (мерцание и яркость)

Рассмотрев возможности монохромной графики, обратимся к многоцветной. На экране SPECTRUMa могут присутствовать одновременно все краски, однако, внутри символьного блока используется не более двух цветов INK и PAPER. Эти цвета могут находиться в состоянии BRIGHT и (или) FLASH, кроме того, могут быть включены особые эффекты, такие как OVER и INVERSE. По команде FLASH для символьного блока цвета сменятся с INK на фоне PAPER на PAPER на фоне цвета INK каждую 1/3секунды.

При включенном состоянии BRIGHT для некоторого блока увеличивается яркость, как цвета INK, так и PAPER. В результате блоки с выключенным состоянием BRIGHT выглядят темнее.

Если вы хотите, чтобы существующее состояние блока не изменилось по команде PRINT, то для FLASH и BRIGHT следует установить значение, равное 8.

Атрибуты

Текущая комбинация состояний FLASH, BRIGHT, PAPER и INK для каждого символьного блока хранится в

памяти в так называемом "файле атрибутов", в котором для каждого блока отводится одна ячейка; этот файл располагается сразу за дисплейным файлом и начинается с ячейки с номером 22528. Модификация программы на листинге 1.1 позволяет непосредственно изменять состояние блоков, как мы это делали ранее для дисплейного файла.

LISTING 1.10

```
10 LET CORNER=22528
20 INPUT "VALUE-BIN ";LINE V$
30 LET VALUE= VAL ("BIN "+V$)
40 PRINT AT 0,0;"*"
50 POKE CORNER,VALUE
60 GO TO 20
```

Упражнение 1.7

Используйте программу на листинге 1.10 для того, чтобы изменить значение битов, хранящихся в первой ячейке файла атрибутов.

Переменной VALUE можно присвоить любое значение первой ячейки файла атрибутов, определяя, будут ли включены состояния FLASH и BRIGHT для первого символьного блока и какие цвета INK и PAPER используются в этом блоке. Состояниям блока соответствуют двоичные числа: FLASH - 0 или 1; BRIGHT - 0 или 1; PAPER -000до 111; INK-000до 111.

Значение VALUE можно вычислить, как и в примере на рис. 7.

		FLASH	BRIGHT	PAPER	INK	
Десятичное значение	=	1	0	5	2	
Двоичное значение	=	1	0	101	010	
BIN 10101010	=	128+	32+	8+	2	=170

Рис. 7.

В предыдущем примере содержимое ячейки соответствует состоянию с включенным FLASH, выключенным BRIGHT, цвет PAPER - циановый, цвет INK - красный. Атрибуты каждого элемента можно узнать, используя функцию ATTR (ROW,COLUMN). Число ROW означает номер строки точки, считая от верхнего края экрана. Те же параметры используются в команде PRINT AT, что позволяет без труда изменять содержимое ячеек файла атрибутов случайным образом (см. листинг 1.11). Такого же эффекта можно добиться, используя команду POKE.

LISTING 1.11

```
10 FLASH INT (RND *2)
20 BRIGHT INT (RND *2)
30 PAPER INT (RND *8)
40 INK INT (RND *8)
50 LET ROW=INT (RND *22): LET COL=INT (RND *32)
60 PRINT AT ROW,COL;"*"
70 BEEP 0.22,RND *40
80 GO TO 10
```

Упражнение.1.8

Измените эту программу так, чтобы случайным образом менялось содержимое одного блока и вычислялось с помощью функции ATTR значение FLASH, BRIGHT, PAPER, INK.

В таблице 1 приводится зависимость между величиной числа в ячейке файла атрибутов и состоянием символьного блока, соответствующего этой ячейке.

Таблица 1
Соответствие между атрибутами и величиной числа в ячейке файла атрибутов

PAPER	INK								MODE
	black	blue	red	magenta	green	yellow	cyan	white	
BLACK	0	1	2	3	4	5	6	7	NORMAL
	64	65	66	67	68	69	70	71	BRIGHT
	128	129	130	131	132	133	134	135	FLASH
	192	193	194	195	196	197	198	199	BRIGHT +FLASH
BLUE	8	9	10	11	12	13	14	15	NORMAL
	72	73	74	75	76	77	78	79	BRIGHT
	136	137	138	139	140	141	142	143	FLASH
	200	201	202	203	204	205	206	207	BRIGHT +FLASH
RED	16	17	18	19	20	21	22	23	NORMAL
	80	81	82	83	84	85	86	87	BRIGHT
	144	145	146	147	148	149	150	151	FLASH
	208	209	210	211	212	213	214	215	BRIGHT +FLASH
MAGENTA	24	25	26	27	28	29	30	31	NORMAL
	88	89	90	91	92	93	94	95	BRIGHT
	152	153	154	155	156	157	158	159	FLASH
	216	217	218	219	220	221	222	223	BRIGHT +FLASH

GREEN	32	33	34	35	36	37	38	39	NORMAL
	96	97	98	99	00	01	02	03	BRIGHT
	160	161	162	163	164	165	166	167	FLASH
	224	225	226	227	228	229	230	231	BRIGHT +FLASH
YELLOW	40	41	42	43	44	45	46	47	NORMAL
	104	05	06	07	08	09	10	11	BRIGHT
	168	169	170	171	172	173	174	175	FLASH
	232	233	234	235	236	237	238	239	BRIGHT +FLASH
CYAN	48	49	50	51	52	53	54	55	NORMAL
	112	113	114	115	116	117	118	119	BRIGHT
	176	177	178	179	180	181	182	183	FLASH
	240	241	242	243'	244	245	246	247	BRIGHT +FLASH
WHITE	56	57	58	59	60	61	62	63	NORMAL
	120	121	122	123	124	125	126	127	BRIGHT
	184	185	186	187	188	189	190	191	FLASH
	248	249	250	251	252	253	254	255	BRIGHT +FLASH

Мы можем считать, что каждый пиксель состоит из трех точек, расположенных в вершинах небольшого треугольника. Каждый элемент изображения содержит одну красную, одну синюю и одну зеленую точку, а ячейки файла атрибутов управляют яркостью свечения этих точек. Положим, что для некоторого пикселя в дисплейном файле содержится информация, что он должен иметь цвет INK. Последние три двоичных разряда значения переменной в ячейке, соответствующей символьному блоку, в котором содержится элемент изображения, определяют, какие из трех люминофоров следует высветить. В глазу человека содержится три типа рецепторов (для зеленого, красного и синего цветов). Цвет трех точек элемента изображения мы воспринимаем как одну точку смешанного типа. Если последние три разряда значения ячейки файла атрибутов - 111-7, то светится зеленая, красная, синяя, что соответствует белому цвету. Другие цвета можно определить по таблице 2.

Таблица 2

COLOUR	NUMBER	BINARY	ILLUMINATED DOTS
BLACK	0	000	
BLUE	1	001	BLUE
RED	2	010	RED
MAGENT	3	011	RED + BLUE
GREEN	4	100	GREEN
CYAN	5	101	GREEN + BLUE
YELLOW	6	110	GREEN + RED
WHITE	7	111	GREEN + RED + BLUE

Информация о цвете фона (PAPER) записана в разрядах 3-5 таким же образом. В шестом разряде содержится информация о том, включено BRIGHT или нет, и это определяет яркость свечения всех трех точек элемента изображения. Когда включено состояние FLASH, то цвета INK и PAPER будут меняться на PAPER и INK с частотой, зависящей от компьютера, для SPECTRUMa такая частота составляет примерно 1 /3 с.

При использовании монохромной графики высокой разрешающей способности мы не встречаем никаких сложностей, но при работе с многоцветной графикой следует помнить, что в символьном блоке одновременно могут присутствовать точки только двух цветов.

Упражнение 1.9

Поэкспериментируйте с различными цветами, используя программы, приведенные в данном разделе. Если вы используете телевизионный экран, а не дорогой монитор, то некоторые сочетания цветов покажутся вам излишними, однако на мониторе некоторые комбинации дисгармонирующих цветов, используемых для программы на листинге 1.5, создают впечатление волн, распространяющихся по экрану.

Простейшее движущееся изображение

Еще одну движущуюся картинку можно построить, используя графику низкой разрешающей способности, цветовые эффекты и команду FLASH. Первая часть программы на листинге 1.12 интересна тем, что после получения изображения им не надо управлять. На границе экрана расположена последовательность блоков с включенным FLASH, красным цветом PAPER и циановым цветом INK, чередующихся с блоками с включенным FLASH, циановым цветом PAPER и красным INK. Когда мы смотрим на эту картинку, то нам кажется, что красный и циановый цвета движутся.

LISTING 1.12

```
10 FLASH 1:PAPER 2: INK 5
20 FOR I=0 TO 1
30 FOR J=0 TO 15
```

```
40 PRINT AT 0,2*J+I;" "
50 PRINT AT 21,2*J+1-I;" "
60 NEXT J
70 FOR J=0 TO 10
80 PRINT AT 2*J+I,0;" "
90 PRINT AT 2*J+1-I,31;" "
100 NEXT J
110 PAPER 5: INK 2
120 NEXT I
130 FLASH 0:LET P=0
140 FOR I=1 TO 20
150 PRINT AT I,15; PAPER P;" "
160 LET P=P+1: IF P=7 THEN LET P=0
170 NEXT I
180 GO TO 140
```

Упражнение 1.10

Перепишите программу "прыгающей точки", используя графику низкой разрешающей способности. Переделайте и другие программы с движущимися объектами.

Команды CIRCLE и DRAW (окружность и чертить)

В SPECTRUMe имеются еще две встроенные команды графики с высокой разрешающей способностью. Это команды CIRCLE и DRAW для вычерчивания кривых.

По команде CIRCLE X,Y,P вычерчивается круг радиуса "P" в пикселях изображения с центром в точке (X,Y). Следует помнить, что после выполнения команды перо находится в правой части окружности, непосредственно под центром.

По команде DRAW X,Y,A чертится кривая из данной точки в точку, смещенную от нее на X,Y и поворачивающуюся на угол "A"; эта кривая будет дугой окружности. Угол поворота задается в радианах и изменяется в диапазоне от -PI до PI.

На листинге 1.13 приведена программа, использующая команду DRAW для построения фигуры на рис. 8.

LISTING 1.13

```
10 CIRCLE 126,88,80: LET N=24
20 LET A=0: LET ADIF=2+PI/N
30 FOR I=1 TO N
40 PLOT 128,88
50 LET X=40*COS A: LET Y=40*SIN A
60 DRAW X,Y,-PI: DRAW X,Y,PI
70 LET A=A+ADIF
80 NEXT I
90 STOP
```

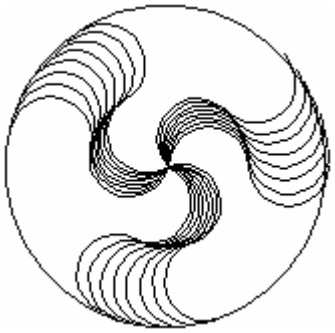


Рис. 8.

Цвета в символьном блоке

Использование различных цветов для INK в графике высокого разрешения может привести к неожиданным результатам. Пропустите программу на листинге 1.14, и вы увидите, к чему может привести непродуманное использование нескольких цветов.

LISTING 1.14

```
10 PAPER 5: INK 7: CLS
20 FOR I=0 TO 175 STEP 8
30 PLOT 0,I: DRAW 255,0
40 NEXT I
50 INK 0
60 CIRCLE 128,88,80
70 STOP
```

Из этого эффекта можно извлечь пользу, употребив его для получения быстроменяющихся сложных изображений, выполненных в графике с низкой разрешающей способностью. Вначале мы располагаем в фиксированных позициях блоки, окрашенные цветом INK. Любая проведенная линия изменит цвет элементов изображения, через которые она проходит. Эта техника работает с внушительной скоростью, в чем вы можете

убедиться, пропустив программу на листинге 1.15.

LISTING 1.15

```
10 INK 7: CLS
20 FOR I=1 TO 704: PRINT " ";: NEXT I
30 LET DIST=80: LET I=0: LET D=2
40 INK I
50 PLOT 127,86+DIST: DRAW -DIST,-DIST: DRAW DIST, DIST
60 PLOT 127,86-DIST: DRAW -DIST, DIST: DRAW DIST, DIST
70 LET DIST=DIST-D: IF DIST=0 OR DIST=80 THEN LET D=-D
80 LET I=I+1: IF I=8 THEN LET I=0
90 GO TO 40
```

Простая игра

Сейчас мы попробуем небольшую программу (листинг 1.16) как пример использования технологии, разобранной в этой главе. Червь может перемещаться по экрану шагами в одну знаковую позицию, горизонтально или вертикально, влево или вправо и управляется с клавиатуры. Цель игры: червь может съедать монетку (мишень). Съев монетку, червь удлиняется. Если хоть раз голова червя упрется в границу экрана или в собственное тело, червь погибает. Червь не может менять направление движения на противоположное. После десяти успешных поеданий монетки червь возвращается к своему первоначальному размеру, и игра продолжается.

Эта игра построена с использованием модульного структурного метода программирования. Этот метод помогает создавать надежные и понятные программы. Проще говоря, мы должны представить программу как последовательность малых задач, которые встраиваются блок за блоком в конечную программу. Для данной игры эти задачи могут быть определены так:

А. Инициализация переменных.

Б. Установка игрового поля.

В. Управление игрой.

Г. Рестарт и вывод счета игры.

Каждую из этих задач мы можем разбить на еще более мелкие. Например, задача "Б" может быть представлена так:

1. Воспроизведение монетки.
2. Использование клавиатуры для изменения направления движения червя.
3. Перемещение червя.
4. Фанфары.

Задача "3" в свою очередь может быть разбита на еще более мелкие подзадачи:

А) рисование червя;

Б) червь упирается в границу или сам в себя и погибает;

В) червь съедает монетку и растет.

Теперь попробуйте самостоятельно найти, где в программе реализованы эти задачи

LISTING 1.16

```
1000 DIM R(55):DIM C(55)
1009 REM Инициализация указателей подпрограмм и установка счета игры в 0
1010 LET FANFARE=2000: LET WORM=3000: LET KEY=4000: LET GOBBLE=5000: LET
STATUS=6000: LET TARGET=7000
1020 BORDER 1 :PAPER 7:INK 0
1030 LET HSC=0
1040 LET SCORE=0: LET WORMS=3: LET LEVEL=1
1048 REM Начало нового червя, длина 5 сегментов, начиная со строки R позиции C
1049 REM P-указатель на сегмент, который должен перемещаться Установка флагов
(0 - нет, не 0 - да)
1050 LET DEAD=0: LET WON=0: LET C=16: LET R=10: LET P=1: LET S=5: LET CMOVE=0:
LET RMOVE=1: LET H$="*"
1059 REM Установка "двигательных" переменных так, что червь перемещается вниз
1060 FOR I=1 TO 55: LET R(I)=-1: NEXT I
1069 REM Установить желтым экран сверху и снизу
1070 CLS: PRINT AT 0,0; PAPER 6: PRINT AT 21,0; PAPER 6
1079 REM ВЫВОД СЧЕТА ИГРЫ
1080 GO SUB STATUS
1090 GO SUB TARGET
1099 REM ГЛАВНЫЙ ЦИКЛ. Контроль за движением червя
1100 GO SUB KEY
1110 GO SUB WORM
1120 IF NOT DEAD AND NOT WON THEN GO TO 1100
1129 REM Если червь съел монету, 10 раз дать фанфары и начать новую игру.
1130 IF WON THEN LET LEVEL=LEVEL+1: GO SUB FANFARE: GO TO 1050
1139 REM Воспроизвести шум разрушения.
1140 IF DEAD THEN FOR I=1 TO 20: BEEP 0.05,15: NEXT I
1149 REM Если у вас остался еще червь, то начать новую игру
```

```
      1150 IF DEAD THEN LET WORMS=WORMS-1: IF WORMS<>0 THEN GO TO 1050
      1159 REM Переместить все сегменты, двигая указатель от конца червя
      1160 FOR I=1 TO S: BEEP 0.01,0: PRINT AT R(P),C(P);PAPER 7;" "
      1170 LET P=P+1: IF P>S THEN LET P=1
      1180 NEXT I
      1189 REM Переместить монетку и заменить строку со счетом
      1190 PRINT AT Y,X;PAPER 7;" ": GO SUB STATUS
      1199 REM Использовать мерцание и нажать Ввод для рестарта игры
      1200 LET I$=CHR$ 18+CHR$ 1+"PRESS ENTER TO START"+"GAME"+CHR$ 18+CHR$ 0
      1210 INPUT (I$+"   ");LINE A$
      1220 GO TO 1040
      2000 REM ФАНФАРЫ
      2010 DATA 0.06,18,0.06,19,0.06,21,0.15,27,0.06,21,0.20,27
      2019 REM Чтение и исполнение фонфар с комбинир. длительности
      2020 RESTORE FANFARE: FOR I=1 TO 6: READ L,T:BEEP L,T: NEXT I
      2030 RETURN
      3000 REM WORM (ЧЕРВЬ)
      3008 REM Червь ползет, перемещая последний сегмент вперед
      3009 REM Если это не приращение сегментов, то убрать старый сегмент
      3010 IF R(P)<>-1 THEN PRINT AT R(P),C(P); PAPER 7;" "
      3019 REM Вычисление новой позиции для головы червя.
      3020 LET R=R+RMOVE: LET C=C+CMOVE
      3029 REM Проверка на столкновение с границами
      3030 IF R>20 OR R<1 OR C>31 OR C<0 THEN LET DEAD=1: RETURN
      3039 REM Проверка на столкновение сам с собой.
      3040 IF ATTR (R,C)=16 THEN LET DEAD=1: RETURN
      3049 REM Установка строки и столбца в новой позиции
      3050 LET R(P)=R: LET C(P)=C
      3059 REM Проверка того, была ли съедена монетка
      3060 IF SCREEN$ (R,C)="$" THEN GO SUB GOBBLE
      3069 REM Добавление на экран еще одного сегмента и продвижение дальше к хвосту
указателя сегментов червя
      3070 PRINT AT R(P),C(P); PAPER 2;H$
      3080 LET P=P+1: IF P>S THEN LET P=1
      3090 RETURN
      4000 REM KEY (КЛАВИШИ)
      4009 REM Проверка используемых для управления клавиш
      4010 LET A$=INKEY$ : IF A$="" THEN RETURN
      4019 REM Установка всех букв, как показано ниже
      4020 IF CODE A$>96 THEN LET A$=CHR$ (CODE A$-32)
      4028 REM Червь может пойти влево или вправо, но не обратно на себя.
      4029 REM "ВВЕРХ" нажато - если вы не шли вниз, то пойдете вверх.
      4030 IF A$="I" AND CMOVE THEN LET RMOVE=-1: LET CMOVE=0: RETURN
      4039 REM Идти ВНИЗ (если червь не шел вверх)
      4040 IF A$="M" AND CMOVE THEN LET RMOVE=1: LET CMOVE=0: RETURN
      4049 REM Идти ВЛЕВО (если червь не шел вправо)
      4050 IF A$="J" AND RMOVE THEN LET RMOVE=0: LET CMOVE=-1: RETURN
      4059 REM Идти ВПРАВО (если червь не шел влево)
      4060 IF A$="L" AND RMOVE THEN LET RMOVE=0: LET CMOVE=1: RETURN
      4070 RETURN
      5000 REM GOBBLE (ПРОГЛАТЫВАНИЕ)
      5009 REM Поедается монетка, раздается звук глотания
      5010 FOR I=2 TO 4 STEP 0.5
      5020 BEEP 0.01,EXP I-10: NEXT I
      5029 REM Добавить к вашему счету и заменить строку со счетом
      5030 LET SCORE=SCORE+1: GO SUB STATUS
      5038 REM Добавить 5 сегментов
      5039 REM Если червь содержит 55 сегментов, это означает, что вы съели 10
монеток и выиграли
      5040 LET S=S+5: IF S=55 THEN LET WON=1: RETURN
      5049 REM Установить новую монетку на экран.
      5050 GO SUB TARGET
      5060 RETURN
      6000 REM СТАТУС
      6009 REM Если ваш счет превосходит наибольший в этой игре (HISCORE), то
заменить наибольший счет
```

```
6010 IF SCORE>HSC THEN LET HSC=SCORE
6019 REM Заменить строку с обеими счетами
6020 PRINT AT 0,0;PAPER 6; "  SCORE ";SCORE," HI-SCORE ";HSC
6030 PRINT AT 21,0;PAPER 6;" LEVEL ";LEVEL," WORMS ";WORMS
6040 RETURN
7000 REM TERCET (МОНЕТКА)
7009 REM Выбор знаковой позиции в игровом поле
7010 LET X=RND*31: LET Y=RND*19+1
7019 REM Проверка того, что это не та же самая позиция
7020 IF X=C AND Y=R THEN GO TO TARGET
7029 REM Проверка того, что эта позиция не находится под червем
7030 IF ATTR (Y,X)=16 THEN GO TO TARGET
7039 REM Ввод новой монетки на экран
7040 PRINT AT Y,X; PAPER 4;"$"
7050 RETURN
```

В следующей главе мы рассмотрим средства, необходимые для изображения реальных картин.

ПРИЛОЖЕНИЕ: управление в программах:

Листинг 1.8. Нажатие клавиш вокруг "F" - перемещает точку в одном из восьми направлений.

Листинг 1.16. Для управления червем используйте клавиши I, J, L, M.

ГЛАВА 2
От реальных координат к пикселям

Для изображения SPECTRUM представляет графический экран, т.е. матрицу адресуемых точек и пикселей. Пиксели сгруппированы в 256 вертикальных столбцов(NXPIX) и 176 горизонтальных строк (NYPIX). Установка NXPIX и NYPIX осуществляется парой целых чисел, называемых вектором пикселей (I,J), где 0<=I<=NXPIX-1 и 0<=J<=NYPIX-1. Этот вектор задает позицию пикселя в I-ом столбце и J-ой строке. Вектор (0,0) идентифицирует самый нижний левый пиксель экрана. SPECTRUM имеет специальные команды бейсика для оперирования матрицей пикселей. Они позволяют "включать" их. Это делает возможным аппроксимировать линии, или полигоны, или другие графические объекты последовательностью цветных точек.

В главе будут описаны возможности создания двумерных и трехмерных графических объектов на SPECTRUMe при помощи программ на бейсике. Эти элементарные подпрограммы приведены в данной главе.

Примитивы, обеспечивающие карту пространства на графическом экране

В целом компьютерная графика состоит из точек, линий, областей и координат вне прерывном двумерном и трехмерном евклидовом пространстве. Пиксельная графика довольно ограничена. Изображение объекта в дискретном поле очень усложняет практическое использование этого способа.

Рассмотрим отношение между двумерным реальным пространством и экранными пикселями. Мы можем вообразить себе двумерное пространство как плоскость. Для использования координатной геометрии (планиметрии) мы должны зафиксировать точку в этом пространстве, которую мы будем называть началом координат. Через начало координат проведем две взаимно перпендикулярные линии, X-ось и Y-ось. Теперь начертим шкалу вдоль каждой из осей. (см. рис. 9)

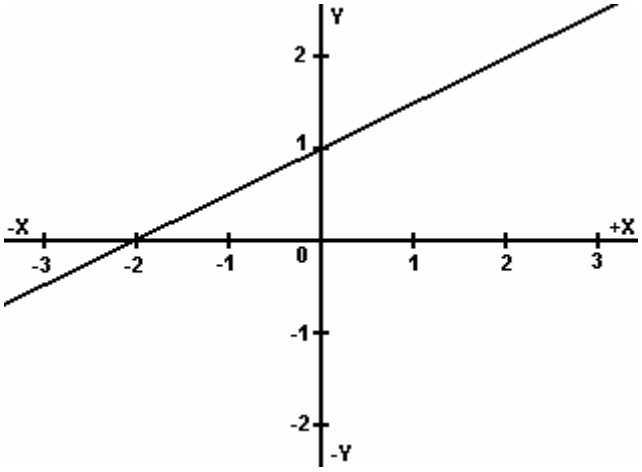


Рис. 9.

Мы полагаем, что значения на X-оси положительны справа, а слева отрицательны; значения на Y-оси положительны выше начала координат и отрицательны ниже.

Теперь позиция любой точки P на плоскости однозначно определяется ее координатами. Координата по X-оси, обозначим ее X, определяется как расстояние от начала координат до точки пересечения прямой, перпендикулярной оси X и проходящей через точку P с осью X, взятое с положительным знаком, если эта точка находится справа от начала координат, и с противоположным, если слева. Аналогичным образом определяется Y-координата. Эти два значения, называемые парой координат двумерного вектора, обычно записываются в скобках: (X,Y). Обратите внимание, что первой записывается X координата, за которой следует Y-координата. Как правило, мы будем называть эту пару вектором, размерность которого будет ясна из контекста. Кроме точки (X,Y), вектор определяет также и направление, которое параллельно прямой, соединяющей начало координат с точкой (X,Y); дополнительные сведения по этим вопросам вы найдете в главе 3.

Сейчас мы должны развить средства (упомянутые выше примитивные функции) для переноса геометрических понятий на состоящую из точек двумерную матрицу.

Здесь мы ограничимся случаем двух измерений: трехмерная геометрия рассматривается, начиная с главы 7. В обоих случаях нам потребуется отображение прямоугольной области двумерной действительной плоскости на дискретную графическую плоскость. Для простоты предположим, что стороны этой области параллельны координатным осям. Вначале предположим, что левый нижний угол прямоугольника находится в начале координат. (0.0,0.0), а длина горизонтальной стороны равна HORIZ,вертикальной - VERT; отождествим также начало координат с пикселем (0,0), а затем выберем такой масштаб прямоугольной области, чтобы она поместилась в дисплейном поле. Область может в точности совпадать с дисплейным окном, если только соотношение HORIZ: VERT и NXPIX: NYPIX равны. Эта ситуация встречается крайне редко, и поэтому мы используем масштабирующий множитель XYSCALE, который помещает точку (HORIZ,VERT) либо на правой стороне рамки дисплейного окна, либо на верхней. Мы можем считать этот прямоугольник окном в декартову плоскость; оно уже не привязано к началу координат, а может перемещаться по плоскости, при этом стороны остаются параллельными координатами осям; при таком движении это окно сканирует области плоскости, по форме совпадающие с первоначальной. Как правило, мы будем выбирать HORIZ примерно в полтора раза больше VERT.

В любой момент выполнения программы начало координат может быть передвинуто из первоначального

положения в левом нижнем углу в любую другую точку с координатами XORG и YORG, обозначающие значение X-координаты и Y-координаты точки. Точка (XORG,YORG) отождествляется с началом координат (0.0,0.0), а точка декартовой плоскости с координатами (XPT.YPT) отображается в элементе изображения с горизонтальной компонентой INT ((XORG+XPT) * XYSCALE+0.5) и вертикальной компонентой INT ((YORG+YPT) * XYSCALE+ 0.5). (INT - функция бейсика, возвращающая целое число, равное целой части вещественного аргумента).

Эти две компоненты хранятся функциями FNX и FNY (см. листинг 2.2). Во время рисования на экране мы имеем дело с точечным пером, управляемым парой целых чисел и перемещающимся по экрану; вначале это перо находится в точке (0,0), а в общем случае в точке (XPEN,YPEN). Константы NXPIX, NYPIX и переменные XYSCALE, XPEN, YPEX, XORIZ, YORIG должны быть доступными для приведенных ниже подпрограмм нанесения точек на экран в любой момент времени, так что эти имена нельзя использовать для других целей. Эти подпрограммы написаны специально для SPECTRUMa, но мы обсуждаем принципы составления аналогичных подпрограмм для других устройств. Перед тем, как использовать иное устройство, нам для начала пришлось бы изменить значения NXPIX и NYPIX.

Наша первая подпрограмма START присваивает значения переменным и подготавливает экран для работы. Листинг 2.1 содержит пример такой подпрограммы для SPECTRUM.

LI STING 2.1

```
9700 REM START
9701 REM IN - HORIZ,VERT
9702 REM OUT - NXPIX,NYPIX,XORIG,YORIG,XYSCALE,XPEN,YPEN
9710 LET XORIG=0: LET YORIG=0
9720 LET XPEN=0: LET YPEN=0
9730 LET NXPIX=256: LET NYPIX=176
9740 LET XYSCALE=NXPIX/HORIZ: LET YSCALE=NYPIX/VERT
9750 IF XYSCALE>YSCALE THEN LET XYSCALE=YSCALE
9760 RETURN
```

Если мы хотим использовать цветовые возможности, то нам следует ввести два дополнительных параметра COLPAP и COLINK (целые числа в диапазоне от 0 до 7) для определения цвета фона и цвета изображения. Соответствующий оператор:

```
9725 PAPER COLPAP: INK COLINK
```

Если мы хотим написать эту программу для другого компьютера, то необходимо заменить команды графики SPECTRUMa на эквивалентные команды для него.

В подпрограмме START и во многих других подпрограммах, приведенных ниже, нам потребуется переводить X и Y координаты точки в их аналог для пикселей, с этой целью мы вводим две функции FNX и FNY на листинге 2.2.

LISTING 2.2

```
9650 DEF FN X(Z)= INT ((XORIG+Z)*XYSCALE+0.5)
9660 DEF FN Y(Z)= INT ((YORIG+Z)*XYSCALE+0.5)
```

Следующая подпрограмма SETORIGIN позволяет передвигать начало координат на величину XMOVE по горизонтали и YMOVE по вертикали (в масштабе координатной системы) и соответствующим образом меняет величину (XORIG.YORIG). После этого перо устанавливается на пиксель, соответствующий новому началу координат.

LISTING 2.3

```
9600 REM SETORIGIN
9601 REM IN - XORIG,YORIG,XMOVE,YMOVE
9602 REM OUT - XORIG,YORIG,XPEN,YPEN
9610 LET XORIG=XORIG+XMOVE: LET YORIG=YORIG+YMOVE
9620 LET XPEN=FN X(0)
9630 LET YPEN=FN Y(0)
9640 RETURN
```

Мы сможем чертить прямые после того, как введем еще две функции: "MOVETO", которая помещает перо в элемент изображения, соответствующий началу отрезка, и "LINETO", которая проводит прямую из первоначального положения (задаваемого обращением к подпрограммам "SETORIGIN", "MOVETO", "LINETO") на пиксель, соответствующий другому концу отрезка. На листингах 2.4 и 2.5 приводятся тексты программ "LINETO" и "MOVETO", написанные специально для SPECTRUMa. Подпрограмма "LINETO" использует машинно-зависимую программу бейсика DRAW, для того, чтобы начертить прямую (обратите внимание на то, что PLOT использует абсолютные координаты, а DRAW - относительные); подпрограмма "MOVETO" является машинно-независимой. Таким образом, если вы хотите использовать эти программы для другого устройства, то вам придется модифицировать только подпрограмму "LINETO".

LISTING 2.4

```
9500 REM MOVETO
9501 REM IN - XPT,YPT
9502 REM OUT - XPEN,YPEN
9510 LET XPEN=FN X(XPT)
9520 LET YPEN=FN Y(YPT)
9530 RETURN
```

LISTING 2.5

```
9400 REM LINETO
```

```
9401 REM IN - XPT,YPT, XPEN.YPEN
9402 REM OUT- XPEN,YPEN
9410 LET NXPEN=FN X(XPT)
9420 LET NYPEN=FN Y(YPT)
9430 PLOT XPEN,YPEN
9440 DRAW NXPEN=XPEN, NYPEN=YPEN
9450 LET XPEN=NXPEN: LET YPEN=NYPEN
9460 RETURN
```

Во всех компьютерах, кроме самых примитивных, эти программы можно поместить в библиотеку или в фоновую память (BACKING STORE), что избавит вас от необходимости впечатывать их в тело каждой новой программы. В SPECTRUMe вы можете хранить эти программы в виде файлов на кассете (загружая их командой MERGE). У нас эти программы хранятся в библиотеке "LIBIA".

Упражнение 2.1

Отождествите прямоугольную область на декартовой плоскости со сторонами 30 ед. x 20 ед. с графическим полем дисплея SPECTRUMa нарисуйте в центре этого прямоугольника квадрат со стороной 15 ед.

Мы помещаем квадрат в центр, помещая начало координат в точке (±7.5; ±7.5). См. листинг 2.6.

LISTING 2.6

```
100 REM SQUARE
109 REM SETUR IDENTIFIERS TO GRAPHICS ROUTINES.
110 LET START=9700: LET SETORIGIN=9600: LET MOVETO=9500: LET LINETO=9400
119 REM DEFINE GRAPHCS AREA.
120 LET HORIZ=30: LET VERT=20
130 GO SUB START
140 LET XMOVE=HORIZ*0.5: LET YMOVE=VERT*0.5
150 GO SUB SETORIGIN
159 REM JOIN CORNERS OF SQUARE IN ORDER.
160 LET XPT=7.5: LET YPT=7.5: GO SUB MOVETO
170 LET XPT=-7.5: LET YET=7.5: GO SUB LINETO
180 LET XPT=-7.5: LET YPT=-7.5: GO SUB LINETO
190 LET XPT=7.5: LET YPT=-7.5: GO SUB LINETO
200 LET XPT=7.5: LET YPT=7.5: GO SUB LINETO
210 STOP
```

Отметим, что порядок соединения вершин в этой задаче существенен.

Теперь мы составим программу "POLIGON", которая чертит многоугольники, используя команды SPECTRUMa для вычерчивания прямых (см. листинг 2.7).

LISTING 2.7

```
100 REM MAIN PROGRAM/CALLING POLIGON
110 LET START=9700: LET SETORIGIN=9600: LET MOVETO=9500: LET LINETO=9400: LET
POLIGON=300
120 LET HORIZ=30: LET VERT=20
130 GO SUB START
140 LET XMOVE=0: LET YMOVE=0
150 GO SUB SETORIGIN
159 REM DECLARE AND INPUT VERTICES OF POLIGON.
160 DIM X(50): DIM Y(50)
170 INPUT "NUMBER OF VERTICES-";NPOL
180 FOR I=1 TO NPOL
190 INPUT ("X("+STR$ I+")-");X(I);(" Y("+STR$ I+")=");Y(I)
200 NEXT I
210 GO SUB POLIGON
220 STOP
300 REM POLIGON
301 REM IN - NPOL,X(POL),Y(POL)
310 FOR I=1 TO NPOL
320 LET XPT=X(I): LET YPT=Y(I): GO SUB MOVETO
330 LET XPT=X(I+1): LET YPT=Y(I+1): IF I=NPOL THEN LET XPT=X(1): LET YPT=Y(1)
340 GO SUB LINETO
350 NEXT I
360 RETURN
```

Упражнение 2.2

Если вы используете SPECTRUM, то вы можете рисовать орнаменты с различным сочетанием цветов, но для этого необходимо вначале установить рисующий цвет INK. Напишите подпрограмму "SETING", выполняющую эту функцию с одним входным параметром COLINK.

Упражнение 2.3

Во всех описанных выше подпрограммах значение масштабного множителя XYSCALE фиксировано и одинаково для обеих осей. Введите два масштабных фактора XSCALE и YSCALE и напишите подпрограмму

"FACTOR", которая бы изменяла горизонтальную шкалу подпрограммой FX, а вертикальную шкалу подпрограммой FY. Вам придется изменить программы ("START", "SETORIGIN", "MOVETO", "LINETO", см. главу 6).

Упражнение 2.4

Нет никаких особых оснований считать X-ось горизонталью, а Y-ось - вертикалью; они могут даже не быть перпендикулярными. Поэкспериментируйте с этим, для этого вам придется изменить все программы "START", "MOVETO", и т.д.

Пример 2.A

Один из первых популярных графических пакетов назывался CALCOMP. Он включает подпрограммы для вычерчивания осей и шкал, для построения графиков и много других полезных подпрограмм.

Структура этих подпрограмм во многом определяется подпрограммой "PLOT" (не путать с командой PLOT), программа PLOT содержит три параметра: два действительных XPT и YPT, координаты точки в пространстве и информацию о движении MOVE - целое число, принимающее значения ±2 и ±3. Одна эта подпрограмма заменяет три наших - "SETORIGIN", "MOVETO", "LINETO". Если MOVE отрицательно, то начало координат помещается в точку (XPT, YPT) старой системы координат, и в этом случае программа "PLOT" эквивалентна "SETORIGIN". Если абсолютное значение MOVE равно 3, то движение пера происходит без вычерчивания прямой, эквивалент MOVETO, если MOVE равно 2, то чертится линия - эквивалент LINETO. Если мы даже не хотим использовать весь пакет CALCOMP, то мы все равно можем заменить подпрограммы "SOTORIGIN", "MOVETO", "LINETO" подпрограммой "PLOT" и использовать ее с последующими в этой главе подпрограммами - см. листинг 2.8.

LISTING 2.8

```
9800 REM PLOT/CALCOMP
9801 REM IN - XPT,YPT,XPEN,YPEN,XORIG,YORIG,MOVE
9802 REM OUT - XPEN,YPEN,XORIG,YORIG
9810 LET NXPEN=FN X(XPT)
9820 LET NYPEN=FN Y(YPT)
9830 IF ABS (MOVE)=2 THEN PLOT XPEN,YPEN: DRAW NXPEN-XPEN,NYPEN-YPEN
9840 LET XPEN=NXPEN: LET YPEN=NYPEN
9850 IF MOVE<0 THEN LET XORIG=XORIG+XPT: LET YORIG=YORIG+YPT
9860 RETURN
```

Чтобы продемонстрировать практическую пользу от этих функций, нарисуем несколько простейших орнаментов. Некоторые считают рисование орнаментов уделом легкомысленных людей, но мы считаем, что это весьма полезное упражнение на первой стадии ознакомления с машинной графикой. Очень часто весьма сложные рисунки являются результатом простых программ. Быстрое получение результатов стимулирует интерес начинающих и помогает им обрести уверенность в своих силах. Более того, на новый дизайн всегда спрос: он используется для оформления обложек книг, буклетов, в рекламе. Не принесет никакого вреда поупражняться в составлении орнаментов, что может оказаться полезным при дальнейшем изучении графического представления информации. Орнаменты также являются идеальным способом введения некоторых фундаментальных идей машинной графики, делая их вполне осязаемыми. В качестве примера приведем программу, иллюстрирующую важную роль тригонометрических функций (синуса и косинуса) и роль измерения углов в радианах. Напомним, что Pi-радиан эквивалентно 180 градусам.

Упражнение 2.5

Фигура на рис. 10 получается, если соединить между собой все вершины правильного N-угольника, где N=10.

Мы помещаем начало координат в центр, а вершины располагаем на единичной окружности с центром в начале координат: значения HORIZ и VERT (3,2.1) выбираются так, чтобы рисунок хорошо поместился на экране. Если одна из вершин лежит на оси X (горизонтали), то все вершины являются векторами вида (COS(ALPHA), SIN(ALPHA)), где ALPHA равно (2*PI/N), а I=1....N. В этой программе впервые в данной книге координаты точек вычисляются в программе, а не задаются программистом, как это делается в программе 2.6, Далее, так как эти числа используются программой неоднократно, то имеет смысл их поместить в массив и обращаться к ним с помощью индекса. Обратите внимание, что в программе на листинге 2.9 точка с индексом J не соединяется с точкой с индексом I, где 0 <=I < J <= N, так как эти точки уже были соединены ранее.

LISTING 2.9

```
100 REM JOINING VERTICES OF REGULAR N-GON
110 LET START=9700: LET SETORIGIN=9600: LET MOVETO=9500: LET LINETO=9400
120 LET HORIZ=3: LET VERT =2.1
130 GO SUB START
140 LET XMOVE=HORIZ*0.5: LET YMOVE=VERT*0.5
150 GO SUB SETORIGIN
160 DIM X(30):DIM Y(30)
169 REM SETUR vertices of regular N-GON in ARRAYS X and Y.
170 INPUT "VALUE OF N= ";N
180 LET ALPHA=0: LET ADIF=2*PI/N
190 FOR I=1 TO N
200 LET X(I)=COS ALPHA: LET Y(I)=SIN ALPHA
210 LET ALPHA=ALPHA-ADIF
220 NEXT I
229 REM JOINT POINT I TO POINT J-1<=I<J<=N.
230 FOR I=1 TO N-1
```

```
240 FOR J=I+1 TO N
250 LET XPT=X(I): LET YPT=Y(I): GO SUB MOVETO
260 LET XPT=X(J): LET YPT=Y(J): GO SUB LINETO
270 NEXT J
280 NEXT I
290 STOP
```

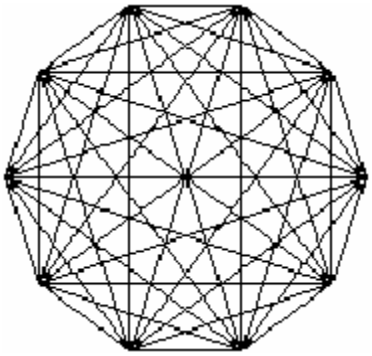


Рис. 10.

В этом простом примере выявляются два обстоятельства, первое относится к разрешающей способности. Так как экран состоит из дискретных точек, то прямые линии приходится аппроксимировать последовательностью точек. К сожалению, разрешающая способность SPECTRUMa, как и большинства микрокомпьютерных систем, низка(что означает, NXPIX и NYPIX величины порядка сотни) поэтому на прямых видны зубчики; на устройствах с более высокой разрешающей способностью эти зубчики становятся неразличимыми невооруженным глазом.

Второе наблюдение - это то, что по мере увеличения в программе на листинге 2.9 контуры N-угольника начинают напоминать окружность. Мы можем использовать это наблюдение с тем, чтобы рисовать окружности, что и делает программа "CIRCLE" (листинг 2.10A), которая рисует окружность радиуса R с центром в точке (XCENT,YCENT) (см. рис. 11).

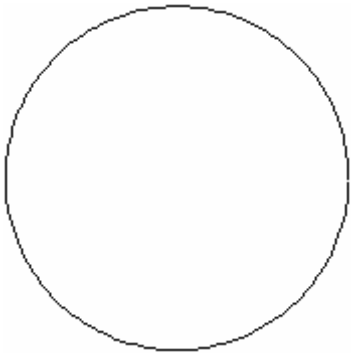


Рис. 11.

Обратите внимание на то, что мы измеряем углы в радианах, увеличивая угол на величину 3/(R*XYSCALE) на каждом шаге цикла; эта величина зависит только от радиуса и позволяет построить окружность без лишних действий. Так как каждую точку мы используем только раз, то мы не используем массивы. Как и в случае с прямой, форма окружности искажена из-за ограниченной разрешающей способности дисплея.

LISTING 2.10A

```
10 REM MAIN PROGRAM/CIRCLE1
20 LET START=9700: LET SETORIGIN=9600: LET MOVETO=9500: LET LINETO=9400: LET
CIRCLE1=300
30 LET HORIZ=30: LET VERT=20
40 GO SUB START
50 INPUT " XCENT=";XCENT;" YCENT=";YCENT;" R=";R
60 GO SUB CIRCLE1
70 STOP
300 REM CIRCLE1
301 REM IN - XCENT,YCENT,R.XYSCALE
310 LET XMOVE=XCENT: LET YMOVE=YCENT: GO SUB SETORIGIN
320 LET ADIF=3/(R*XYSCALE)
330 LET XPT=R: LET YPT=0: GO SUB MOVETO
339 REM CALCULATE and JOIN points (XPT,YPT) around the circle.
340 FOR A=ADIF TO 2*PI STEP ADIF
350 LET XPT=R*COS A: LET YPT=R*SIN A: GO SUB LINETO
360 NEXT A
370 RETURN
```

LISTING 2.10Б

```
10 REM MAIN PROGRAM/CIRCLE2
20 LET START=9700: LET circle2=400
```

```
30 LET HORIZ=30: LET VERT=20
40 GO SUB START
50 INPUT " XCENT= ";XCENT;" YCENT= ";YCENT;" R= ";R
60 GO SUB circle2
70 STOP
400 REM CIRCLE2
401 REM IN- XCENT,YCENT, R,XYSCALE
410 CIRCLE FN X(XCENT), FN Y(YCENT), R*XYSCALE
420 RETURN
```

Мы видели, что SPECTRUM - бейсик содержит функцию CIRCLE, с помощью которой можно чертить окружности. Мы можем использовать эту функцию для составления программы CIRCLE2 (листинг 2.10Б), более эффективной, чем CIRCLE 1.

Упражнение 2.6

Напишите подпрограмму, вычерчивающую эллипс с большой осью "A" единиц (горизонтальной) и малой осью "B" единиц (вертикальной); точка на эллипсе задается, как (A*COS ALPHA, B*SIN ALPHA), где 0<=ALPHA<=2PI.

Следует помнить, однако, что ALPHA - не угол между прямой, проходящей через точку и положительным направлением оси X, это просто параметр для задания эллипса. Используйте эту подпрограмму для вычерчивания фигуры.

Пример 2.Б

Выполнив предыдущие упражнения, естественно перейти к построению спирали. Эта кривая также имеет вид (R*COS ALPHA, R*SIN ALPHA), но в данном случае изменяется от B до B+2*N*PI, где B (параметр BETA) - начальный угол, который спираль образует с положительным направлением оси X, а N - число витков спирали. Теперь радиус R - это не константа, R вычисляется по формуле

```
R=RMAX*(ALPHA-BETA)/2*N*PI
```

Отметим, что эта подпрограмма, помещая центр спирали в точку (XCENT, YCENT), не вносит побочных эффектов, т.к. перед выходом из подпрограммы мы возвращаем начало координат в исходную точку.

LISTING 2.11A

```
10 REM MAIN PROGRAM/SPIRAL1
20 LET START=9700: LET SETORIGIN=9600: LET LINETO=9400: LET SPIRAL1=300
30 LET HORIZ=30: LET VERT=20
40 GO  SUB START
50 INPUT " XCENT= ";XCENT; " YCENT= ";YCENT;" RMAX= ";RMAX
60 INPUT " N=" ;N; " BETA=";BETA
70 GO SUB SPIRAL1
80 STOP
300 REM SPIRAL1
301 REM IN - XCENT,YCENT,RMAX,N,BETA
310 LET XMOVE=XCENT: LET YMOVE=YCENT: GO SUB SETORIGIN
320 LET ADIF=PI/50: LET ALPHA=BETA
330 LET RDIF=RMAX/(N*100)
340 FOR R=RDIF TO RMAX STEP RDIF
350 LET XPT=R*COS ALPHA: LET YPT=R*SIN ALPHA: GO SUB LINETO
360 LET ALPHA=ALPHA+ADIF
370 NEXT R
380 LET XMOVE=XCENT: LET YMOVE=YCENT: GO SUB SETORIGIN
390 RETURN
```

LISTING 2.11Б

```
10 REM MAIN PROGRAM/CELTIC
20 LET START=9700: LET SETORIGIN=9600: LET CELTIC=300
30 LET HORIZ=30: LET VERT=20
40 GO SUB START
50 INPUT " XCENT= ";XCENT;" YCENT= ";YCENT;" RMAX= ";RMAX
60 INPUT " N= ";N;" SIGN= ";SIGN
70 GO SUB CELTIC
80 STOP
300 REM CELTIC/SPIRAL
301 REM IN - XCENT,YCENT,RMAX,N,SIGN
310 LET XMOVE=XCENT: LET YMOVE=YCENT: GO SUB SETORIGIN
320 PLOT XPEN,YPEN
330 LET R=0: LET S=1: LET RDIF=RMAX/(N*2)
339 REM Construct the spiral using draw to produce a series of semicircles
340 FOR I=1 TO N*2
350 LET R=R+RDIF: LET XPIX=S*R*XYSCALE
360 DRAW XPIX,XPIX,SIGN*PI
370 LET S=-S
```

```
380 NEXT I
390 RETURN
```

Упражнение 2.7

Программа на листинге 2.11А рисует спираль (рис. 12) с XCENT=15, YCENT=10, N=6, BETA=2, RMAX=5. Что произойдет, если положить RMAX=-3? Обратите внимание на то, что если вращать голову перед этой спиралью, оставляя линию глаз параллельной горизонтали экрана, то кажется, что спираль вращается относительно центра.

Рис. 12.

Пример 2.В

Спирали более тысячи лет используются в искусстве и дизайнерских работах. Большая часть древних спиралей, однако, не являются спиралями, а составлены из полуокружностей. Программа на листинге 2.11Б (использующая параметры RMAX, N, SIGN=±1 - параметр, определяющий ориентацию кривой) позволяет рисовать такие составные спирали; программа использует функцию DRAW и много эффективнее точного метода программы на листинге 2.11 А.

Упражнение 2.8

Просмотрите программу на листинге 2.11Б и измените её так, чтобы конец кривой лежал не на оси X, а образовывал с ней угол BETA. В книге кельтов содержатся рисунки кривой, называющейся трискелом, состоящей из последовательности дуг 1/3 окружности. Попробуйте составить кривую, состоящую из дуг 1/4 окружности, и т.д.

Пример 2.Г

Напишите подпрограмму (листинг 2.12), которая рисует фигуру, изображенную на рис. 13.

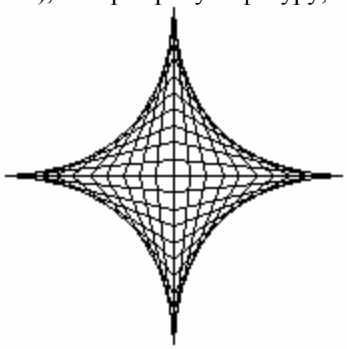


Рис. 13.

LISTING 2.12

```
100 REM AN ENVELOPE
110 LET START=9700: LET SETORIGIN=9600: LET MOVETO=9500: LET LINETO=9400
120 LET HORIZ=3: LET VERT=2.1
130 GO SUB START
140 LET XMOVE=HORIZ*0.5: LET YMOVE=VERT*0.5
150 GO SUB SETORIGIN
159 REM DRAW UNIT AXES IN GRAPHICS AREA.
160 INPUT "N=";N
170 LET XPT=1:LET YPT=0: GO SUB MOVETO
180 LET XPT=-1: LET YPT=0: GO SUB LINETO
190 LET XPT=0: LET YPT=1: GO SUB MOVETO
200 LET XPT=0: LET YPT=-1: GO SUB LINETO
209 REM PRODUCE N SETS EACH OF FOUR POINTS. ONE ON EACH AXIS. JOIN THE POINTS
OF EACH SET IN ORDER.
210 FOR I=1 TO N
220 LET ID1=I/N: LET ID2=(N+1-I)/N
230 LET XPT=ID1: LET YPT=0: GO SUB MOVETO
240 LET XPT=0: LET YPT=ID2: GO SUB LINETO
250 LET XPT=-ID1: LET YPT=0: GO SUB LINETO
260 LET XPT=0: LET YPT=-ID2: GO SUB LINETO
270 LET XPT=ID1: LET YPT=0: GO SUB LINETO
280 NEXT I
290 STOP
```

Здесь мы вводим понятие развертки, вместо того, чтобы составлять кривую из последовательности небольших отрезков (как это делается на листинге 2.9), мы рисуем кривую с помощью касательных к кривой. N точек наносится с единичным интервалом на каждый из четырех лучей, которые делят плоскость на четыре части. Таким образом, координаты этих четырех точек (±I/N,0.) и (0.0,±I/N), где I=1,2…,N.

Упражнение 2.9

Обобщите результаты предыдущего упражнения, считая, что плоскость разбита M-лучами, выходящими из центра, на сектора с равными углами.

Упражнение 2.10

Напишем программу, рисующую такую фигуру: Входным параметром программы служит целое число N. Нанесем 4N точек по периметру квадрата с единичной стороной; каждая из этих точек P(I): (1-1,2…, 4N) расположены на расстоянии 1/N друг от друга. Затем эти точки соединяются друг с другом по такому правилу: точка P(I) соединяется с точкой P(J) для всех положительных I, не больших 4, так что числа J-I (вычитание по модулю 4) принадлежат последовательности I,I+2,I+2+3,… Например, если N-10, то P(20) соединяется с P(21),P(23),P(26),P(30),P(35),P(1),P(8),P(16). Следует вначале начертить квадрат, что избавит от необходимости соединять точки, находящиеся на одной стороне.

Пример 2.Д

Создайте программу, имитирующую спирограф, рисующий фигуры, подобные фигуре, приведенной на рис. 14.

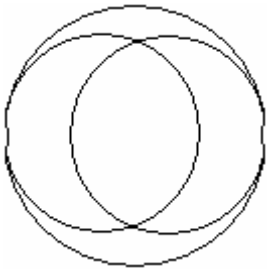


Рис. 14.

Спирограф - это устройство, состоящее из зубчатого диска, расположенного внутри зубчатой окружности, находящейся на листе бумаги.

Пусть радиус внешней окружности равен целому A, а радиус внутренней – целому числу B. В диске имеется небольшое отверстие, расположенное на расстоянии D (D - тоже целое число) от центра. В это отверстие вставляется кончик карандаша. Диск двигают внутри окружности против часовой стрелки, так что он все время касается окружности; зубцы гарантируют отсутствие проскальзывания. Кончик карандаша вырисовывает орнамент, который считается нарисованным, как только кончик карандаша возвращается в исходную точку.

Предположим, что вначале центр диска, окружности и отверстия в диске расположены на оси X, причем центр окружности совпадает с началом координат.

Для того, чтобы имитировать работу спирографа, мы параметрическим образом зададим кривую, которую вычерчивает кончик карандаша. Пусть ALPHA - угол между положительным направлением оси X и лучом, соединяющим точку касания диска и окружности. Тогда координаты точки касания - (A*COS ALPHA, A*SIN ALPHA), а координаты центра окружности ((A-B) GOS ALPHA, (A-B) SIN ALPHA). Пусть BETA обозначает угол, который образует положительное направление оси X и прямая, соединяющая отверстие с центром диска, тогда координаты отверстия:

```
((A-B) COS ALPHA + D*COS BETA, (A-B) SIN ALPHA + D*SIN BETA)
```

Когда точка соприкосновения между диском и окружностью перемещается по окружности на расстояние A*ALPHA, то оно перемещается на -B*BETA по диску (знак минус появляется из-за того, что у ALPHA и BETA противоположная ориентация). Так как проскальзывание отсутствует, то эти расстояния должны совпадать и, следовательно, BETA=-(A/B)*ALPHA. Карандаш вернется в исходную точку, когда ALPHA и BETA оба кратны 2*PI. Когда ALPHA=2*N*PI, тогда BETA=-N*(A/B)*2*PI; и мы видим, что кончик карандаша вернется в исходное положение в первый раз, когда N*(A/B) в первый раз станет целым; это произойдет, когда станет равным B, деленному на наибольший общий множитель A и B. Подпрограмма "EUCLID" (листинг 2.13) использует алгоритм Евклида для вычисления наибольшего общего множителя (целая HCF) двух целых положительных величин A и B.

Эта функция используется в подпрограмме "SPIRO" (листинг 2.13), вычисляющей значение N, а затем вычисляющей значение BETA для каждого значения ALPHA, где ALPHA изменяется в диапазоне от 0 до 2*PI с шагом PI/100, после чего вычерчивается кривая.

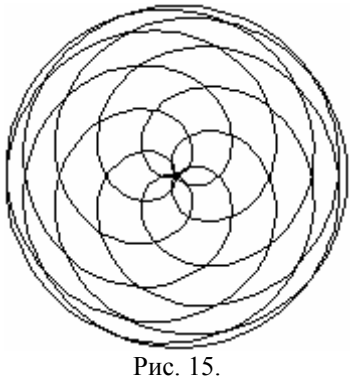


Рис. 15.

Рисунок 15 был получен обращением к подпрограмме "SPIRO" со значениями параметров A=12, B=7, D=5. Значения HORIZ и VERT выбираются такими, чтобы рисунок уместился на экране; в данном случае HORIZ=30, а VERT=20.

LISTING 2.13

```
10 REM MAIN PROGRAM/SPIRO
20 LET START=9700: LET SETORIGIN=9600: LET MOVETO=9500: LET LINETO=9400: LET EUCLID=200: LET SPIRO=300
30 LET HORIZ=30: LET VERT=20
40 GO SUB START
50 LET XMOVE=HORIZ*0.5: LET YMOVE=VERT*0.5: GO SUB SETORIGIN
60 INPUT" A=";A;" B=";B;" D=";D
70 GO SUB SPIRO
80 STOP
200 REM EUCLID
201 REM IN - A,B
202 REM OUT - HCF
210 LET I=A: LET HCF=B
220 IF A<B THEN LET I=B: LET HCF=A
230 LET J=I-INT (I/HCF)*HCF
240 IF J=0 THEN RETURN
250 LET I=HCF: LET HCF=J: GO TO 230
300 REM SPIRO
301 REM IN - A,B,D
310 LET RAB=A-B: LET ALPHA=0: LET ADIF=PI/50: LET ACB=A/B
320 GO SUB EUCLID: LET N=B/HCF: LET NC=100*N
330 LET XPT=RAB+D: LET YPT=0: GO SUB MOVETO
339 REM CALCULATE AND JOIN POINTES(XPT,YPT).
340 FOR I=1 TO NC
350 LET ALPHA=ALPHA+ADIF
360 LET BETA=ALPHA*ACB
370 LET XPT=RAB*COS ALPHA+D*COS BETA
380 LET YPT=RAB*SIN ALPHA+D*SIN BETA
390 GO SUB LINETO
400 NEXT I
410 RETURN
```

Эта программа показывает, что построение орнаментов не такое бесхитростное занятие, каким оно кажется на первый взгляд. Даже такая простая фигура, как на рис. 15, требует привлечения алгоритма Евклида. Углубляясь в предмет машинной графики, мы убедимся, что весьма полезным может оказаться знание геометрии, алгебры, теории чисел. Будьте готовы к тому, что вам придется зайти в библиотеку (или к знакомому математику) для получения дополнительной информации.

Список программ

На этом шаге мы объединим листинги 2.1 ("START"), 2.2 (две функции FNX, FNY), 2.3 ("SETORIGIN"), 2.4 ("MOVETO"), 2.5 ("LINETO") подзаголовком "LIBl". Позже мы заменим листинг 2.5 листингом 3.3 ("CLIP" и новая версия программы "LINETO").

1. "LIBl" и листинг 2.6 ("SQUARE" - вычерчивание квадрата): никаких входных данных;

2. "LIB1" и листинг 2.7 (MAIN PROGRAM - основная программа и "POLIGON" - многоугольник): требуется задать количество вершин многоугольника 3<=N<=50 и их координаты X, Y (-15<=X<=15 и -10<=Y<=10).

3. "LIBl" и листинг 2.9 ("JOINING VERTICES OF REGULAR N-GON" – соединить вершины правильного N-угольника): требуется задать целое число 3<=N<=50.

4. " LIBl" и ваша собственная "MAIN PROGRAM"- основная программа (в качестве примера см. листинг2.7), вызывающая листинги 2.10А "CIRCLE1" и 2.10Б "CIRCLE2". Для каждой подпрограммы должны быть указаны центр (XCENTER, YCENTER) и радиус R. Выбирайте эти значения так, чтобы они были согласованы со значениями HORIZ, VERT, XMOVE, YMOVE; например, присвойте последним значения 30, 20, 15 и 10 соответственно. В качестве примера вызовите "CIRCLE1" с центром (1,-1), радиусом 8 и "CIRCLE2" с центром (1,2) и радиусом 5.

5. "LIB1" и ваша собственная основная программа, вызывающая листинг 2.11А ("SPIRAL1") и 2.11Б ("CELTIC"). Для каждой из этих подпрограмм следует задать центр (XCENT, YCENT), максимальный радиус RMAX и количество витков спирали N. Для листинга 2.11А ("SPIRAL1") требуется задать также угол BETA, в то время, как для программы 2.11Б требуется задать SIGN, которое принимает значения ±1. Выбирайте значения параметров такими, чтобы они согласовывались со значениями HORIZ, VERT, XMOVE, YMOVE (например, 30, 20, 15 и 10). Например, вызовите "SPIRAL1" с центром (1,-1), RMAX=5, N=5, SIGN=-1. Попробуйте SIGN=±1.

6. "LIB1" и листинг 2.12 ("ENVELORE" - развертка): задайте число N: 2<=N<=30.

7. "LIB1" и листинг 2.13 ("EUCLID" и "SPIRO"): требуется задать три числа A, B и D. Выберите значения HORIZ, VERT и т.д. такими, чтобы изображение уместилось на экране: положите XMOVE=HORIZ*0.5, YMOVE=0.5*VERT (для подпрограммы "SETORIGIN", где HORIZ и VERT больше 2*(A-B+D).

ГЛАВА 3
Геометрия декартовой плоскости

Во второй главе мы ввели понятие двумерной прямоугольной системы координат; мы определили точки пространства как вектора и научились проводить через них прямые. Строго говоря, прямая простирается в бесконечность в обоих направлениях, так что нам необходимо указать способы записи того, что точка принадлежит такой линии.

Мы знаем, что уравнение прямой есть

Y-M*X+C,

где M - тангенс угла между прямой и положительным направлением оси X, C – точка пересечения прямой с осью Y; что означает, что Y-C когда XЧ). Эта формула широко известна, но она не всегда пригодна. Что произойдет, если прямая вертикальна? M обратится в бесконечность! Более общей формулой является:

A*Y-B*X+C

С помощью этой формулы можно задавать любые прямые; если прямая вертикальна A-0; (B/A) равно тангенсу угла между прямой и положительным направлением оси X; если A-0, то прямая пересекает ось Y в точке (C/A), а если B-0, то ось X в точке (-C/B). Прямая параллельна оси Y, если A-0 и параллельна оси X, если B-0.

Мы будем часто пользоваться этой формулой в дальнейшем, однако введем еще один, быть может более практичный способ определения прямой.

Определим сначала умножение вектора на скаляр и операцию сложения векторов. Пусть даны два вектора P(1) - (X(1),Y(1)) и P(2) - (X(2),Y(2)), тогда произведение вектора на скаляр K*P(1) - (K*X(1),K*Y(1)) - равно вектору, составленному из компонента вектора, умноженных на скаляр K.

Сумма векторов определяется так:

P(1)+P(2)-(X(1)+X(2),Y(1)+Y(2))

Складываем X-компоненты друг с другом и Y-компоненты друг с другом, абсолютные значения P(1)=X(l)+Y(1) -это расстояние от начала координат до точки P (оно также называется длиной вектора, нормой вектора). Для того, чтобы задать прямую, выберем сначала на ней две точки P(1)=(X(1),Y(1)) и P(2)=(X(2),Y(2)). Произвольная точка прямой P(MU)=(X,Y), задается как комбинация умножения на скаляр и сложения векторов:

(1-MU)*P(1)+MU*P(2),

где MU-действительное число, что дает вектор

((1-MU)*X(1)+MU*X(2),(1-MU)*Y(1)+MU*Y(2))

Мы поместили MU в скобках после P, чтобы показать, что P зависит от MU. После того, как мы разберемся в этой зависимости получше, мы будем опускать символ (MU). Если 0<=MU<=1, тогда P(MU) лежит между P(1)и P(2). Для любой точки P(MU) значение MU задается с помощью отношения:

$$\frac{\text{расстояние от P(MU) до P(1)}}{\text{расстояние от P(2) до P(1)}}$$

где расстояние считается положительным, если P(MU) расположена по ту же сторону от P(1), что и P(2), и отрицательным - в противоположном случае. Абсолютное значение расстояния между точками задается формулой (Пифагора):

ABS(P(1)-P(2))=SQRT((X(1) - X(2))^2+(Y(1)-Y(2))^2)

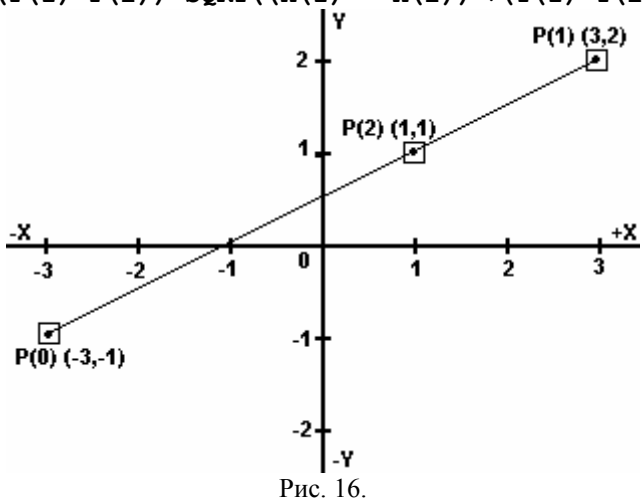


Рис. 16.

См. рис. 16, на котором показан отрезок, соединяющий точки (-3,-1)=P(0) и (3,2)=P(1); точка (1,1) лежит на прямой и соответствует точке P(2). Заметьте, что (3,2) находится от (-3,-1) на расстоянии 3*SQRT(5), тогда как (1,1) на расстоянии 2*SQRT(5). С этого места мы опускаем обозначение (MU).

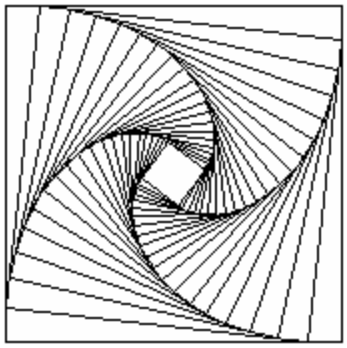


Рис. 16A.

Пример 3.A

Воспользовавшись таким способом параметрического представления прямой, начертим орнамент, показанный на рис. 16A. На первый взгляд орнамент кажется сложным, но, если присмотреться повнимательнее, видно, что он состоит из последовательности вложенных друг в друга квадратов. Стороны квадратов монотонно убывают, и каждый из квадратов повернут относительно своего предшественника на фиксированный угол. Для того, чтобы нарисовать этот орнамент, нам нужно научиться вписывать в квадрат другой квадрат, повернутый на фиксированный угол. Предположим, что четыре вершины квадрата - это отрезок, соединяющий точку (X(I), Y(D) (1-1,2,3,4); 1-ая сторона квадрата - это отрезок, соединяющий точку (X (I), Y(I) с точкой (X(I+1), Y(I+1), где сложение производится по модулю 4. Координаты (X''(I), Y''(I) произвольной точки, лежащей на этой стороне, есть

```
((1-MU)*X(I)+MU*X(I+1),(1-MU)*Y(I)+MU*Y(I+1)),
```

где 0<=MU<=1

Эта точка делит сторону в отношении MU:(1-MU). Если MU фиксировано и четыре точки (X''(I), Y''(I) (I=1,……4) вычисляются по приведенной формуле, то стороны нового квадрата образуют угол ALFA=ARCTG(MU/(1-MU)) со сторонами старого квадрата. Таким образом, если MU остается одним и тем же для всех квадратов, то мы получим необходимую последовательность. Программа, приведенная на листинге 3.1, рисует орнамент, состоящий из 21 -го квадрата с MU=0.1.

LISTING 3.1

```
100 REM SQUARE OUTSIDE SQUARE ETC.
110 LET START=9700: LET SETORIGIN=9600: LET MOVETO=9500: LET LINETO=9400
120 LET HORIZ=3: LET VERT=2.1
130 GO SUB START
140 LET XMOVE=HORIZ*0.5: LET YMOVE=VERT*0.5
150 GO SUB SETORIGIN
160 DIM X(4):DIM Y(4): DIM V(4):DIM W(4)
170 DATA 1,1,1,-1,-1,-1,-1,1
179 REM INITIALISE FIRST SQUARE.
180 FOR I=1 TO 4: READ X(I),Y(I): NEXT I
189 REM SET MU VALUE AND DRAW 20 SQUARES.
190 LET MU=0.1: LET UM=1-MU
200 FOR I=1 TO 21
208 REM JOIN FOIR VERTICES OF SQUARE (X(J),Y(J)).
209 REM CALCULATE NEXT FOIR VERTICES(V(J),W(J)).
210 LET XPT=X(4): LET YPT=Y(4): GO SUB MOVETO
220 FOR J=1 TO 4
230 LET XPT=X(J): LET YPT=Y(J): GO SUB LINETO
240 LET NJ=J+1: IF NJ=5THEN LET NJ=1
250 LET V(J)=UM*X(J)+MU*X(NJ)
260 LET W(J)=UM*Y(J)+MU*Y(NJ)
270 NEXT J
279 REM COPY ARRAYS V AND W INTO X AND Y.
280 FOR J=1 TO 4
290 LET X(J)=V(J): LET Y(J)=W(J)
300 NEXT J
310 NEXT I
320 STOP
```

Иногда оказывается полезной следующая параметрическая форма прямой:

```
P(1)+MU*(P(2)-P(1))
```

Здесь P(1) называется базовым вектором, а (P(2)-P(1)) - направляющим вектором. Любая точка прямой может выступать в роли базового вектора; она просто выделяет из семейства прямых, параллельных направляющему вектору, прямую, проходящую через эту точку. Следует дополнительно обсудить возможность использования вектора для указания направления. Мы знаем, что вектор (X,Y) может использоваться для обозначения точек на плоскости. Тогда прямая, проходящая через начало координат и эту точку, определяет направление - по определению, любая прямая, параллельная этой прямой, имеет тот же направляющий вектор. Отметим, что положительным направлением на прямой считается направление от начала координат к точке (X,Y). Двойственное представление вектора как точки

и как направления находит применение в следующем примере.

Пример 3.Б

Нарисуйте линию, состоящую из 13 штрихов и проходящую через точки P(1)=(X(1),Y(1)) и P(2)=(X(2),Y(2)). Эта задача решается при помощи 26 точек, равноудаленных друг от друга; точка этого семейства задается выражением P(1)+(I/25)*(P(2)-P(1)), где I принимает значения от 0 (точка P(1)) до 25 (точка P(2)). Используя подпрограмму "PLOT" (листинг 2.8), мы попеременно то движемся по прямой, то вычерчиваем участок этой прямой. Хранить значения координат необязательно; P(1) нам задано, и, прибавляя каждый раз по 1/25*(P(2)-P(1)), мы побываем во всех необходимых точках (см. листинг 3.2).

LISTING 3.2

```
100 REM DASHED LINES
110 LET START=9700: LET PLOT=9800
120 LET HORIZ=30: LET VERT=20
130 GO SUB START
140 LET XPT=HORIZ*0.5: LET YPT=VERT*0.5: LET MOVE=3
150 GO SUB PLOT
160 INPUT " X1=";X1;" Y1=";Y1
170 INPUT " X2=";X2;" Y2=";Y2
179 REM MOVE TO FIRST POINT.
180 LET XPT=X1:LET YPT=Y1: LET MOVE=3: GO SUB PLOT
190 LET XD=(X2-X1)/25: LET YD=(Y2-Y1)/25
199 REM Alternately draw and move to next 25 points on the line.
200 FOR I=1 TO 25
210 LET XPT=XPT+XD: LET YPT=YPT+YD: LET MOVE=5-MOVE: GO SUB PLOT
220 NEXT I
230 STOP
```

Упражнение 3.1

Поэкспериментируйте с различными видами пунктирных линий:

А. Длина штриха в два ряда превосходит длину промежутка между кривыми.

Б. Длина штриха фиксирована, а количество штрихов неизвестно.

В. Тип штриха меняется от длинного штриха до короткого, в зависимости от значения входного параметра; еще один входной параметр можно ввести для того, чтобы задавать величину промежутка.

Данная параметрическая форма прямой удобна для вычисления точки пересечения двух прямых - задачи, часто встречающейся в двумерной графике. Предположим, что заданы две прямые

```
P+MU*Q и R+LMD*S
```

где P=(X(l),Y(l)), Q=(X(2),Y(2)), R=(X(3),Y(3)), S=(X(4),Y(4)),

Минус Бесконечность < MU, S < Плюс Бесконечность

Мы должны найти такие значения MU и LMD, что

```
P + MU*Q-R + LMD*S
```

то есть точку, лежащую на обеих прямых. Это векторное уравнение распадается на два скалярных:

```
X(1)+MU*X(2)=X(3)+LMD*X(4)     (3.1)
Y(1)+MU*Y(2)=Y(3)+LMD*Y(4)     (3.2)
```

Перепишем эти уравнения так:

```
MU*X(2)-LMD*X(4)=X(3)-X(1)     (3.3)
MU*Y(2)-LMD*Y(4)=Y(3)-Y(1)     (3.4)
```

Умножим (3.3) на Y(4), а (3.4) на X(4) и, вычитая, получим:

```
MU*(X(2)*Y(4)-Y(2)*X(4))=(X(3)-X(1))*Y(4)-(Y(3)-Y(1))*X(4)
```

Если (X(2)*Y(4)-Y(2)*X(4))=0, тогда эти прямые параллельны, и точки пересечения не существует (MU не существует), в противном случае

$$MU = \frac{(X(3)-X(1))*Y(4)-(Y(3)-Y(1))*X(4)}{X(2)*Y(4)-Y(2)*X(4)}$$

и аналогично

$$LMD = \frac{(X(3)-X(1))*Y(3)-(Y(3)-Y(1))*X(3)}{X(2)*Y(4)-Y(2)*X(4)}$$

Это решение упрощается, если одна из прямых параллельна координатной оси. Предположим, что X=D, тогда R=(D,0) и S=(0,l),

```
MU=(D-X(1))/X(2)
```

Аналогично для прямой Y=D, получим:

```
MU=(D-Y(1))/Y(2)
```

Если прямые параллельны, то детерминант уравнения обращается в бесконечность, так как параллельные прямые не пересекаются.

Пример 3.В

Найдите точку пересечения двух прямых:

А. Соединяет (1,-1) и (-1.-3);

В. Соединяет (1,2) и (2,2).

Эти прямые задаются выражениями

```
(1-MU)*(1,-1) + MU*(-1,3) -Бесконечность<MU<+Бесконечность (3.7)
(1-(LMD))*(1,2) + LMD*(2,-2) -Бесконечность<LMD<+Бесконечность (3.8)
```

или, переходя к форме базовая точка - направление:

```
(1,-1)+MU*(-2,-2) (3.9)
(1,2)+LMD*(2,-4) (3.10)
```

Подставляя эти значения в уравнение (3.5), получим:

$$MU = \frac{(1-1)(4)-(2+1)2}{(-2(-4)-(-2)2)} = -\frac{1}{2}$$

откуда значения координат точки пересечения (1,-1) - 1/2* (-2,2) = (2,0).

Упражнение 3.2

Поэкспериментируйте с такой формой представления двумерной геометрии. Вы можете составить свои собственные задачи. Легко проверить правильность решения этих задач. Рассмотрим пример 3.Б. Мы знаем, что точка (2,0) лежит на первой прямой, так как мы использовали MU=-1/2; наше решение верно, если она также лежит и на второй прямой; нетрудно убедиться, что это так - значение LMD=1/2;

Упражнение 3.3

Напишите программу, которая считывает информацию о двух прямых; они могут быть заданы в форме уравнений или в форме базовая точка-направление и вычисляет точку их пересечения (если таковая имеется).

Кадрирование

Нам известно, что для точек (X,Y), расположенных вне экрана, команды PLOT и DRAW недействительны, так что мы ограничены значениями координат 0<=X<=255 и 0<=Y<=175. Двумерные и трехмерные построения удобно проводить на больших областях, чем область экрана, но тогда нам необходимо найти алгоритм, удаляющий всю лишнюю информацию без потери полезной информации.

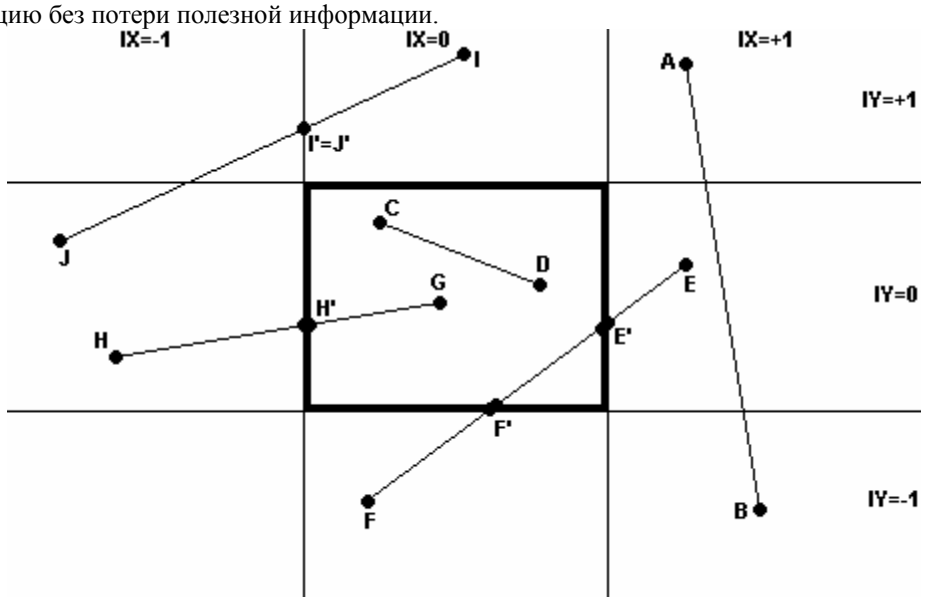


Рис. 17.

Предположим, что центр экрана задается точкой (не пикселем!) с действительными координатами (255/2, 175/2) = (127.5, 87.5) и, следовательно, угловые точки экрана имеют координаты (127.5±127.5, 87.5±87.5). Наша задача сводится к нахождению отрезка прямой, соединяющего пиксели (XA,YA) и (XB,YB) и лежащего в поле экрана. Для простоты предположим, что начало координат совпадает с центром экрана, что достигается вычитанием вектора (127.5,87.5) из векторов, соответствующих точкам в старой системе координат. Теперь координаты угловых точек экрана равны (±127.5,±87.5). Мы продолжаем стороны прямоугольника, разбивая таким образом плоскость на девять участков; см. рис. 17, на котором видны графическое поле и его граница. Для иллюстрации работы алгоритма на рисунке изображены несколько сегментов. Каждая точка пространства может быть теперь охарактеризована двумя параметрами, IX и IY, где

(1) IX=-1,0,+1 в зависимости от того, будет ли значение X-координаты точки лежать справа, в, или левее графической области;

(2) IY=-1,0,+1 в зависимости от того, будет ли значение Y-координаты точки лежать ниже графической области, в графической области или выше её.

Эти значения при необходимости вычисляются внутри программы.

Пусть концам сегмента с координатами (XA,YA), (XB,YB) соответствуют значения параметров IXA и IYA и IXB и IYB соответственно, тогда имеются следующие возможности:

(I) если IXA=IXB не=0 и IYA=IYB не=0, тогда весь сегмент лежит вне экрана и его можно не рассматривать; см. сегмент AB на рис. 17.

(II) если IXA=IYA=IXB=IYB=0, тогда весь сегмент лежит в графическом поле и его следует изобразить целиком; см. сегмент CD.

(III) оставшиеся случаи следует рассмотреть подробнее. Если IXA не=0 и (или) IYA не=0, тогда точка (XA,YA) лежит вне прямоугольника и поэтому следует вычислить новые значения XA и YA - чтобы избежать

недоразумений, мы обозначим их XA',YA'. (XA',YA') - ближняя к (XA,YA) точка пересечения прямой с границей графической области. Формула для вычисления координат точки пересечения рассматривалась выше: это формула для нахождения точки пересечения прямой с другой прямой, параллельной координатной оси. Если прямая не пересекает прямоугольник, то принимаем за точку (XA',YA') точку пересечения прямой с одной из продолженных вертикальных прямых. Если IXA=IYA=0, то (XA',YA')=(XA,YA). Точка (XB',YB') вычисляется аналогично; см. алгоритм подпрограммы CLIP на листинге 3.3. Требуемый отрезок - это отрезок, соединяющий точки (XA',YA') с точкой (XB',YB'). Если прямая не пересекает прямоугольник, то (XA',YA')=(XB',YB'), и новый отрезок вырождается в точку.

Например, EF обрезается до E'F'. GH обрезается до GH' (G=G'), и IJ сводится к точке I'=J'.

Программа "CLIP" переводит координаты концов отрезка (XA,YA) и (XB,YB) в координаты центрированной системы координат. Затем проверяется, какая из трех ситуаций имеет место, и в случае (I) программа тут же прекращает работу, в случае (II) соединяет две точки, а в случае (III) - вычисляет "штрихованные" координаты.

Листинг 3.3 содержит новую версию подпрограммы "LINETO", вызывающей "CLIP" вместо PLOT и DRAW, так что теперь мы можем соединять любые точки. Начиная с этого момента, используйте эту версию "LINETO". Она окажется неоценимым подспорьем, особенно при исследовании трехмерных объектов.

Упражнение 3.4

Используйте эту модифицированную программу в программах главы 2. Выбирайте значения HORIZ и VERT таким образом, чтобы некоторые участки кривых выходили за пределы графического поля.

LISTING 3.3

```
8400 REM CLIP
8401 REM IN - XA,YA,XB,YB
8409 REM CHANGE COORDINATE SYSTEM.
8410 LET XA=XA-127.5: LET YA=YA-87.5: LET XB=XB-127.5: LET YB=YB-87.5
8419 REM Fire the sector values of two points (XA,YA) and (XB,YB).
8420 LET IXA=0: IF ABS XA>127.5 THEN LET IXA=SGN XA
8430 LET IYA=0: IF ABS YA>87.5 THEN LET IYA=SGN YA
8440 LET IXB=0: IF ABS XB>127.5 THEN LET IXB=SGN XB
8450 LET IYB=0: IF ABS YB>87.5 THEN LET IYB=SGN YB
8459 REM POINTS IN SAME OFF-SCREEN SECTOR THEN RETURN.
8460 IF IXA*IXB=1 OR IYA*IYB=1 THEN RETURN
8470 IF IXA=0 THEN GO TO 8500
8479 REM MOVE 1"ST POINT TO NEARER X-EDGE.
8480 LET XX=127.5*IXA: LET YA=YA+(YB-YA)*(XX-XA)/(XB-XA): LET XA=XX
8490 LET IYA=0: IF ABS YA> 87.5 THEN LET IYA=SGN YA
8500 IF IYA=0 THEN GO TO 8520
8509 REM MOVE 1"ST POINT TO NEARER Y-EDGE.
8510 LET YY=87.5*IYA: LET XA=XA+(XB-XA)*(YY-YA)/(YB-YA): LET YA=YY
8520 IF IXB=0 THEN GO TO 8550
8529 REM MOVE 2"ND POINT TO NEARER X-EDGE.
8530 LET XX=127.5*IXB: LET YB=YB+(YB-YA)*(XX-XB)/(XB-XA): LET XB=XX
8540 LET IYB=0: IF ABS YB>87.5 THEN LET IYB=SGN YB
8550 IF IYB=0 THEN GO TO 8570
8559 REM MOVE 2"ND POINT TO NEARER Y-EDGE.
8560 LET YY=87.5*IYB: LET XB=XB+(XB-XA)*(YY-YB)/(YB-YA): LET YB=YY
8570 IF ABS (XA-XB)<0.000001 AND ABS (YA-YB)<0.000001 THEN RETURN
8579 REM PLOT NOT-COINCIDENT POINTS.
8580 LET XA=INT (XA+128): LET YA=INT (YA+88): LET XB=INT (XB+128): LET YB=INT
(YB+88)
8590 PLOT XA,YA: DRAW XB-XA,YB-YA: RETURN
9400 REM LINETO/CLIPPING
9401 REM IN - XPT,YPT,XPEN,YPEN
9402 REM OUT - XPEN,YPEN
9410 LET CLIP=8400
9420 LET XA=XPEN: LET YA=YPEN
9430 LET XPEN=FN X(XPT): LET YPEN=FN Y(YPT)
9440 LET XB=XPEN: LET YB=YPEN
9450 GO SUB CLIP
9460 RETURN
```

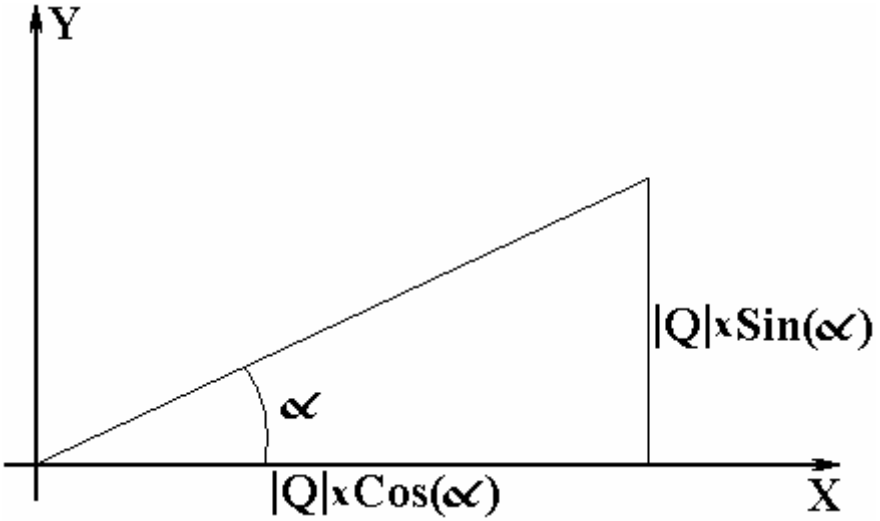


Рис. 18.

Возвращаясь к использованию направляющего вектора Q=(X,Y) не= (0,0) для определения направления, заметим, что вектор KQ, где K>0, определяет то же направление и ориентацию, что и Q (если K отрицательно, то ориентация направления противоположная). Если мы возьмем K=1/ABS(Q), то получим вектор (X/SQRT(X**2+Y**2), Y/SQRT(X**2+Y**2)) единичной длины.

Расстояние точки прямой P+MU*Q до базового вектора P есть ABS(MU*Q), и если ABS(Q)=1, то это расстояние равно ABS(MU).

Предположим, что ALFA - угол между прямой, соединяющей 0 (начало координат) с точкой Q=(X,Y) и положительной полуосью X. Тогда X=ABS(Q)*COS(ALFA), Y-ABS(Q)*SIN(ALFA), см. рис. 18; для других квадрантов картина аналогичная.

Если Q - вектор единичной длины, то Q=(COS(ALFA).SIN(ALFA)). Заметим, что SIN(ALFA)=COS(PI/2-ALFA) для всех значений ALFA. Поэтому мы можем записать Q=(COS(ALFA),COS(PI/2-ALFA)), но PI/2 - ALFA это угол между векторами и положительной полуосью Y, что дает основание называть координаты единичного вектора направляющими косинусами.

Перед тем, как продолжить, мы рассмотрим имеющиеся в бейсике тригонометрические функции: SIN и COS и обратную тригонометрическую функцию ATAN. SIN и COS - однопараметрические (угол, выраженный в радианах) и однозначные функции с областью значений между -1 до +1.

Функция ATAN имеет областью определения всю числовую ось, и вычисляет угол в радианах (так называемую главную ветвь с диапазоном значений от -PI/2 до PI/2), тангенс которого совпадает со значением аргумента.

Программа "ANGLE" вычисляет угол, который образует направление Q=(X,Y) с положительной полуосью X. Эта подпрограмма приведена на листинге 3.4 и окажет нам большую помощь в последующих главах, посвященных трехмерной геометрии.

LISTING 3.4

```
8800 REM ANGLE
8801 REM IN - AX.AY
8802 REM OUT- THETA
8809 REM THETA IS THE ANGLE MADE BY LINE TO (AX,AY) WITH+VE X-AXIS.
8810 IF ABS AX>0.00001 THEN GO TO 8860
8819 REM LINE IS VERTICAL.
8820 LET THETA=PI /2
8830 IF AY<0 THEN LET THETA=THETA+PI
8840 IF ABS AY<0.00001 THEN LET THETA=0
8850 RETURN
8859 REM LINE NOT VERTICAL SO IT HAS FINITE TANGENT.
8860 LET THETA=ATN (AY/AX)
8870 IF AX<0 THEN LET THETA=THETA+PI
8880 RETURN
```

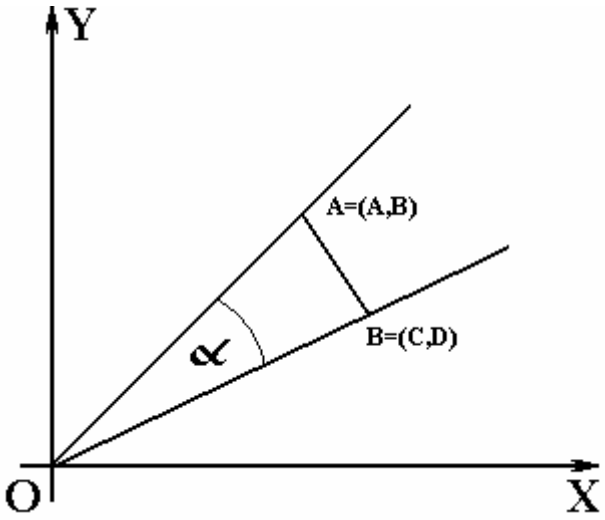


Рис. 19

Предположим, что нам заданы два вектора (A,B) и (C,D); ограничимся для простоты случаем векторов единичной длины (см. рис. 19). Мы хотим вычислить острый угол ALFA между этими векторами. Из рисунка видно, что

```
OA=SQRT(A**2+B**2)=1,
OB=SQRT(C**2+D**2)=1.
```

По теореме косинусов имеем:

```
AB**2=OA**2+OB**2-2*OA*OB*COS(ALFA)=2*(1-COS(ALFA))
```

Но по теореме Пифагора:

```
AB**2=(A-C)**2+(B-D)**2=(A**2+B**2)+(C**2+D**2)-2(A*C+B*D)=2-2(A*C+B*D).
```

Таким образом

```
COS(ALFA)=A*C+B*D.
```

Если A*C+B*D отрицательно, то ARCCOS(A*C+B*D) - тупой угол, и требуемый острый угол равен PI-ALFA. Так как COS(PI-ALFA)=-COS(ALFA), то мы приходим к выводу, что острый угол вычисляется по формуле

```
ARCCOS(ABS(A*C+B*D))
```

Рассмотрим, например, две прямые с направляющими косинусами

```
(SQRT(3)/2,1/2) и (-1/2,-SQRT(3/2);
```

мы видим, что A*C+B*D=-3/2 и, следовательно,

```
ALFA=ARCCOS(SQRT(3)/2)=PI/6.
```

Этот простой пример приводится для того, чтобы ввести понятие скалярного произведения двух векторов (A,B)*(C,D)=A*C+B*D. Понятие скалярного произведения распространяется на случай пространств больших измерений, так что и в этих случаях оно помогает ввести понятие угла между прямыми.

Задание кривых с помощью функциональной зависимости

Кривая в двумерном пространстве задает зависимость между значениями X координаты и Y координаты, которая называется функциональной зависимостью. С другой стороны, значения координат точек вдоль кривой можно задавать с помощью параметра.

Мы знаем, что прямую (которая является окружностью бесконечно большого радиуса) можно задавать с помощью AY=BX+C. Если перенести правую часть в левую, то мы придем к уравнению (A*Y-B*X-C=0), правая часть которого:

```
F(X,Y)=A*X-B*Y-C
```

называется функциональной формулой прямой. Точка принадлежит прямой тогда и только тогда, когда F(X,Y)=0. Таким образом, вся плоскость оказывается разделенной на три области:

- область, в которой F(X,Y)>0 (область положительных значений),
- область, в которой F(X,Y)<0,
- собственно прямая F(X,Y)=0.

Если кривая делит плоскость на две связанные компоненты (область называется связанной, если любые две точки, принадлежащие области, можно соединить кривой, не пересекающей кривую F(X,Y)=0), тогда каждую из этих компонент можно отождествить с областью положительных значений.

Следует помнить, что некоторые, даже вполне элементарные, кривые делят область не на две, а на бесконечное число компонент (например, F(X,Y)=COS(Y)-SIN(X)), что не исключает возможности обеим несвязанным компонентам принадлежать к типу областей положительных значений.

Обратите внимание, что функциональное представление не единственно. Ту же прямую задает функция

```
F(X,Y)=B*X+C-A*Y.
```

для которой область положительных значений совпадает с областью отрицательных значений предыдущей функции.

Случай, когда кривая разбивает плоскость на две компоненты, оказывается весьма полезным для машинной графики, в особенности для случая трех измерений. Например, рассмотрим прямую

```
F(X,Y)=A*Y-B*X-C,
```

где точка (X1,Y1) лежит по ту же сторону от прямой, если и только если знаки F(X1,Y1) и F(X2,Y2) совпадают.

Функциональное представление прямой позволяет не только указать, с какой стороны от прямой лежит точка, но и вычислить расстояние от этой точки до прямой.

Направляющий вектор прямой равен (A,B). Направляющий вектор (-B,A). Почему? Произведение тангенсов взаимно перпендикулярных прямых равно -1 . Мы видим, что ближайшая к точке P точка Q задается:

```
Q=(X(1),Y(1))+MU*(-B,A),
```

что означает, что прямая, проходящая через P и Q, перпендикулярна исходной прямой. Так как Q лежит на исходной прямой, то

```
F(Q)-F(X(1),Y(1)H/IU*(-B,A)-0
```

и следовательно

```
A*(Y(1)+MU*A)-B*(X(1)-MU*B)-C=F(X(1),Y(1))+MU*(A**2+B**2)=0
```

откуда

```
MU=F(X(1),Y(1))/(A**2+B**2).
```

Точка Q лежит на расстоянии MU*ABS(-B,A) от точки (X(1),Y(1)); это означает, что расстояние от (X(1),Y(1)) до прямой есть

```
MU*SQRT(A**2+B**2)=-F(X(1),Y(1))/SQRT(A**2+B**2)
```

Знак указывает, по какую сторону от прямой лежит точка. Если A**2+B**2=1, то ABS(F(X1,Y1)) задает расстояние от точки (X1,Y1) до прямой.

Мы подошли к понятию выпуклой области. По определению, область называется выпуклой, если отрезок, соединяющий любые две точки этой области, целиком принадлежит этой области. В нашем исследовании мы ограничимся случаем выпуклого многоугольника, так как очевидно, что любая выпуклая область может быть аппроксимирована таким многоугольником с достаточно большим числом сторон.

Рассмотрим выпуклый N-угольник; предположим что нам задано множество его вершин в том порядке, в каком они встречаются при обходе многоугольника по (против) часовой стрелки. В дальнейшем мы будем называть такой набор вершин упорядоченным. Задачу выяснения того, является ли набор упорядоченным по часовой стрелке или против, мы отложим до главы 7. N сторон многоугольника являются отрезками прямых

```
F(I)(X,Y)=(X(I+1)-X(I))*(Y-Y(I))-(Y(I+1)-Y(I))*(X-X(I)),
```

где I=l,…,N, и сложение производится по модулю N (что означает N+J=J для I<=J<=N). Попробуйте объяснить, почему эти формулы описывают отрезки. Последовательное рассмотрение этих отрезков позволит нам определить внутренность этой выпуклой области. Каждый из этих сегментов, например, соединяющий точки P(I) и P(I+1) для некоторого I, обладает тем свойством, что точки внутри области должны лежать с той же стороны от прямой, на которой лежит отрезок и остальные вершины многоугольника, например, вершина P(I+2). Таким образом, не трудно вывести формулы для точки, лежащей вне области, на границе области и внутри области.

А теперь перейдем к примеру.

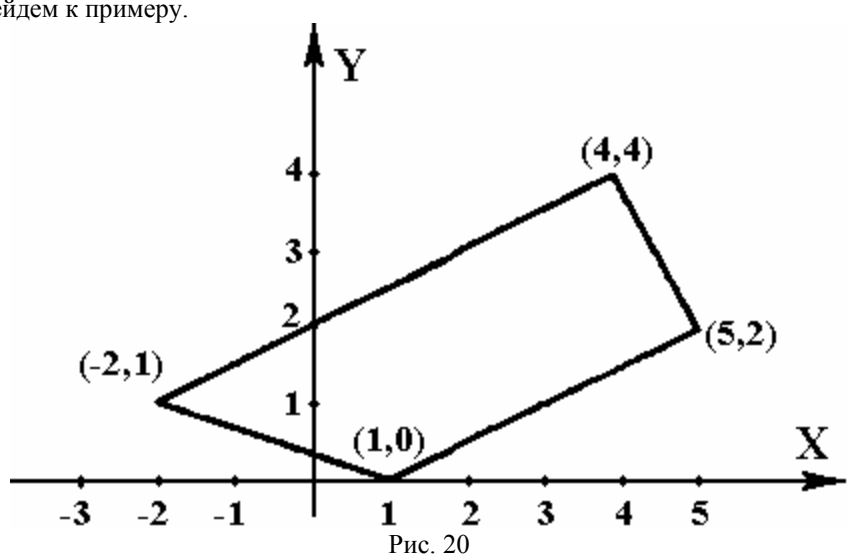


Рис. 20

Пример 3.Г

Предположим, что задан выпуклый многоугольник с вершинами (1,0), (5,2), (4,4) и (-2,1) (см. рис. 20). Очевидно, что точки обходились против часовой стрелки. Находятся ли точки (3,2), (1,4), (3,1) внутри многоугольника, вне многоугольника или лежат на границе? Каково расстояние от точки (4,4) до первой прямой?

```
F(1)(X,Y)     =     (5-1)*(Y-0)-(2-0)*(X-1)       =       4*Y-2*X+2
F(2)(X,Y)     =     (4-5)*(Y-2)-(4-2)*(X-5)       =       -Y-2*X+12
F(3)(X,Y)     =     (-2-4)*(Y-4)-(1-4)*(X-4)      =       -6*Y+3*X+12
F(4)(X,Y)     =     (1+2)*(Y-1)-(0-1)*(X+2)       =       3*Y+X-1
```

Мы видим, что точка (3,2) лежит внутри многоугольника, так как F(l)(3,2)=4 и F(l)(4,4)=10; F(2)(3,2)=4 и F(2)(2,1)=15; F(3)(3,2)=9 и F(3)(1,0)=15; F(4)(3,2)=8 и F(4)(5,2)=10 - все точки имеют одинаковый положительный знак.

Точка (1,4) лежит вне области, так как F(3)(1,4)=-9 и F(3)(1,0)=15 - знаки противоположны.

Точка (3,1) лежит на границе, так как F(1)(3,1)=0, F(2)(3,l)=5, F(3)(3,l)=15, F(4)(3,4)=5.

На самом деле нет необходимости вычислять F(I)(X(I+2,Y(I+2)) для каждого значения I, они все имеют тот же знак, так что вычислив F(1)(X(3),Y(3)), мы можем использовать это значение в наших вычислениях.

(4,4) находится на расстоянии

```
F(1)(4,4)/SQRT(4**2+2**2)=10/SQRT(20)=SQRT(5)
```

Упражнение 3.5

Представьте, что два выпуклых многоугольника пересекаются. Их общий участок является выпуклым многоугольником. Используя методы этой главы, найдите вершины нового многоугольника.

Теперь рассмотрим параметрическое представление кривой. Еще раз отметим, что в случае параметрического задания кривой координаты точки, принадлежащие прямой, определяются с помощью параметра с учетом области допустимых значений этого параметра. Мы уже знаем параметрический вид прямой: это просто задание прямой с помощью базовой точки и направляющего вектора:

```
B+MU*Q=(X(1),Y(1)+MU*(X(2),Y(2))=(X(1)+MU*X(2),Y(1)+MU*Y(2)),
```

где -Бесконечность<MU<+Бесконечность.

MU - параметр, X(1)+MU*X(2) и Y(1)+MU*Y(2) - соответственно X- и Y-координаты, которые зависят только от значений MU.

Мы приведем также функциональный и параметрический виды простейших кривых. Например, кривая синуса задается в функциональной форме

```
F(X,Y)=Y-SIN(X)
```

и в параметрической

```
(X,SIN(X))
```

где -бесконечность<X<+бесконечность.

Общий вид конического сечения (эллипс, парабола, гипербола) задается функцией

```
F(X,Y)=A*X**2+B*Y**2+H*X*Y+F*X+G*Y+C.
```

где коэффициенты A, B, H, F, G, C однозначно определяют кривую. Окружность радиуса R с центром в начале координат имеет значение параметров A=B=1, F=G=H=0 и C=R**2, F(X,Y)>0. В параметрической форме координаты окружности имеют вид:

```
(R*COS(ALFA),R*SIN(ALFA)),
```

где 0<=ALFA<=2*PI (мы уже встречались с параметрической формой эллипса, окружности и спирали в главе 2).

Очень полезно повозиться с этими (и другими) представлениями двумерной геометрии. Во многих случаях окажется полезным оформлять ваши идеи в виде программ, а что касается построения диаграмм и графиков, то там требуется делать это постоянно.

Пример 3.Д

Предположим, что мы хотим нарисовать шар (радиуса R), исчезающий в эллипсной дыре (большая ось-A, малая ось-B). Некоторые участки окружности и эллипса скрыты от наблюдателя.

Пусть центр эллипса совпадает с началом координат, оси параллельны координатным осям, большая ось горизонтальна, а центр шара расположен над центром эллипса на расстоянии D. Функциональное представление эллипса:

```
F(X,Y)=X**2/A**2+Y**2/B**2-1
```

и параметрический вид:

```
(A*COS(ALFA),B*SIN(ALFA)),
```

где 0<=ALPHA<=2*PI.

Для окружности:

```
F(X,Y)=X**2+(Y-D)**2-R**2
```

и параметрическая форма:

```
(R*COS(LMDA)*D+R*SIN(LMDA)),
```

где 0<=LMDA<=2*PI.

Чтобы нарисовать такую картинку, нам надо найти общие точки (если они есть) эллипса и окружности. Для большей поучительности мы используем параметрическое представление эллипса и функциональное для окружности.

Таким образом, мы ищем точки эллипса

```
(X,Y)=(A*COS(ALFA),B*SIN(ALFA)),
```

удовлетворяющие F(C)(X,Y)=0, что означает:

```
A**2(COS(ALFA))**2+(B*SIN(ALFA)-D)**2-R**2=0
A**2*COS(ALPHA)**2+B**2*SIN(ALPHA)**2-2*B*D*SIN(ALPHA)+D**2-R**2=0;
```

и так как (COS(ALFA))**2=1-(SIN(ALFA))**2, то:

```
(B**2-A**2)*(SIN(ALFA))**2-2*B*D*SIN(ALFA)+A**2+D**2-R**2=0
```

Это простое квадратное уравнение легко разрешить относительно SIN(ALFA) (квадратное уравнение A*X**2+B*X+C=0 имеет корни (-B+-SQRT(B-4*A*C)/2*A)). Для каждого значения SIN(ALFA) мы можем вычислить точки значения ALFA, 0<=ALFA<=2*PI (если таковые существуют) и затем вычислить точки пересечения (A*COS(ALFA),B*SIN(ALFA)).

Не существует жестких правил о том, каким представлением лучше пользоваться в том или ином случае; это надо чувствовать, и это достигается только практикой.

Список программ

1. LIB1 и листинг 3.1.: входные данные отсутствуют.
2. Листинги 2.1, 2.8 и 3.2: входные данные - две координатные пары (X(1),Y(1)) и (X(2),Y(2))

Замечание: с этого момента листинг 3.3 ("CLIP) и новая версия "LINETO" заменят листинг 2.5 в "LIB1A".

ГЛАВА 4
Матричное представление преобразований двумерного пространства

В главе 2 у нас возникла необходимость перемещать объекты по экрану. Представляется более удобным сначала определять объект в простейших терминах (таких, как вершины в терминах пикселей или значений координат, вместе с информацией о линиях и областях, связанных с этими вершинами), а затем помещать эти объекты в нужное место на экране, оставляя при этом систему координат экрана неподвижной. Мы ограничимся случаем линейных перемещений (преобразований).

Иногда бывает необходимо переместить большое количество вершин, и в этом случае чрезвычайно полезным оказывается понятие матрицы. Перед тем, как ввести матричное представление преобразований, определим сначала, что мы подразумеваем под словом матрица, и введем понятие вектора-столбца.

На самом деле мы ограничимся случаем матрицы 3x3 (произносится: три на три) для изучения двумерной геометрии, а позже, при изучении геометрии, мы будем использовать матрицы 4X4. Такая 3X3 матрица - это не что иное, как таблица чисел, состоящая из трех строк и трех столбцов, а вектор-столбец - это столбец чисел, состоящий из трех чисел.

```
A(1,1)      A(1,2)      A(1,3)            D1
A(2,1)      A(2,2)      A(2,3)      и     D2
A(3,1)      A(3,2)      A(3,3)            D3
```

Элемент матрицы имеет вид A(I,J), где I - это номер строки, в которой расположен этот элемент, J - номер столбца (например, A(2,3) расположен во второй строке и третьем столбце). Все элементы являются числами, и важно помнить, что информация, записанная в столбце матрицы, состоит из значений чисел, составляющих этот столбец и расположения столбца в матрице. Символы в программе, написанной, на бейсике, расположены в строчку, и полому матрицы и вектора записываются также в строчку с индексами, заключенными и скобки и следующими за идентификатором матрицы.

Матрицы можно складывать. Матрица A+B, сумма матриц A и B, элемент которой C(I,J) по определению равен:

```
C(I,J)=A(I,J)+B(I,J)  1<=I, J<=3
```

Умноженная на скаляр K матрица B есть

```
B(I,J)=K*A(I,J)
```

Мы можем определить умножение матрицы A на вектор-столбец, результатом которого является вектор-столбец E, следующим образом:

```
E(I)=A(I,1)*D1+A(I,2)*D2+A(I,3)*D3=SUM(K)A(I,K)*D(K)
```

Элемент столбца, стоящий в I-ой строке, равен произведению элементов I-ой строки матрицы, умноженных на соответствующие элементы столбца. Далее, мы можем вычислить произведение двух матриц A и B.

```
C(I,J)=A(I,1)*B(1,J)+A(I,2)*B(2,J)+A(I,3)*B(3,J)=SUM(K) A(I,K)*B(K,J),
```

где I<=l, J<=3

Для получения элемента, стоящего в I-ой строке и J-ом столбце, мы вычисляем сумму произведений элементов I-ой строки матрицы A на соответствующие элементы матрицы B. Следует отметить, что произведение матриц необязательно коммутативно, это означает, что произведение A*B не обязательно равно произведению B*A. Например:

```
010          001          010
001    X     010    =     100
100          100          001
```

но

```
001          010          100
010    X     001    =     001
100          100          010
```

Поупражняйтесь с операциями над матрицами с тем, чтобы владеть этими понятиями в совершенстве.

Среди матриц выделяется так называемая единичная матрица I:

```
                  100
I      =          010
                  001
```

Для каждой матрицы можно вычислить ее детерминант:

```
DET(A)=A(1,1)*(A(2,2)*A(3,3)-A(2,3)*A(3,2))+A(1,2)*(A(2,3)*A(3,1)-
    -A(2,1)*A(3,3))+A(1,3)*(A(2,1)*A(3,2)-A(2,2)*A(3,1));
```

Матрица, у которой детерминант не равен нулю, называется невырожденной, а матрица, у которой детерминант равен нулю, называется вырожденной. Невырожденные матрицы A обладают обратными матрицами A**(-1) такими, что A*(A**(-1))=1. Мы приводим программу (листинг 7.5, глава 7), которая вычисляет обратную матрицу методом сопряженных миноров.

Теперь перейдем к рассмотрению преобразования точек в пространстве. Пусть (X,Y) - координаты точки до преобразования, а (X',Y') - ее координаты после преобразования. Мы определяем преобразование полностью, если задаем формулы для вычисления координаты точек после преобразования по координатам до такового. В случае

линейного преобразования, по определению, эта зависимость линейна. Это означает, что правые части уравнений содержат X и Y, умноженные на некоторые множители, и дополнительные константы. Такие уравнения можно написать в виде

```
X'=A(1,1)*X+A(1,2)*Y+A(1,3)
Y'=A(2,1)*X+A(2,2)*Y+A(2,3)
```

Значения A называются коэффициентами уравнений. Мы видим, что линейное преобразование получается в результате умножения на коэффициенты значений X, Y и единицы. Мы можем добавить к нашим уравнениям еще одно

```
I=A(3,1)*X+A(3,2)*Y+A(3,3)
```

Так как оно должно выполняться для всех X и Y, то мы пилим, что A(3,1)=A(3,2)=0 и A(3,3)=1. С первого взгляда, кажется, что это не более чем формальная выкладка, но мы позже убедимся в пользе наших рассмотрении. Теперь, если мы представим вектор (X,Y) (который называется вектор-строкой) в виде

```
X
Y
I
```

то приведенные выше три уравнения запишутся в виде умножения столбца на матрицу:

```
X'          A(1,1)      A(1,2)      A(1,3)          X
Y'    =     A(2,1)      A(2,2)      A(2,3)    =     Y
I'          A(3,1)      A(3,2)      A(3,3)          I
```

Таким образом, если мы храним информацию о линейном преобразовании в виде матрицы, то мы можем найти образ каждой точки при преобразовании, если представим ее первоначальные координаты в виде столбца, а затем умножим на эту матрицу.

Многие авторы книг по компьютерной графике предпочитают этому вектору-столбцу вектор-строку (X,Y,I). Тогда преобразование запишется в виде

```
                                    A(1,1)      A(1,2)      A(1,3)
(X',Y',I)   =     (X,Y,I)     *     A(2,1)      A(2,2)      A(2,3)
                                    A(3,1)      A(3,2)      A(3,3)
```

Обратите внимание, что в этом случае матрица преобразования получается из нашей матрицы A с помощью так называемой операции транспонирования. Этот факт приводит к множеству недоразумений среди тех, кто не знаком с матричным исчислением. Именно поэтому в нашей книге мы придерживаемся обозначений с вектор-столбцами.

Как только вы освоитесь с матрицами получше, вы убедитесь в том, что неплохо переписать некоторые (или все) из программ, составленных ранее, в матричных обозначениях. Неважно, какой из методов применять, важно, что после выбора какого-либо метода следует придерживаться его раз и навсегда (обратите внимание, что матрица B, транспонированная по отношению к A, задается как B(I,J)=A(J,I), где 1<=I,J<=3.

Композиция преобразований

Важным свойством матриц является то, что если мы произведем сначала преобразование с матрицей A, а затем преобразование, которому соответствует матрица B, то результирующему преобразованию будет соответствовать матрица C=B*A. Обратите внимание на порядок сомножителей! Такой порядок сомножителей возникает потому, что мы выбрали представление векторов в виде столбцов; если бы представляли вектора строками, то порядок сомножителей был бы более естественным - сначала матрица A, а затем B.

Мы должны написать подпрограмму "MULT2" для умножения двух матриц. Язык бейсик не позволяет передавать векторные переменные в качестве параметров подпрограммы, так что нам придется разработать эффективные методы обхода этого недостатка. Предположим, что все матричные умножения производятся над матрицами A и R с результатом - матрицей B, которая записывается обратно в R. Выбор обозначений и операция копирования станут понятны позже. Нам также потребуется подпрограмма ("IDR2"), присваивающая R значение единичной матрицы. Если нам потребуется вычислить произведение последовательности матриц, то мы вначале положим R=1, а затем каждую матрицу в последовательности справа налево, будем называть A и вызывать подпрограмму "MULT2". По окончании этой процедуры результат произведения последовательности матриц будет храниться в R (см. листинг 4.1)

LISTING 4.1

```
9100 REM MULT2
9101 REM IN - A(3,3)
9102 REM OUT - R(3,3)
9110 FOR I=1 TO 3
9120 FOR J=1 TO 3
9130 LET AR=0
9140 FOR K=1 TO 3
9150 LET AR=AR+A(I,K)*R(K,J)
9160 NEXT K
9170 LET B(I,J)=AR
9180 NEXT J
9190 NEXT I
9200 FOR I=1 TO 3
9210 FOR J=1 TO 3
```

```
9220 LET R(I,J)=B(I,J)
9230 NEXT J
9240 NEXT I
9250 RETURN
9300 REM IDR2
9302 REM OUT - R(3,3)
9310 FOR I=1 TO 3
9320 FOR J=1 TO 3
9330 LET R(I,J)=0
9340 NEXT J
9350 LET R(I,I)=1
9360 NEXT I
9370 RETURN
```

Все естественные преобразования плоскости могут быть представлены в виде произведения трех преобразований: переноса, растяжения и вращения относительно начала координат. Следует подчеркнуть, что матрицы всех этих преобразований невырождены.

Следующие подпрограммы формируют матрицу A для каждого из этих преобразований, что делает возможным использование подпрограммы "MULT2" для того, чтобы составить композицию этих преобразований.

Перенос

Точка с координатами (X,Y) переводится в точку с новыми координатами (X',Y') с помощью вектора (TX,TY):

```
X′=1*X+0*Y+TX
Y′=0*X+1*Y+TY
```

так что матрица этого преобразования

```
1     0     TX
0     1     TY
0     0      1
```

Программа "TRAN2" генерирует такую матрицу по заданным значениям параметров TX и TY (см. листинг 4.2).

LISTING 4.2

```
9000 REM TRAN2
9001 REM IN - TX,TY
9002 REM OUT - A (3,3)
9010 FOR I=1 TO 3
9020 FOR J=1 TO 3
9030 LET A(I,J)=0
9040 NEXT J
9050 LET A(I,I)=1
9060 NEXT I
9070 LET A(1,3)=TX: LET A(2,3)=TY
9080 RETURN
```

Растяжение

При таком преобразовании координата X умножается на SX, а Y-координата точки на SY;

```
X′=SX*X+0*Y+0
Y′=0*X+SY*Y+0
```

и соответствующая матрица

```
SX    0     0
0     SY    0
0     0     1
```

Обычно SX и SY положительны, но если одно из них отрицательно, то это приводит дополнительно к отражению, в частности если SX=-1 и SY=1, то точка отражается относительно оси Y. Подпрограмма "SCALE2" генерирует матрицу A по заданным значениям X и Y.

LISTING 4.3

```
8900 REM SCALE2
8901 REM IN - SX,SY
8902 REM OUT - A(3,3)
8910 FOR I=1 TO 3
8920 FOR J=1 TO 3
8930 LET A(I,J)=0
8940 NEXT J
8950 NEXT I
8960 LET A(1,1)=SX: LET A(2,2)=SY: LET A(3,3)=1
8970 RETURN
```

Вращение относительно начала координат

Если поворачиваем плоскость на угол TETA против часовой стрелки (такой способ отсчета является общепринятым в математической литературе) относительно начала координат, то уравнения, определяющие это

преобразование, имеют вид:

```
X'=COS(TETA)*X-SIN(TETA)*Y+0
Y'=SIN(TETA)*X+COS(TETA)*X+0
```

и матрица:

```
COS(TETA)    -SIN(TETA)   0
SIN(TETA)    COS(FKTA)    0
0            0            1
```

Подпрограмма "ROT2" генерирует матрицу вращения A по заданному значению TETA (см. листинг 4.4) .

LISTING 4.4

```
8600 REM ROT2
8601 REM IN - THETA
8602 REM OUT - A(3,3)
8610 FOR I=1 TO 3
8620 FOR J=1 TO 3
8630 LET A(I,J)=0
8640 NEXT J
8650 NEXT I
8660 LET A(3,3)=1
8670 LET CT=COS THETA: LET ST=SIN THETA
8680 LET A(1,1)=CT: LET A(2,2)=CT
8690 LET A(1,2)=-ST: LET A(2,1)=ST
8700 RETURN
```

Обратное преобразование

Для каждого преобразования имеется преобразование, которое восстановит исходные позиции точек. Если матрице соответствует матрица A, то обратному преобразованию соответствует матрица A**(-1). Если преобразование разложено в композицию элементарных, то нет необходимости в вычислении обратной матрицы использовать подпрограмму на листинге 7.5, мы можем найти ее, используя программы на листингах 4.2, 4.3, 4.4 со значениями параметров, которые вычисляются по значениям соответствующих параметров исходного преобразования.

- Преобразование переноса с (TX,TY) обращается преобразованием переноса с (-TX,-TY);
- Растяжение со значением коэффициентов растяжения SX и SY обращается растяжением со значением коэффициентов 1/SX, 1/SY (ни одно из чисел SX и SY не обращается в нуль, в противном случае двумерное пространство выродилось бы в прямую).
- Поворот на угол TETA обращается поворотом на угол -TETA;
- Если преобразование есть произведение преобразований вращения, переноса и растяжения, например, A*B*C*…*M, то матрица обратного преобразования:

```
M**(-1)*...*C**(-1)*B**(-1)*A**(-1)
```

Обратите внимание на порядок сомножителей!

Размещение объекта

Нам часто приходится рисовать предмет в различных участках экрана и по разному повернутым. Чрезвычайно неудобно было бы каждый раз вычислять вручную координаты точек предмета, а затем вводить их в программу.

Вместо этого введём сначала произвольную, но фиксированную систему координат, которую мы назовем абсолютной системой координат (ABSOLUTE). Затем мы выбираем наиболее удобную систему координат, в которой мы задаем координаты вершин фигуры; такое положение фигуры мы назовем SETUP.

Используя матрицы, можно преобразовать вершины объекта из положения SETUP в положение ACTUAL (фактическое положение) в системе координат ABSOLUTE. Положение соответствующих линий и областей устанавливается положением вершин фигуры. Матрицу преобразования, переводящую SETUP в ACTUAL, мы всюду в тексте будем обозначать буквой P (иногда у этой буквы будет появляться индекс, с тем чтобы можно было идентифицировать матрицы, относящиеся к различным таким преобразованиям) . Из-за ограничения на передачу векторных параметров подпрограмм, обычно мы не будем генерировать непосредственно P, а будем использовать эту матрицу для модификации R.

Рассматривание объекта

Мы ввели метод перемещения предмета по неподвижной координатной системе ABSOLUTE. Предположим, что взгляд наблюдателя направлен в точку X,Y, системы координат ABSOLUTE, а голова наклонена под углом ALPHA. Удобнее полагать, что взгляд направлен на начало координат, а голова не наклонена (такое положение называется OBSERVED - наблюдаемое положение). Таким образом, нам приходится ввести еще одну матрицу Q, переводящую позицию ACTUAL в OBSERVED.

Эта матрица получается комбинацией трансляции точек пространства на вектор (-DX,-DY), которому соответствует матрица A, и последующим поворотом на угол -ALPHA, которому соответствует матрица B (обратите внимание на знак минус!). Так матрица Q=B*A генерируется программой "LOOK2", приведенной на листинге 4.5. Как правило, мы не вычисляем матрицу Q явно, а используем ее только для преобразования матрицы P. Эту матрицу имеет смысл вычислять явно и сохранить в памяти.

LISTING 4.5

```
8200 REM LOOK2
```

```
8202 REM OUT - R(3,3)
8210 INPUT "(DX,DY)=";DX;",";DY
8220 INPUT "ALPHA=";ALPHA
8229 REM LOOK AT (DX,DY)
8230 LET TX=-DX: LET TY=-DY
8240 GO SUB TRAN2: GO SUB MULT2
8249 REM TILT HEAD THROUGH ALPHA RADIANS.
8250 LET THETA=ALPHA
8260 GO SUB ROT2: GO SUB MULT2
8270 RETURN
```

Вычерчивание объекта

Комбинируя матрицу преобразования P, переводящую положение SETUP в положение ACTUAL с матрицей Q, переводящей ACTUAL в OBSERVED, получим матрицу R=Q+P, переводящую SETUP в OBSERVED (мы всегда будем обозначать эту матрицу R; следует иметь в виду, что R всегда получается использованием подпрограммы "MULT2"). Преобразовывая все вершины фигуры в положении R и производя соответствующее преобразование линий и областей, мы получим координаты точек фигуры в системе координат наблюдателя, который смотрит в начало системы координат ABSOLUTE и держит голову вертикально, то есть такого наблюдателя, который смотрит на экран.

Таким образом, мы отождествляем систему координат ABSOLUTE с системой, соответствующей полю экрана для определения координат точек фигуры на графическом поле экрана с тем, чтобы нанести на экран вершины, прямые и области, из которых состоит объект. На практике это достигается с помощью конструирующей подпрограммы, использующей матрицу R. Эта подпрограмма установит необходимую информацию о вершинах фигуры, линиях и областях, преобразует вершины с помощью матрицы R, а затем, возможно, нарисует объект (см. пример 4.A).

Позже мы увидим, что существуют ситуации, в которых удобнее сохранить в памяти информацию о вершинах, линиях и областях фигуры. Например, координаты вершин могут храниться в одномерных массивах X и Y; информация о прямых – в двумерном массиве I. Координаты вершин могут запоминаться в положении исходное, фактическое, или наблюдаемое - это зависит от программы. Этот метод используется для создания динамической последовательности сцен: объекты могут перемещаться относительно системы координат абсолютно, относительно самих себя и, кроме того, обозреватель может перемещаться по экрану. Рассмотрим вначале простейший случай неподвижного изображения.

Сложные изображения – метод построения изображения "по кирпичикам"

У нас есть метод, позволяющий генерировать последовательность подобных изображений объекта. Для этого достаточно создать программу, рисующую объект, а затем вызывать ее с различными значениями матрицы P, соответствующей преобразованию из исходного положения в наблюдаемое. Изображение создается подпрограммой "SCENE2", которая вызывается из стандартной главной программы (см. листинг 4.6). Эта главная программа используется для установления меток различных подпрограмм, объявляет массивы, устанавливает значения параметров VERT и HORIZ, а затем вызывает подпрограмму "SCENE2".

LISTING 4.6

```
100 REM MAIN PROGRAM
109 REM Define identifiers for initialisation and 2-D PLOT routines.
110 LET START=9700: LET SETORIGIN=9600: LET MOVETO=9500: LET LINETO=9400: LET
CLIP=8400
120 LET ROT2=8600: LET ANGLE=8800: LET SCALE2=8900: LET TRAN2=9000: LET
MULT2=9100: LET IDR2=9300
130 LET SCENE2=6000: LET LOOK2=8200
139 REM INITIALISE AND CENTRE GRAPHICS AREA.
140 INPUT "HORIZ=";HORIZ;",";"VERT=";VERT
150 GO SUB START
160 LET XMOVE=HORIZ*0.5: LET YMOVE=VERT*0.5
170 GO SUB SETORIGIN
179 REM SET THE SCENE.
180 GO SUB SCENE2
190 STOP
```

"SCENE2" сначала вызовет "LOOK2" и вычислит матрицу Q, и если ей предстоит начертить более одной фигуры, то эта матрица заносится в память. Для каждого отдельного элемента ("кирпичика") мы вычисляем матрицу P и затем вызываем необходимую подпрограмму построения изображения, используя R=Q+P. В итоге из кирпичиков выстраивается необходимая картинка. Для того, чтобы различать матрицы, относящиеся к разным элементам, мы добавляем индексы.

Может быть, предложенный нами модульный подход к решению и не является самым эффективным, но наш опыт показывает, что этот метод чрезвычайно проясняет решение для начинающих и позволяет им задавать правильные вопросы о процессе построения того или иного объекта. Мы увидим, что при мультиплицировании изображения наш подход позволяет уменьшить количество подзадач, так как в этом случае не только элементы перемещаются относительно друг друга, но и движется наблюдатель.

Естественно, что если считать, что наблюдатель держит голову вертикально, то вычисление матрицы можно

свести к обращению к подпрограмме "SETORIGIN" (установить начало), которая изменяет положение начала координат системы координат, связанной с экраном. Если наблюдатель смотрит на начало координат и его голова вертикальна, то матрица Q равна 1; в этом случае она не принимает участия в генерации изображения и тогда подпрограммой "LOOK2" можно пренебречь. Однако, мы будем иметь дело с общим случаем. Мы ставим своей целью объяснить понятие преобразования пространства в наиболее общем случае, используя самые прямые методы, несмотря на то, что это может потребовать от читателя больших усилий при чтении. К этим подпрограммам следует возвращаться по мере изучения книги. Позже мы наметим пути изменения этих программ так, чтобы они были эффективными в конкретных приложениях.

Однако основная причина использования модульного метода построения программ станет понятной при изучении трехмерной графики. Перед тем как изучать методы трехмерной графики, следует хорошо изучить двухмерную графику, так как многие методы являются общими для этих случаев.

Пример 4.А

Рассмотрим простейшее изображение космического корабля, который в исходном положении направлен параллельно оси X. Контур корабля получается последовательным соединением точек (3,0), (0,0), (-1,1), (2,0), (-1,-1) и снова (0,0). См. рис. 21, на котором изображен корабль; размеры графического поля 5 блоков на 3 блока, а матрица, переводящая исходное положение в фактическое, единичная. Матрица, соответствующая преобразованию исходное в наблюдаемое, такова, что взгляд наблюдателя направлен в точку (1,0), а его голова вертикальна.

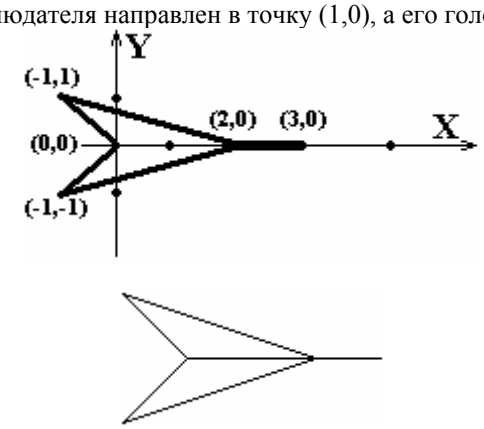


Рис.21.

На листинге 4.7 приведена подпрограмма "SCENE2" в том виде, в каком она необходима для установки объекта в правильное положение; на листинге 4.8 приведена подпрограмма, вычерчивающая контуры корабля.

Обратите внимание, что подпрограмма "SHIP", которая использует матрицу P для перемещения вершин корабля (а следовательно, и самого корабля) в положение координат последних, а вместо этого сохраняются значения X(6), Y(6) с тем, чтобы можно было вызывать подпрограмму повторно для генерации изображения в других положениях.

LISTING 4.7

```
6000 REM SCENE2/LOOK2 - SHIP (NOT STORED)
6010 DIM X(6):DIM Y(6)
6020 DIM A(3,3): DIM B(3,3): DIM R(3,3)
6030 LET SHIP=6500
6039 REM PLACE THE OBSERVER.
6040 GO SUB IDR2: GO SUB LOOK2
6049 REM DEFINE AND DRAW OBJECT.
6050 GO SUB SHIP
6060 RETURN
```

LISTING 4.8

```
6500 REM SHIP/ NOT STORED
6501 REM IN - R(3,3)
6510 DATA 3,0,0,0,-1,1,2,0,-1,-1,0,0
6520 RESTORE SHIP
6530 FOR I=1 TO 6
6539 REM READ COORDINATES OF SETUP OBJECT.
6540 READ XX,YY
6549 REM MOVE OBJECT INTO OBSERVED POSITION.
6550 LET X(I)=XX*R(1,1)+YY*R(1,2)+R(1,3)
6560 LET Y(I)=XX*R(2,1)+YY*R(2,2)+R(2,3)
6570 NEXT I
6579 REM JOIN VERTICES IN ORDER.
6580 LET XPT=X(1): LET YPT=Y(1): GO SUB MOVETO
6590 FOR I=2 TO 6
6600 LET XPT=X(I): LET YPT=Y(I): GO SUB LINETO
6610 NEXT I
6620 RETURN
```

Пример 4.Б

Предположим, что мы хотим начертить рисунок, состоящий из четырех космических кораблей A, B, C, D, размещенных на экране 60x40. Для простоты мы полагаем, что Q - единичная матрица, так что голова наблюдателя вертикальна, а взгляд направлен в начало координат.

Корабль A расположен так же, как и в положении исходное; это означает, что R(A)=1, в то время как корабль B переводится из положения исходное в фактическое с помощью следующих преобразований:

1. Фигура растягивается; параметры растяжения X=4, Y=2; соответствует этому преобразованию матрица A;
2. Фигура поворачивается на PI/6 радиан матрицей B;
3. Производится параллельный перенос с параметрами TX=6, TY=4 - матрица C.

$$A = \begin{matrix} 4 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{matrix} \qquad B = \begin{matrix} \frac{SQRT(3)}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & \frac{SQRT(3)}{2} & 0 \\ 0 & 0 & 1 \end{matrix}$$

$$C = \begin{matrix} 1 & 0 & 6 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{matrix}$$

Результирующее преобразование задается матрицей

R(B)=Q*P(B)=I*P(B)=P(B)=C*B*A

Обратите внимание на то, как используется нижний индекс для указания того, что матрица относится к элементу B.

Если мы используем другой порядок сомножителей A*B*C (что дает матрицу P), тогда

$$P(B) = \begin{matrix} 2*SQRT(3) & -1 & 6 \\ 2 & SQRT(3) & 4 \\ 0 & 0 & 1 \end{matrix} \qquad P(D) = \begin{matrix} 2*SQRT(3) & -2 & 12*SQRT(3)-8 \\ 1 & SQRT(3) & 4*SQRT(3)+6 \\ 0 & 0 & 1 \end{matrix}$$

откуда видно, что мы пришли к совершенно другому преобразованию. Матрице P(D)=Q*P(D)=I*P(D) соответствует корабль D. Обратите внимание, что корабль ассиметричен; пользуясь операцией растяжения, следует помнить, что она определена относительно начала координат. Так что форма предмета, который удален от начала координат, может быть искажена.

Корабль C получается из корабля B отражением относительно прямой 3*Y=-4*X-9. Эта прямая пересекает ось Y в точке (0,3) и образует с положительной полуосью X угол GAMMA

GAMMA=ARCCOS(-3/5)=ARCSIN(4/5)=ARCTG(-3/4)

Если мы совершим параллельный перенос на вектор (0,3), которому соответствует матрица D, то эта прямая пройдет через начало координат; если мы повернем плоскость на угол ALPHA матрицей E, то наша прямая совпадет с осью X. Матрица F тогда отразит корабль относительно оси X, матрица E**(-1) повернет прямую так, что она снова будет составлять угол с осью X, и матрица D**(-l) вернет прямую в исходное положение. Таким образом, матрица G=D**(-1)*E**(-1)*F*E*D отражает вершины корабля относительно прямой 3*Y=-4*X-9 и R(C)=I*P©=G*P(B) может использоваться для генерации изображения C, что означает, что мы сначала матрицей P устанавливает корабль в положение B, а затем матрицей G в положение C.

$$D = \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{matrix} \qquad E = \begin{matrix} -\frac{3}{5} & \frac{4}{5} & 0 \\ -\frac{4}{5} & -\frac{3}{5} & 0 \\ 0 & 0 & 1 \end{matrix}$$

$$F = \begin{matrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{matrix} \qquad R(C) = \frac{1}{25}\begin{pmatrix} -48-14*SQRT(3) & 7 & -24*SQRT(3)-210 \\ -14-48*SQRT(3) & 24 & 7*SQRT(3)-170 \\ 0 & 0 & 25 \end{pmatrix}$$

На листинге 4.9 приведена новая программа для рисования нашей фигуры. Обратите внимание, что "SCENE2" не вызывает "LOOK2", так как мы полагаем, что взгляд наблюдателя направлен на начало координат, и голову он держит прямо. Основная программа и подпрограмма "SHIP", а также все остальные графические подпрограммы остаются неизменными.

LISTING 4.9

```
6000 REM SCENE2/NO LOOK2 4 SHIP (NOT STORED).
6010 DIM X(6): DIM Y(6)
6020 DIM A(3,3): DIM B(3,3): DIM R(3,3)
6030 LET SHIP=6500
6038 REM OBSERVED=ACTUAL. NO NEED TO CALL "LOOK2".
6039 REM SHIP A.
6040 GO SUB IDR2: GO SUB SHIP
```

```
6049 REM SHIP B.
6050 LET SX=4: LET SY=2
6060 GO SUB SCALE2: GO SUB MULT2
6070 LET THETA=PI/6
6080 GO SUB ROT2: GO SUB MULT2
6090 LET TX=6:LET TY=4
6100 GO SUB TRAN2: GO SUB MULT2
6110 GO SUB SHIP
6119 REM SHIP C.
6120 LET AX=-3:LET AY=4
6130 GO SUB ANGLE
6140 LET TX=0:LET TY=3
6150 GO SUB TRAN2: GO SUB MULT2
6160 LET THETA=-THETA
6170 GO SUB ROT2: GO SUB MULT2
6180 LET SX=1: LET SY=-1
6190 GO SUB SCALE2: GO SUB MULT2
6200 LET THETA=-THETA
6210 GO SUB ROT2: GO SUB MULT2
6220 LET TX=0: LET TY=3
6230 GO SUB TRAN2: GO SUB MULT2
6240 GO SUB SHIP
6249 REM SHIP D.
6250 GO SUB IDR2
6260 LET TX=6: LET TY=4
6270 GO SUB TRAN2: GO SUB MULT2
6280 LET THETA=PI/6
6290 GO SUB ROT2: GO SUB MULT2
6300 LET SX=4:LET SY=2
6310 GO SUB SCALE2: GO SUB MULT2
6320 GO SUB SHIP
6330 RETURN
```

Упражнение 4.1

Для того, чтобы убедиться, что программу можно использовать в общей ситуации, пропустите эту программу с матрицей Q, отличной от единицы, что достигается ненулевым значением DX, DY или ALPHA.

Ваша программа "SCENE2" должна вызывать "LOOK2" для вычисления Q, которая хранится в памяти. Затем для каждого объекта на экране вычисляется матрица P перехода из исходного положения в фактическое (эта матрица хранится подпрограммой "MULT2" в R), умножается на матрицу Q, которая копируется в матрицу A для использования в подпрограмме "MULT2", а затем вызывается подпрограмма построения с матрицей R=Q+P.

Используйте подпрограмму "LINETO", которая вызывает подпрограмму "CLIP", так как в противном случае при вычерчивании участков фигур, выходящих за графическое поле, произойдет сбой программы.

Упражнение 4.2

Используйте предыдущие подпрограммы для вычерчивания фигур кораблей, но числа, положение и направление кораблей вводится с клавиатуры. Вы можете написать свои подпрограммы для генерирования более сложных изображений; мы выбрали очень простую картинку, так как не хотели усложнять алгоритм сложностью самих объектов.

Этим методом можно манипулировать с любым числом точек и линий, которыми может справиться SPECTRUM с учетом ограничений на память и время. Объекты могут быть не просто состоящими из линий, но и имеющими разноцветные участки (многоугольники, вершины которых являются преобразованными точками).

Упражнение 4.3

Используя циклы, можно генерировать последовательности объектов, например, они могут быть одинаково направлены, а их центры расположены на прямой P+MU*Q на одинаковом расстоянии друг от друга.

Мы можем генерировать значения MU с помощью цикла и рисовать по одному кораблю на каждом шаге; при этом мы можем изменять параметры преобразования регулярным образом, используя MU, P и Q.

Эти новые значения параметров используются для вычисления матрицы преобразования из исходного положения в фактическое на каждом шаге, передвигая таким образом объект в новое положение. R=Q*P=I*P используется для наблюдения за объектом и для его вычерчивания. Используя эти идеи, нарисуйте картины боев между кораблями.

Эффективное использование матриц

Очевидно, что какую бы мы комбинацию преобразований ни использовали, третьей строкой матрицы всегда будет (0,0,1). Если мы будем использовать только две верхних строки, то это сделает нашу подпрограмму значительно более эффективной.

Мы продолжаем хранить матрицы 3x3, так как процедуры, написанные нами ранее, используют матрицы 3x3. Изменение размерности массивов привело бы к нарушению границ массивов в предыдущих подпрограммах - лишние три столбца, добавленные к каждой матрице; это малая цена, которую мы платили, чтобы избежать ошибки.

Обратите внимание, что когда мы объявляем массив оператором DIM, то элементам массива присваивается

нулевое значение. Перепишем с учетом этого обстоятельства, подпрограммы на листингах 4.1, 4.2, 4.3, 4.4 и обозначим их 4.1A, 4.2A, 4.3A, 4.4A соответственно.

LISTING 4.1 A

```
9100 REM MULT2
9101 REM IN - A(3,3),R(3,3)
9102 REM OUT - R(3,3)
9110 FOR I=1 TO 2
9120 FOR J=1 TO 3
9130 LET B(I,J)=A(I,1)*R(1,J)+A(I,2)*R(2,J)
9140 NEXT J
9150 LET B(I,3)=B(I,3)+A(I,3)
9160 NEXT I
9170 FOR J=1 TO 3
9180 LET R(1,J)=B(1,J): LET R(2,J)=B(2,J)
9190 NEXT J
9200 RETURN
9300 REM IDR2
9302 REM OUT - R(3,3)
9310 DIM R(3,3)
9320 LET R(1,1)=1: LET R(2,2)=1
9330 RETURN
```

LI STING 4.2 A

```
9000 REM TRAN2
9001 REM IN - TX,TY
9002 REM OUT - A(3,3)
9010 DIM A(3,3)
9020 LET A(1,1)=1: LET A(2,2)=1: LET A(1,3)=TX: LET A(2,3)=TY
9030 RETURN
```

LISTING 4.3 A

```
8900 REM SCALE2
8901 REM IN - SX.SY
8902 REM OUT - A(3,3)
8910 DIM A(3,3)
8920 LET A(1,1)=SX: LET A(2,2)=SY
8930 RETURN
```

LISTING 4.4A

```
8600 REM ROT2
8601 REM IN - THETA
8602 REM OUT - A(3,3)
8610 DIM A(3,3)
8620 LET CT=COS THETA: LET ST=SIN THETA
8630 LET A(1,1)=CT: LET A(2,2)=CT
8640 LET A(1,2)=ST: LET A(2,1)=ST
8650 RETURN
```

Может показаться, что пример с нашими кораблями несколько надуман, так как координаты объектов выбраны произвольно. Однако при построении большинства диаграмм положение объектов определяется явным образом, исходя из строения диаграммы (см. приведенные ниже примеры).

Пример 4. B

Напишите программу, которая чертит эллипс с главными осями A и B и центром в точке (CX,CY). Большая ось образует с осью X угол TETA.

Обратите внимание на то, что порядок операций здесь существенен: сначала поворачивайте, а затем переносите. Если мы хотим генерировать эллипсы с горизонтальной главной осью, то мы можем не использовать матрицы, используя подпрограмму упражнения 2.5 и методы подпрограмм на листинге 2.9.

На листинге 4.10 приводится подпрограмма "SCENE2", которая считывает информацию об эллипсе, вычисляет матрицу преобразования исходного положения в фактическое, а затем вызывает подпрограмму "ELLIPCE" для вычерчивания эллипса (см. рис.22).

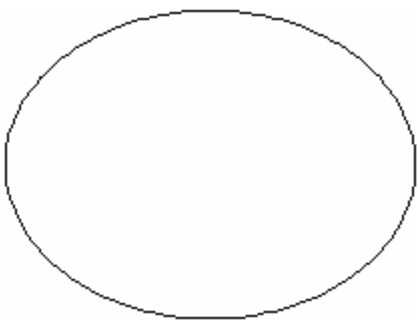


Рис. 22.

LISTING 4.10

```
6000 REM SCENE2/ELLIPSE
6010 DIM A(3,3): DIM B(3,3): DIM R(3,3)
6020 LET ELLIPSE=6500
6030 INPUT "(CX,CY)=";CX;",";CY;" A=";A;" B=";B;" THETA=";THETA
6040 LET THETA=-THETA
6049 REM Ellipse centred at (CX,CY) and tilted through angle THETA.
6050 GO SUB IDR2
6060 GO SUB ROT2: GO SUB MULT2
6070 LET TX=CX: LET TY=CY
6080 GO SUB TRAN2: GO SUB MULT2
6090 GO SUB ELLIPSE
6100 RETURN
6500 REM ELLIPSE
6501 REM IN - A,B,R(3,3)
6509 REM ELLIPSE. MAJOR AXIS A. MINOR AXIS B.
6510 LET XPT=A*R(1,1)+R(1,3)
6520 LET YPT=A*R(2,1)+R(2,3)
6530 GO SUB MOVETO
6540 LET ALPHA=0: LET ADIF=PI/100
6549 REM Calculate points (XPT.YPT) on ellipse, in observed position.
6550 FOR I=1 TO 200
6560 LET ALPHA=ALPHA+ADIF
6570 LET AA=A*COS ALPHA: LET BB=B*SIN ALPHA
6580 LET XPT=AA*R(1,1)+BB*R(1,2)+R(1,3)
6590 LET YPT=AA*R(2,1)+BB*R(2,2)+R(2,3)
6600 GO SUB LINETO
6610 NEXT I
6620 RETURN
```

Упражнение 4.4

Напишите подпрограмму, рисующую единичный перемещаемый объект, а затем используйте матричную технику для составления комбинаций этих объектов. Для примера возьмите следующую фигуру, заданную в параметрической форме:

```
(R(COS(TETA)**3)/(SIN(TETA)**3)),
```

где 0<=TETA<=2*PI, а R - радиус (максимальное расстояние точки на кривой до начала координат)

Параметры, необходимые для работы этой подпрограммы - это радиус R и матрица преобразования.

Упражнение 4.5

Поэкспериментируйте с этой матричной техникой. Напишите подпрограмму, которая поворачивает плоскость относительно точки (X,Y) на угол TETA. Напишите программу, которая осуществляет отражение плоскости относительно прямой A*Y=B*X+C.

Хранение информации об изображении

Ранее мы упоминали, что иногда приходится запоминать всю информацию об изображении перед тем, как выйти из подпрограммы построения изображения. В этом случае информация хранится в одномерных массивах X и Y длины большей или равной NOV, где NOV - количество вершин, которые необходимо запомнить (эти точки могут соответствовать позиции SETUP, ACTUAL или OBSERVED, все зависит от контекста). Нам также потребуется хранить информацию о прямых, в двумерном массиве которого первый индекс изменяется от 1 до 2, а второй изменяется от 1 до величины, большей или равной NOL, где NOL - количество прямых в изображении. I-ая прямая соединяет точки с индексами L(1,I) и L(2,I); мы видим, что эта информация зависит от предыдущей, она просто сообщает нам номера точек, которые соединяет I-ая прямая. Значениям параметров NOV и NOL присваиваются начальные значения в подпрограмме "SCENE2" и увеличиваются подпрограммами построения изображения.

Теперь мы можем в программах построения изображения не чертить прямые, а использовать их только для записи в память информации о вершинах, прямых и т.д. (преобразованных матрицей R). После того как вся информация об изображении записана в память подпрограммой "SCENE2", вызывается подпрограмма "DRAWIT" ("РИСУЙ"), которая и вычерчивает изображение. Программа для создания изображения наших кораблей, использующая новый метод, получается из подпрограммы на листинге 4.9 внесением трех небольших изменений:

```
6010 DIM X(20): DIM Y(20): DIM L(2,20)
6030 LET NOV=0: LET NOL=0: LET SHIP=6500: LET DRAWIT=7000
6330 GO SUB DRAWIT: RETURN
```

Эта подпрограмма используется вместе с листингом 4.11, где имеется текст подпрограммы, записывающей информацию об изображении и рисующую подпрограмму.

LISTING 4.11

```
6500 REM SHIP/STORED
6501 REM IN - NOV,NOL,R(3,3),X(NOV),Y(NOV)
6502 REM OUT - NOV,NOL,X(NOV), Y(NOV),L(2,NOL)
6510 DATA 3,0,0,0,-1,1,2,0,-1,-1,1,2,2,3,3,4,4,5,5,2
```

```
6520 RESTORE SHIP
6530 LET NV=NOV
6539 REM Read vertices and move into position using matrix R.
6540 FOR I=1 TO 5
6550 READ XX,YY
6560 LET NOV=NOV+1
6570 LET X(NOV)=XX*R(1,1)+YY*R(1,2)+R(1,3)
6580 LET Y(NOV)=XX*R(2,1)+YY*R(2,2)+R(2,3)
6590 NEXT I
6599 REM READ AND STORE LINE INFORMATION.
6600 FOR I=1 TO 5
6610 READ L1,L2
6620 LET NOL=NOL+1
6630 LET L(1,NOL)=L1+NV: LET L(2,NOL)=L2+NV
6640 NEXT I
6650 RETURN
7000 REM DRAWIT
7001 REM IN - NOV,X(NOV),Y(NOV),NOL,L(2,NOL)
7009 REM DRAW LINE BY JOINING PAIRS OF VERTICES.
7010 FOR I=1 TO NOL
7020 LET L1=L(1,I): LET L2=L(2,I)
7030 LET XPT=X(L1): LET YPT=Y(L1): GO SUB MOVETO
7040 LET XPT=X(L2): LET YPT=Y(L2): GO SUB LINETO
7050 NEXT I
7060 RETURN
```

Предположим, что мы хотим создать несколько видов одной и той же картины (мы опять используем в качестве примера наш космический корабль); это означает, что при той же матрице преобразования P SETUP в ACTUAL, но с различными матрицами Q преобразования из ACTUAL в OBSERVED. Очевидное решение этой задачи – записать в память информацию о вершинах в положении ACTUAL.

Теперь для каждого нового положения OBSERVED мы вычисляем Q и вводим это значение в другую подпрограмму "DRAWIT" (см. листинг 4.12, который отличается от листинга 4.11), которая переводит каждую вершину из положения ACTUAL в OBSERVED с помощью матрицы Q, записывает координаты этих вершин в массивы V и W и использует эти значения, когда они требуются для генерации изображения. Когда мы используем этот метод создания изображения, мы изменяем только подпрограммы "SCENE2" и "DRAWIT" и то лишь слегка (см. листинг 4.12).

LISTING 4.12

```
6000 REM SCENE2/VARIABLE LOOK2. 4 SHIPS (STORED).
6009 REM CONSTRUCT 4 SHIPS STORED IN ACTUAL POSITION.
6010 DIM X(20): DIM Y(20): DIM V(20): DIM W(20): DIM L(2,20)
6020 DIM A(3,3): DIM B(3,3): DIM R(3,3)
6030 LET SHIP=6500: LET DRAWIT=7000
6039 REM SHIP A.
6040 LET NOV=0:LET NOL=0: GO SUB IDR2: GO SUB SHIP
6049 REM SHIP B.
6050 LET SX=4: LET SY=2
6060 GO SUB SCALE2: GO SUB MULT2
6070 LET THETA=PI/6
6080 GO SUB ROT2: GO SUB MULT2
6090 LET TX=6:LET TY=4
6100 GO SUB TRAN2: GO SUB MULT2
6110 GO SUB SHIP
6119 REM SHIPC.
6120 LET AX=-3:LET AY=4
6130 GO SUB ANGLE
6140 LET TX=0:LET TY=3
6150 GO SUB TRAN2: GO SUB MULT2
6160 LET THETA=-THETA
6170 GO SUB ROT2: GO SUB MULT2
6180 LET SX=1:LET SY=-1
6190 GO SUB SCALE2: GO SUB MULT2
6200 LET THETA=-THETA
6210 GO SUB ROT2: GO SUB MULT2
6220 LET TX=0:LET TY=-3
6230 GO SUB TRAN2: GO SUB MULT2
6240 GO SUB SHIP
6249 REM SHIP D.
```

```
6250 GO SUB IDR2
6260 LET TX=6:LET TY=4
6270 GO SUB TRAN2: GO SUB MULT2
6280 LET THETA=PI/6
6290 GO SUB ROT2: GO SUB MULT2
6300 LET SX=4:LET SY=2
6310 GO SUB SCALE2: GO SUB MULT2
6320 GO SUB SHIP
6329 REM LOOP THROUGH OBSERVATION POINTS.
6330 GO SUB IDR2: GO SUB LOOK2
6340 CLS: GO SUB DRAWIT
6350 GO TO 6330
6360 RETURN
7000 REM DRAWIT
7001 REM IN - NOV,NOL,R(3,3),X(NOV),Y(NOV),L(2,NOL)
7009 REM Transform vertices from actual to observed position.
7010 FOR I=1 TO NOV
7020 LET V(I)=X(I)*R(1,1)+Y(I)*R(1,2)+R(1,3)
7030 LET W(I)=X(I)*R(2,1)+Y(I)*R(2,2)-R(2,3)
7040 NEXT I
7049 REM DRAW LINES BY JOINING PAIRS OF VERTICES.
7050 FOR I=1 TO NOL
7060 LET L1=L(1,I): LET L2=L(2,I)
7070 LET XPT=V(L1): LET YPT=W(L1): GO SUB MOVETO
7080 LET XPT=V(L2): LET YPT=W(L2): GO SUB LINETO
7090 NEXT I
7100 RETURN
```

Упражнение 4.6

Создайте движущиеся изображения. Каждая новая сцена изображает плывущие относительно друг друга корабли. Движение кораблей должно задаваться каким-нибудь удобным образом. Наблюдатель тоже должен двигаться; например, в начальный момент времени взгляд наблюдателя направлен в начало координат, затем, через пять кадров, он смотрит в точку (10,10) и с каждым новым кадром голова наклоняется на 0.1 радиан.

Теперь у вас нет необходимости вводить значения (DX,DY) и ALPHA в подпрограмму "LOOK2", вместо этого они вычисляются в программе. После того, как вы прочитаете главу 13, вы сможете поместить эти пять сцен в память, а затем извлекать их с помощью подпрограммы "MOVIE" ("мультипликация"), если у вас 48-ми килобайтовый вариант SPECTRUMa.

Упражнение 4.7

Сгенерируйте диаграматизированное изображение комнаты, в которой вы живете - двумерное схематическое изображение столов, стульев и т.д. Различные типы предметов используют различные программы генерации изображения, и "SCENE2" должна считывать информацию о предметах в комнате. Как только ваша комната воссоздана в машинной форме, сгенерируйте последовательность видов вашей комнаты, которые получаются различными выборами места обзора и направления взгляда.

Вы можете пойти другим путем и сначала нарисовать схему ее под разными углами. Выбор различных сцен огромен!

Так как мы используем "CLIP" в подпрограмме "LINETO", то мы можем задавать небольшие значения HORIZ и VERT, что производит эффект "зума" – визуального увеличения участка схемы, при этом внешности участка прямых будут обрезаны.

Список программ

Мы сгруппируем листинги 3.4 (ANGLE-угол), 4.1A ("MULT2" и "IDR2"), 4.2A ("TRAN2"), 4.3A ("SCALE2"), 4.4A ("ROT2"), 4.5 ("LOOK2") и 4.6 ("MAIN PROGRAM") подзаголовком "LIB2".

1. "LIB1", "LIB2", листинг 4.7 ("SCENE2") и 4.8 "SHIP". Необходимая входная информация: HORIZ, VERT, DX, DY и ALPHA. Попробуйте значения 8, 5, 1, 1, 0.5. Зафиксируйте четыре из этих значений и изменяйте пятое небольшими шагами.
2. "LIB1", "LIB2", листинг 4.9A ("SCENE2") и 4.8 ("SHIP"). Входные данные: HORIZ, VERT. Попробуйте 30, 20, 60, 40.
3. "LIB1", "LIB2" и листинг 4.10 ("SCENE2EL"). Входные данные: HORIZ, VERT, CX, CY, A, B, THETA. Попробуйте 30, 20, 0, 0, 12, 9, 0. Снова зафиксируйте значения параметров, кроме одного, и медленно изменяйте значения оставшегося параметра.
4. "LIBI", "LIB2", листинги 4.9A (вариант "SCENE2N") и 4.11 ("SHIP" и "DRAWIT"). Входные данные: те же, что и в пункте 2.
5. "LIBI", "LIB2", листинги 4.11 ("SHIP") и 4.12 ("SCENE2" и "DRAWIT"). Входные данные: HORIZ, VERT, DX, DY, ALPHA. Попробуйте значения: 60, 40, 0, 0, 0. Последовательно изменяйте значения переменных.

ГЛАВА 5
Символьная графика ZX SPECTRUM

В первой главе (листинг 1.2) мы видели, что небольшой блок или символ задается восемью двоичными числами. Такой блок восемь пикселей на восемь называется символьным блоком. В SPECTRUMe имеются три вида символьных блоков: 96 символов стандартного набора, 16 графических символов и 21 блок-символ, определяемых пользователем. Последние два вида блоков доступны в графическом режиме. Эти символы появляются на экране при использовании оператора PRINT, так что мы должны познакомиться с этой командой.

Команда PRINT позволяет поместить символ в любую из 22 строк с 0-ой (верхней) по 21-ую (нижнюю), и в любой из 32 позиций этой строки с 0-ой (левой) по 31-ую (правую); символ находится в символьном блоке и каждый из этих блоков содержит восемь строк по восемь пикселей в каждом. Каждой линии соответствует содержимое одной ячейки памяти; каждая точка соответствует разряду восьмизначного двоичного числа. Это число соответствует десятичному числу от 0 до 255. В памяти хранится таблица символов. Для каждого символа команда PRINT либо находит восемь значений из таблицы или вычисляет их, и записывает их в соответствующую ячейку дисплей-файла. Последнее приводит к появлению на экране изображения символа.

Стандартный набор символов

Таблица набора стандартных символов хранится в ПЗУ, постоянном запоминающем устройстве компьютера. Для каждого из 96 символов отводится по восемь ячеек памяти, так что таблица занимает 768 (96x8) последовательных ячеек, начиная с ячейки с номером 15616. Каждый символ имеет свой уникальный кодовый номер. В таблице содержится информация, начиная с символа "пробел" (код 32) и кончая символом "COPYRIGHT" (код 127).

Когда команда PRINT запрашивает для некоторого числа восемь чисел, то сначала находится системная переменная CHAR$, где содержится адрес ячейки 256 (восемь раз значение кода пробела) меньше адреса начала таблицы символов.

Для нахождения адреса, с которого начинается информация для генерации изображения символа, код символа умножается на восемь и к нему прибавляется CHAR$. После этого PRINT копирует информацию в дисплей файл, и символ появляется на экране.

Пропустите следующую программу (листинг 5.1), основанную на листинге 1.2. Она показывает, какие вычисления проводятся в ходе этого процесса. Подробное объяснение, как определить ячейки, в которых находится содержимое дисплейного файла, мы даем в главе 13, а в данном примере мы используем только блок (0,0).

Рис.23 показывает работу программы для входного параметра "?". Для выхода из программы на вопрос "ANOTHER GO ?" ответьте "N".

LISTING 5.1

```
10 CLS : INPUT "UHTCH CHARACTER";A$
20 LET A$=A$(1): PRINT AT 2,0;"""";A$;""SELECTED.CODE(";A$;")=";CODE A$
30 PRINT AT 4,0;"CALCULATION OF DATA LOCATION"
40 LET CHARS=PEEK 23606+256*PEEK 23607
50 PRINT AT 6,2;"CHARS POINTS TO ";CHARS
60 LET TABLE=CODE A$*8
70 PRINT AT 7,2;"CODE A$*8=        ";TABLE
80 LET START=CHARS+TABLE
90 PRINT AT 8,17;"-------": PRINT AT 9,2;"DATA STARTS AT  ";START
100 LET CORNER=16384: PRINT AT 11,0;"TABLE LOC. VALUE";"    SCREEN LOC."
110 FOR I=0 TO 7
120 LET VALUE=PEEK (START+I): LET MEMORY=CORNER+256*I
130 PRINT AT I+13,2;START+I: PRINT AT I+13,22;MEMORY
140 PRINT AT I+13,13;VALUE
150 POKE MEMORY,VALUE
160 NEXT I
170 INPUT "ANOTHER GO ?";Y$: IF Y$="N" THEN STOP
180 GO TO 10
```

```
?
"?"SELECTED.CODE(?)=63
CALCULATION OF DATA LOCATION
  CHARS POINTS TO 15360
  CODE A$*8=        504
                 -------
  DATA STARTS AT  15864
TABLE LOC. VALUE    SCREEN LOC.
  15864       0         16384
  15865       60        16640
  15866       66        16896
  15867       4         17152
  15868       8         17408
  15869       0         17664
  15870       8         17920
  15871       0         18176

ANOTHER GO ?"■"
```

Рис. 23.

Результат работы программы для символа"?" приведен на рис. 23.

Упражнение 5.1

Перепишите программу на листинге 5.1 с тем, чтобы вы могли решать: помещать или не помещать значение переменной VALUE командой POKE в дисплейный файл. Если какое-либо из восьми значений VALUE вами отбраковывается, то введите замену для этого VALUE. Поэкспериментируйте с заменой одного или двух значений в VALUE символе.

Графический режим работы

Естественно было бы предположить, что с помощью адреса, значение которого хранится в CHARS, можно узнать о символах, соответствующих коду 0, однако кодам со значением от 0 до 31 либо соответствуют управляющие символы, либо вообще никакие символы не соответствуют. Хотя с помощью CHAR$ можно определить адреса ячеек, соответствующих этим значениям кода, но эти 256 (32*8) бит не содержат информации о символах и, следовательно, их можно использовать для других целей. Если задать графический символ в качестве параметра для подпрограммы на листинге 5.1, то мы либо не получим никакой информации о символах, либо она будет совершенно произвольной. Это связано с тем, что в таблице символов хранится информация о символах с кодами от 32 до 127, а у символов графического режима номер кода больше 127, что приведет к тому, что программа рассматривает место за концом таблицы символов. Эта таблица хранится в ПЗУ сразу перед началом памяти прямой адресации, которая используется для временного хранения информации, так что программа будет просматривать дисплейный файл, занимающий первые 6К RAM (ОЗУ).

Блочная графика

Информация для графического блока не хранится в ПЗУ, а каждый раз вычисляется по коду символа. Каждый графический блок состоит из четырех квадратов, состояние каждого представлено двоичным "вкл" или "выкл" одного из последних четырех двоичных разрядов номера квадрата. Четыре строки по 8 пикселей считаются идентичными, то же относится и к последним четырем строкам.

```
      1 0 0 0 1 1 0 1  =  CODE  (" --I ")
      1 0 0 0 X X X X  =  BLOCK   GRAPHIS
                             CHARACTER,
      X X X X 1 1 0 1  =  FOUR  CORNERS OF
                             CHARACTER,
1 1 1 1    1 1 1 1    0 0 0 0    1 1 1 1
0 0 0 0 1 1 1 1  =  UALUE  FOR TOP  4 LINE3
1 1 1 1 1 1 1 1  =  UALUE  BOTTOM   4 LINE3
        00001111              □□□□■■■■
        00001111              □□□□■■■■
        00001111              □□□□■■■■
        00001111              □□□□■■■■
        11111111              ■■■■■■■■
        11111111              ■■■■■■■■
        11111111              ■■■■■■■■
        11111111              ■■■■■■■■
```

Рис. 24.

Если мы знаем, что символ является графическим блоком, то мы можем использовать два вида данных, четырежды используя которые мы создаем блок. Пример того, как это делается, дается на рис. 24. Графические блоки можно использовать для получения графического изображения средней разрешающей способности с графическим полем 64x44 блоков. Используя такую технику, можно очень быстро создавать сложные двухцветные картинки. Изображение, приведенное на рис. 25, создано с помощью программы на листинге 5.2. Программа рисует последовательность небольших концентрических окружностей в левом верхнем углу, а затем создает крупное изображение "укрупненными" элементами изображения, вызывая для этого специальную подпрограмму. Для этой подпрограммы требуется определить символьный блок, который будет левым верхним углом нашей начальной позиции. Информация с поля пять блоков вниз от этого символа и девять блоков вправо записывается подпрограммой из дисплейного файла в массив S. Для каждой пары линий - эта подпрограмма собирает восемь информационных частей и преобразует их в бинарные строчки (см. строку 1130 и далее). Эти строки используются для построения двоичного представления кода графического символа: из строк выбираются двухбитовые сегменты, перед которыми вставляется значение кода (см. строку 1230).

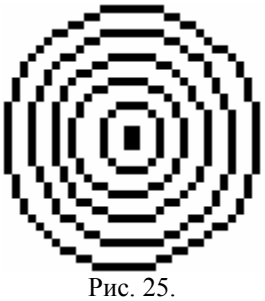


Рис. 25.

LISTING 5.2

```
100 REM MAIN PROGRAM
110 LET BIG PIXELS=1000
```

```
120 FOR I=1 TO 18 STEP 3: CIRCLE 32,156,I: NEXT I
130 GO SUB BIG PIXELS
140 STOP
1000 REM BIG PIXELS
1010 DIM P(8):DIM S(40,8)
1020 FOR I=1 TO 8: READ P(I): NEXT I: DATA 128,64,32,16,8,4,2,1
1030 DEF FN S(R,C)=16384+INT(R/8)*2048+(R-INT(R/8)*8)*32+C
1040 INPUT "TOP LEFT CORNER ","BLOCK IS AT ROW.COL ";ROW;",";COL
1050 FOR I=0 TO 4
1060 FOR J=0 TO 7
1070 LET P=FN S(ROW+I,COL)
1080 FOR K=0 TO 7
1090 LET S(I*8+J+1,K+1)=PEEK (P+256*J+K)
1100 NEXT K: NEXT J: NEXT I
1109 REM SET SCREEN TO 20*32 WHICH IS FOUR TIMES AREA TO BE EXPANDET.
1110 BORDER 1: PAPER 7: INK 0: CLS
1120 PRINT AT 21,0; PAPER 1;,,: PRINT AT 0,0; PAPER 1;,,
1129 REM TAKE TWO LINES OF PIXELS AT A TIME FOR TRANSLATION TO QUARTER BLOCKS.
1130 FOR I=1 TO 39 STEP 2
1140 LET A=I: LET B=I+1
1150 FOR J=1 TO 8
1160 LET A$="00000000": LET B$=A$
1170 LET T=S(A,J): LET U=S(B,J)
1179 REM CALCULATE BINARY FORMS OF PIXELS FOR THE BLOCK ACROSS.
1180 FOR K=1 TO 8
1190 IF T>=P(K) THEN LET T=T-P(K): LET A$(K)="1"
1200 IF U>=P(K) THEN LET U=U-P(K): LET B$(K)="1"
1210 NEXT K
1219 REM TAKE TWO PIXELS FROM EACH OF THE PAIR OF BYTES TO MAKE FOUR QUARTERS.
1220 FOR K=1 TO 4: LET D=2*K: LET C=D-1
1230 LET C$="BIN 1000"+B$(C TO D)+A$(C TO D)
1239 REM CONVERT BINARY FORM OF QUATER BLOCKS TO CHARACTER.
1240 LET C=VAL C$: PRINT CHR$ C;
1250 NEXT K: NEXT J: NEXT I
1260 RETURN
```

Графические символы, сгенерированные предыдущей программой, высвечиваются на экране и создают точную копию содержимого графического поля 5*8. Это изображение получается четырехкратным растяжением по каждой оси, так что теперь оно занимает 20 блоков на 32 блока, и каждому элементу изображения соответствует квадрат четыре элемента изображения на четыре.

Используя команду COPY и листинги принтера X, четырехкратное увеличение может быть достигнуто 16 - кратным использованием подпрограммы "BIGPIXELS", а затем склеиванием отдельных частей изображения.

Упражнение 5.2

Напишите программу, которая печатает саму себя, копируя каждое из полей 5x8 с помощью подпрограммы "BIG PIXELS".

Графика, определяемая пользователем

Третий набор символов - это символы, определяемые пользователем (номера кодов от 144 до 164). Еще одна таблица для этого набора хранится в последних 168 ячейках в конце памяти. Когда компьютер включают, то 21 символ от "A" до "U" копируется в эту таблицу. В графическом режиме эти символы печатаются с клавиатуры. Информация в ячейках этой таблицы может быть изменена пользователем, так что необходимый символ можно вводить прямо с клавиатуры. Для этой цели имеется встроенная функция USR, которая при введении символа (от "A" до "U") выдает адрес первой ячейки в которой хранится информация об эквиваленте этого символа. Адрес начала этой таблицы хранится в системной переменной UDG. Код символа минус 144 и умноженный на восемь даст относительное положение первой из восьми ячеек, относящихся к символу в этой таблице. Функция USR вычисляет эту позицию и прибавляет ее к UDG для получения абсолютного адреса этого символа в памяти. UDG можно модифицировать так, что станет доступным меньшее количество символов, например: только символы от A до H, что оставляет места в памяти для программ, хотя в этом возникает необходимость только тогда, если у вас есть 16 килобайт памяти ОЗУ для того, чтобы познакомиться с использованием этой функции для изменения изображения символа, введите следующую команду:

```
POKE USR "A", BIN 00100100
```

Теперь, если нажать клавишу "A" в графическом режиме, вы увидите над ней две точки.

Программа на листинге 5.3 позволяет вам переопределить все восемь двоичных чисел, представляющих символ. После этого программа создает графический режим, что позволяет печатать определяемые пользователем символы на экране. Для выхода из программы используется символ "Z".

LISTING 5.3

```
10 INPUT "CHARACTER TO BE","REDEFINED ";U$: IF U$="" THEN GO TO 10
15 IF U$=CHR$ 90 THEN GO TO 210
```

```
20 IF U$<"A" OR U$>"U" THEN GO TO 10
30 LET MEMORY=USR U$: CLS
40 FOR I=0 TO 7
50 INPUT ("VALUE"+STR$ I+"=BIN ");LINE B$
60 LET VALUE=VAL ("BIN "+B$)
70 POKE MEMORY+I,VALUE
80 IF LEN B$<8 THEN LET B$="0"+B$: GO TO 80
90 PRINT B$: NEXT I
100 PRINT AT 10,0;"TYPE CHARACTER OR ENTER"
109 REM SIMULATE GRAPHICS MODE FOR LETTERS A TO U.
110 LET R=12:LET C=0
120 PRINT AT R,C; FLASH 1;"G"
130 IF INKEY$ <>"" THEN GO TO 130
140 LET A$=INKEY$ : IF A$="" THEN GO TO 140
149 REM IF "ENTER" IS PRESSED THEN RESTART PROGRAM.
150 IF A$=CHR$ 13 THEN GO TO 10
160 IF A$<"A" OR A$>"U" THEN GO TO 130
169 REM CONVERT CHARACTER TO EQUIVALENT USER-DEFINED GRAPHIC AND PRINT.
170 LET A$=CHR$ (CODE A$+79)
180 PRINT AT R,C;A$
189 REM MOVE TAKE CURSOR POINTERS.
190 LET C=C+1: IF C=32 THEN LET C=0: LET R=R+1: IF R=22 THEN LET R=12
200 GO TO 120
210 STOP
```

Упражнение 5.3

Используйте программу на листинге 5.3 для генерации графического символа, у которого однотонные участки заменяются на участки с орнаментом в клеточку, как на шахматной доске. Используйте различные цвета PAPER (ФОН) и INK (РИСУЮЩИЙ ЦВЕТ), например, сочетание красного и желтого дадут оранжевый.

Определяемая пользователем графика может непосредственно использоваться в ваших программах для ускорения построения изображения. Тот же символ может быть создан последовательностью команд PLOT и DRAW, но в большинстве случаев это требует больше времени. Например, с помощью символов можно создавать орнаменты, что занимает значительно меньше времени, чем при использовании PLOT и DRAW для создания такого же орнамента. На листинге 5.4 приведен текст программы, создающей орнамент, используя определенные пользователем символы от "A" до "D", которые набираются в символьных константах (строки 240 и 250) в графическом режиме. После переопределения на экране дисплея будут появляться новые символы, а не соответствующие им символы клавиатуры.

LISTING 5.4

```
200 REM ORNAMENT
209 REM FILL SCREEN WITH A DIAGONAL PATTERN OF FIRST FOUR GRAPHICS CHARACTER.
210 OVER 0
220 INPUT "INK=";I: INK I
230 INPUT "PAPER=';;I: PAPER I
240 LET A$="ABCD": LET B$="BCDA"
250 LET C$="CDAB": LET D$="DABC"
260 FOR J=0 TO 7
270 FOR I=0 TO 4
280 PRINT AT I*4,J*4;A$
290 PRINT AT 1*4+1,J*4;B$
300 PRINT AT I*4+2,J*4;C$
310 PRINT AT I*4+3,J*4;D$
320 NEXT I
330 PRINT AT I*4,J*4;A$
340 PRINT AT 1*4+1,J*4;B$
350 NEXT J
360 STOP
```

Альтернативные множества символов

До сих пор мы использовали 37 графических символов для создания рисунков, но, используя альтернативные стандартные множества, их число можно довести до 96. Для того, чтобы использовать альтернативные множества, мы просто изменяем значение, хранящееся в CHAR, так, что это значение указывает на новое множество, хранящееся в запоминающем устройстве прямой адресации, вместо обычной таблицы.

Для этого нам нужна программа, которая бы упрощала эту задачу, а также позволяла бы изменять символы после их генерации.

LISTING 5.5

```
1 CLEAR 62294:INK 0: PAPER 7: BORDER 7: OVER 1 :LET N=0: REM FOR 16K MACHINES
USE CLEAR 255+THE VALUE OF S(I) WHERE I IS THE INDEX OF THE LOWEST CHARACTER SET
WHICH WILL FIT IN THE AVAILABLE STORE.
2 DIM Z$(8,8):DIM T$(8,8): DIM S(6): DIM P(8): DIM B(8): DIM G(7): REM LINE
```

```
NUMBERS TO GO FOR MENU OPTIONS.
      3 FOR I=1 TO 7: READ G(I): NEXT I: DATA 61,67,71,100,106,112,119: REM POWERS OF
2 READY FOR USE IN BINARY CONVERSIONS.
      4 FOR I=1 TO 8: READ P(I): NEXT I: DATA 128,64,32,16,8,4,2,1: REM VALUES WHICH
MUST BE PLACED IN CHARS TO USE EACH ALTERNATIVE SET. FOR 16K MACHINES THESE NUMBERS
SHOULD BE: 15360,29271,30039,30807,31575,32080, (SEE APPENDIX).
      5 FOR I=1 TO 6: READ S(I): NEXT I: DATA 15360,62039,62807,63575,64343,64848
      6 LET S=1: GO SUB 17: REM CHECK SET 1 IS WEING USED AND PRINT MENU.
      7 CLS : PRINT AT 2,2;"***CHARACTER GENERATOR***"
      8 PRINT AT 5,6;PAPER 5;"1";PAPER 7;"...PRINT ALL SETS"
      9 PRINT AT 7,6;PAPER 5;"2";PAPER 7;"...PRINT ONE SET"
      10 PRINT AT 9,6;PAPER 5;"3";PAPER 7;"...EDIT CHARACTER"
      11 PRINT AT 11,6;PAPER 5;"4";PAPER 7;"...COPY SET TO SET"
      12 PRINT AT 13,6;PAPER 5;"5";PAPER 7;"...SAVE SET"
      13 PRINT AT 15,6;PAPER 5;"6"; PAPER 7;"...LOAD SET"
      14 PRINT AT 17,6;PAPER 5;"7";PAPER 7;"...RUN YOUR PROGRAM": REM WAIT FOR VALID
OPTION.
      15 INPUT "WHICH OPTION ";OP: IF OP<1 OR OP>7 THEN GO TO 15: REM JUMP TO
APPROPRIATE ROUTINE.
      16 GO TO G(OP): REM AOST SUBROUTINES ARE LOCATED AT START OF PROGRAM FOR
EFFICIENCY. ROUTINE TO CHANGE TO SET S BY ALTERING CHARS TO POINT TO NEW TABLE OF
DATA.
      17 LET HI=INT (S(S)/256): LET LO=S(S)-HI*256
      18 POKE 23606,LO: POKE 23607,HI: RETURN: REM GENERAL ROUTINE TO WAIT BEFORE
CLEARING SCREEN AND RETURNING TO MENU.
      19 LET S=1: GO SUB 17: INPUT "PRESS ENTER TO","RETURN TO MENU";LINE A$: RETURN
: REM CLEAR BINARY CHARACTERS ARRAY.
      20 FOR I=1 TO 8: LET Z$(I)="00000000": NEXT I: RETURN: REM INPUT A VALID SET
NUMBER AND GIVE UPPER AND LOWER BOUNDS FOR CHARACTERS
      21 INPUT "WHICH SET";S: IF S<1 OR S>6 THEN GO TO 21
      22 LET A=32: LET B=127:IF S=6 THEN LET A=65: LET B=85
      23 RETURN: REM INPUT A VALID CHARACTER FOR CHOSEN SET.
      24 INPUT "WHICH CHARACTER";C$: LET C=CODE C$: IF C<A OR C>B THEN GO TO 24
      25 RETURN: REM DRAW EIGHT BY EIGHT BLOCK GRID AT HORIZONTAL POSITION ALTERED BY
N.
      26 LET NN=64+N: FOR I=0 TO 8
      27 PLOT I*8+NN,64: DRAW 0,64
      28 PLOT NN,I*8+64: DRAW 64,0
      29 NEXT I: RETURN: REM PRODUCE GRAPHICAL EQUIVALENT OF BINARY STRING ARRAY
INSIDE GRIB.
      30 FOR I=1 TO 8: FOR J=1 TO 8
      31 LET P=7: IF N=0 AND Z$(I,J)="1" OR N=96 AND T$(I,J)="1" THEN LET P=4
      32 GO SUB 48: NEXT J: NEXT I: RETURN
```

Подпрограммы программы CHARACTER GENERATOR (ГЕНЕРАТОР СИМВОЛОВ) независимы друг от друга и предназначены для выполнения подготовки к использованию и использования определяемых пользователем символов: они позволяют вам редактировать и управлять символами, использовать альтернативные множества символов и графические символы, сохранять и перезагружать определяемые символы и мгновенно проверять ваши собственные программы. Программы написаны для SRECTRUMa с 48К памяти и располагаются в нижней ее части (занимая строки с 1-ой по 118-ую), затем идет ваша собственная программа, начинающаяся со строки за 118-ой строкой, за которой следуют подпрограммы перенумерации и удаления (см. главу 13), которые начинаются с 9900-ой строки. Если у вас 16К памяти, то эта подпрограмма должна быть независимой от других подпрограмм - см. приложение А.

Подпрограмма вначале предлагает вам семь возможных опций, к рассмотрению которых мы и приступим. (Сразу оговоримся, что в листинге 5.5 не приведены строки 33 - 118 подпрограмм обработки опций, поэтому программа в приведенном виде лишь выводит на экран меню опций, не обрабатывая их).

1. Первая опция - PRINT ALL SETS (НАПЕЧАТАЙТЕ ВСЕ НАБОРЫ). В этом режиме сначала напечатаются нормальные символы (1-ый набор), за которыми следует четыре альтернативных набора символов (со 2-го по 5-ый), затем символы, определяемые пользователем (шестой набор). В нескольких последних символах 5-го набора будет содержаться бессмысленная информация, остаток стека GO SUB, вытесненный в безопасное место.

2. Вторая опция - PRINT ONE SET - (НАПЕЧАТАЙТЕ ОДИН НАБОР). Эти наборы пронумерованы числами от 1 до 6 следующим образом: 1 набор стандартных символов, не может быть изменен; наборы со второго по пятый - альтернативные, могут на время заменить стандартный набор; 6 - набор символов, определяемых пользователем, доступен всегда. После выбора какого-нибудь из этих чисел на экране появятся символы стандартного набора, после каждого из которых будет стоять знак -; как символ, так и этот знак печатаются на желтом фоне, за которыми следует соответствующий символ альтернативного набора. Для 6-го набора будут напечатаны только значения символов от "A" до "U".

3. Третья опция - РЕДАКТИРОВАНИЕ. Это самая сложная опция с большим числом команд, которые вводятся набором первого символа команды. Во-первых, вам придется ответить, с какого набора и с какого символа вы хотите начать. Если вы хотите начать с незаполненной таблицы 8 пикселей на 8, то вам следует ввести 1 и символ пробела (если вы хотите использовать апостроф, то его следует набирать дважды.

Теперь вы находитесь в режиме редактирования, и символ, который вы указали, будет воспроизведен в зеленых квадратах размером восемь пикселей на восемь. Крест в левом верхнем углу - это курсор, который управляется стандартными курсорными клавишами ("5", "6", "7", "8") с или без клавиши SHIFT.

Первые команды - PLOT (НАНЕСТИ) и OFF (ВЫКЛ). Они изменяют один квадрат в сетке символов. Для этого подведите курсор к квадрату и введите "P" или "O".

"P" делает квадрат зеленым, что эквивалентно двоичному "ВКЛ", а "O" стирает изображение в квадрате, делая его белым.

Следующие три команды определяют преобразования, аналогичные описанным в главе 4. ROTATE поворачивает символ на 90 градусов против часовой стрелки относительно центра квадрата. Символ отражается относительно координатных осей по команде X-AXIS (ось X) или Y-AXIS (ось Y). Эти команды используются для создания текстов, расположенных на экране произвольным образом.

И, наконец, команды MERGE, SAVE, DOUBLE.

Команда MERGE (ПОМЕСТИТЬ) позволяет поместить любой символ в сетку, поверх того, что уже было отредактировано. Это очень удобная команда для создания набора букв иностранного алфавита: например, для того, чтобы поместить черточку через O для скандинавских языков, добавлять акценты для французского и т.д.

Команда SAVE (СОХРАНИТЬ) запрашивает, в каком наборе мы желаем сохранить вновь созданный символ, и какому стандартному символу он соответствует. Если вводится набор 1, то символ будет уничтожен, так как первый набор хранится в ПЗУ; эта операция может оказаться полезной при уничтожении ненужных сеток.

DOUBLE - это мощное вспомогательное средство, позволяющее увеличить вдвое любую из четвертей символа, который вы редактируете, и сохранить это изображение в памяти командой SAVE, после чего к нему можно обращаться как к новому символу. Эта команда не меняет символа, который редактируется, так что все четыре четвертинки символа могут быть записаны в память, что можно использовать для увеличения символов.

4. COPY SET TO SET (КОПИРУЙТЕ НАБОР В НАБОР) - эта опция используется для копирования наборов целиком для облегчения выполнения действий с этим набором; эта операция сводится к перемещению в памяти одного из альтернативных наборов. Наборы со 2-го по 5-ый хранятся в конце памяти в ячейках, так что конец 5-го набора непосредственно предшествует графике, определяемой пользователем. В 48-килобайтном варианте SRECTRUMa область памяти, отведенная для наборов 2 - 5, охраняется от бейсика командой CLEAR 62294. Число 62294 на единицу меньше номера первой ячейки набора 2. Чтобы защитить только области для наборов 1, нам нужно вычислить S(1)+255 и поместить это значение в команду CLEAR в начале программы. S(l) - это значение, присваиваемое системной переменной CHARS, для того, чтобы команда PRINT могла использовать 1-ый набор. Эти значения вводятся командой READ оператора DATA третьей строки программы. Обратите внимание на изменения, которые следует внести, если у вас 16K памяти (см. приложение A).

5 и 6. Пятая опция позволяет записать набор на ленту, а шестая - загрузить его обратно в память. Если на вопрос "WHICH SET?" (КАКОЙ НАБОР?) мы отвечаем числом от 1 до 6, то этот набор записывается или перезагружается. Если напечатать 5 6, то 5 и 6 наборы будут записаны или перезагружены в одном блоке.

Для того, чтобы другие программы могли использовать альтернативные наборы символов, следует скопировать строки 112-117 программы "CHARACTER GENERATOR" вместе с подпрограммой (на 17 и 18), позволяющей переключаться с одного набора на другой; также следует включить множество и соответствующий ему оператор DATA на 5-ой строке.

7. Последняя опция - RUN YOUR PROGRAM (ЗАПУСТИТЬ ВАШУ ПРОГРАММУ), которая передает управление первой строчке, следующей за программой, генерирующей символы, или, если за этой программой ничего не следует, то останавливает программу.

Для того чтобы освоиться с программой "CHARACTER GENERATOR", следуйте инструкциям, приведенным ниже.

Используя MERGE, введите листинги 5.4 и 5.5 на 48K-байтном SPECTRUMe; на 16K-байтном SPECTRUMe пропускайте их по отдельности. Нарисуйте что-нибудь вроде кляксы и сохраните это изображение под символом "A" шестого набора. Отредактируйте символ: поверните его и сохраните под символом "B" шестого набора. Примените команду X-AXIS к символу "A" шестого набора и сохраните результат под символом "C" шестого набора. Теперь используйте 7-ую опцию для того, чтобы посмотреть, что у вас получилось.

Упражнение 5.4

Поэкспериментируйте с различными видами симметрии символов и орнаментов для размещения символов на экране. Измените программу так, чтобы она перебирала, с помощью двух вложенных циклов FOR…NEXT все возможные сочетания цветов для INK и PAPER.

Упражнение 5.5

Теперь создайте символы костяшек для игры в домино, используя программу "CHARACTER GENERATOR".

Упражнение 5.6

Напишите подпрограмму, копирующую один набор символов в другой, используя при этом команды режима редактирования для модификации символов во время этой перезаписи.

В следующей главе вы познакомитесь с тем, как можно использовать символьную графику и накопленные нами сведения о двумерной геометрии для наглядного представления информации.

Список программ

1. Листинг 5.1. Входные данные A$. A$ - любой символ не графического режима, попробуйте "X" или "*".

2. Листинг 5.2. ("MAIN PROGRAM" и "BIG PIXELS"). Входные данные ROW и COL, где 0<=ROW<=COL<=24. Попробуйте (0,0) и (1,1).

3. Листинг 5.3. Входные данные: U$ и восемь двоичных чисел, которые задают изображение, соответствующее символу U$. Попробуйте "A", и III, III000, I0I0II, I0I, IIII00I, I00I, IIII, I0II0, и "B", "C", "D", каждую вместе с перестановкой этих восьми чисел. Затем нажмите клавиши от "A" до "U" для того, чтобы посмотреть, что получилось, или клавишу "ENTER" для рестарта программы.

4. Листинг 5.4: никаких данных. Программа 5.3 должна использоваться для создания символов "A", "B", "C", "D".

5. Листинг5.5: CHARACTER GENERATOR (ГЕНЕРАТОР СИМВОЛОВ). Прочитайте описание и пример использования.

ГЛАВА 6
Диаграммы и графики

Компьютеры могут помочь нам разобраться в потоке информации, который обрушивается на нас каждый день; компьютеры позволяют придать этой информации более наглядный и более простой для восприятия вид, используя для этой цели круговые диаграммы, гистограммы, графики и т.д. С тех пор, как появились компьютеры, которые можно устанавливать на рабочем столе, спрос на программы, придающие информации наглядный вид, сделал эго направление одним из самых быстроразвивающихся в области машинной графики. В этой главе мы увидим, как можно создавать такие программы с помощью небольшого числа вспомогательных средств.

Курсоры

Предстаете себе, что экран перед вами - это лист бумаги, на который мы будем наносить изображение объектов. Нам необходим метод, при помощи которого мы могли бы аккуратно управлять положениями этих предметов. Обычно для этих целей используется курсор, движением которого можно управлять с помощью джойстика или аналогичного внешнего устройства, однако мы будем управлять движением курсора с клавиатуры. Это не так удобно, как, например, джойстиком, но зато не требует затрат периферийных устройств и в конечном итоге дает тот же эффект, что и использование таковых.

Подпрограмма "CURSOR" (курсор) на листинге 6.1 выводит на экран две пересекающиеся прямые, нарисованные прозрачными чернилами (INK цвет 8), перекрестие которых выделяет символьный блок. Курсор передвигается под действием любой из восьми клавиш вокруг клавиши "F". Положение этих клавиш относительно "F" определяет направление движения. Если в вашем распоряжении имеется джойстик, то переделайте подпрограмму "CURSOR" с тем, чтобы движением курсора можно было управлять джойстиком, а не с клавиатуры.

Нажатием "F" вы помещаете курсор в центр экрана. Если держать клавишу нажатой, то скорость движения курсора постепенно возрастает: с курсором, который перемещается на один элемент изображения при каждом нажатии клавиши, чрезвычайно утомительно работать! Для облегчения установки курсора имеется сетка (LATTICE), которая зажигается нажатием буквы "L" и гасится повторным нажатием "L". Она автоматически исчезает, когда вы нажимаете клавишу ENTER для ввода другой точки.

Клавиши управления курсором для программы "CURSOR"

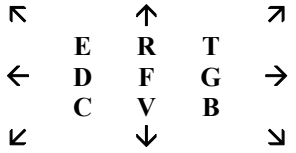


Рис. 26.

LISTING 6.1

```
5700 REM CURSOR
5701 REM IN - PX,PY
5702 REM OUT - PX,PY,ROW,COL
5709 REM THE NEXT LINE IS ONLY EXECUTED ON THE FIRST CALL TO CURSOR.
5710 LET PX=128: LET PY=88: LET CURSOR=5720: LET GRID=5900
5719 REM START OF MAIN CURSOR ROUTINE.
5720 INK 8: LET A=1:LET FLAG=1: OVER 1
5729 REM WAIT FOR KEYBOARD TO BE CLEAR BEFORE LOOKING FOR COMMANDS.
5730 IF INKEY$<>"" THEN GO TO 5730
5740 PLOT PX,0: DRAW 0,175: PLOT 0,PY: DRAW 255,0
5750 LET A$=INKEY$
5760 IF A$="" THEN LET A=1: GO TO 5750
5770 PLOT PX,0: DRAW 0,175: PLOT 0,PY: DRAW 255,0
5780 IF (A$="E" OR A$="D" OR A$="C") AND PX>=A THEN LET PX=PX-A
5790 IF (A$="C" OR A$="V" OR A$="B") AND PY>=A THEN LET PY=PY-A
5800 IF (A$="T" OR A$="G" OR A$="B") AND PX<=255-A THEN LET PX=PX+A
5810 IF (A$="E" OR A$="R" OR A$="T") AND PY<=175-A THEN LET PY=PY+A
5820 IF A$="F" THEN LET PX=128: LET PY=88
5830 IF A$="L" THEN GO SUB GRID
5840 IF A$<>CHR$ 13 THEN LET A=A+1: GO TO 5740
5850 LET COL=INT(PX/8): LET ROW=INT((175-PY)/8)
5860 IF FLAG=-1 THEN GO SUB GRID
5870 INK 0: OVER 0
5880 RETURN
5900 REM GRID
5909 REM ROUTINE CHANGES FLAG TO REMEMBER WHETHER GRID IS SHOWING ON SCREEN.
5910 FOR J=0 TO 255 STEP 8: PLOT J,0: DRAW 0,175: IF J<176 THEN PLOT 0,J: DRAW
255,0
5920 NEXT J: LET FLAG=-FLAG: PLOT 255,0: DRAW -255,0
5930 RETURN
```

Упражнение 6.1

Напишите основную программу "MAIN PROGRAM", которая вызывает программу "CURSOR", а затем печатает цветной квадрат на экране в указанном месте. Программа должна также печатать номер столбца и строки, в которых находится курсор. Измените подпрограмму "CURSOR" так, чтобы стандартными клавишами курсора ("5", "6", "7", "8") можно было бы задавать перемещение курсора на один символьный блок.

Атрибуты

Можно развить идеи предыдущего пункта и написать подпрограммы, изменяющие атрибуты заданного блока или группы блоков без изменения информации, которая им соответствует в дисплейном файле. На листинге 6.2 показаны две подпрограммы, которые делают это для цвета PAPER и INK определенного множества блоков

LISTING 6.2

```
4000 REM PAPER
4010 GO SUB CURSOR: INPUT "WHAT COLOUR ";P: INPUT "NO.OF BLOCKS "; "(ROW*COL)
";R;"*";C
4020 FOR I=ROW TO ROW+R-1: FOR J=COL TO COL+C-1
4030 PRINT AT I,J;PAPER P;INK 8;OVER 1;" ": NEXT J: NEXT I
4040 RETURN
4100 REM INK
4110 GO SUB CURSOR: INPUT "WHAT COLOUR ";P: INPUT "NO.OF BLOCKS ";"(ROW*COL)
";R;"*";C
4120 FOR I=ROW TO ROW+R-1: FOR J=COL TO COL+C-1
4130 PRINT AT I,J;PAPER 8;INK P;OVER 1;" ": NEXT J: NEXT I
4140 RETURN
```

Упражнение 6.2

Напишите "MAIN PROGRAM", которая вводит часть своего текста, а затем позволяет вам выделять участки распечатки, используя различные цвета INK и PAPER. Напишите подпрограммы, которые бы размещались в строках от 4050 до 4150 и изменяли бы атрибуты FLASH и BRIGHT блока (почему это сведется к добавлению команд BRIGHT 8 и FLASH 8 к подпрограммам "INK" и "PAPER"?). Составьте из четырех подпрограмм одну под названием "ATRIBUTE".

Точки и линии

Достигнув простейшего интерактивного управления видом нашей диаграммы, нам предстоит заполнить детали этой диаграммы. Для этого требуются подпрограммы для нанесения точек (PLOT) и проведения прямых (DRAW). Мы могли бы использовать программу на листинге 1.8, но наносить все точки на прямой слишком утомительно. Вместо этого мы используем две подпрограммы - "POINT", которая наносит отдельные точки, и "LINE", для которой мы просто определяем две точки прежде, чем их соединить. Обе подпрограммы приведены на листинге 6.3.

LISTING 6.3

```
4200 REM POINT
4210 INPUT "PRESS ENTER FOR CURSOR";LINE A$
4220 GO SUB CURSOR: INPUT "OVER (1 OR 0) ";O: INPUT "COLOUR ";C
4230 PLOT INK C;OVER O;PX,PY
4240 RETURN
4300 REM LINE
4310 INPUT "PRESS ENTER FOR CURSOR ";LINE A$
4320 GO SUB CURSOR: LET SX=PX: LET SY=PY
4330 INPUT "PRESS ENTER FOR CURSOR ";LINE A$
4340 GO SUB CURSOR: INPUT "OVER (1 OR 0) ";O: INPUT "COLOUR ";C,
4350 PLOT PAPER 8;INK C; OVER O;SX,SY
4360 DRAW PAPER 8;INK 0; OVER 0;PX-SX,PY-SY
4370 RETURN
```

Упражнение 6.3

Напишите программу "MAIN PROGRAM", которая с помощью подпрограмм "POINT" и "LINE" помогла бы вам сделать набросок вашего SPECTRUMa. Составьте подпрограмму, которая по параметрам, вводимым с помощью курсора, чертила бы круг; используйте подпрограмму "CIRCLE1" или "CIRCLE2", приведенные на листинге 2.10. Переделайте вашу программу из упражнения 1.3, которая чертит многоугольник так, чтобы вершины можно было задавать курсором. Добавьте опцию к подпрограмме "LINE" с тем, чтобы она могла строить кривые линии. Вам будет удобнее использовать восемь точек вокруг стороны, а не двенадцать.

Если у вас есть графическая прокладка (PAD), то вы можете перенести, используя модификации "LINE" и "POINT", грубые наброски с этой прокладки в машину. После этого вы должны написать программы, приводящие изображение в порядок, т.е. выпрямляющее прямые и сглаживающие кривые.

Запись и загрузка

Потратив много времени на создание той или иной картинки, возможно, вам захочется ее сохранить. Это достигается использованием программы "SAVE". С другой стороны, может возникнуть необходимость в загрузке картинки в ходе выполнения какой-либо программы, а затем в изменении ее; для этой цели следует использовать подпрограмму "LOAD". "LOAD" и "SAVE" приведены на листинге 6.4.

LISTING 6.4

```
4500 REM SAVE
```

```
4510 INPUT "NAME OF PICTURE? ";N$: IF N$="" THEN RETURN
4520 SAVE (N$)SCREEN$
4530 RETURN
4600 REM LOAD
4610 INPUT "NAME OF PICTURE? ";N$: IF N$="" THEN RETURN
4620 LOAD (N$)SCREEN$
4630 RETURN
```

Упражнение 6.4

Запустите программу на листинге 3.1 (или загрузите с помощью подпрограммы "LOAD", если вы его сохранили), а затем измените цвет INK на зеленый, а PAPER на черный. Для экономии места соедините "SAVE" и "LOAD" в одной подпрограмме.

Надписи

Программы на листингах 6.1-6.4 образуют основу пакета программ для создания графиков, но пока нам не хватает подпрограмм, позволяющих делать надписи. Такая подпрограмма приведена на листинге 6.5. Обратите внимание, что она содержит обращения к еще не написанным программам. Это типичный пример принятого в книге модульного метода создания программ. Это означает, что когда мы сталкиваемся с задачей, которую не можем решить с ходу, то мы присваиваем ей имя с тем, чтобы вернуться к ней позднее. В подпрограмме "LABEL" мы полагаем, что символы из пятого набора будут печататься в последовательности один над другим. По причинам, которые станут, понятны позже, определим специальный префикс (7:), который указывает, что цепочка символов для надписи должна быть напечатана в узких символах, используя подпрограмму "THIN". Мы предполагаем также наличие подпрограммы "SET", которая позволяет переключать наборы Символов.

LISTING 6.5

```
5400 REM LABEL
5410 GO SUB CURSOR: INPUT "SET:STRING ";A$: IF A$="" THEN GO TO 5410
5420 INPUT "COLOUR ";C$: INK VAL ("0"+C$)
5430 IF A$(TO 2)="1:" THEN PRINT AT ROW,COL;A$(3 TO): GO TO 5510
5439 REM CHECK FOR SPECIAL SET NUMBERS, SET 5 IS PRINTED VERTICALLY.
5440 IF A$(TO 2)<>"5:" THEN GO TO 5470
5450 LET S=5: GO SUB SET: LET A$=A$(3 TO)
5460 FOR I=1 TO LEN A$: PRINT AT ROW-I+1,COL;A$(I): NEXT I: GO TO 5510
5469 REM SET 7 SPECIFIES THAT THE THIN ROUTINE SHOULD BE USED FOR PRINTING.
5470 IF A$(TO 2)<>"7:" THEN GO TO 5490
5480 LET A$=A$(3 TO): GO SUB THIN: GO TO 5510
5489 REM CHECK VALID NUMBER IS GIVEN.
5490 LET S=VAL A$(1): IF S>6 OR S<1 OR A$(2)<>":" THEN GO TO 5510
5499 REM PRINT OUT STRING IN APPROPRIATE SET.
5500 GO SUB SET: PRINT AT ROW,COL;A$(3 TO)
5510 LET S=1: GO SUB SET
5520 INK 0: RETURN
```

Данные для графиков

Для окончательной отделки нашей программы, мы вводим программу "MAIN PROGRAM" и подпрограмму "QUERY" ("запрос"), текст которых приведен на листинге 6.6, хотя эти программы и ставят перед нами новые нерешенные задачи, но все же они полезны тем, что позволяют нам проследить, какие именно задачи остались нерешенными. Мы вводим идентификаторы (нижний регистр) для имен подпрограмм, которые нам еще предстоит написать, которые, кроме всего прочего, позволят нам создавать специальные виды диаграмм, например, гистограммы, круговые диаграммы и графики. Часть листинга 6.6, до "MAIN PROGRAM", - это подчищенная версия программы CHARACTER GENERATOR, которая необходима для использования альтернативных множеств.

Все диаграммы в этой книге, за исключением тех, которые являются непосредственным результатом работы программ-образцов, были созданы программой, содержащей подпрограммы на листингах 6.1-6.7; кроме того, эта программа использовалась для добавления надписей ко многим из рисунков, созданных программами.

LISTING 6.6

```
8 REM FOR 16K MAKE CHANGEN AS GIVEN IN LISTING 5.5.
9 REM SET UP TO USE ALTERNATE CHARACTER SETS.
10 CLEAR 62294
20 INK 0: PAPER 7: CLS : DIM S(6)
30 FOR I=1 TO 6: READ S(I): NEXT I: DATA 15360,62039,62807,63575,64343,64848
40 LET SET=50: LET S=1: GO SUB SET: GO TO 200
50 REM SET/CHANGE TO SET S
60 LET HI=INT(S(S)/256): LET LO=S(S)-HI*256
70 POKE 23606,LO: POKE 23607,HI: RETURN
80 REM CHARLOAD
90 INPUT "WHAT FILE NAME? ";N$: IF N$="" THEN RETURN
100 INPUT "LOAD WHICH SET? ";S: IF (S<2 OR S>6) AND S<>11 THEN GO TO 80
110 INPUT "PRESS ENTER ";LINE X$
120 IF S=11 THEN LOAD N$ CODE (S(5)+256),768+168: RETURN
```

```
      130 IF S=6 THEN LOAD N$ CODE (S(S)+512),168: RETURN
      140 LOAD N$ CODE (S(S)+256),768: RETURN
      200 REM MAIN PROGRAM
      210 LET RESTART=280: LET QUERY=500
      220 LET HISTO=1000: LET PIE=2000: LET GRAPH=3000
      230 LET PAPER=4000: LET INK=4100: LET POINT=4200: LET LINE=4300: LET SAVE=4500:
LET LOAD=4600: LET NUMBER=4700
      240 LET CHARLOAD=80: LET CREATE=5000: LET THIN=5200
      250 LET LABEL=5400: LET CURSOR=5700
      260 LET I$="DEFINE CHARACTERS? ": GO SUB QUERY: IF YES THEN GO SUB CREATE
      270 CLS : LET I$="LOAD CHARACTERS? ": GO SUB QUERY: IF YES THEN GO SUB
CHARLOAD: GO TO 270
      280 LET I$="LOAD PICTURE? ": GO SUB QUERY: IF YES THEN GO SUB LOAD
      290 LET I$="DRAW DIAGRAM? ": GO SUB QUERY: IF NOT YES THEN GO TO 360
      300 LET I$="HIST.,PIE-CHART OR GRAPH? ": GO SUB QUERY
      310 LET DIAGRAM=0: IF A$="H" THEN LET DIAGRAM=HISTO
      320 IF A$="P" THEN LET DIAGRAM=PIE
      330 IF A$="G" THEN LET DIAGRAM=GRAPH
      340 IF DIAGRAM=0 THEN GO TO 280
      350 GO SUB DIAGRAM
      360 LET I$="LABEL PICTURE ?": GO SUB QUERY: IF YES THEN GO SUB LABEL: GO TO 360
      370 LET I$="COLOUR PAPER ? ": GO SUB QUERY: IF YES THEN GO SUB PAPER: GO TO 370
      380 LET I$="COLOUR INK ? ": GO SUB QUERY: IF YES THEN GO SUB INK: GO TO 380
      390 LET I$="DRAW POINT ? ": GO SUB QUERY: IF YES THEN GO SUB POINT: GO TO 390
      400 LET I$="DRAW LINE ? ": GO SUB QUERY: IF YES THEN GO SUB LINE: GO TO 400
      410 LET I$="END PICTURE ? ": GO SUB QUERY: IF NOT YES THEN GO TO RESTART
      420 LET I$="SAVE PICTURE ? ": GO SUB QUERY: IF YES THEN GO SUB SAVE
      430 STOP
      500 REM QUERY
      501 REM IN - I$
      502 REM OUT - A$,YES
      510 LET YES=0
      520 INPUT (I$);LINE A$: IF A$="" THEN RETURN
      530 LET A$=A$(1)
      540 LET YES=(A$="Y")
      550 RETURN
```

Специальные символы

Для завершения подготовительной работы для программы "LABEL" нам необходимо создать ряд символов, получаемых поворотом из стандартных; эти символы используются в вертикальных надписях. Такие символы создаются программой "CREATE" (листинг 6.7), которая использует некоторые технические приемы подпрограммы "CHARACTER GENERATOR" для записи повернутых символов набора 1 в набор 5.

LISTING 6.7

```
      5000 REM CREATE/CHARACTERS
      5009 REM CREATE CHARACTERS FOR HISTOGRAM ROUTINE.
      5010 DIM P(8): DIM D(3): LET D(1)=15: LET D(2)=255: LET D(3)=240
      5020 FOR I=0 TO 6: FOR J=0 TO 7
      5030 LET P(1)=USR "A"+I*8+J: LET P(2)=USR "H"+I*8+J
      5040 LET P(3)=USR "O"+I*8+J
      5050 FOR K=1 TO 3: POKE P(K),0: NEXT K
      5060 IF J<I THEN FOR K=1 TO 3: POKE P(K),D(K): NEXT K
      5070 NEXT J: NEXT I
      5078 REM COPY SET 1 TO 5 WITH A ROTATION TO THE SIDEWAYS CHARACTERS.
      5079 REM FOR 16K MACHINES Q=31831.
      5080 DIM B$(8,8): LET T=256: FOR I=1 TO 8: LET T=T/2: LET P(I)=T: NEXT I: LET
P=15616: LET Q=64599
      5090 FOR I=32 TO 127: FOR J=1 TO 8: LET B$(J)="00000000": NEXT J
      5100 FOR J=1 TO 8: LET T=PEEK P: FOR K=1 TO 8
      5110 IF T>=P(K) THEN LET T=T-P(K): LET B$(9-K,J)="1"
      5120 NEXT K: LET P=P+1: NEXT J
      5130 FOR J=1 TO 8: POKE Q,VAL ("BIN"+B$(J)): LET Q=Q+1: NEXT J
      5140 PRINT AT 10,10;"ROTATING ";CHR$ I
      5150 NEXT I: CLS
      5160 RETURN
      5200 REM THIN
      5201 REM IN - ROW,COL,A$
      5209 REM PRINT EVERY OTHER CHARACTER IN LEFT HALF OF BLOCKS.
```

```
5210 LET S=4: GO SUB SET: OVER 1: LET TC=COL
5220 FOR I=1 TO LEN A$ STEP 2: PRINT AT ROW,TC;A$(I): LET TC=TC+1: NEXT I
5229 REM PRINT OTHER CHARACTERS IN BETWEEN IN RIGHT HALF OF BLOCKS.
5230 LET S=3: GO SUB SET: LET TC=COL
5240 FOR I=2 TO LEN A$ STEP 2: PRINT AT ROW,TC;A$(I): LET TC=TC+1: NEXT I
5250 LET S=1: GO SUB SET: OVER 0
5260 RETURN
```

Перед тем как произвести эту операцию, подпрограмма "CREATE" переопределяет набор символов, определяемых пользователем, так что для каждого символа оказывается закрепленной снизу либо левая часть, либо правая, либо по всей ширине блока. Надпечатывая эти символы командой OVER, можно проводить горизонтальные линии различной толщины, но в основном эти символы предназначены для создания гистограмм.

В оставшейся части главы мы будем обсуждать некоторые распространенные способы представления информации. Из-за большого разнообразия вариантов каждого из трех основных видов графического представления информации, кажется неразумным обсуждать их все. Вместо этого мы детально обсудим несколько представителей трех основных видов графического представления информации гистограмм, круговых диаграмм и графиков. Пользуясь этими простейшими примерами, несложно построить варианты подпрограмм, организующие выдачу информации в нужном вам виде. Еще раз напомним вам, что изменения, которые необходимо внести для машин с объемом памяти 16 К, содержатся в приложении А.

Определенные пользователем символы для использования в гистограммах

A-	**W-**	**C-**	**D-**	**E-**	**F-**	**G-**
H-	**I-**	**J-**	**K-**	**L-**	**M-**	**N-**
O-	**P-**	**Q-**	**R-**	**S-**	**T-**	**U-**

Рис. .27

Гистограммы

Создаваемые нашими подпрограммами гистограммы могут быть любой высоты и цвета, если только соблюдается условие, что ширина столбцов гистограммы и промежуток между ними не меньше половины блока. Так как для гистограмм это условие обычно имеет место, то мы можем сформулировать метод нахождения толщины столбцов и промежутка между ними, если только нам известно количество столбцов. Первая часть программы "HISTO" проводит оси, наносит надписи и запрашивает количество столбцов. Ширина столбцов (X) и величина промежутка между ними (GAP) вычисляются (в половинах блоков) так, что гарантируется, что 1/2<=GAP<=X.

По мере получения информации подпрограмма использует символы шириной в полблока и в блок, а также символы, определяемые пользователем (из "CREATE"), для построения цепочек, высота которых пропорциональна высоте гистограммы. Эти цепочки используются для заполнения символьного блока или половины символьного блока, соответствующего этому столбцу; для этого перед каждой цепочкой ставится префикс "5:" и вызывается подпрограмма "BAR".

Обратите внимание, что оси вынесены за поле гистограммы для того, чтобы они не менялись ни при каких изменениях цветов внутри этого поля. На рис. 28 приведена диаграмма, на которой показано годовое количество осадков в ЭГХАМЕ; эта диаграмма была начерчена с помощью подпрограммы "HISTO" (TYPED на листинге 6.8, а затем надписана с помощью "LABEL".

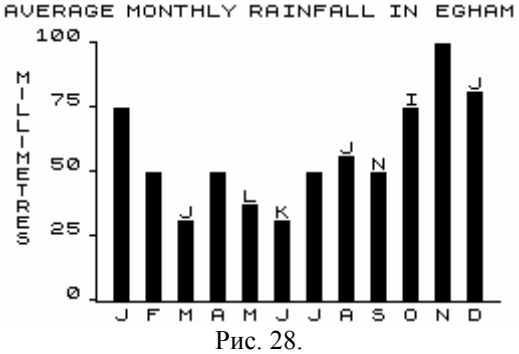


Рис. 28.

Упражнение 6.5

В качестве вариантов стандартной подпрограммы "HISTO" можно писать заменяющиеся подпрограммы, которые можно загружать по мере необходимости. Напишите подпрограмму, вычисляющую позиции пар столбцов, где парные блоки отделяются друг от друга просветом в полблока, а расстояние между соседними парами не менее одного блока. Используйте эту подпрограмму для создания диаграмм, аналогичных диаграмме на рис. 28.

LISTING 6.8

```
1000 REM HISTO/TYPE1
1009 REM SET POINTER TO SUBROUTINE WHICH PRINTS BARS.
1010 LET BAR=1320
1019 REM FIND SCALE AND DRAW AXES.
1020 INPUT "RANGE OF VERTICAL ";YB;" TO ";YT
1030 IF YB>=YT THEN GO TO 1020
1040 LET YSCALE=128/(YT-YB)
1050 PLOT 47,152: DRAW 0,-129: DRAW 201,0
1059 REM LABEL VERTICAL AXIS.
```

```
      1060 LET YDIF=(YT-YB)/4: LET TICK=YB
      1070 FOR I=1 TO 5: LET TK=INT(TICK+0.5)
      1080 LET Y=32*I-8: PLOT 47,Y: DRAW -3,0: LET ROW=INT ((176-Y)/8)-1
      1090 LET A$=STR$ TK: IF LEN A$>3 THEN LET A$=A$(TO 3): IF TK>99 OR TK<-99 THEN
LET A$="***"
      1100 LET COL=1: IF TK>=0 THEN LET COL=COL+1
      1110 IF ABS TK<10 THEN LET COL=COL+1
      1120 IF ABS TK<100 THEN LET COL=COL+1
      1130 PRINT AT ROW,COL;A$: LET TICK=TICK+YDIF
      1140 NEXT I
      1149 REM FIND NUMBER OF BARS REQUIRED AND CALCULATE HOW IHEY CAN FIT IN.
      1150 INPUT "NO. OF BARS=";NB
      1160 LET X=INT(25/NB+0.5)/2: LET GAP=INT(50/NB-2*X)/2
      1170 LET GP=(25-NB*X-(NB-1)*GAP)*2: LET GP=INT((GP+1)/2)/2
      1179 REM POSITION COLUMN POINTER AT FIRST BAR THEN REPEATE LOOP FOR EACH BAR.
      1180 LET COL=6+GP: FOR I=1 TO NB
      1190 LET I$="DATA FOR BAR "+STR$ I+"=": INPUT (I$);D;"  COLOUR=";C: INK C: LET
W=X
      1199 REM VALUES BELOW THE HORIZONTAL AXIS ARE TREATED AS SINGLE PIXEL HEIGHT.
      1200 IF D<=YB THEN LET D=0: LET H=1: GO TO 1220
      1209 REM CALCULATE THE NUMBER OF WHOLE BLOCKS OF HEIGHT FOR BAR.
      1210 LET D=D-YB: LET H=INT(D*YSCALE+0.5)
      1220 LET IH=INT(H/8): LET L$="": LET M$="": LET R$=""
      1229 REM CONSTRUCT STRINGS TO HEIGHT FOR LEFT.MIDDLE AND RIGHT OF BARS.
      1230 FOR J=1 TO IH: LET L$=L$+CHR$ 133: LET M$=M$+CHR$ 143: LET R$=R$+CHR$ 138
      1239 REM FIND NUMBER OF PIXELS OF HEIGHT FOR SPECIAL CHARACTER AT TOP.
      1240 NEXT J: LET RH=H-IH*8: IF RH=0 THEN GO TO 1260
      1249 REM ADD SPECIAL CHARACTERS TO TOPS OF BARS.
      1250 LET L$=L$+CHR$(151-RH): LET M$=M$+CHR$(158-RH): LET R$=R$+CHR$(165-RH)
      1259 REM IF COLUMN POINTER IS ON A HALF-BLOCK PRINT RIGHT BIASED PART OF BAR.
      1260 IF COL=INT (COL) THEN GO TO 1280
      1270 LET A$=L$: GO SUB BAR: LET W=W-0.5: LET COL=COL+0.5
      1279 REM PRINT AS MANY WHOLE BLOCK BARS AS NEEDED.
      1280 IF W>0.1 THEN LET A$=M$: GO SUB BAR: LET W=W-1: LET COL=COL+1: GO TO 1280
      1289 REM SEE IF LEFT BIASED HALF-WIDTH BAR IS NEEDED.
      1290 IF W>0 THEN LET A$=R$: GO SUB BAR: LET COL=COL+0.5
      1299 REM MOVE COLUMN POINTER TO NEXT BAR AND RE-GO LOOP
      1300 LET COL=COL+GAP: NEXT I
      1310 INK 0: RETURN
      1320 REM BAR
      1321 REM IN - COL,A$
      1329 REM PRINT AS VERTICALLY FROM ROW 19 UPWARDS IN COLUMN.
      1330 FOR K=1 TO LEN A$: PRINT AT 19-K,INT COL;A$(K): NEXT K
      1340 RETURN
```

На листинге 6.9 приведен пример подпрограммы, заменяющей подпрограмму "HISTO". Эта подпрограмма "HISTO" (TYPE2) составлена так, что она гарантирует - толщина столбцов и промежуток между ними состоит из целого числа символьных блоков (X и GAP - целые числа). Для каждого столбца требуется задавать два числа, определяющие максимальную и минимальную высоту столбца. Используя команду OVER для печати столбцов, можно создавать диаграммы, подобные той, что на рис. 29, на которой показаны месячные диапазоны температур в ЭГХАМЕ. Операторы PRINT в этой подпрограмме, а также в последующих подпрограммах рассредоточены по тексту таким образом, чтобы читатель мог лучше проследить за логикой алгоритма по мере его выполнения.

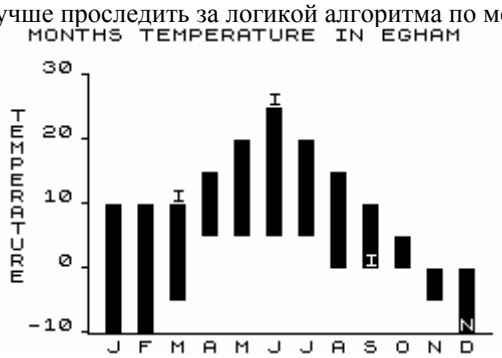


Рис. 29.

LISTING 6.9

```
      1000 REM HISTO/TYPE2
      1009 REM SET POINTER TO SUBROUTINE WHICH PRINTS BARS.
```

```
      1010 LET BAR=1320: DIM O$(2,3): LET O$(1)="MAX": LET O$(2)="MIN"
      1019 REM FIND SCALE AND DRAW AXES.
      1020 INPUT "RANGE OF VERTICAL ";YB;" TO ";YT
      1030 IF YB>=YT THEN GO TO 1020
      1040 LET YSCALE=128/(YT-YB)
      1050 PLOT 47,152: DRAW 0,-129: DRAW 201,0
      1059 REM LABEL VERTICAL AXIS.
      1060 LET YDIF=(YT-YB)/4: LET TICK=YB
      1070 FOR I=1 TO 5: LET TK=INT (TICK+0.5)
      1080 LET Y=32*I-8: PLOT 47,Y: DRAW -3,0: LET ROW=INT ((176-Y)/8)-1
      1090 LET A$=STR$ TK: IF LEN A$>3 THEN LET A$=A$(TO 3): IF TK>99 OR TK<-99 THEN
LET A$="***"
      1100 LET COL=1: IF TK>=0 THEN LET COL=COL+1
      1110 IF ABS TK<10 THEN LET COL=COL+1
      1120 IF ABS TK<100 THEN LET COL=COL+1
      1130 PRINT AT ROW,COL;A$: LET TICK=TICK+YDIF
      1140 NEXT I
      1149 REM FIND NUMBER OF BARS REQUIRED AND CALCULATE AN INTEGER WIDTH FOR THEN.
      1150 INPUT "NO. OF BARS=";NB: OVER 1
      1160 LET GAP=INT(13/NB): LET X=INT((26-GAP*NB)/NB)
      1170 LET GP=(25-NB*X-(NB-1)*GAP): LET GP=INT ((GP+1)/2)
      1179 REM POSITION COLUMN POINTER AT FIRST BAR THEN REPEAT LOOP FOR EACH BAR.
      1180 LET COL=6+GP: FOR I=1 TO NB: INPUT ("COLOUR FOR BAR "+STR$ I+"=");C: INK C
      1189 REM LOOP TO INPUT MAXIMUM AND MINIMUM VALUES FOR BAR.
      1190 FOR O=1 TO 2: LET I$=O$(O)+" FOR BAR "+ STR$ I+"=": INPUT (I$);D: LET W=X
      1200 IF D<=YB THEN LET D=0: LET H=1: GO TO 1220
      1209 REM CALCULATE THE NUMBER OF WHOLE BLOCKS OF HEIGHT FOR BAR.
      1210 LET D=D-YB: LET H=INT(D*YSCALE+0.5)
      1220 LET IH=INT(H/8): LET M$=""
      1229 REM CONSTRUCT STRING TO HEIGHT FOR BAR.
      1230 FOR J=1 TO IH: LET M$=M$+CHR$ 143
      1239 REM ADD SPECIAL CHARACTER FOR REMAINING HEIGHT.
      1240 NEXT J: LET RH=H-IH*8: IF RH=0 THEN GO TO 1260
      1250 LET M$=M$+CHR$ (158-RH)
      1259 REM START AT SAME COLUMN FOR BOTH BARS IN PAIR.
      1260 IF O=1 THEN LET OCOL=COL
      1270 IF O=2 THEN LET COL=OCOL
      1279 REM OUTPUT REQUIRED NUMBER OF WHOLE BLOCK WIDTH BARS.
      1280 IF W>=1 THEN LET A$=M$: GO SUB BAR: LET W=W-1: LET COL=COL+1: GO TO 1280
      1290 LET COL=COL+GAP: NEXT O: NEXT I
      1300 INK 0: OVER 0: RETURN
      1320 REM BAR
      1321 REM IN - COL,A$
      1329 REM PRINT AS VERTICALLY FROM ROW 19 UPWARDS IN COLUMN.
      1330 FOR K=1 TO LEN A$: PRINT AT 19-K,INT COL;A$(K): NEXT K
      1340 RETURN
```

Одним из больших удобств бейсика является то, что в программу можно вставлять дополнительные операторы STOP в программы, выяснить значение переменных, а затем продолжить выполнение оператором CONT, без изменения статуса программы. Это очень полезное средство отладки программ, которое также помогает понять принципы построения программы.

Упражнение 6.6

Используйте два различных цвета PAPER во время вывода столбцов гистограммы с помощью оператора OVER, для печати двух цветных столбцов. Будьте внимательны при использовании значений параметров столбца, разнящихся меньше чем на восемь элементов изображения!

При использовании двух цветов для одного столбца (упражнение 6.6) возникнут сложности. Если значения максимума и минимума попадает в один символьный блок, то для этого блока потребуется три цвета (два - цвет столбца, один - фоновый). Проверить, возникнут ли осложнения при раскрашивании столбца, можно, сравнив количество символов в цепочке, соответствующей минимальному значению, с количеством блоков, соответствующих максимальномы значению. Если они одинаковы, то простейший выход - укоротить цепочку, соответствующую мин. значению, на последний блок, что избавит от проблем с раскраской, но сделает диаграмму менее точной.

Эта проблема отпадает, если нижний столбец окрашивать в цвет фона (PAPER), что приведет к тому, что в каждом блоке используются только два цвета.

Существует еще множество вариантов: например, столбики могут располагаться относительно центральной прямой, что является хорошим способом иллюстрировать колебания валютного курса. Фундаментальные идеи, с которыми мы вас познакомили, помогут вам построить любую необходимую гистограмму.

Круговые диаграммы

Круговые диаграммы особенно полезны в биологии и экономике. Обычно к программам построения круговых диаграмм предъявляются следующие требования: они должны генерировать диаграммы различных размеров; должна быть предусмотрена возможность извлекать сектора из диаграммы и наносить на сектора штриховки и надписи различных типов. Первые два требования удовлетворены в программе (см. листинг 6.10 и рис. 30).

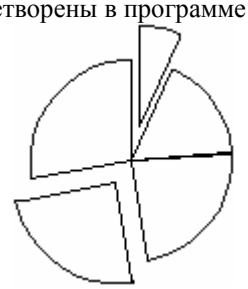


Рис. 30.

LISTING 6.10

```
2000 REM PIE/CHART
2009 REM SET POINTERS TO ROUTINES.
2010 LET HATCH=2300: LET IN=2800
2019 REM ALL SEGMENTS ARE INPUT AND TOTALLED TO CALCULATE ANGULAR SCALE.
2020 INPUT "NO.OF SEGMENTS=";NB: INPUT "COLOUR=";C$: LET C=VAL ("0"+C$): LET
TOT=0
2030 DIM D(NB): FOR I=1 TO NB: INPUT ("DATA "+STR$ I+": ");D(I): LET
TOT=TOT+D(I): NEXT I
2039 REM USE CURSOR TO SPECIFY CENTRE OF PIE.
2040 INPUT "PRESS ENTER FOR","CENTERING PIE ";LINE Y$
2050 GO SUB CURSOR: LET XC=PX: LET YC=PY
2060 INPUT "RADIUS (IN PIXELS) "; RAD
2070 LET ASCALE=2*PI/TOT: LET A1=PI/2
2079 REM ANY WEDGE MAY BE PULLED OUT BY MOVING THE CURSOR AWAY FROM THE CENTRE.
2080 FOR I=1 TO NB
2090 INPUT "PRESS ENTER FOR","CENTERING WEDGE";LINE Y$
2100 LET PX=XC: LET PY=YC: GO SUB CURSOR: LET ANG=ASCALE*D(I): LET A2=A1-ANG
2109 REM IF WEDGE IS TO MOVE OUT CALCULATE DISPLACEMENT ALONG IT'S BISECTOR.
2110 IF PX=XC AND PY=YC THEN GO TO 2140
2120 LET A3=A1-ANG/2: LET DIST=SQR ((PX-XC)*(PX-XC)+(PY-YC)*(PY-YC))
2130 LET PX= INT (XC+DIST*COS A3+0.5): LET PY= INT (YC+DIST*SIN A3+0.5)
2139 REM ENQUIRE WHETHING IS REQUIRED AND WHAT TYPE.
2140 INPUT "HATCH (X,Y,B,N)";H$
2149 REM IF HATCHING IS WANTED INPUT GAP SIZE BETWEEN LINES AND OFFSET.
2150 IF H$<>"N" THEN INPUT "JUMP="; JUMP, "REM="; REM
2159 REM DRAW SEGMENT OF PIE.
2160 INK C: PLOT PX,PY: LET X1= INT (RAD*COS A1-0.5): LET Y1= INT (RAD*SIN
A1+0.5): DRAW X1,Y1
2170 LET A2S=A2: LET ASTO=ANG: LET XA=PX+X1: LET YA=PY+Y1
2180 LET XB= INT (RAD*COS A2-0.5): LET YB= INT (RAD*SIN A2-0.5)
2190 FOR T=A1 TO A2 STEP -3/RAD
2200 LET X2=INT (RAD*COS T+0.5): LET Y2= INT (RAD*SIN T+0.5): DRAW X2-X1,Y2-Y1:
LET X1=X2: LET Y1=Y2
2210 NEXT T: DRAW XB-X2,YB-Y2: LET X2=XB: LET Y2=YB
2220 DRAW -XB,-YB: IF H$="N" THEN GO TO 2250
2228 REM CALL HATCHING ROUTINE IF NEEDED.
2229 REM IF ANGLE IS GREATER THEN HALF CIRCLE THEN DO HATCHING IN TWO PARTS.
2230 IF ASTO>PI THEN LET A2=A1-PI: LET XB=2*PX-XA: LET YB=2*PY-YA: GO SUB
HATCH: LET XA=XB: LET YA=YB: LET A1=A2: LET A2=A2S
2240 LET XB=PX+X2: LET YB=PY+Y2: GO SUB HATCH
2249 REM LOOP BACK FOR NEXT SEGMENT.
2250 LET A1=A2: NEXT I: RETURN
```

На листинге 6.10, используя подпрограмму "CURSOR", центр диаграммы устанавливается курсором, после чего вводится радиус диаграммы в пикселях. После этого вводятся данные о каждом элементе, затем эти данные суммируются и устанавливается угловой масштаб. Каждый сектор центруется курсором, при этом расстояние от центра диаграммы до перекрестья курсора задает величину сдвига вершины сектора по лучу, являющемуся биссектрисой угла, ограничивающего сектор. Для каждого нового сектора курсор устанавливается в центр диаграммы.

Штриховка секторов

Для штриховки секторов нам придется сначала определить точки пересечения прямых, штрихующих сектор с его границами, поэтому для упрощения вычислений мы ограничимся простейшим случаем штриховки сектора

вертикальными прямыми, горизонтальными прямыми и горизонтальными и вертикальными прямыми одновременно. Мы будем штриховать только сектора с величиной центрального угла меньшим 180 градусов. Сектора с центральным углом больше 180 градусов раскрашиваются как два сектора, первый из которых имеет центральный угол 180 градусов. Подпрограмма "PIE" запрашивает, будет ли штриховка горизонтальной (ответ "X"), вертикальной (ответ "Y"), оба вида штриховки (ответ "B"), или никакой штриховки (ответ "N").

Сектора диаграммы образованы двумя отрезками и дугой окружности. Мы должны определить, какая часть прямой, штрихующей сектор, находится в секторе (если такая часть существует). Так как центральный угол сектора не более 180 градусов, то имеются только четыре возможности:

1. Линия не попадает в сектор;
2. Пересекает дугу в двух точках;
3. Пересекает дугу и отрезок;
4. Пересекает оба отрезка.

Специальный случай пересечения прямой дуги и отрезка в одной точке может рассматриваться, как один из перечисленных выше случаев. Алгоритм штриховки объясняется на примере горизонтальной штриховки; вертикальная штриховка производится аналогичным образом. Сначала мы определяем максимальное и минимальное значение Y для прямых, пересекающих сектор. Затем мы рассматриваем все горизонтальные прямые, штрихующие сектор, задаваемые уравнениями вида Y=K*JUMP+REM и лежащие в этих пределах (0<=REM<=JUMP-1). Для каждой линии штриховки мы вычисляем две точки пересечения с продолженными отрезками, а затем проверяем, лежат ли у них значения MU между 0 и 1; что означает проверку, лежит ли точка пересечения между центром и дугой. После этого мы находим две точки пересечения штрихующей прямой полной окружности, содержащей дугу, ограничивающую сектор. Из этих точек можно выбрать те, которые соответствуют пересечению с сектором, после чего они соединяются штриховой линией (листинг 6.11).

LISTING 6.11

```
2300 REM HATCH
2309 REM IF CROSS HATCHING IS REQUIRED THEN RUN HATCH ROUTINE TWICE.
2310 IF H$="B" THEN LET H$="X": GO SUB 2320: LET H$="Y": GO SUB 2320: LET
H$="B": RETURN
2319 REM SET HATCHING VARIABLES TO CONTROL DIRECTION OF LINES.
2320 IF H$="Y" THEN LET PZ=PX: LET PT=PY: LET ZA=XA: LET TA=YA: LET ZB=XB: LET
TB=YB
2330 IF H$="X" THEN LET PZ=PY: LET PT=PX: LET ZA=YA: LET TA=XA: LET ZB=YB: LET
TB=XB
2340 DIM Z(3)
2349 REM FIND MAX AND MIN COORDINATES FOR LINES WHICH PASS THROUGH SEGMENT.
2350 LET T=PI/2: LET MAX=0: LET MIN=0
2360 LET VAL=SIN A1: IF H$="X" THEN LET VAL=COS A1
2370 IF MAX<VAL THEN LET MAX=VAL
2380 IF MIN>VAL THEN LET MIN=VAL
2390 IF T>A1 THEN LET T=T-PI/2: GO TO 2390
2400 IF T<A2 THEN GO TO 2450
2410 LET VAL=SIN T: IF H$="X" THEN LET VAL=COS T
2420 IF MAX<VAL THEN LET MAX=VAL
2430 IF MIN>VAL THEN LET MIN=VAL
2440 LET T=T-PI/2: GO TO 2400
2450 LET VAL=SIN A2: IF H$="X" THEN LET VAL=COS A2
2460 IF MAX<VAL THEN LET MAX=VAL
2470 IF MIN>VAL THEN LET MIN=VAL
2480 LET NEWMIN= INT (INT (RAD*MIN+1)/JUMP)*JUMP+REM
2489 REM FOR LINES WHICH CROSSES SEGMENT FIND INTERSECTIONS WITH SACIF AND ARC.
2490 FOR E=NEWMIN TO MAX*RAD STEP JUMP
2499 REM STORE INTERSECTION COORDINATE INFORMATION IN ARRAW Z(l:4).
2500 LET IC=0
2510 LET DEMON=TA-PT: IF DEMON=0 THEN GO TO 2540
2520 LET MU=E/DEMON: IF MU<0 OR MU>1 THEN GO TO 2540
2530 LET IC=IC+1: LET Z(IC)=PZ+MU*(ZA-PZ)
2540 LET DEMON=TB-PT: IF DEMON=0 THEN GO TO 2580
2550 LET MU=E/DEMON: IF MU<0 OR MU>1 THEN GO TO 2580
2560 LET IC=IC+1: LET Z(IC)=PZ+MU*(ZB-PZ)
2569 REM IF MORE THEN TWO POINTS OF INTERSECTION FOUNDE, DELETE DUPLICATES.
2570 IF IC=2 AND Z(1)=Z(2) THEN LET IC=1
2580 IF IC<>2 THEN GO TO 2610
2589 REM DRAW HATCH LINES.
2590 IF H$="Y" THEN PLOT Z(1),E+PT: DRAW Z(2)-Z(I),0: GO TO 2710
2600 IF H$="X" THEN PLOT E+PT,Z(1): DRAW 0,Z(2)-Z(1): GO TO 2710
2610 LET DISC=RAD*RAD-E*E: IF DISC<0 THEN GO TO 2710
2620 LET DISC=INT (SQR DISC+0.5)
```

```
2630 LET ZZ=PZ+DISC: LET AZ=DISC: GO SUB IN: IF OUT THEN GO TO 2650
2640 LET IC=IC+1: LET Z(IC)=ZZ
2650 LET ZZ=PZ-DISC: LET AZ=DISC: GO SUB IN: IF OUT THEN GO TO 2670
2660 LET IC=IC+1: LET Z(IC)=ZZ
2670 IF IC<2 THEN GO TO 2710
2680 IF IC=2 THEN GO TO 2590
2690 IF Z(1)=Z(2) THEN LET Z(2)=Z(3)
2700 GO TO 2590
2710 NEXT E
2720 RETURN
2800 REM IN
2808 REM CALCULATE ANGLE TO POINT OF INTERSECTION.
2809 REM IF ANGLE LINES BETWEEN ANGLES OF SEGMENT THEN POINT OF INTERSECTION IS
ON THE ARE OF THE SEGMENT.
2810 LET BZ=AZ: LET EZ=E
2820 IF H$="X" THEN LET B2=E: LET EZ=AZ
2830 IF BZ=0 THEN LET PHI=-PI/2: IF EZ>0 THEN LET PHI=-PHI
2840 IF B2=0 THEN GO TO 2860
2850 LET PHI=ATN (EZ/BZ): IF BZ<0 THEN LET PHI=PHI-PI
2859 REM SET FLAG TO INDICATE WHETHER OR NOT POINT IS VALID.
2860 LET OUT=(PHI>=A1) OR (PHI<=A2)
2870 RETURN
```

Графики

Для построения графиков необходимы подвижные координатные оси, обладающие широким диапазоном масштабов. Мы используем довольно распространенный метод расположения осей: если область графика проходит через ноль, то и ось проходит через ноль, а если нет - то ось располагается по ту сторону экрана, которая ближе к нулю. После этого на ось наносится пять штрихов, около каждого из которых наносится соответствующее этому штриху значение на оси. Требование точности научных графиков заставляет нас искать способы размещения возможно большего количества надписей в графическом поле. Так как обычных символов на экране помещается 32 в длину и 22 в высоту, то нам приходится пользоваться узкими символами, создаваемыми подпрограммой "THIN". Мы использовали CHARACTER GENERATOR для генерации двух видов узких символов: первый тип расположен в левой половине символьного блока, а второй - в правой. Когда цепочка печатается узкими символами, то подпрограмма "THIN" сначала печатает каждый второй символ цепочки в левой части блока, а затем командой OVER помещает оставшиеся символы в правые половины блока, так что в каждом блоке оказывается по два узких символа, а всего таких символов в строке экрана - 64. Подпрограмма "LABEL" требует преобразования узких чисел, печатаемых "THIN", и приведения их к одной длине (или числу десятичных знаков после запятой). Это достигается использованием подпрограммы "NUMBER", приведенной на листинге 6.12.

LISTING 6.12

```
3000 REM GRAPH
3009 REM SYMBOL ROUTINE IS USED TO MARK DATA POINTS ON DISCRETE GRAPHS.
3010 LET SYMBOL=3500: LET NUMBER=4700: LET THIN=5200
3019 REM CALCULATE SCALES AND DRAW AXES.
3020 INPUT "X GOES FROM ";XB;" TO ";XT: IF XT<=XB THEN GO TO 3020
3030 INPUT "Y GOES FROM ";YB;" TO ";YT: IF YT<=YB THEN GO TO 3030
3040 LET XSCALE=192/(XT-XB): LET YSCALE=128/(YT-YB)
3050 LET XO= INT (-XB*XSCALE+32.5): LET YO= INT (-YB*YSCALE+24.5)
3059 REM IF ZERO IS POT IN RANGE OF GRAPH MOVE AXIS TO APPROPRIATE EDGE.
3060 IF YT<0 THEN LET YO=153
3070 IF YB>0 THEN LET YO=23
3080 IF XT<O THEN LET XO=224
3090 IF XB>O THEN LET XO=31
3100 PLOT XO,23: DRAW 0,128: PLOT 31,YO: DRAW 192,0
3109 REM USE THIN ROUTINE TO PRINT LABEL ON AXES WITH FOUR FIGURE ACCURACY.
3110 LET XDIF=(XT-XB)/4: LET YDIF=(YT-YB)/4
3120 LET X=XB: LET Y=YB: FOR J=1 TO 5
3130 LET PX= INT ((X-XB)*XSCALE+32.5): LET PY=YO
3140 PLOT PX,PY-2:DRAW 0,4
3150 LET COL=INT (PX/8)-1: LET ROW=INT ((175-PY)/8)+1
3160 LET A=X: GO SUB NUMBER: GO SUB THIN
3170 LET PY= INT ((Y-YB)*YSCALE+24.5): LET PX=XO
3180 PLOT PX-2,PY: DRAW 4,0
3190 LET COL=INT (PX/8)+1: LET ROW=INT ((175-PY)/8)
3200 LET A-Y: GO SUB NUMBER: GO SUB THIN
3210 LET X=X+XDIF: LET Y=Y+YDIF: NEXT J
3220 INPUT "CONTINIUS OR ","DISCRETE GRAPH? ",D$: IF D$<>"C" OR D$<>"D" THEN GO
TO 3220
```

```
3230 IF D$="D" THEN GO TO 3320
3239 REM INPUT THE FUNCTION TO BE PLOTTED.
3240 INPUT "F(X):Y=";LINE F$
3250 LET X=XB: LET Y=VAL(F$): LET OY= INT ((Y-YB)*YSCALE+24.5)
3260 PLOT 32,OY
3270 FOR I=33 TO 224
3280 LET X=(I-32)*YSCALE+XB
3290 LET Y=VAL (F$): LET IY=INT ((Y-YB)*YSCALE+24.5)
3300 DRAW 1,IY-OY: LET OY=IY: NEXT I
3310 RETURN
3319 REM DISCRETE GRAPH REQUIRED SO INPUT SET OF POINTS.
3320 INPUT "NO. OF POINTS=";NP: DIM X(NP):DIM Y(NP)
3330 FOR I=1 TO NP: INPUT ("X("+STR$ I+") ");X(I);("Y("+STR$ I+") ");Y(I): NEXT
I
3339 REM SORT POINTS INTO ASCENDING ORDER OF X COORDINATE.
3340 FOR I=1 TO NP-1: FOR J=I+1 TO NP
3350 IF X(J)<X(I) THEN LET T=X(I): LET X(I)=X(J): LET X(J)=T: LET T=Y(I): LET
Y(I)=Y(J): LET Y(J)=T
3360 NEXT J: NEXT I
3370 LET X= INT ((X(1)-XB)*XSCALE+32.5): LET Y=INT ((Y(1)-YB)*YSCALE+24.5)
3380 PLOT X,Y: LET OX=X: LET OY=Y: GO SUB SYMBOL: PLOT OX,OY
3389 REM JOIN UP POINTS AND PLACE A "SYMBOL" AT EACH POINT.
3390 FOR I=2 TO NP
3400 LET X=INT ((X(I)-XB)*XSCALE+32.5): LET Y=INT ((Y(I)-YB)*YSCALE+24.5)
3410 PLOT X-OX,Y-OY: LET OX=X: LET OY=Y: GO SUB SYMBOL: PLOT OX,OY
3420 NEXT I: RETURN
3500 REM SYMBOL
3510 DRAW 0,1: DRAW 1,0: DRAW 0,-2: DRAW -2,0: DRAW 0,2
3520 RETURN
4700 REM NUMBER
4710 LET A$=STR$ A: IF LEN A$<=4 THEN RETURN
4720 LET A$=A$(TO 4)
4730 RETURN
```

Упражнение 6.7

Напишите усовершенствованный вариант программы "NUMBER", которой можно было бы определять формат возвращаемой цепочки. Один из способов задать формат числа - это задать его в виде шаблона, например, шаблон "#.##" определяет число с двумя десятичными знаками до запятой и тремя десятичными знаками после.

Упражнение 6.8

Составьте набор узких чисел, который необходим для составления надписей. Используйте его для надписывания диаграмм. При составлении набора вы можете руководствоваться рисунком 31:

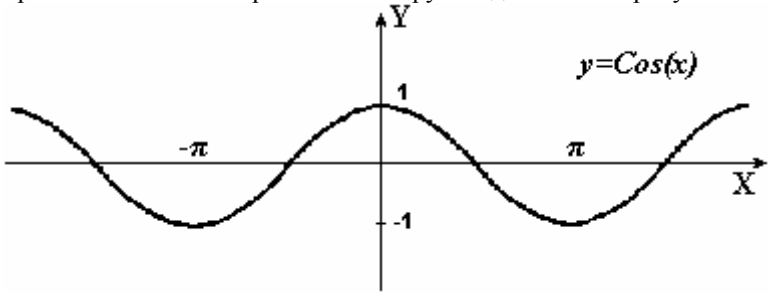


Рис. 31.

Кривые можно строить двумя способами: задавать кривые функционально и задавать кривые в виде набора дискретных точек, через которые проводится ломаная кривая. В первом случае по заданной функции вычисляется Y-координата над каждым элементом изображения, лежащим на оси X, и затем эти точки соединяются линией. Во втором случае сначала вводятся имеющиеся пары X и Y координат, которые записываются в память в порядке возрастания координаты X. Эти точки соединяются прямой. Первый вариант иллюстрирует рис. 31 - типичная континуальная кривая - график функции косинуса.

Упражнение 6.9

Мы видим, что для построения графиков единственная необходимая информация - это последовательность координат, расположенных по возрастанию X; соответствующие точки соединяются кривой. Эта последовательность может быть создана любым способом: последовательностью операторов READ или последовательностью вызова подпрограммы вычисления значения координат.

Вместо вычисления цепочки значений FS, мы могли бы написать подпрограмму, которая используется каждый раз, когда возникнет необходимость вычисления точки кривой. Напишите подпрограмму, позволяющую нарисовать график функции SIN(X)/X, чтобы избежать вычисления SIN(O)/O.

Список программ

Мы группируем листинг 6.6 ("MAIN PROGRAM", "CHARLOAD", "SET" и "QUERY"), 6.1 ("CURSOR" и "GRID"), 6.4 ("SAVE" и "LOAD") под именем "LIBDIAG" и используем их совместно с 6.2 ("PAPER" и "INK"), 6.3 ("POINT" и "LINE"), 6.5 ("LABEL"), 6.7 ("CREATE" и "THIN"). Обратите внимание на изменения, которые необходимо внести в программы, если вы используете 16-ти килобайтный вариант SPECTRUMa (см. приложение А).

1. "LIBDIAG", 6.2, 6.3, 6.5, 6.7.

Необходимая информация:

"DEFINE CHARACTER? ("Определить символы?"). Напечатайте Y (да) для генерации поворота символов и генерации блоков для гистограмм; в остальных случаях – M (нет).

"LOAD CHARACTER? ("Загружать символы?") - Y, если специальные наборы символов (включая вышеприведенные) вводятся с ленты; в противном случае - N.

"LOAD PICTURE? ("Загрузить картинку?"): Y - загрузить картинку, созданную ранее, N - нет.

"DRAW DIAGRAM?" ("Подписывать рисунок?"): - Y, если используется подпрограмма "LABEL", N - если нет.

*"COLOUR PAPER?" ("Окрашивать фон?"). Если Y (да), то задайте "WHICH COLOUR?" ("Какой цвет"), например - I; установите курсор в необходимое положение, используя при необходимости подпрограмму "GRID" (сетка), и напечатайте размер участка в блоках, длина по горизонтали, длина по вертикали (например, 2*4).*

*"COLOUR INK?" ("Цвет туши?"). Если Y (да), то определите " WHICH COLOUR?" ("Какого цвета?"), например, 5; установите курсор в необходимое положение, используя по необходимости программу "GRID", и впечатайте размер области в блоках (например, 3*1).*

"DRAW POINT?" ("Нанести точку?"). Если Y (да), то используйте курсор для указания точки; "WHICH COLOUR?" ("Какого цвета?")', например, 6 и параметр команд OVER (1 или 0).

"DRAW LINET' ("Начертить прямую?"). Если да, то курсором укажите концы отрезка;" WHICH COLOUR?" ("Какого цвета?"), (например, 5) и параметр для OVER (1 или 0).

"END PICTURE?" ("Конец рисунка?"). Если Y, то повторите последовательность сначала;

"SAVE PICTURE?" ("Сохранить рисунок?"). Если Y, то введите имя рисунка.

Поэкспериментируйте с построением графиков.

2. "LIBDIAG" и листинг 6.8 ("HISTO/TYPEl). Пример данных:

```
"RANGE OF VERTICAL" .....0   "TO"..............100
"NO.OFBARS"  ............6
"DATA FOR BAR1"........ 56   "COLOUR"............2
"DATA FOR BAR2"........ 95   "COLOUR"............6
"DATA FOR BAR3"........ 20   "COLOUR"............4
"DATA FOR BAR4"........ 77   "COLOUR"............5
"DATA FOR BAR5"........ 54   "COLOUR"............1
"DATA FOR BAR6"........ 33   "COLOUR"............3
```

3. "LIBDIAG" и листинг 6.9 ("HISTO/TYPE2). Пример данных:

```
"RANGE OF VERTICAL".....0   "TO"................50
"NO.OFBARS".............4
"COLOUR FOR BAR 1"......2
"DATA MAX FOR BAR 1"...44
"DATA MIN FOR BAR 1"...22
"COLOUR FOR BAR2".......6
"DATA MAX BAR2".........36
"DATAMINBAR2"...........5
"COLOURFORBAR3".........4
"DATA MAX BAR3".........42
"DATA MIN BARS".........29
"COLOUR FOR BAR4".......1
"DATA MAX BAR4".........31
"DATA MIN BAR4".........12
```

4. "LIBDIAG" и листинг 6.10 и 6.11 ("PIE", "HATCH" и т.д.). Пример данных:

```
"NO. OF SEGMENT"........3
"COLOUR"................0
"DATA 1"................1
"DATA 2"................2
"DATA 3"................3
```

Укажите курсором центр диаграммы, затем:

```
"RADIUS (IN PIXELS)"...75
```

Используйте курсор для указания положения вершины сектора

```
"HATCH (X, Y, B, N)"....B   "JUMP"..............8
"REM"...................5
```

Укажите курсором вершину сектора

```
"HATCH (X, Y, B, N)"....Y   "JUMP"..............I:
"REM"...................0
```

Укажите курсором вершину сектора

```
"HATCH" (X, Y, B, N)....P
```

5. "LIBDIAG" и листинг 6.12 ("GRAPH" и т.д.)
Вопросы программы содержат всю информацию, необходимую для ответа на них.

ГЛАВА 7
Трёхмерная декартова геометрия

Перед тем, как изучать методы трехмерной графики, нам необходимо ознакомиться с методами декартовой трехмерной геометрии. Так же, как и в двумерном варианте, мы начинаем с того, что выбираем в пространстве точку, которую считаем началом координат, и проводим через неё три взаимно перпендикулярные прямые, которые мы называем осью X, осью Y и осью Z. На этих осях следует указать положительное и отрицательное направления, так что расстояние от точки, расположенной по положительную сторону от начала координат, положительно, а отрицательную - отрицательно.

Для осей X и Y положительное направление можно указать так же, как и в двумерном случае - предположим, что плоскость, в которой лежат эти оси, совпадает с плоскостью страницы, ось X горизонтальна, ось Y - вертикальна, тогда положительное направление на оси X - вправо от начала координат, а положительное направление по оси Y - вверх от начала координат. Для оси Z остается только две возможности: она должна идти перпендикулярно плоскости листа (это равносильно перпендикулярности обеим координатным осям), и может быть направлена положительной полуосью в страницу, или положительной полуосью из страницы.

Первый случай соответствует так называемой правосторонней координатной системе, а второй - левосторонней. Всюду в этой книге мы будем пользоваться только левосторонней системой координат. Все, что относится к левосторонним координатам, относится и к правосторонним системам, так что неважно, какие из них использовать, важно только придерживаться какой-то одной системы.

С помощью координатной системы мы отождествляем точки P пространства с тройками действительных чисел (X,Y,Z), где эти числа называются координатными точками и являются расстояниями от проекции точки на координатную ось до начала координат. Проекцией точки на координатную ось называется точка, которая однозначно определяется тем, что прямая, соединяющая проекцию точки и точку перпендикулярна этой оси.

Над трёхмерными векторами можно производить две операции:

- Умножить на число - K*(X,Y,Z) = (K*X,K*Y,K*Z);
- Складывать - сумма векторов P1 = (X1,Y1,Z1) и P2 = (X2,Y2,Z2) есть вектор (X1+X2,Y1+Y2,Z1+Z2), то есть при сложении векторов соответствующие координаты складываются.

Определение прямой

Теперь мы определим прямую, проходящую через две заданные точки (X1,Y1,Z1), (X2,Y2,Z2), что можно сделать, задав уравнение этой прямой:

```
(X-X1)*(Y2-Y1)=(Y-Y1)*(X2-X1)
(Y-Y1)*(Z2-Z1)=(Z-Z1)*(Y2-Y1)
(Z-Z1)*(X2-X1)=(X-X1)*(Z2-Z1)
```

Хотя здесь приведены три уравнения, их решение определяет прямую, а не единственную точку, что связано с тем, что эти уравнения линейно зависимы так, что, задавая одну из координат, можно однозначно получить значение двух других (см. пример 7.A).

Также, как и в случае двух измерений, это не единственный способ задавать прямую. Существует ещё параметрический способ задания прямой, проходящей через точки P1 и P2:

`P(MU)=(1-MU)*P1+MU*P2`, или

```
P(MU)=(1-MU)*X1+MU*X2,(1-MU)*Y1+MU*Y2,(1-MU)*Z1+MU*Z2,
```

где MU - действительный параметр.

Параметрический вид прямой в точности совпадает с двумерным случаем. Символ MU в скобках указывает на то, что точка на прямой параметрически зависит от MU. Начиная с этого момента, будем опускать MU. Обратите внимание, что MU=1 дает точку P2, а MU=0 - точку P1.

Мы можем записать параметрическое уравнение прямой в таком виде:

```
P(MU)=P1+MU*(P2-P1)
```

Так же, как и в двумерном варианте, вектор P1 называется базовым вектором, а вектор (P2-P1) - направляющим вектором. Мы, опять-таки, имеем двойственность в представлении векторов: с одной стороны, вектор можно отождествлять с точкой, проекции которой на координатные оси равны координатам вектора, а с другой стороны - вектор задает направление, совпадающее с направлением прямой, проведенной из начала координат в эту точку. При этом направление на прямой от начала координат к точке считается положительным, а обратное направление - отрицательным. Таким образом, вектора (X,Y,Z) и (-X,-Y,-Z) определяют одну и ту же прямую в пространстве, но с разной ориентацией.

Определим норму вектора (которую также называют модулем вектора, его длиной), обозначаемую ABS(D), как расстояние от точки, определяемой вектором, до начала координат

```
ABS(D)=SQRT(X**2+Y**2+Z**2)
```

Точку P+MU*D на прямой можно определить следующим образом: найти точку P и сдвинуть по прямой на расстояние MU*ABS(D) в положительном направлении, если MU положительно, и в отрицательном направлении - если MU отрицательно; при этом мы попадаем в точку P+MU*D. Заметьте, что любая точка прямой может служить базовым вектором, и что направляющий вектор можно умножить на любое, отличное от нуля, число.

Если вектор D=(X,Y,Z) составляет с осью X,Y,Z, углы TETX, TETY и TETZ, то

```
X:Y:Z=COS(TETX):COS(TETY):COS(TETZ),
```

это означает, что

```
D=(LMD*COS(TETX),LMD*COS(TETY),LMD*COS(TETZ))
```

Из свойств трёхмерного пространства следует:

(COS(TETX))2+(COS(TETY))**2+(COS(TETZ))**2=1**

Отсюда следует, что LMD=ABS(D), и что если направляющий вектор единичной длины, то его компоненты: (COS OX, COS OY, COS OZ) и LMD=-1. Координаты единичного вектора называются направляющими косинусами. В общем случае, направляющие косинусы прямой с направляющим вектором D=(X,Y,Z) есть

$$\left(\frac{X}{ABS(D)},\frac{Y}{ABS(D)},\frac{Z}{ABS(D)}\right)$$

Пример 7.А

Используя три приведенных метода, описать прямую, соединяющую точки (1,2,3) и (-1,0,2).

Произвольная точка (X,Y,Z), лежащая на прямой, удовлетворяет уравнению

(X-1)×(0-2)=(Y-2)×(-1-1),
(Y-2)×(2-3)=(Z-3)×(0-2),
(Z-3)×(-1-1)=(X-1)×(2-3),

что эквивалентно:

-2*X+2*Y =2
-Y+2*Z =4
-2*Z+X =-5

Мы можем ограничиться рассмотрением только последних двух уравнений, так как первое уравнение получается из суммы второго и третьего уравнений, умноженных на -2

Y= 2*Z-4 и
X= 2*Z-5,

так что прямая задается одним параметром, в нашем случае Z, что дает точки прямой в параметрическом виде (2*Z-5, 2*Z-4,Z). Этот результат легко проверить: значение Z=3 соответствует точке (1,2,3), а значение Z=2 соответствует точке (-1,0,2), а эти точки и определяют прямую.

В векторной форме точка прямой задается:

P(MU)=(1-MU)(1,2,3)+MU(-1,0,2)=(1-2*MU,2-2*MU,3-MU)

Координаты точки и в этом случае зависят от одного параметра. Значению MU=0 отвечает точка (1,2,3), значению MU=1 - точка (-1,0,2).

Если задавать прямую с помощью базового и направляющего вектора, то

P'(MU)=(-1,0,2)+MU(-2,-2,1)

Мы можем заменить координаты направляющего вектора направляющими косинусами (-2/3,-2/3,-1/2), что дает другую форму представления прямой с помощью базового и направляющего векторов:

P''(MU)=(1,2,3)+MU(-2/3,-2/3,-1/3)

Естественно, что одно и тоже значение задает различные точки прямой; например, P(3)=(-5,4,0), P'(3)=(-7,-6,-1) и P''(3)=(-1,0,2). Эта прямая составляет:

131.81º(ARCCOS(-2/3)),
131.81º(ARCCOS(-2/3)),
109.47º(ARCCOS(-1/3))

с осями X, Y и Z соответственно.

Угол между двумя направляющими векторами

Для вычисления угла между двумя векторами введем понятие скалярного произведения двух векторов:

P*Q=(X1,Y1,Z1)(X2,Y2,Z2)=X1*X2+Y1*Y2+Z1*Z2

Если P и Q - оба вектора единичной длины (что означает, что они записаны как направляющие косинусы), а TETA - угол между прямыми, то COS(TETA)=P*Q (см. главу 3, где приведён двумерный аналог этой формулы). В общем случае угол между прямыми с направляющими векторами P и Q (мы предполагаем, что обе прямые проходят через начало координат):

ARCCOS((P/ABS(P))*(Q/ABS(Q)))

Очевидно, что P и Q перпендикулярны, только если P*Q=0.

Определение плоскости

Перейдем к определению плоскости. Точки плоскости задаются уравнением

N*X=K,

где K - скаляр, а N - вектор, перпендикулярный плоскости (см. пример 7.Б). Если A - принадлежит плоскости, то N*A=K, заменяя K в предыдущем уравнении, получим

N*X=N*A
N*(X-A)=0

Последнее уравнение станет очевидным, если вспомнить, что скалярное произведение двух перпендикулярных векторов равно нулю. Для любой точки X плоскости, не совпадающей с A, X-A можно рассматривать, как направляющий вектор прямой, лежащей в плоскости, а так как вектор N перпендикулярен плоскости, то он перпендикулярен и каждой прямой, лежащей в плоскости, а следовательно, N*(X-A)=COS(PI/2)=0.

Расписывая это уравнение по координатам, мы получим обычное уравнение плоскости:

(N1,N2,N3)*(X,Y,Z)=N1*X+N2*Y+N3*Z=K

Обратите внимание, что две плоскости параллельны тогда и только тогда, когда вектора, нормальные к этим плоскостям, параллельны.

Точка пересечения прямой и плоскости

Предположим, что прямая задана как A+MU*D, а плоскость N*X=K. Тогда для нахождения точки пересечения прямой и плоскости нам необходимо найти такое значение MU, чтобы

N*(A+MU*D)=K, откуда

MU=(K-N*A)/(N*D)

где (N*D) не =0.

Если N*D=0, то прямая параллельна плоскости, и точек пересечения нет.

Расстояние от точки до прямой

Расстоянием от точки P до плоскости N*X-K называется наименьшее из расстояний от точки P до точек, лежащих на этой плоскости, откуда следует, что прямая, соединяющая точку P с точкой P2, лежащей на плоскости N, такой, что расстояние от точки P до P2 равно расстоянию от точки P до плоскости, перпендикулярна этой плоскости.

Эта прямая может быть записана в виде P+MU*N;

MU определяется точкой P2:

MU=(K-N*P)/(N*N)

откуда расстояние от точки до плоскости:

MU*ABS(N)=ABS(K-N*P)/ABS(N)

В частности, если P - начало координат, то расстояние от плоскости до начала координат есть ABS(K)/ABS(N). Более того, если N - единичный вектор, то K - расстояние от плоскости до начала координат.

Пример 7.Б

Найдем точку пересечения прямой, соединяющей точки (1,2,3) с (-1,0,2), и плоскости (0,-2,1)*X=5, а также расстояние от плоскости до начала координат

B =(1,2,3)
N =(0,-2,1)
D =(-1,0,2)-(1,2,3)=(-2.-2,-1)
N*B =(0*1,-2*2,1*3)=-1
N*D =(0*-2,-2*-2,1*-1)=3

откуда значение MU, соответствующее точке пересечения, есть (5-(-1))/3=2, а точка пересечения (1,2,3)+2(-2,-2,-1)=(-3,-2,1); расстояние от плоскости до начала координат есть

5/(N)=5/SQRT(5)=SQRT(5)

Программа на листинге 7.1 позволяет вычислить точку пересечения (массив P) плоскости и прямой. Базовый вектор прямой - B, направляющий вектор прямой - D, нормаль к плоскости - N, константа плоскости K. Обратите внимание на тот факт, что из-за использования десятичной арифметики мы не можем проверить, равно ли нулю скалярное произведение, так как следует учитывать ошибки округления. Всё, что мы можем определить - это является ли скалярное произведение пренебрежимо малым; величина того, что считать бесконечно малым, определяется программистом (на SPECTRUMe это примерно шесть знаков после запятой).

LISTING 7.1

```
100 REM INTERSECTION OF LINE AND PLANE
110 DIM B(3): DIM D(3): DIM N(3):DIM P(3): DIM A$(8)
120 INPUT "BASE VECTOR OF LINE","(";B(1);",";B(2);",";B(3);")"
130 PRINT AT 1,0;"BASE VECTOR OF LINE ","(";B(1);",";B(2);",";B(3);")"
140 INPUT "DIRECTION VECTOR OF LINE ","(";D(1);",";D(2);",";D(3);")"
150 PRINT AT 4,0;"DIRECTION VECTOR OF LINE ","(";D(1);",";D(2);",";D(3);")"
160 INPUT "NORMAL TO PLANE ,,"(";N(1);",";N(2);",";N(3);")"
170 PRINT AT 7,0;"NORMAL TO PLAN ,,"(";N(1);",";N(2);",";N(3);")"
180 INPUT "PLAN CONSTANT ";K
190 PRINT AT 10,0;"PLAN CONSTANT ";K
199 REM CALCULATE POINT OF INTERSECTION (P(1), P(2),P(3)) OF LINE AND PLANE,
INPUT ABOVE.
200 LET DOT=N(1)*D(1)+N(2)*D(2)+N(3)*D(3)
209 REM ZERO DOT PRODUCT SO NO INTERSECTION.
210 IF ABS DOT<0.000001 THEN PRINT AT 15,0;"NO POINT OF INTERSECTION": STOP
220 LET MU=(K-N(1)*B(1)-N(2)*B(2)-N(3)*B(3))/DOT
230 FOR I=1 TO 3: LET P(I)=B(I)+MU*D(I): IF ABS P(I)<0.000001 THEN LET P(I)=0
240 NEXT I: PRINT AT 15,0;"POINT OF INTERSECTION","("
249 REM TIDY UP OUTPUT.
250 FOR I=1 TO 3: LET A$=STR$ P(I): FOR J=1 TO 8
260 IF A$<>" " THEN PRINT A$(J);
270 NEXT J: IF I<>3 THEN PRINT",";
280 NEXT I: PRINT ")"
290 STOP
```

Точка пересечения двух прямых

Предположим, что точка пересечения двух прямых задаваемых, как B1+MU*D1, и B2+LMD*D2, существует (если прямые не компланарны или они параллельны, то точки пересечения не существует). Точка пересечения определяется таким значением MU и LMD, что

B1+MU*D1=B2+LMD*D2 **(7.1)**

Это уравнение векторное и состоит их трёх скалярных уравнений.

Для разрешимости этого уравнения необходимо, чтобы только два из них были независимыми, а оставшееся являлось бы их следствием. Две прямые параллельны, если вектора D1 и D2 пропорциональны. Возьмём два независимых уравнения, решим их относительно MU и LMD (у нас два уравнения и два неизвестных) и подставим эти значения в третье уравнение для проверки совместимости этих уравнений.

Пример 7.B иллюстрирует этот метод, а на листинге 7.2 содержится программа, находящая этим методом точку пересечения прямых. Базовый и направляющий вектор первой прямой хранится в массивах B и D, а вектора второй прямой - в массивах C и E; найденная точка пересечения записывается в массив P.

Обратите внимание, что если два независимых уравнения есть:

```
A(1,1)*MU+A(1.2)*LMD=B1                    (7.2)
A(2,1)*MU+A(2,2)*LMD=B2                    (7.3)
```

то их детерминант DELTA=A(1,1)*A(2,2)-A(1,2)*A(2,1) отличен от нуля (так как уравнения независимы), и решения можно записать в виде:

```
MU=(A(2,2)*B1-A(1,2)*B2)/DELTA;
LMD=(A(1,1)*B2-A(2,1)*B1)/DELTA.
```

LISTING 7.2

```
100 REM INTERSECTION OF TWO LINES
110 DIM B(3): DIM D(3): DIM C(3): DIM E(3): DIM P(3): DIM A$(8)
120 INPUT "BASE VECTOR OF FIRST LINE ","("; B(1);",";B(2);",";B(3);"),"
130 PRINT AT 1,0;"BASE VECTOR OF FIRST LINE ","(";B(1);",";B(2);",";B(3);")"
140 INPUT "DIRECTION VECTOR OF FIRST LINE ","(";D(1);",";D(2);",";D(3);")"
150 PRINT AT 4,0;"DIRECTION VECTOR OF FIRST LINE ","(";D(1);",";D(2);",";D(3);
")"
160 INPUT "BASE VECTOR OF SECOND LINE ","(";C(1);",";C(2);",";C(3);")"
170 PRINT AT 7,0;"BASE VECTOR OF SECOND LINE ","(";C(1);",";C(2);",";C(3);")"
180 INPUT "DIRECTION VECTOR OF SECOND LINE ","(";E(1);",";E(2);",";E(3);")"
190 PRINT AT 10,0;"DIRECTION VECTOR OF SECOND LINE ","(";E(1);",";E(2);",";
E(3);")"
199 REM CALCULATE POINT OF INTERSECTION (P(1),P(2),P(3)) OF TWO LINES, DATA
INPUT ABOVE. FIND ANY TWO INDEPENDENT LINE EQUATIONS FROM THE THREE (X/Y/Z).
200 FOR I=1 TO 3
210 LET J=I+1: IF J=4 THEN LET J=1
220 LET DELTA=E(I)*D(J)-E(J)*D(I)
230 IF ABS DELTA>0.000001 THEN GO TO 260
240 NEXT I
249 REM CANNOT FIND TWO INDEPENDENT EQUATIONS-LINES DO NOT INTERSECT.
250 PRINT AT 15,0;"LINES DO NOT INTERSECT": STOP
259 REM CALCULATE MU AND LAMBDA VALUES OF POINT OF INTERSECTION.
260 LET MU=(E(I)*(C(J)-B(J))-E(J)*(C(I)-B(I)))/DELTA
270 LET LAMBDA=(D(I)*(C(J)-B(J)-D(J))*(C(I)-B(I)))/DELTA
279 REM NO SOLUTION IF MU AND LAMBDA DO NOT MATISFY THIRD EQUATION.
280 LET K=J+1: IF K=4 THEN LET K=1
290 IF ABS(B(K)+MU*D(K)-C(K)-LAMBDA*E(K))>0.000001 THEN GO TO 250
299 REM CALCULATE (P(1),P(2),P(3)) USING MU VALUE.
300 FOR I=1 TO 3: LET P(I)=B(I)+MU*D(I): IF ABS P(I)<0.000001 THEN LET P(I)=0
310 NEXT I: PRINT AT 15,0;"POINT OF INTERSECTION ","(";
319 REM TIDY UP OUTPUT.
320 FOR I=1 TO 3: LET A$=STR$ P(I): FOR J=1 TO 8
330 IF A$(J)<>" " THEN PRINT A$(J);
340 NEXT J: IF I<>3 THEN PRINT",";
350 NEXT I: PRINT ")"
360 STOP
```

Пример 7.B.

Найдите точки пересечения (если они имеются)

(А) (1,1,1)+MU(2,1,3) с уравнением (0,0,1)+LMD(-1,1,1)

(Б) (2,3,4)+MU (1,1,1) с уравнением (-2,-3,-4)+LMD(1,2,3)

В (А) эти три уравнения есть:

```
1+2*MU=0-LMD                               (7.4)
1+MU=0+LMD                                 (7.5)
1+3*MU=1+LMD                               (7.6)
```

Из уравнений 7.4 и 7.5 мы получаем MU=-2/3 и LMD=1/3, подставляя в уравнение 7.6, получаем 1+3×(-2/3)=-1 в левой части уравнения и 1+1×(1/3)=4/3 - в правой части, так, что прямые не пересекаются.

Из (Б) следуют уравнения:

```
2+MU=-2+LMD                                (7.7)
3+MU=-3+2LMD                               (7.8)
4+MU =-4+3LMD                              (7.9)
```

Из 7.7 и 7.8 получаем MU=-2 и LMD=2, и эти значения удовлетворяют 7.9 (правая и левая части равны 2). Так что точка пересечения:

(2,3,4)-2(1,1,1)=(-2,3,4)+2(1,2,3)=(0,1,2)

Плоскость, проходящая через три неколлинеарные точки

Для решения задачи мы вводим понятие векторного произведения двух векторов, результатом которого является вектор

P*Q=(P1,P2,P3)*(Q1,Q2,Q3)=(P2*Q3-P3*Q2,P3*Q1-P1*Q3,P2*Q3-P3*Q2)

Если P и Q не параллельны, то вектор P*Q перпендикулярен как P, так и Q. Следует отметить, что векторное произведение не коммутативно, то есть P*Q≠Q*P. При изменении порядка сомножителей векторного произведения вектор результата меняет направление на противоположное. Например, (1,0,0)*(0,1,0)=(0,0,1), но (0,1,0)*(1,0,0)=(0,0,-1); вектора (0,0,1) и (0,0,-1) оба параллельны оси Z. На листинге 7.3 приведена программа, вызывающая подпрограмму "VECPROD" (для векторов L и M вычисляется их векторное произведение N, и "DOTPROD", (вычисляющее значение DOT скалярного произведения L и M); обе эти программы приведены на листинге 7.4.

LISTING 7.3

```
100 REM DOT PRODUCT AND VECTOR PRODUCT
110 LET VECPROD=300: LET DOTPROD=400
120 DIM L(3): DIM M(3): DIM N(3)
130 INPUT "VECTOR L ","(";L(1);",";L(2);",";L(3);")"
140 PRINT AT 1,O;"VECTOR L ","(";L(1);",";L(2);",";L(3);")"
150 INPUT "VECTOR M ","(";M(1);",";M(2);",";M(3);")"
160 PRINT AT 4,0;"VECTOR M ","(";M(1);",";M(2);",";M(3);")"
170 GO SUB VECPROD
180 PRINT AT 8,0;"VECTOR PRODUCT ","(";N(1);",";N(2);",";N(3);")"
190 GO SUB DOTPROD
200 PRINT AT 11,0;"DOT PRODUCT ";DOT
210 STOP
```

LISTING 7.4

```
300 REM VECPROD
301 REM IN - L(3),M(3)
302 REM OUT- N(3)
309 REM N IS THE VECTOR PRODUCT OF L AND M.
310 LET NI=2: LET NNI=3
320 FOR I=1 TO 3
330 LET N(I)=L(NI)*M(NNI)-L(NNI)*M(NI)
340 LET NI=NNI: LET NNI=NI+1: IF NNI=4 THEN LET NNI=1
350 NEXT I
360 RETURN
400 REM DOTPROD
401 REM IN - L(3),M(3)
402 REM OUT - DOT
409 REM DOT IS THE DOT PRODUCT OF L AND M.
410 LET DOT=0
420 FOR I=1 TO 3: LET DOT=DOT+L(I)*M(I): NEXT I
430 RETURN
```

Предположим, что даны три неколлинеарные точки – P1, P2 и P3, тогда векторы P2-P1 и P3-P1 представляют два направления, пересекающиеся в P1 и лежащие в плоскости, содержащей эти три точки. Мы знаем, что нормаль к плоскости перпендикулярна каждой линии, лежащей в плоскости, в частности и двум упомянутым выше. Так как эти точки неколлинеарны, то P2-P1=P3-P1 и нормаль к плоскости есть (P2-P1)*(P3-P1), а уравнение прямой есть ((P2-P1)*(P3-P1)*(X-P1))=0

Пример 7.Г

Найдите уравнение плоскости, проходящей через точки (0,1,1), (1,2,3) и (-2,3,-1).

Точка такой плоскости удовлетворяет

(((1,2,3)-(0,1,1)*((-2,3,-1)-(0,1,1)))*((X,Y,Z)-(0,1,1))=0,

что даёт

((1,1,2)*(-2,2,-2)*(X,Y-1,Z-1)=0 или
(-6,-2,4)*(X,Y-1,Z-1)=0

Это равносильно -6X -2Y + 4Z -2 - 0, а это равносильно 3X + Y - 2Z - -1.

Точка пересечения трёх плоскостей

Предположим, что плоскости заданы уравнениями 7.10-7.12, приведенными ниже:

N1*X=K1 (7.10)
N2*X=K2 (7.11)
N3*X=K3 (7.12)

где N1=(N(1,1),N(1,2),N(1,3)), N2=(N(2,1),N(2,2),N(2,3)), N3=(N(3,1),N(3,2),N(3,3)). Мы можем переписать эти уравнения в матричном виде:

N(1,1) N(1,2) N(1,3) X K1

```
N(2,1) N(2,2) N(2,3)    *    Y    =    K2
N(3,1) N(3,2) N(3,3)         Z         K3
```

Решение задается в виде столбца

```
X         N(1,1) N(1,2) N(1,3)         K1
Y    =    N(2,1) N(2,2) N(2,3)    *    K2
Z         N(3,1) N(3,2) N(3,3)         K3
```

Мы видим, что вычисление точки пересечения трёх плоскостей требует нахождения обратной матрицы размера (3*3). На листинге 7.5 приводится программа, вычисляющая обратную матрицу присоединённым методом. Обращаемая матрица - N, обратная - M.

LISTING 7.5

```
500 REM INVERSE OF 3*3 MATRIX
501 REM IN - N(3,3)
502 REM OUT- M (3,3), SINGULAR
510 LET SINGULAR=1
520 LET DET=0: LET NI=2:LET NNI=3
530 FOR I=1 TO 3
540 LET DET=DET+N(1,I)*(N(2,NI)*N(3,NNI)-N(2,NNI)*N(3,NI))
550 LET NI=NNI: LET NNI=NI+1: IF NNI=4 THEN LET NNI=1
560 NEXT I
569 REM DETERMINANT OF SINGULAR MATRIX IS ZERO, THERE IS NO INVERSE.
570 IF ABS DET<0.000001 THEN RETURN
579 REM CALCULATE M, THE INVERSE OF N, BY THE ADJOINT METHOD.
580 LET NI=2: LET NNI=3
590 FOR I=1 TO 3
600 LET NJ=2: LET NNJ=3
610 FOR J=1 TO 3
620 LET M(J,I)=(N(NI,NJ)*N(NNI,NNJ)-N(NI,NNJ)*N(NNI,NJ))/DET
630 LET NJ=NNJ: LET NNJ=NJ+1: IF NNJ=4 THEN LET NNJ=1
640 NEXT J
650 LET NI=NNI: LET NNI=NI+1: IF NNI=4 THEN LET NNI=1
660 NEXT I
670 LET SINGULAR=0
680 RETURN
```

В программе на листинге 7.6 векторы представляются одномерными массивами; массив B(3) содержит решение уравнений, а K(3) - константы плоскостей. Нормали N1, N2 и N3 задаются в матрице N размерности (3*3). Если две из этих плоскостей параллельны или если эти плоскости пересекаются по прямой, то точка пересечения не единственна; в этом случае переменная DET, равная детерминанту N, равна нулю, и переменная SINGULAR=1.

LISTING 7.6

```
100 REM INTERSECTION OF THREE PLANES
110 DIM N(3,3): DIM M(3,3): DIM K(3) :DIM B(3)
120 LET INV=500
129 REM INPUT DATA OF THREE PLANES.
130 PRINT AT 2,4;"COEFFICIENTS CONSTANT"
140 FOR I=1 TO 3
150 PRINT AT 2+2*I,0;"PLANE(";I;")-(   ,   ,   )"
160 FOR J=1 TO 3
170 LET I$="INPUT N("+STR$ I+","+STR$ J+") "
180 INPUT (I$);N(I,J): PRINT AT 2+2*I,7+4*J;N(I,J)
190 NEXT J
200 LET I$="INPUT K("+STR$ I+") "
210 INPUT (I$);K(I): PRINT AT 2+2*I,28;K(I)
220 NEXT I
229 REM IF MATRIX OF NORMAL IS SINGULAR THEN NO INTERSECTION.
230 GO SUB INV
240 PRINT AT 12,4;"POINT OF INTERSECTION"
250 IF SINGULAR THEN PRINT AT 12,0;"NO": STOP
259 REM POINT OF INTERSECTION IS (B(1),B(2),B(3)).
260 FOR i=1 TO 3
270 LET B(I)=0
280 FOR J=1 TO 3
290 LET B(I)=B(I)+M(I,J)*K(J)
300 NEXT J
310 IF ABS B(I)<0.000001 THEN LET B(I)=0
320 PRINT AT 12+2*1,7;"B(";I;") = ";B(I)
330 NEXT I
340 STOP
```

Пример 7.Д

Найдите точку пересечения трёх плоскостей (0,1,1)*X=2, (1,2,3)*X=4 и (1,1,1)*X=0

В матричной форме

$$\begin{matrix}0 & 1 & 1\\ 1 & 2 & 3\\ 1 & 1 & 1\end{matrix} \times \begin{matrix}X\\ Y\\ Z\end{matrix} = \begin{matrix}2\\ 4\\ 0\end{matrix}$$

Матрица, обратная к

$$\begin{matrix}0 & 1 & 1\\ 1 & 2 & 3\\ 1 & 1 & 1\end{matrix}$$

есть

$$\begin{matrix}-1 & 0 & 1\\ 2 & -1 & 1\\ -1 & 1 & -1\end{matrix}$$

и следовательно

$$\begin{matrix}X\\ Y\\ Z\end{matrix} = \begin{matrix}-1 & 0 & 1\\ 2 & -1 & 1\\ -1 & 1 & -1\end{matrix} * \begin{matrix}2\\ 4\\ 0\end{matrix} = \begin{matrix}-2\\ 0\\ 2\end{matrix}$$

Это решение легко проверить:

(0,1,1)*(-2,0,2) =2,
(1,2,3)*(-2,0,2) =4,
(1,1,1)*(-2,0,2) =0,

что означает: точка (-2,0,2) принадлежит всем трём плоскостям и, следовательно, является их точкой пересечения.

Линия пересечения двух плоскостей

Пусть две плоскости задаются уравнениями

P*X=(P1,P2,P3)*X=K1
Q*X=(Q1,Q2,Q3)*X=K2

Мы полагаем, что плоскости не параллельны и, следовательно, P не- LMD*Q для любого LMD. Прямая, лежащая в обеих плоскостях должна быть перпендикулярна обеим нормалям к этим плоскостям. Это означает, что направление прямой D=P*Q и прямая может быть записана в виде B+MU*Q, где B - любая точка прямой. Для задания прямой мы должны найти этот вектор B. Мы находим точку пересечения этих двух плоскостей третьей, которая не параллельна ни одной из них и не пересекает их по общей прямой. Плоскость с нормалью P*Q удовлетворяет этим условиям (помните, что мы уже вычислили это произведение). Нам остается задать K3. Выберем K3=0, так, что третья плоскость проходит через начало координат. Тогда вектор B задается выражением

$$B = \begin{pmatrix} P1 & P1 & P1 \\ Q1 & Q2 & Q3 \\ P2*Q2-P3*Q3 & P3*Q1-P1*Q3 & P1*Q2-Q1*P2 \end{pmatrix}^{-1} \times \begin{pmatrix} K1 \\ K2 \\ 0 \end{pmatrix}$$

Пример 7.Е

Найдите прямую пересечения плоскостей (0,1,1)*X=2 и (1,2,3)*X=2. Так как P=(0,1,1) и Q=(l,2,3), то P*A=(1*3-1*2,1*1-0*3,0*2-1*1)=(1,1,-1). Обратная матрица

$$\begin{pmatrix} 0 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 1 & -1 \end{pmatrix}^{-1} = \frac{1}{3} \times \begin{pmatrix} -5 & 2 & 1 \\ 4 & -1 & 1 \\ -1 & 1 & -1 \end{pmatrix}$$

следовательно, точка пересечения трех плоскостей есть

$$\frac{1}{3} \times \begin{pmatrix} -5 & 2 & 1 \\ 4 & -1 & 1 \\ -1 & 1 & -1 \end{pmatrix} \times \begin{pmatrix} 2 \\ 2 \\ 0 \end{pmatrix} = \frac{1}{3} \times \begin{pmatrix} -6 \\ 0 \\ 0 \end{pmatrix} = \begin{matrix} -2 \\ 2 \\ 0 \end{matrix}$$

и прямая есть (-2,2,0)+MU*(1,1,-1).

Ответ легко проверить, так как все точки прямой должны удовлетворять уравнению обеих плоскостей

(0,1,1)*((-2,2,0)+MU(1,1,-1))=(0,1,1)*(-2,2,0)
MU(0,1,1)*(1,1,-1)=2

для всех MU и

(1,2,3)*((-2,2,0)+MU(1,1,-1))=(0,1,1)*(-2,2,0)

MU(1,2,3)*(1,1,-1)=2

для любых MU.

Программа для решения этой задачи приводится на листинге 7.7. Обратите внимание, что она очень похожа на предыдущую программу. Обратите внимание, что для P и Q не выделяются специальные массивы, они хранятся в первых двух строках матрицы N. В массиве B хранится значение базового вектора прямой, значение направляющего вектора прямой D хранится в третьей строке матрицы N.

LISTING 7.7

```
100 REM LINE OF INTERSECTION OF TWO PLANES
110 DIM N(3,3): DIM M(3,3): DIM K(3): DIM B(3): DIM A$(2)
120 LET INV=500: LET A$="PQ"
129 REM INPUT DATA OF TWO PLANES.
130 PRINT AT 2,4;"COEFFICIENTS CONSTANT"
140 FOR I=1 TO 2
150 PRINT AT 2+2*I,0;"PLANE(";I;")=(   ,   ,   )"
160 FOR J=1 TO 3
170 LET I$="INPUT "+A$(I)+"("+STR$ J+") "
180 INPUT (I$);N(I,J): PRINT AT 2+2*I,7+4*J;N(I,J)
190 NEXT J
200 LET I$="INPUT K("+STR$ I+")"
210 INPUT (I$);K(I): PRINT AT 2+2*I,28;K(I)
220 NEXT I
229 REM FOR THIRD PLANE.
230 LET N(3,1)=N(1,2)*N(2,3)-N(1,3)*N(2,2)
240 LET N(3,2)=N(1,3)*N(2,1)-N(1,1)*N(2,3)
250 LET N(3,3)=N(1,1)*N(2,2)-N(1,2)*N(2,1)
260 LET K(3)=0
269 REM IF MATRIX OF NORMAL IS SINGULAR THEN NO  INTERSECTION.
270 GO SUB INV
280 PRINT AT 10,4;"LINE OF INTERSECTION"
290 IF SINGULAR THEN PRINT AT 10,0;"NO": STOP
300 PRINT AT 12,0;"BASE VECTOR";"            DIRECTION"
309 REM LINE OF INTERSECTION - BASE (B(1),B(2),B(3))) AND DIRECTION
(N(3,1),N(3,2),N(3,3)).
310 FOR I=1 TO 3
320 LET B(I)=0
330 FOR J=1 TO 3
340 LET B(I)=B(I)+M(I,J)*K(J)
350 NEXT J
360 IF ABS B(I)<0.000001 THEN LET B(I)=0
370 PRINT AT 12+2*I,0;"B(";I;") - ";B(I)
380 PRINT AT 12+2*I,20;"D(";I;") - ";N(3,I)
390 NEXT I
400 STOP
```

Функциональное представление поверхности

Мы видели, что в двумерном случае кривые можно задавать с помощью функциональных зависимостей. Этот метод можно распространить на случай трёх измерений для исследования поверхностей в пространстве. Простейшей формой поверхности является плоскость с нормалью N=(N1,N2,N3), которая, как мы знаем, задается уравнением:

N*X-K=N1*X1+N2*Y+N3*Z-K=0,

которое можно переписать в функциональной форме

F(X)=F(X,Y,Z)=N1*X+N1*Y+N*Z-K=N*X-K,

где **X=(X,Y,Z)**.

Это простое выражение позволяет нам разбить пространство на три множества, множество точек X, таких, что F(X)<0 - отрицательная область, множество точек X, лежащих на плоскости таких, что F(X)=0 и множество точек X, таких, что F(X)>0 - положительная область.

Если поверхность разбивает пространство на две связные области (область называется связной, если любые две точки области можно соединить кривой, целиком лежащей в этой области), то эти области можно отождествлять с областью положительных и отрицательных значений. Следует иметь в виду, что многие поверхности делят пространство на большое число компонент; примером такой поверхности является поверхность, задаваемая уравнением F(X,Y,Z)=COS(Y)-SIN(X**2+Z**2). Однако, имеется ряд поверхностей, удовлетворяющих требуемому условию. Примером такой поверхности является сфера

F(X)=R2-ABS(X)**2**

или по компонентам:

F(X,Y,Z)=R2-X**2-Y**2-Z**2**

Если P(X)=0, то X лежит на сфере; если F(X)<0, то точка лежит вне сферы; если F(X)>0, то точка лежит внутри сферы.

Функциональное представление поверхностей весьма полезно при нахождении точек пересечения

поверхностей. Однако, это представление особенно ценно при выяснении того, лежат ли две точки P и Q по одну сторону от поверхности. Для этого достаточно сравнить знаки F(P) и F(Q); если они одного знака, то точки P и Q лежат по одну сторону от поверхности, а если разного - то по разные, что означает, что любая кривая, соединяющая P и Q, пересекает поверхность хотя бы в одной точке. Ниже приведен пример, иллюстрирующий вышесказанное.

Лежит ли точка по ту же сторону от плоскости, что и начало координат?

Предположим, что плоскость задана тремя неколлинеарными точками. Тогда уравнение плоскости:

```
((P1-P2)*(P3-P1))*(X-P1)=0
```

Это уравнение можно переписать в функциональном виде

```
F(X)=((P2-P1)*(P3-P1))*(X-P1)
```

Все, что нам надо сделать, это сравнить значение F(0) с F(E), где 0 – начало координат, а E - выбранная нами точка. Мы предполагаем, что ни E, ни 0 не лежат в плоскости.

Этот метод является чрезвычайно полезным при исследовании алгоритмов скрытых линий.

Пример 7.Ж

Лежат ли точка (1,1,3) и начало координат по одну сторону от плоскости, проходящей через точки (0,1,1), (1,2,3) и (-2,3,-1)?

Из примера 7.Г мы видим, что функция, определяющая плоскость, есть:

```
F(X)=((-6,-2,4)*(X-(0,1,1))
```

Отсюда

```
F(0,0,0)=-(-6,-2,4)*(0,1,1)=-2
F(1,1,3)=(-6,2,4)*(1,1,3)-(0,1,1))=2
```

Мы видим, что начало координат и точка (1,1.3) лежат по разные стороны от плоскости, так что отрезок, соединяющий эти две точки, пересечет плоскость в точке (1-MU)*(0,0,0)+MU(1,1,3), где 0<MU<1.

Каково направление обхода двумерного многоугольника, заданного последовательностью своих вершин?

Начнём со случая треугольника, заданного вершинами P1=(X1,Y1), P2=(X2,Y2), P3=(X3,Y3). Хотя эти точки заданы на двумерной плоскости, мы можем считать их лежащими в пространстве, полагая, что они принадлежат плоскости X/Y и полагая, что координата Z этих точек равна нулю. Направление сторон треугольника есть (P2-P1), (P3-P2), (P1-P3). Так как эти прямые лежат в X/Y плоскости, то для каждого I=1,2,3 существует действительное R(I) такое, что

```
(P(I+1)-P(I))*(P(I+2)-P(I+1))=(0,0,R(I))
```

Это справедливо потому, что вектор, перпендикулярный плоскости X/Y, имеет одну отличную от нуля координату - Z. Сложение в индексах считается по модулю 3. Так как считается, что точки многоугольника задаются в последовательности обхода многоугольника, то знаки всех R(I) совпадают. Более того, если этот знак положителен, то обход производится против часовой стрелки, а если отрицателен - то по часовой стрелке.

Для заданного многоугольника достаточно проверить три точки для того, чтобы установить последовательность обхода многоугольника. Этот метод окажется незаменимым при исследовании алгоритмов скрытых прямых. На листинге 7.8 приведена программа, устанавливающая порядок обхода трех точек.

Пример 7.И

Почему многоугольник, приведенный в примере 3.Г, обходится против часовой стрелки? Вершины этого многоугольника есть (1,0,0), (5,2,0), (4,4,0) и (-2,1,0). Направление сторон - (4,2,0), (-1,2,0), (-6,-3,0), (3.-1.0). Тогда:

```
(4,2,0)*(-1,2,0)     =(0,0,10)
(-1,2,0)*(-6,-3,0)   =(0,0,15)
(-4,-3,0)*(3,-1,0)   =(0,0,15)
(3,-1,0)*(4,2,0)     =(0,0,10)
```

Мы видим, что все R(I) положительны и, следовательно, многоугольник обходится против часовой стрелки. Будьте внимательны, соблюдая последовательность обхода, так как нарушение последовательности может привести к неверному результату, например:

(-6,-3,0)*(4,2,0)=(0,0,0) - прямые параллельны! или

(-1,2,0)*(3,-1,0)=(0,0,-5) - мы пропустили вершину.

LISTING 7.8

```
100 REM ORIENTATION OF 2-D TRIANGLE
110 DIM X(3): DIM Y(3)
120 LET K$="TYPE IN ": LET J$="COORDINATES OF TRIANGLE:"
130 FOR I=1 TO 3
140 LET I$="VERTEX ("+STR$ I+") - ("
150 INPUT (K$+J$+I$);X(I);",";Y(I);")"
160 PRINT AT 2+2*1,0;I$;X(I);",";Y(I);")"
170 NEXT I
180 PRINT AT 12,0;"THE TRIANGLE IS ";
188 REM (DX1,DY1) IS 2-D DIRECTOIN VECTOR JOINING POINT 1 TO POINT 2.
189 REM (DX2,DY2) IS 2-D DIRECTION VECTOR JOINING POINT 2 TO POINT 3.
190 LET DX1=X(2)-X(1): LET DY1=Y(2)-Y(1)
200 LET DX2=X(3)-X(2): LET DY2=Y(3)-Y(2)
209 REM USE 3-D VECTOR PRODUCT TO CHECK ON ORIENTATION OF TRIANGLE.
210 IF DX1*DY2-DX2*DY1>0 THEN PRINT "ANTI-";
```

```
220 PRINT "CLOCKWISE"
230 STOP
```

Список программ

1. Листинг 7.1 (пересечение прямой и плоскости). Необходимая информация: базовый вектор (B(1),B(2),B(3)) и направляющий вектор (D(l),D(2),D(3)), задающие прямую; нормаль (N(1),N(2),N(3)) и константа K, задающая плоскость. В качестве входных данных попробуйте (1,2,3), (1,1,-1), (1,0,1) и 2 соответственно.

2. Листинг 7.2 (пересечение двух прямых). Необходимая информация: базовые векторы и направляющие векторы этих прямых: (B(1),B(2),B(3)), (D(1),D(2),D(3)) и (C(1),C(2),C(3)), (E(1),E(2),E(3)). Попробуйте (1,2,3), (1,1,-1) и (-1,1,3), (1,0,1).

3. Листинг 7.3 и 7.4 (MAIN PROGRAM "VECPROD" - "векторное произведение", "DOTPROD" - "скалярное произведение"). Входные данные: два вектора (L(l),L(2),L(3)) и (M(1),M(2),M(3)). Попробуйте (1,2,3), (1,1,-1).

ГЛАВА 8
Матричное представление преобразования трехмерного пространства

Представление преобразования переноса, вращения, растяжения (отражения) трёхмерного пространства с помощью матриц четыре на четыре

- Обратное преобразование
- Композиция преобразований
- Вращение вокруг произвольной оси

ВНИМАНИЕ! Тем нашим читателям, которые слабо представляют себе матрицы и операции над ними, советуем сначала внимательно прочитать главу 4 данной книги, где популярно описываются основы матричных преобразований и подробно рассматриваются вопросы матричного представления двумерного пространства.

Здесь же мы предлагаем читателю дальнейшее развитие теории матричного представления пространства уже для трёх измерений. Ниже мы приводим готовую программу умножения двух матриц для трёхмерного случая (см. листинг 8.1).

LISTING 8.1

```
9100 REM MULT3
9101 REM IN - A(4,4),R(4,4)
9102 REM OUT - R(4,4)
9110 FOR I=1 TO 4
9120 FOR J=1 TO 4
9130 LET AR=0
9140 FOR K=1 TO 4
9150 LET AR=AR+A(I,K)*R(K,J)
9160 NEXT K
9170 LET B(I,J)=AR
9180 NEXT J
9190 NEXT I
9200 FOR I=1 TO 4
9210 FOR J=1 TO 4
9220 LET R(I,J)=B(I,J)
9230 NEXT J
9240 NEXT I
9250 RETURN
9300 REM IDR3
9302 REM OUT - R(4,4)
9310 FOR I=1 TO 4
9320 FOR J=1 TO 4
9330 LET R(I,J)=0
9340 NEXT J
9350 LET R(I,I)=1
9360 NEXT I
9370 RETURN
```

Рассмотрим уравнение: 1=A*X+A*Y+A*Z+A

Его справедливость для всех X,Y означает A(4,1)=A(4,2)=A(4,3)=0 и A(4,4)=1. Эти уравнения могут быть записаны в матричной форме, где вектор-столбец, соответствующий точке "после", есть произведение матрицы на столбец "до".

$$\begin{pmatrix} X' \\ Y' \\ Z' \\ 1 \end{pmatrix} = \begin{pmatrix} A(1,1) & A(1,2) & A(1,3) & A(1,4) \\ A(2,1) & A(2,2) & A(2,3) & A(2,4) \\ A(3,1) & A(3,2) & A(3,3) & A(3,4) \\ A(4,1) & A(4,2) & A(4,3) & A(4,4) \end{pmatrix} \times \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Так что если мы запишем преобразование в виде матрицы и сохраним её, то мы можем преобразовать любые точки умножением соответствующих им вектор-столбцов на матрицы. Как и ранее, имея матричную запись преобразований, легко составить преобразование, соответствующее композиции этих преобразований. Если матрицы преобразований есть (в порядке совершения преобразований) A,B,C…L,M,N, то матрица результирующего преобразования есть N*M*L*C*B*A. Запомните последовательность. Она связана с тем, что мы оперируем с вектор-столбцами, а не вектор-строками.

Параллельный перенос

Пусть совершается параллельный перенос на вектор (TX,TY,TZ). Этому преобразованию соответствуют уравнения:

X’ = 1*X+0*Y+0*Z+TX
Y’ = 0*X-1*Y+0*Z+TY

$$Z' = 0*X-0*Y+1*Z+TZ$$

которым в свою очередь соответствует матрица

$$\begin{pmatrix} 1 & 0 & 0 & TX \\ 0 & 1 & 0 & TY \\ 0 & 0 & 1 & TZ \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Подпрограмма "TRAN3" генерирует такую матрицу по заданным параметрам TX, TY, TZ (см. листинг 8.2).

LJSTING 8.2

```
9000 REM TRAN3
9001 REM IN - TX,TY,TZ
9002 REM OUT - A(4,4)
9010 FOR I=1 TO 4
9020 FOR J=1 TO 4
9030 LET A(I,J)=0
9040 NEXT J
9050 LET A(I,I)=1
9060 NEXT I
9070 LET A(1,4)=TX: LET A(2,4)=TY: LET A(3,4)=TZ
9080 RETURN
```

Растяжение

Если масштабные множители растяжения по оси X, Y и Z есть X, Y и Z соответственно, то уравнения преобразования:

$$X'=SX*X+0*Y+0*Z+0$$
$$Y'=0*X+SY*Y+0*Z+0$$
$$Z'=0*X+0*Y+SZ*Z+0$$

и соответствующая ему матрица:

$$\begin{pmatrix} SX & 0 & 0 & 0 \\ 0 & SY & 0 & 0 \\ 0 & 0 & SZ & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

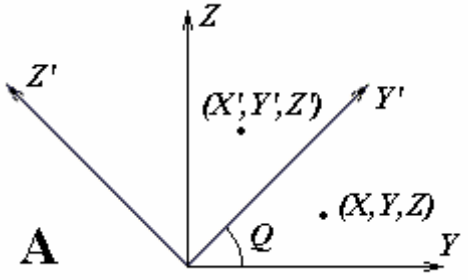
Обычно масштабные множители положительны, однако, если какой-либо из этих множителей отрицателен, то преобразование является композицией отражения и растяжения. Например, значениям SX=-1, SY=1, SZ=1 соответствует отражение относительно плоскости X/Y. Подпрограмма "SCALES" генерирует матрицу преобразования A по заданным SX, SY, SZ (см. листинг 8.3).

LISTING 8.3

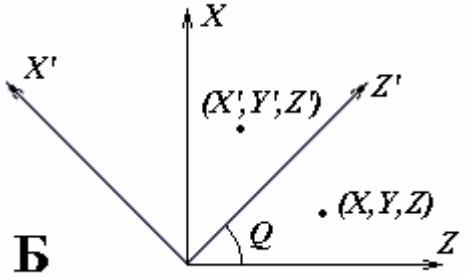
```
8900 REM SCALE3
8901 REM IN - SX,SY,SZ
8902 REM OUT - A(4,4)
8910 FOR I=1 TO 4
8920 FOR J=1 TO 4
8930 LET A(I,J)=0
8940 NEXT J
8950 NEXT I
8960 LET A(1,1)=SX: LET A(2,2)=SY: LET A(3,3)=SZ
8970 LET A(4,4)=1
8980 RETURN
```

Вращение относительно координатной оси

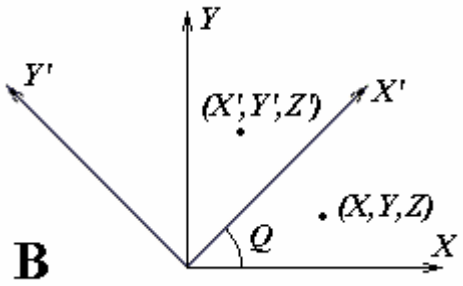
Прежде чем изучать вращение относительно оси P-MU*Q на заданный угол рассмотрим более простой случай вращения относительно координатной оси.



X – ось в страницу



Y – ось в страницу



Z – ось в страницу

Рис. 32.

(А) X-ось направлена в плоскость страницы;

(Б) Y-ось направлена в плоскость страницы;

(В) Z-ось направлена в плоскость страницы.

(А) Вращение на угол TETA относительно оси X

На рис. 32А ось вращения направлена перпендикулярно плоскости страницы (положительной полуосью от читателя); на рисунке показана точка (X',Y',Z'), в которую переходит точка (X,Y,Z) в результате вращения. Мы видим, что вращение сводится к вращению в двумерной плоскости, что означает, что X-координаты точки при таком преобразовании не меняются. Используя методы главы 4, мы получим уравнение:

```
X'  =  X
Y'  =  COS TETA * Y - SIN TETA * Z
Z'  =  SIN TETA * Y + COS TETA * Z,
```

которому соответствует матрица:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & COS\ \ TETA & -SIN\ \ TETA & 0 \\ 0 & SIN\ \ TETA & COS\ \ TETA & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

(Б) Вращение на угол TETA относительно оси Y

На рисунке 32Б показано вращение плоскости на угол Q относительно оси Y, положительная полуось которой направлена от читателя и перпендикулярна плоскости страницы. Это вращение задается уравнением:

```
X'  =  SIN TETA *Z + COS TETA *X
Y'  =  Y
Z'  =  COS TETA * Z - SIN TETA * X,
```

и соответствующая этому преобразованию матрица:

$$\begin{pmatrix} COS\ \ TETA & 0 & SIN\ \ TETA & 0 \\ 0 & 1 & 0 & 0 \\ -SIN\ \ TETA & 0 & COS\ \ TETA & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

(В) Вращение на угол TETA относительно оси Z

Согласно рисунку имеем:

```
X'  =  COS TETA * X - SIN TETA * Y
Y'  =  SIN TETA * X + COS TETA * Y
Z'  =  Z
```

и матрица:

$$\begin{pmatrix} COS\ \ TETA & -SIN\ \ TETA & 0 & 0 \\ SIN\ \ TETA & COS\ \ TETA & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Подпрограмма "ROT3" генерирует необходимую матрицу по значениям двух параметров - углу THETA и номеру оси AXIS (1 - соответствует оси X, 2 - оси Y, 3 - оси Z). Программа приведена на листинге 8.4.

LISTING 8.4

```
8600 REM ROT3
8601 REM IN - THETA,AXIS
8602 REM OUT - A(4,4)
8610 FOR I=1 TO 4
8620 FOR J=1 TO 4
8630 LET A(I,J)=0
```

```
8640 NEXT J
8650 NEXT I
8660 LET A(4,4)=1: LET A(AXIS,AXIS)=1
8670 LET AX1=AXIS+1: IF AX1=4 THEN LET AX1=1
8680 LET AX2=AX1+1: IF AX2=4 THEN LET AX2=1
8690 LET CT=COS THETA: LET ST=SIN THETA
8700 LET A(AX1,AX1)=CT: LET A(AX2,AX2)=CT
8710 LET A(AX1,AX2)=-ST: LET A(AX2,AX1)=ST
8720 RETURN
```

Обратное преобразование

Прежде чем изучать общий вид вращения трёхмерного пространства, мы ознакомимся с понятием обратного преобразования. Обратное преобразование возвращает точки, переведенные заданным преобразованием, в исходные. Если преобразование задается матрицей A, то обратное преобразование задается матрицей A**(-1), обратной к A. Нам не потребуется явное вычисление обратной матрицы такими методами, как, например, присоединенный метод (листинг 7.5). Мы можем использовать программы на листингах 8.2, 8.3, 8.4 с параметрами, полученными из значений параметров исходного преобразования.

1) Параллельный перенос на вектор (TX,TY,TZ), обратное преобразование - параллельный перенос на вектор (-TX,-TY,-TZ);

2) Растяжение на SX, SY, SZ; обратное преобразование - растяжение на 1/SX, 1/SY, 1/SZ;

3) Поворот на угол THETA относительно оси; обратное преобразование – поворот на угол THETA относительно той же оси;

4) Если преобразование является композицией нескольких преобразований параллельного переноса, вращения, растяжения матриц A*B*C*…*L*M*N, тогда обратное преобразование есть $N^{-1}*M^{-1}*L^{-1}*\ldots*C^{-1}*B^{-1}*A^{-1}$.

*Поворот на угол NU относительно произвольной оси P+MU*Q*

Положим P=(PX,PY,PZ), Q=(QX,QY,QZ). Разобьём задачу на следующие этапы:

А) Преобразуем, точки пространства так, чтобы ось вращения проходила через начало координат. Это достигается параллельным переносом на вектор (-PX,-PY,-PZ); матрица этого преобразования E генерируется вызовом процедуры "TRAN3" со значением параметров (-PX,-PY,-PZ). Обратная к этой матрице, матрица E, генерируется вызовом процедуры "TRAN3" со значением параметров (PX,PY,PY).

$$E=\begin{pmatrix}1&0&0&-PX\\0&1&0&-PY\\0&0&1&-PZ\\0&0&0&1\end{pmatrix}\qquad E^{-1}=\begin{pmatrix}1&0&0&PX\\0&1&0&PY\\0&0&1&PZ\\0&0&0&1\end{pmatrix}$$

Б) Теперь повернем точки пространства на угол ALPHA, где ALPHA=ARCTG (QY/QX); этому преобразованию соответствует матрица G, которая генерируется вызовом подпрограммы "ROT3" со значением параметров TETA=-ALPHA, AXIS=3. После выполнения такого преобразования ось вращения лежит в плоскости X/Z и проходит через точку (Y,0,QZ).

$$G=\frac{1}{NU}\begin{pmatrix}QX&QY&0&0\\-QY&QX&0&0\\0&0&NU&0\\0&0&0&NU\end{pmatrix}$$

где NU>0: NU**2=QX**2+QY**2

В) Теперь повернем точки плоскости относительно оси Y на угол -BETA, где BETA=ARCTG(NU/QZ); соответствующую матрицу обозначим H. Эта матрица генерируется вызовом "ROT3" со значением параметров AXIS=2 и THETA=BETA.

$$H=\frac{1}{W}\begin{pmatrix}QZ&0&-NU&0\\0&W&0&0\\NU&0&QZ&0\\0&0&0&1\end{pmatrix}\qquad H^{-1}=\frac{1}{W}\begin{pmatrix}QZ&0&NU&0\\0&W&0&0\\-NU&0&QZ&0\\0&0&0&1\end{pmatrix}$$

где W задается соотношением:

W**2=NU**2+QZ**2=QX**2+QY**2+QZ**2. Точка (NU,0,QZ) переводится в точку (0,0,W) и, следовательно, ось вращения теперь совпадает с осью Z.

Г) Теперь мы можем повернуть точки пространства на угол GAMMA относительно оси вращения, используя матрицу W, сгенерированную вызовом "ROT3" (со значением

$$W=\begin{pmatrix} COS(GAMMA) & -SIN(GAMMA) & 0 & 0 \\ SIN(GAMMA) & COS(GAMMA) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

AXIS=3 и THETA=GAMMA).

Д) Установим ось вращения в исходное положение с помощью преобразований с матрицами H^{-1}, G^{-1} и, наконец, F^{-1}.

В окончательном виде матрица P вращения точек пространства относительно оси P+M*Q на угол GAMMA задается в виде

$P=F^{-1}*G^{-1}*H^{-1}*W*H*F.$

В некоторых случаях эти матрицы равны тождественной и не влияют на вид преобразования. Например, если ось вращения проходит через начало координат, то матрицы F и F^{-1} равны тождественной, и ими можно пренебречь.

Программа "GENROT" (листинг 8.5) вычисляет матрицу вращения P по заданным входным параметрам GAMMA, (PX,PY,PZ) и (QX.QY.QZ).

LISTING 8.5

```
5800 REM GENROT
5801 REM IN - PX,PY,PZ, QX,QY,QZ, GAMMA,R(4,4)
5802 REM OUT - R(4,4)
5809 REM PLACE ORIGIN ON AXIS OF ROTATION.
5810 LET TX=PX: LET TY=PY: LET TZ=-PZ: GO SUB TRAN3: GO SUB MULT3
5819 REM ROTATE AXIS OF ROTATION INTO X/Z PLANE.
5820 LET AX=QX: LET AY=QY: GO SUB ANGLE
5830 LET ALPHA=THETA: LET THETA=-THETA: LET AXIS=3: GO SUB ROT3: GO SUB MULT3
5839 REM ROTATE AXIS OF ROTATION INTO Z-AXIS.
5840 LET AX=QZ: LET AY=SQR(QX*QX+QY*QY): GO SUB ANGLE
5850 LET BETA=THETA: LET THETA=-THETA: LET AXIS=2: GO SUB ROT3: GO SUB MULT3
5859 REM ROTATE BY GAMMA ABOUT AXIS OF ROTATION.
5860 LET THETA=GAMMA: LET AXIS=3: GO SUB ROTS: GO SUB MULT3
5869 REM REPLACE AXIS BACK TO ORIGINAL POSITION.
5870 LET THETA=BETA: LET AXIS=2: GO SUB ROT3: GO SUB MULT3
5880 LET THETA=ALPHA: LET AXIS=3: GO SUB ROT3: GO SUB MULT3
5890 LET TX=PX: LET TY=PY: LET TZ=PZ: GO SUB TRAN3: GO SUB MULT3
5900 RETURN
```

Пример 8.A

Куда перейдут точки (0,0,0), (1,0,0), (0,1,0) и (0,0,1), (1,1,1) при повороте на PI/4 относительно оси (1,0,1) (3,4,5).

Используя изложенную выше теорию, получим:

$$F=\begin{pmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad F^{-1}=\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$G=\frac{1}{5}\begin{pmatrix} 3 & 4 & 0 & 0 \\ -4 & 3 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 5 \end{pmatrix} \qquad G^{-1}=\frac{1}{5}\begin{pmatrix} 3 & -4 & 0 & 0 \\ 4 & 3 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 5 \end{pmatrix}$$

$$H=\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & 2 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix} \qquad H^{-1}=\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}$$

$$W=\frac{1}{2}\begin{pmatrix} 1 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}$$

где $P=F^{-1}*G^{-1}*H^{-1}*W*H*G*F$ - матрица, соответствующая искомому преобразованию. Умножая вектор-столбцы,

соответствующие (0,0,0), (1,0,0), (0,0,1), (1,1,1), на матрицу P, переводя получившиеся столбцы в строчную форму и вынося фактор 1/50*SQRT(2), получим новые координаты (-26+6SQRT2, 32-42SQRT2, -10+30SQRT2), (15+15SQRT2, 20-5SQRT2, -25+25SQRT2), (-38-7SQRT2, 66-26SQRT2, -3+65SQRT2), (-41+41SQRT2, 12-37SQRT2, 15+55SQRT2), (12+37SQRT2, 34+16SQRT2, -20+85SQRT2) соответственно.

Естественно, что вращение не меняет взаимного расположения точек; в частности, углы между прямыми не изменяются (что неверно для растяжения, которое в общем случае меняет взаимное расположение прямых). В исходном положении направления от (0,0,0) к (1,0,0), (0,1,0), (0,0,1) взаимно перпендикулярно (что означает равенство нулю скалярного произведения). Скалярное произведение преобразованных направлений должно быть равным нулю; действительно, три направляющих вектора (с точностью до множителя 1/50SQRT2) есть (41+9SQRT2, -12+37SQRT2, -15-5SQRT2), (-12-13SQRT2, 34+16SQRT2, -20+35SQRT2) и (-15+35SQRT2, -20+5SQRT2, 25+25SQRT2) и скалярное произведение любой пары равно нулю.

Аналогично, скалярное произведение направляющего вектора (1,1,1) с любым из направлений, перечисленных выше, равно 1. То же верно и для направлений после преобразования: четвёртое направление - (14+31SQRT2, 2+58SQRT2, -10+55SQRT2), а скалярное произведение с любым из трех направлений есть 5000, что после деления на (59SQRT2) даёт 1.

Программа, которая считывает параметры, задающие ось вращения (PX,PY,PZ)+M(QX,QY,QZ) и угол вращения GAMMA и поворачивает точку (XX,YY,ZZ), приведена на листинге 8.6.

LISTING 8.6

```
100 REM ROTATION OF A POINT ABOUT A GIVEN AXIS
110 DIM A(4,4): DIM B(4,4): DIM R(4,4)
120 DIM P(3): DIM A$(8)
130 INPUT "BASE VECTOR OF AXIS ","(";PX;",";PY;",";PZ;")"
140 PRINT AT 1,0;"BASE VECTOR OF AXIS","(";PX;",";PY;",";PZ;")"
150 INPUT "DIRECTION VECTOR OF AXIS ","(";QX;",";QY;",";QZ;")"
160 PRINT AT 4,0;"DIRECTION VECTOR OF AXIS","(";QX;",";QY;",";QZ;")"
170 INPUT "ANGLE OF ROTATION ";GAMMA
180 PRINT AT 7,0;"ANGLE OF ROTATION ";GAMMA
190 LET MULT3=9100: LET IDR3=9300: LET ROT3=8600: LET TRAN3=9000: LET
ANGLE=8800: LET GENROT=5800
199 REM CALCULATE R(4,4) FOR ROTATING POINT BY ANGLE GAMMA ABOUT AN AXIS WITH
BASE VECTOR (PX,PY,PZ) AND DIRECTION VECTOR (QX,QY,QZ)
200 GO SUB IDR3: GO SUB GENROT
210 PRINT AT 14,0;"BECOMES"
219 REM INPUT AND TRANSFORM POINT (XX,YY,ZZ).
220 INPUT "POINT VECTOR ","(";XX;",";YY;",";ZZ;")"
230 PRINT AT 12,0;"POINT (";XX;",";YY;",";ZZ;")",,
240 LET P(1)=XX*R(1,1)+YY*R(1,2)+ZZ*R(1,3)+R(1,4)
250 LET P(2)=XX*R(2,1)+YY*R(2,2)+ZZ*R(2,3)+R(2,4)
260 LET P(3)=XX*R(3,1)+YY*R(3,2)+ZZ*R(3,3)+R(3,4)
269 REM TIDY UP OUTPUT.
270 PRINT AT 16,0;" (";
280 FOR I=1 TO 3: LET A$=STR$ P(I): FOR J=1 TO 8
290 IF A$(J)<>" "THEN PRINT A$(J);
300 NEXT J: IF I<>3 THEN PRINT",";
310 NEXT I: PRINT ")",,
320 GO TO 220
```

Упражнение 8.1

Поэкспериментируйте с этими идеями. Вы всегда можете проверить правильность своего решения, выбрав значения, для которых легко найти решение. Для дальнейшего важно, чтобы вы освоились в совершенстве с матричным исчислением, а это достигается только опытом. Сначала вас будут преследовать арифметические ошибки, но через некоторое время вы привыкнете думать о преобразованиях в терминах матриц, что значительно облегчает изучение способов изображения трехмерных объектов.

Упражнение 8.2

Заметим, что, так же как и в двумерном случае, нижняя строка любой матрицы, соответствующей преобразованию трёхмерного пространства, всегда есть (0,0,0,1) и не используется при вычислениях. Эта строка позволяет нам представлять преобразования в виде квадратных матриц, а для квадратных матриц, в свою очередь, определить операцию умножения. Мы можем изменить это определение с тем, чтобы перемножать матрицы и умножать вектор-столбцы на матрицы, используя только верхние три строки матриц 4×4 (в главе 4 мы использовали в программах, приведенных на листингах 4.2A, 4.3A, 4.4A, 4.5A, две верхние строки матриц 3×3). Измените листинги 8.1, 8.2, 8.3 и 8.4 с учётом нового определения.

Упражнение 8.3

Подпрограмма "ROT3" вычисляет величину THETA, вызывая подпрограмму "ANGLE", входными параметрами которой являются величины AX и AY. Подпрограмма "ROT3" вычисляет косинус и синус угла THETA, но мы знаем, что последние равны AX/SQRT(AX**2+AY**2) и AY/SQRT(AX**2+AY**2) соответственно.

Напишите другую подпрограмму, осуществляющую вращение "ROTXY", которая вычисляла бы матрицу вращения непосредственно AX и AY, не обращаясь к подпрограмме "ANGLE".

Обратите также внимание на то, что в начале своей работы подпрограммы "ROT", "TRAM", "SCALE", и "IDR" очищают массив. SPECTRUM производит эту операцию быстрее, если используется команда DIM. Кроме того, зачастую непосредственные присвоения элементам массива значений выполняется быстрее, чем присвоения в циклах FOR…NEXT.

Упражнение 8.4

Ещё раз отметим, что некоторые авторы предпочитают использовать вектор-строки вместо вектор-столбцов, что приводит к иному порядку сомножителей при композиции преобразований. Наши обозначения имеют то преимущество, что элементы матрицы есть в точности коэффициенты уравнения, определяющего преобразования.

Полезно также помнить и о другом способе представления преобразований; используйте его и перепишите все подпрограммы этой главы, а также оставшейся части книги. Помните, что не важно, какой метод использовать, важно только остановить свой выбор на одном из них раз и навсегда. Мы используем вектор-столбцы потому, что считаем его наиболее подходящим на ранней стадии обучения.

Список программ

1. Все листинги в этой главе, 8.1 ("MULT3" и "IDR3"), 8.2 ("TRAN3"), 8.3 ("SCALES"), 8.4 ("ROT3"), 8.5 ("GENROT"), 8.6 (MAIN PROGRAM) и листинг 3.4 ("ANGLE"). Входные данные: базисный вектор (PX,PY,PZ) и направляющий вектор (QX,QY,QZ) оси вращения, угол, на который производится вращение - GAMMA, а также любой вектор с тремя компонентами - (XX.YY.ZZ). Введите, например, (0,0,0), (1,1,1) и PI/4иточки (1,0,1), (1,1,1), (1,2,3).

ГЛАВА 9
Ортогональные проекции

Теперь мы можем перейти к проблеме изображения трёхмерных объектов на нашем двумерном экране. Простейший метод, описанный здесь, является естественным обобщением метода главы 4, в которой рассматривается изображение двумерных объектов. Нам опять придется использовать до трёх положений объекта. Для иллюстрации этих идей мы сначала даём сжатое описание метода, а затем приводим множество поясняющих примеров. Мы начинаем с введения произвольной, но фиксированной тройки осей в пространстве, которую мы называем абсолютной системой. После этого мы рассмотрим 3 положения предмета: (1) исходное, (2) фактическое, (3) наблюдаемое.

(1) Исходное положение

Большая часть рисунков состоит из простых предметов, расположенных в пространстве определенным образом. Вычислять координаты каждой вершины и вносить их в программу является неэффективным. Вместо этого мы вначале задаем информацию об объекте в абсолютной системе координат, располагая его наиболее удобным образом - как правило, возле начала координат. В эту информацию входит: информация о вершинах (X-координата, Y-координата, Z-координата), возможно, информация о линиях (соединяющих вершины) или (после того как мы познакомимся с алгоритмами удаления невидимых линий и поверхностей) информация о видимых поверхностях, которые являются плоскими многоугольниками, ограниченными заданными линиями. Положение объекта, в котором он определяется, называется исходным. Нам может потребоваться дополнительная информация об объекте, например, его цвет.

(2) Фактическое положение

Мы используем развитую в предыдущих главах матричную технику для вычисления матрицы, переводящей объект из положения исходное в положение фактическое; вычисления идут в системе абсолютных координат. Обозначим эту матрицу P.

(3) Наблюдаемое положение

Понятие наблюдения предполагает наличие наблюдателя (глаза наблюдателя и, заметьте, не только глаза!), находящегося в точке (EX, EY, EZ) относительно системы координат абсолютная, который смотрит в направлении, задаваемом любой точкой, находящейся на прямой, определяющей направление взгляда. Кроме того, голова может быть наклонена, но об этом позже. Воспринимаемое глазом изображение объекта есть не что иное, как проекция вершин, линий и поверхностей на двумерную плоскость (визуальную плоскость), перпендикулярную линии взгляда. Для того, чтобы строить подобные проекции, нам необходимо разработать соответствующие методы. Мы используем матричный метод для преобразования точек пространства таким образом, чтобы глаз располагался в начале координат, а направление взгляда было параллельно оси Z. Мы будем называть такое положение наблюдаемым, а матрицу, переводящую фактическое положение в наблюдаемое, мы всюду в этой книге обозначаем Q. Метод вычисления матрицы Q будет приведен позже, а пока мы считаем, что глаз наблюдателя уже находится в начале координат, а взгляд направлен параллельно оси Z: в этом простейшем случае матрица Q равна тождественной.

Когда все точки пространства переведены в наблюдаемое положение, то визуальной плоскостью является плоскость, параллельная плоскости X/Y. Поместив глаз в параллельное положение, мы готовы к построению проекции объекта на визуальную плоскость. Но обратите внимание, что мы пока что не определили ни положение плоскости проекции (нам задана только её нормаль), ни тип проекции трёхмерного пространства в двухмерное. Эти два требования взаимосвязаны. В данной книге рассматриваются два типа проекций: в дальнейших главах мы познакомимся с перспективной проекцией, а пока ограничимся случаем ортогональной проекции.

Ортогональная проекция

Это простейший вид проекций, и в этом случае положение визуальной плоскости не играет роли, лишь бы она была перпендикулярна направлению взгляда. После перевода объекта в наблюдаемое положение, визуальной плоскостью может служить любая плоскость, параллельная плоскости X/Y, задаваемой уравнением Z=0. Для простоты мы выбираем в качестве визуальной плоскости плоскость, проходящую через начало координат. Для того, чтобы получить проекцию вершины объекта на визуальную плоскость, следует всего лишь отбросить координату Z этой вершины. Таким образом, любые две точки объекта в наблюдаемом положении с координатами (X,Y,Z), (X',Y',Z'), (Z=Z'), проектируются в одну и ту же точку визуальной плоскости с координатами (X,Y,Z). Используя методы главы 2, мы ставим в соответствие с точками X/Y плоскости точки системы координат, связанной с экраном (как правило, начало координат этой системы находится в центре экрана). Как только вершины спроецированы на визуальную плоскость, мы можем построить проекции линий и поверхностей объекта. Последние находятся в таком же отношении к спроецированным вершинам, в каком они были в самом объекте.

Прежде чем перейти к общему случаю, продемонстрируем сказанное на простом примере.

Пример 9.A

Используйте изложенный выше метод для построения проекции куба. Подобные фигуры, по очевидным причинам называются каркасными фигурами.

В исходном положении куб можно задавать с помощью восьми вершин (1,1,1), (1,1,-1), (1,-1,-1), (1,-1,1), (-1,1,1), (-1,1,-1), (-1,-1,-1), (-1,-1,1); эти вершины нумеруются числами от 1 до 8. Двенадцать линий, составляющих каркас куба, соединяют вершины 1 и 2, 2 и 3, 3 и 4, 4 и 1, 5 и 6, 6 и 7, 7 и 8, 8 и 1, 1 и 5, 2 и 6, 3 и 7, 4 и 8.

Рассмотрим простейший случай проекции куба - даже матрица, переводящая куб из исходного положения в фактическое, тождественна единичной. В результате проекции мы получили квадрат: параллельные ребра куба проецируются в одну точку.

На рисунке 33 приведено изображение куба, переведенного в фактическое положение при помощи следующих трёх преобразований:

1. Поворота относительно оси Z на ALFA=-0.92729518 рад - матрица A. Величина ALFA выбиралась с тем, чтобы COS(ALFA)=3/5, а SIN(ALFA)=-4/5, и матрица поворота имеет простой вид;
2. Переноса на вектор (-1,0,0) - матрица B;
3. Поворота на угол -ALFA относительно оси Y - матрица C.

Матрица, переводящая куб из исходного положения в фактическое, есть P=C*B*A, где

$$A=\begin{pmatrix} 3/5 & 4/5 & 0 & 0 \\ -4/5 & 3/5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad B=\begin{pmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad C=\begin{pmatrix} 3/5 & 0 & 4/5 & 0 \\ 0 & 1 & 0 & 0 \\ -4/5 & 0 & 3/5 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

и P равна:

$$P=\frac{1}{25}\begin{pmatrix} 9 & 12 & 20 & 15 \\ -20 & 15 & 0 & 0 \\ -12 & -16 & 15 & 20 \\ 0 & 0 & 0 & 25 \end{pmatrix}$$

Перечисленные выше трёхмерные векторы, соответствующие вершинам куба в исходном положении, переходят в восемь векторов, соответствующие этим вершинам в фактическом положении: (26/25,-5/25,7/25), (-14/25,-5/25,-23/25), (-38/25,-35/25,9/25), (2/25,-35/25,39/25), (8/25,35/25,31/25), (-32/25,35/25,1/25), (-56/25,5/25,33/25), (-16/25,5/25,63/26).

Например, точка (1,1,1) переходит в точку (26/25,-5/25,7/25) так как

$$\frac{1}{25}\begin{pmatrix} 9 & 12 & 20 & 15 \\ -20 & 15 & 0 & 0 \\ -12 & -16 & 15 & 20 \\ 0 & 0 & 0 & 25 \end{pmatrix} * \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \frac{1}{25}\begin{pmatrix} 26 \\ -5 \\ 7 \\ 25 \end{pmatrix}$$

Так как матрица Q, переводящая куб из фактического положения в наблюдаемое, единичная, то координаты проекций точек на визуальную плоскость есть (26/25,-5/25), (-14/25,5/25), (-32/25,-35/25), (2/25,-35/25), (8/25,35/25), (-32/25,35/25), (56/25,5/25), (-16/25,5/25). Мы можем нанести эти точки на экран и соединить их в той же последовательности, в какой они определялись в исходном положении.

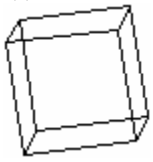


Рис. 33.

Вычисление матрицы Q, переводящей объект из фактического положения в наблюдаемое

Предположим, что глаз наблюдателя находится в точке (EX,EY,EZ) относительно абсолютной оси координат, а направление взгляда - в точку (DX,DY,DZ). Тогда наблюдаемое положение получается так:

1. Матрица D осуществляет перенос точек пространства на вектор (-DX,-DY,-DZ) так, что глаз находится в точке (FX,FY,FZ), равной (EX-DX,EY-DY,EZ-DZ), а взгляд направлен в сторону начала координат.

$$D=\begin{pmatrix} 1 & 0 & 0 & -DX \\ 0 & 1 & 0 & -DY \\ 0 & 0 & 1 & -DZ \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

2. Матрица E переводит (FX,FY,FZ) в (R,0,FZ) вращением пространства на угол ALPHA, где ALPHA=ARCTG(FY/FX), относительно оси Z. Здесь R**2=X**2+Y**2 и R>0

$$E=\frac{1}{R}\begin{pmatrix} FX & FY & 0 & 0 \\ -FY & FX & 0 & 0 \\ 0 & 0 & R & 0 \\ 0 & 0 & 0 & R \end{pmatrix}$$

3. Матрица F переводит (R,0,FZ) в (0,0,-S) поворотом относительно оси Z на угол PI-TETA, где TETA=ARCTG(R/FZ); здесь S**2=R**2+FZ**2=FX**2+FY**2+FZ**2 при S>0

$$F=\frac{1}{S}\begin{pmatrix}-FZ & 0 & R & 0\\ 0 & S & 0 & 0\\ -R & 0 & -FS & 0\\ 0 & 0 & 0 & S\end{pmatrix}$$

Матрицами D, E, F глаз помешается в (0,0,-S), направление взгляда - в сторону начала координат. Глаз находится на том же расстоянии от начала координат, на котором находятся относительно друг друга точки (EX,EY,EZ) и (DX,DY,DZ).

$$G=\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & S\\ 0 & 0 & 0 & 1\end{pmatrix}$$

Теперь мы можем выписать матрицу, помещающую глаз в начало координат.

4. Если мы теперь умножим P=A*B*C на наше первое приближение матрицы Q (=G*F*E*D), переводящей фактическое положение в наблюдаемое, то мы получим матрицу R=Q*P=G*F*E*D*C*B*A, которая переводит объект из исходного положения в наблюдаемое. Изображение куба, полученное с помощью матрицы R, нельзя считать удовлетворительным, так как матрица Q помещает куб на плоскость так, что его ориентация оказывается случайной. Самым распространенным методом, позволяющим избежать этот эффект, является метод фиксации вертикали (или направления, параллельного оси Y), при котором прямая, вертикальная в фактическом положении, остается таковой после перевода объекта в наблюдаемое положение матрицей Q. Рассмотрим прямую, соединяющую точки (DX,DY+1,DZ) и (DX,DY,DZ). Мы знаем, что после преобразования эта прямая перейдет в прямую, соединяющую точки (K(1,2),K(2,2),K(3,2))=(P,Q,R), так что если мы повернём объект относительно оси Z на угол BETA=ARCTG(K(1,2)/K(2,2))=ARCTG(P/Q)=ARCTG(-FY*FZ/(S*PX)), используя для этого поворота умножение на матрицу E (умножение производится перед умножением на матрицу G), то мы установим вертикаль в нужное положение:

$$H=\frac{1}{T}\begin{pmatrix}Q & P & 0 & 0\\ P & -Q & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}$$ где $T^2=P^2*Q^2$

и таким образом:

$$H=\begin{pmatrix}P\\ Q\\ R\\ 1\end{pmatrix}=\frac{1}{T}\begin{pmatrix}Q & P & 0 & 0\\ P & -Q & 0 & 0\\ 0 & 0 & T & 0\\ 0 & 0 & 0 & T\end{pmatrix}*\begin{pmatrix}P\\ Q\\ R\\ 1\end{pmatrix}=\begin{pmatrix}0\\ T\\ R\\ 1\end{pmatrix}$$

Преобразование (рис. 33) осуществляется с помощью матрицы P=Q*R=G*H*F*E*D*C*B*A. Проекцией прямой, соединяющей точки (DX,DY,DZ) с точкой (DX,DY+1,DZ) будет прямая, соединяющая точку экрана (0,0) с точкой (0,T), то есть вертикальная прямая. Матрица G не меняет значений координат X и Y. Обратите внимание на то, что предложенный нами метод применим во всех случаях, за исключением ситуации, когда точка (DX,DY,DZ) находится в точности над точкой (EX,EY,EZ); в этом случае фиксирование вертикали не имеет смысла. Подпрограмма "LOOK3" (листинг 9.1) генерирует матрицу преобразования, переводящего фактическое положение в наблюдаемое положение; эта генерация состоит из нескольких операций, в соответствии с приведенным выше описанием алгоритма; каждой из этих операций соответствует вычисление очередного сомножителя, участвующего в выражении для вычисления матрицы R, так что в конце процесса матрица R кажется домноженной на матрицу Q. Если мы хотим занести в память непосредственно значение матрицы Q, то для этого мы должны присвоить матрице R значение единичной матрицы (используя подпрограмму "IDR3"), затем вызвать "LOOK3" и, наконец, занести массив R в массив Q. Подпрограмму "LOOK3" можно существенно сократить, если условиться считать, что взгляд наблюдателя всегда направлен на начало координат (что эквивалентно DX=0, DY=0, DZ=0). В случае ортогональной проекции нет необходимости помещать глаз в начало координат в наблюдаемом положении, что также позволяет сократить программу. Мы приводим наиболее общий подход к задаче, что окажется весьма полезным при рассмотрении перспективной проекции.

LISTING 9.1

```
8200 REM LOOK3
8210 INPUT "(EX,EY,EZ)   ";EX;",";EY;",";EZ
8220 INPUT "(DX,DY,DZ)   ";DX;",";DY;",";DZ
8229 REM MOVE ORIGIN TO (DX,DY,DZ).
8230 LET TX=DX: LET TY=DY: LET TZ=DZ
```

```
8240 GO SUB TRAN3: GO SUB MULT3
8249 REM MOVE EYE INTO NEGATIVE Z-AXIS, LOOKING AT THE ORIGIN.
8250 LET FX=EX-DX: LET FY=EY-DY: LET FZ=EZ-DZ
8260 LET AX=FX: LET AY=FY: GO SUB ANGLE
8270 LET AXIS=3: LET THETA=-THETA: GO SUB ROT3: GO SUB MULT3
8280 LET AX=FZ: LET AY=SQR(FX*FX+FY*FY): GO SUB ANGLE
8290 LET AXIS=2: LET THETA=PI-THETA: GO SUB ROT3: GO SUB MULT3
8299 REM MAINTAIN VERTICAL.
8300 LET TX=0: LET TY=0: LET TZ=SQR(FX*FX+FY*FY+FZ*FZ)
8310 LET AX=TZ*FX: LET AY=-FY*FZ: GO SUB ANGLE
8320 LET AXIS=3: GO SUB ROT3: GO SUB MULT3
8329 REM MOVE EYE TO THE ORIGIN. SPACE IS ROW IN THE OBSERVED POSITION.
8330 GO SUB TRAN3: GO SUB MULT3
8340 RETURN
```

При необходимости мы можем расширить нашу программу с тем, чтобы включить случай, когда голова наклонена на угол GAMMA относительно вертикали. Для этого необходимо дополнительное вращение пространства на угол GAMMA относительно оси Z, что означает, что матрица H должна повернуть пространство на угол BETA-GAMMA.

Очевидно, что матрица преобразования, переводящего фактическое положение в наблюдаемое положение, зависит только от точки, в которой расположен глаз наблюдателя, направления взгляда и наклона головы, следовательно, матрица Q одинакова для всех объектов, и её имеет смысл занести в память.

Как задавать объект

Настало время обсудить проблему представления объектов в компьютере. Не существует общего решения этой проблемы, все зависит от вида объекта и способа его проецирования. В данном разделе мы приводим различные структуры данных, необходимые для генерации изображения объекта, но не обсуждаем, насколько они удачны. Это делается в оставшейся части книги, где мы приводим примеры использования различных методов, иллюстрирующие их пригодность для практического использования.

Для хранения больших объёмов информации об объекте мы используем массивы, и естественно, что чем больше информации требуется для описания объекта, тем больше памяти следует отводить для массивов; объявляя в программе величину массива, вы должны быть уверены, что он достаточно велик для того, чтобы хранить всю необходимую информацию, пусть он лучше будет большим, чем меньшим.

Нам потребуется хранить информацию о координатах точек в пространстве; для этой цели мы используем три массива X, Y, Z для каждой координаты. Когда число точек не указано явно, то мы полагаем, что оно равно величине Q. Таким образом, в каждом таком массиве должно содержаться не менее NOV элементов. Эти точки могут принадлежать объекту в исходном, фактическом или наблюдаемом положениях, что зависит от решаемой задачи. Нам также встретится ситуация (перспективная проекция, например), в которой будет необходимо хранить значения X и Y координат проекций этих NOV вершин. В случае ортогональной проекции такой проблемы не возникает, так как эти значения хранятся в массивах X и Y. Выбор структуры данных существенно зависит от типа проекции.

Пусть изображение объекта состоит из NOL отрезков. Эту формацию мы можем хранить в массиве L: первый отрезок определяется двумя индексами, принимающими значение от 1 до NOL и соответствующими номерами вершин, являющимися концами этого отрезка; эти числа мы храним как L(1,1), L(2,1). Координаты этих точек могут быть определены с помощью массивов X, Y, Z.

Гранью объекта мы считаем плоскую область в пространстве, ограниченную многоугольником. Имеется несколько способов задания таких областей. Большая часть граней, рассматриваемых в этой книге, является треугольными или квадратными; мы предполагаем, для уменьшения объёма хранимой информации, что максимальное количество граней области равно шести.

Каждая из NOF граней может быть определена с помощью индексов вершин, образующих эту грань, хранящихся в массиве F: F(I,J) - индекс I-ой вершины J-ой грани. Если грань не шестиугольная, то некоторые из элементов массива F, соответствующие грани, не несут никакой информации, в связи с этим мы должны ввести массив H, в котором записано количество вершин каждой грани.

Можно также хранить информацию о цвете каждой грани в массиве C. Однако будьте внимательны, так как в SPECTRUMe в каждом символьном блоке разрешается использовать не более двух цветов. Другой способ хранения информации о гранях – это хранить индексы прямых, образующих грань; в этом случае F(I,J) это индекс I-го ребра J-ой грани. Существует много способов хранения этих и других элементов трёхмерных объектов; определите сами, какой из них соответствует вашей задаче наилучшим образом.

Конструирующие подпрограммы и метод построения блоков

Для каждого объекта мы определяем конструирующую подпрограмму, которая использует в качестве параметров матрицу R для установки вершин в необходимое положение и другую информацию о размерах объекта (если объект заносится в память в исходном положении, то никакой матрицы не требуется). Исходя из этих данных, подпрограмма определяет вершины, ребра, грани и все остальные элементы объекта, а затем с помощью матрицы R подпрограмма устанавливает вершины в необходимое положение. В зависимости от ситуации, подпрограмма может либо вычертить объект, либо записать в память информацию об этом объекте. Мы приведем примеры использования обеих возможностей.

Мы можем сгенерировать изображение, состоящее из множества подобных объектов (в этом случае информация будет о фактическом или наблюдаемом положении). В этом случае нет необходимости создавать каждый

из объектов заново, достаточно вычислять для каждого объекта свою матрицу R, соответствующую преобразованию, переводящему объект из исходного положения в фактическое (для фактического положения) или Q*F (для наблюдаемого положения) и вводить её в ту же подпрограмму. Для каждого нового типа объектов нам потребуется конструирующая подпрограмма.

Целиком рисунок генерируется с помощью основной программы (листинг 9.2), объявляющей метки всех подпрограмм, приготавливающей графический экран, используя входные параметры HORIZ и VERT и, наконец, вызывающей подпрограмму "SCENES", располагающую объекты в пространстве и вычисляющую их. Приведенная ниже подпрограмма используется во всех программах трёхмерной графики, которые приводятся ниже, так что не следует её менять без особой на то необходимости.

LISTING 9.2

```
100 REM MAIN PROGRAM
110 LET START=9700: LET SETORIGIN=9600: LET MOVETO=9500: LET LINETO=9400: LET
CLIP=8400
120 LET ROT3=8600: LET ANGLE=8800: LET SCALES=8900: LET TRAN3=9000: LET
MULT3=9100: LET IDR3=9300
130 LET SCENE3=6000: LET LOOK3=8200
139 REM DIMENSION AND CENTRE THE GRAPHICS AREA.
140 INPUT "HORIZ=";HORIZ;" VERT=";VERT
150 GO SUB START
160 LET XMOVE=HORIZ*0.5: LET YMOVE=VERT*0.5
170 GO SUB SETORIGIN
179 REM SET THE SCENE.
180 GO SUB SCENE3
190 STOP
```

В подпрограмме "SCENES" объявляются все массивы, необходимые для хранения информации об изображении, а также матрицы A, B, R и (возможно) матрица Q, предназначенные для помещения объекта в необходимое положение. При необходимости переменным, которые изменяются в дальнейших конструирующих подпрограммах.

Для каждого объекта ("блока") подпрограмма "SCENES" должна вычислить матрицу P, помещающую блок в фактическое положение, а затем, вычисляя соответствующую матрицу P (переводящую исходное положение в фактическое или исходное положение в наблюдаемое). В результате из этих блоков складывается изображение. Иногда это изображение вычерчивается конструирующей подпрограммой, а иногда с помощью подпрограмм, предназначенных для вычерчивания объектов особого вида (таких, как объекты с удаленными невидимыми линиями или поверхностями) – все зависит от того, изображение чего создается и какого типа должно быть это изображение.

Как и раньше, мы не генерируем матрицы P и Q в явном виде, но вместо этого изменяем значение P, что связано с запретом на передачу подпрограмме массивов в качестве параметров. Если требуется матрица Q, то для этого достаточно вызвать подпрограмму "IDR3", которая делает матрицу R единичной; полезно иметь в виду, что все матричные операции производятся с помощью матриц A и R с использованием матрицы B для хранения промежуточных результатов.

В своих программах используйте вариант подпрограммы "LINETO", удаляющий части прямых, выходящие за рамки графического экрана. Часто встречается ситуация, когда выбранные значения HORIZ и VERT слишком малы для того, чтобы весь объект было возможно разместить на экране, и в этом случае немодифицированный вариант подпрограммы "LINETO" не сможет быть использован. Иногда значениям HORIZ и VERT умышленно задают заниженные значения, что позволяет с помощью последующего увеличения получать увеличенные изображения отдельных участков объекта.

Первый пример применения изложенного выше метода приведен на листинге 9.3, где приведена подпрограмма "SCENES". На листинге 9.4 приводится конструирующая подпрограмма "CUBE", генерирующая данные куба с длиной ребра, равной 2; координаты вершин заносятся в массивы X, Y, Z. В данном случае можно не хранить информацию о рёбрах куба: мы получаем необходимую информацию из оператора DATA, а затем чертим линии.

Входные данные для рис. 33 есть:

HORIZ=9, VERT=6,
(EX,EY,EZ)=(-2,2,2), (DX,DY,DZ)=(-l,0,0).

LISTING 9.3

```
6000 REM SCENES/EXAMPLE 1
6010 DIM X(8): DIM Y(8): DIM Z(8)
6020 DIM A(4,4): DIM B(4,4): DIM R(4,4)
6030 LET CUBE=6500
6039 REM CALCULATE THE SETUP TO ACTUAL MATRIX R.
6040 GO SUB IDR3
6050 LET THETA=-0.92729522: LET AXIS=3: GO SUB ROT3: GO SUB MULT3
6060 LET TX=-1: LET TY=0: LET TZ=0: GO SUB TRAN3: GO SUB MULT3
6070 LET THETA=-THETA: LET AXIS=2: GO SUB ROT3: GO SUB MULT3
6079 REM CHANGE R - PREMULTIPLY IT BY THE ACTUAL TO OBSERVED MATRIX.
6080 GO SUB LOOK3
6089 REM CALL CONSTRUCTION ROUTINE DRAW THE CUBE.
```

```
6090 GO SUB CUBE
6100 RETURN
```

LISTING 9.4

```
6500 REM CUBE/LINES (NOT STORED)
6501 REM IN - R(3,3)
6510 DATA 1,1,1,1,1,-1,1,-1,-1,1,-1,1,-1,1,1,-1,1,-1,-1,-1,-1,-1,-1,1
6520 DATA 1,2,2,3,3,4,4,1,5,6,6,7,7,8,8,5,1,5,2,6,3,7,4,8
6530 RESTORE CUBE
6539 REM INPUT SETUP VERTICES OF CUBE AND MOVE THEM INTO OBSERVED POSITION.
6540 FOR I=1 TO 8
6550 READ XX,YY,ZZ
6560 LET X(I)=XX*R(1,1)+YY*R(1,2)+ZZ*R(1,3)+R(1,4)
6570 LET Y(I)=XX*R(2,1)+YY*R(2,2)+ZZ*R(2,3)+R(2,4)
6580 LET Z(I)=XX*R(3,1)+YY*R(3,2)+ZZ*R(3,3)+R(3,4)
6590 NEXT I
6599 REM INPUT LINE INFORMATION. DRAW LINES BY JOINING PAIRS OF VERTICES.
6600 FOR I=1 TO 12
6610 READ L1,L2
6620 LET XPT=X(L1): LET YPT=Y(L1): GO SUB MOVETO
6630 LET XPT=X(L2): LET YPT=Y(L2): GO SUB LINETO
6640 NEXT I
6650 RETURN
```

На одном рисунке можно поместить изображение более чем одного куба. Например, если мы перепишем подпрограмму "SCENES", как это показано на листинге 9.5, а все остальные подпрограммы оставим теми же, то в результате мы получим фигуру, изображенную на рисунке 34.

Обратите внимание на то, что значения X, Y, Z, вычисленные для первого куба, переписываются при повторном вызове программы "CUBE". Кроме того, так как матрица, соответствующая переводу объекта из фактического положения в наблюдаемое, в обоих случаях одна и та же (у них разные матрицы преобразований из исходного положения в фактическое), то мы заносим в память матрицу Q с тем, чтобы использовать её вторично. Помните, что матрица Q умножается на матрицу P, переводящую второй куб в фактическое положение. Параметры для рис. 34 таковы: HORIZ=9, VERT=6, (EX,EY,EZ)=(3,2,1) и (DX,DY,DZ)=(0,0,0).

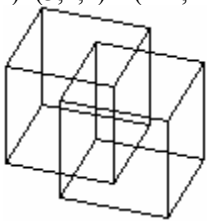


Рис. 34.

LISTING 9.5

```
6000 REM SCENE3/EXAMPLE2
6010 DIM X(8):DIM Y(8): DIM Z(8)
6020 DIM A(4,4): DIM B(4,4): DIM R(4,4): DIM Q(4,4)
6030 LET CUBE=6500
6039 REM DRAW 1'ST CUBE IN OBSERVED POSITION.
6040 GO SUB IDR3: GO SUB LOOK3
6050 GO SUB CUBE
6059 REM DRAW 2'ND CUBE IN OBSERVED POSITION.
6060 FOR I=1 TO 4: FOR J=1 TO 4
6070 LET Q(I,J)=R(I,J)
6080 NEXT J: NEXT I
6090 GO SUB IDR3
6100 LET TX=3: LET TY=1.5: LET TZ=2: GO SUB TRAN3: GO SUB MULT3
6110 FOR I=1 TO 4: FOR J=1 TO 4
6120 LET A(I,J)=Q(I,J)
6130 NEXT J: NEXT I
6140 GO SUB MULT3
6150 GO SUB CUBE
6160 RETURN
```

Упражнение 9.1

Модифицируйте программу "CUBE" так, чтобы можно было вводить данные о прямоугольном параллелепипеде, что позволяло бы строить параллелепипед высоты HT, ширины BH и длины LH. Для этого умножьте значения X-координаты куба в исходном положении на LH/2, Y - на HT/2, и Z - на BH/2.

Ещё раз подчеркнём, что принятый нами модульный подход, возможно, не самый эффективный метод решения поставленной задачи - генерации изображения трёхмерных объектов. Мы выбрали этот подход с тем, чтобы разбить глобальную проблему на множество более простых задач, допускающих сравнительно несложное решение.

Как только читатель достиг хорошего понимания изложенных в книге методов, ему следует взяться за энергичную переделку приведенных здесь подпрограмм с целью их оптимизации. Однако для демонстрации преимуществ модульного подхода мы приведём пример того, как с помощью незначительных модификаций старых подпрограмм удается создавать новые рисунки.

Пример 9.Б

Мы хотим осмотреть некоторую фигуру (например, ту, что на рис. 34) с нескольких различных точек.

В этом случае удобнее запоминать координаты вершин фигуры в фактическом, а не в наблюдаемом положении, и занести информацию о линиях в массив. Программа "SCENES" (листинг 9.6) должна сначала присвоить переменным NOV и NOL нулевые значения, а затем, используя матрицу R=P, поместить объекты в их фактическое положение. Конструкционная подпрограмма "CUBE" (листинг 9.7) тогда должна быть изменена так, чтобы она могла менять значения данных (но обратите внимание, что та же программа могла бы хранить координаты вершин в наблюдаемом положении, что только потребовало бы другой R=Q*P).

Для каждой новой точки, из которой предмет рассматривается, и каждого направления взгляда подпрограмма "SCENES" должна очистить экран, присвоить P значение единичной матрицы, вызвать подпрограмму "LOOK3" и затем вызвать подпрограмму "DRAWIT" (листинг 9.8), использующую матрицу R (в этой матрице хранится значение матрицы Q, переводящей фактическое положение в наблюдаемое) для расположения точек в наблюдаемом положении и последующей ортогональной проекции их в массивы V и W (мы не можем использовать для этой цели массивы X и Y, так как это повлекло бы искажение информации об объекте в фактическом положении). Подпрограмма "DRAWIT" может использовать информацию, хранящуюся в массиве L, для генерации изображения на экране.

LISTING 9.6

```
6000 REM SCENE3/EXAMPLE2 (VARIETY OF VIEWS)
6010 DIM X(16): DIM Y(16): DIM Z(16)
6020 DIM V(16): DIM W(16): DIM L(2,24)
6030 DIM A(4,4): DIM B(4,4): DIM R(4,4)
6040 LET CUBE=6500: LET DRAWIT=7000
6050 LET NOV=0: LET NOL=0
6059 REM STORE 1'ST CUBE IN ACTUAL (=SETUP) POSITION.
6060 GO SUB IDR3
6070 GO SUB CUBE
6079 REM STORE 2'ND CUBE IN ACTUAL POSITION.
6080 LET TX=3: LET TY=1.5: LET TZ=2: GO SUB TRAN3: GO SUB MULT3
6090 GO SUB CUBE
6099 REM LOOP THROUGH DIFFERENT VIEWING POSITIONS.
6100 GO SUB IDR3: GO SUB LOOK3
6109 REM DRAW THE TWO CUBES IN OBSERVED POSITION.
6110 CLS: GO SUB DRAWIT
6120 GO TO 6100
6130 RETURN
```

LISTING 9.7

```
6500 REM CUBE/VERTICES AND LINES (STORED)
6501 REM IN - NOV,NOL,R(4,4),X(NOV),Y(NOV),Z(NOV),L(2,NOL)
6502 REM OUT- NOV,NOL,X(NOV),Y(NOV),Z(NOV),L(2,NOL)
6510 DATA 1,1,1,1,1,-1,1,-1,-1,1,-1,1,-1,1,1,-1,1,-1,-1,-1,-1,-1,-1,1
6520 DATA 1,2,2,3,3,4,4,1,5,6,6,7,7,8,8,5,1,5,2,6,3,7,4,8
6530 RESTORE CUBE
6540 LET NV=NOV
6549 REM INPUT AND STORE VERTICES IN POSITION (ACTUAL OR OBSERVED) USING R.
6550 FOR I=1 TO 8
6560 READ XX,YY,ZZ: LET NOV=NOV+1
6570 LET X(NOV)=XX*R(1,1)+YY*R(1,2)+ZZ*R(1,3)+R(1,4)
6580 LET Y(NOV)=XX*R(2,1)+YY*R(2,2)+ZZ*R(2,3)+R(2,4)
6590 LET Z(NOV)=XX*R(3,1)+YY*R(3,2)+ZZ*R(3,3)+R(3,4)
6600 NEXT I
6609 REM INPUT AND STORE LINE INFORMATION.
6610 FOR I=1 TO 12
6620 READ L1,L2: LET NOL=NOL+1
6630 LET L(1,NOL)=L1+NV: LET L(2,NOL)=L2+NV
6640 NEXT I
6650 RETURN
```

LISTING 9.8

```
7000 REM DRAWIT
7001 REM IN - NOV,NOL,R(4,4), X(NOV),Y(NOV), Z(NOV),L(2,NOL)
7009 REM MOVE VERTICES INTO OBSERVED POSITION AND DRAW OBJECT.
7010 FOR I=1 TO NOV
7020 LET V(I)=X(I)*R(1,1)+Y(I)*R(1,2)+Z(I)*R(1,3)+R(1,4)
7030 LET W(I)=X(I)*R(2,1)+Y(I)*R(2,2)+Z(I)*R(2,3)+R(2,4)
```

```
7040 NEXT I
7050 FOR I=1 TO NOL
7060 LET L1=L(1,I): LET L2=L(2,I)
7070 LET XPT=V(L1): LET YPT=W(L1): GO SUB MOVETO
7080 LET XPT=V(L2): LET YPT=W(L2): GO SUB LINETO
7090 NEXT I
7100 RETURN
```

Если наблюдатель перемещается по прямой линии и при этом смотрит в одном направлении, то нет необходимости каждый раз заново вычислять матрицу Q, достаточно произвести такое преобразование пространства, чтобы взгляд наблюдателя был направлен вдоль оси Z, а затем использовать подпрограмму "SETORIGIN" для того, чтобы перемещать наблюдателя, вместо того, чтобы перемещать объект. После того как вы освоитесь с методами построения проекций трёхмерных объектов, вам следует тщательно выбирать метод обзора и вычерчивания. Почти всегда существуют способы избежать использования общего метода построения проекций трёхмерных объектов и использовать более простые методы.

Упражнение 9.2

Составьте конструирующие программы для тетраэдра, пирамиды, и т.д.

Например,

(А) Тетраэдр: вершины - (1,1,1), (4,-1,-1), (-1,1,-1) и (-1,-1,1) + ребра - точка 1 соединяется с 2, 1 с 3, 1 с 4, 2 с 3, 2 с 4 и 3 с 4.

(Б) Пирамида с квадратом в основании и высотой HT: вершины (0,HT,0), (1,0,1), (1,0,-1), (-1,0,-1) и (-1,0,1); прямые соединяют 1 с 2, 1 с 3, 1 с 4, 1 с 5, 2 с 3, 3 с 4, 4 с 5 и 5 с 1.

Упражнение 9.3

Нанесите на плоскость X/Y изображение любого плоского объекта, например, букв алфавита, и получите изображение этого объекта с различных точек наблюдения. Вы можете поместить их на грани куба. Все, что вам надо сделать - это дополнить программу "CUBE", с тем, чтобы стало возможным хранить дополнительные вершины и линии, определяющие эти символы.

Пока что мы ограничились изображением простейшего куба. Это делалось для того, чтобы не загромождать наши примеры определением сложных объектов. Наши программы пригодны для любых объектов, лишь бы они не вызывали переполнения памяти. Для сложных объектов мы просто увеличиваем размеры наших массивов, хотя некоторые объекты будут обладать свойствами, позволяющими нам уменьшить количество требуемой памяти. Например, реактивный самолет, показанный на рис. 35, обладает зеркальной симметрией. Предположим, что плоскость симметрии совпадает с плоскостью Y/Z, тогда каждой точке самолета (X,Y,Z) соответствует точка самолёта (-X,Y,Z). Для получения изображения на рис. 35 мы используем листинг 9.1, 9.2 и 9.3 совместно с конструирующей подпрограммой "JET", генерирующей все вершины самолёта с положительными значениями координаты X; это позволяет хранить в памяти информацию только об одной половине самолёта.

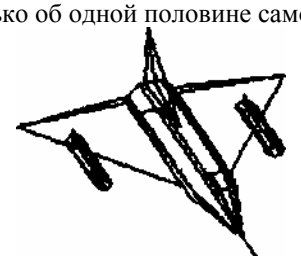


Рис. 35.

Для того чтобы нарисовать самолёт, нам требуется подпрограмма "DRAWIT" (листинг 9.9), которая рисует одну половину самолета, а затем, сменив знак X координаты на противоположный, другую.

Строить подобные фигуры несложно. Нарисуйте интересующие вас фигуры на листе клетчатой бумаги. Пронумеруйте наиболее важные точки фигуры и отметьте, какие из точек соединены линиями. Значения координат легко подсчитать с помощью клеток. Входные данные для рис 35: HORIZ=160, VERT=120, (EX,EY,EZ)=(1,2,3) и (DX,DY,DZ)=(0,0,0).

Тела вращения

До сих пор при построении изображения объекта мы использовали информацию, которая целиком вводилась вручную. Теперь мы рассмотрим тип объектов, для построения которых требуется только небольшой объём информации при сравнительной сложности самих объектов. Таким объектом является тело вращения, приведенное на рис. 36.

Наш метод заключается в том, чтобы определить NUMV линий в плоскости X/Y, которые мы называем набором образующих отрезков. Затем мы вращаем эти отрезки относительно вертикальной оси (оси Y) NUMH-1 раз с тем, чтобы получить новые множества вертикальных отрезков. NUMV отрезков обращающего набора получаются последовательным соединением NUMV+1 вершин (S(I),T(D) (где 1<=I<=NUMV+1). Из этого набора M получаем NUMH различных вертикальных наборов, J-ый вертикальный набор получается вращением набора образующих отрезков на угол PHI+2*PI*(J-1)/NUMH вокруг оси Y, для некоторого значения PHI(0), являющегося входным параметром.

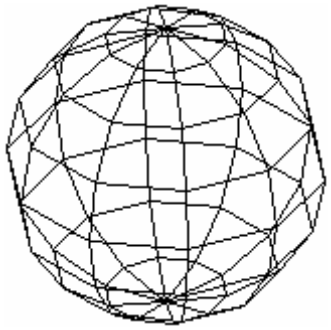


Рис. 36.

Наряду с NUMH*NUMV вертикальными отрезками мы вводим так же горизонтальные прямые. Рассмотрим одну из концевых точек (S(I),T(I),0) отрезка, принадлежащего множеству образующих отрезков: по мере того, как мы вращаем образующие отрезки вокруг оси, он занимает одно из NUMH положений (если только эта точка не расположена на оси вращения).

(S(I)*COS(TETA+0),T(I),S(I)*SIN(TETA+0)),

где TETA=2*PI*(J-1) и 1<=J<=NUMH.

Эти NUMH точек последовательно соединяются, а затем NUMH-ая точка соединяется с первой, что дает набор горизонталей. Всего получается (NUMH-N)*NUMV горизонтальных линий, где N-число вершин на оси вращения. На листинге 9.10 приведена программа "ROTBOD", рисующая тело вращения, если заданы NUMV, NUMH, PHI, первоначальный набор вершин в массивах T и S и матрица R, определяющая положение тела. На листинге 9.11 приведена программа "SCENES", генерирующая изображение сферы, приведенной на рис. 36; начальный набор точек состоит из восьми точек, расположенных на половине полной дуги окружности: HORIZ=3.2, VERT=2.2, PHI=PI/25, NUMH=10, NUMV=8, глаз наблюдателя расположен в точке (1,2,3), а взгляд направлен в точку (0,0,0). В подпрограмме на листинге 9.11 предполагается, что NUMV меньше или равен 15.

LISTING 9.9

```
6000 REM SCENE3/JET
6010 DIM X(37): DIM Y(37): DIM Z(37)
6020 DIM V(37): DIM W(37): DIM L(2,46)
6030 DIM A(4,4): DIM B(4,4): DIM R(4,4)
6040 LET JET=6500: LET DRAWIT=7000
6050 GO SUB IDR3: GO SUB LOOK3
6060 GO SUB JET
6070 GO SUB DRAWIT
6080 RETURN
6500 REM JET
6502 REM OUT - NOV,NOL,X(NOV),Y(NOV),Z(NOV),L(2,NOL)
6510 DATA 0,0,80,0,0,64,0,8,32,4,8,32,8,4,32,8,0,32,4,-4,32,0,8,-32,4,8,-32,8,
4,-32,8,0,-32,4,-4,-32,0,-4,-32,8,0,24
6520 DATA 1,2,2,3,2,4,2,5,2,6,2,7,3,4,4,9,5,10,6,11,7,12,8,9,9,10,10,11,11,12,
12,13,14,15,15,10,15,16,14,16,17,18,17,19,18,19,20,21,21,22,22,23,23,24,24,25
6529 REM SETUP VERTEX AND LINE INFORMATION FOR HALF THE JET.
6530 LET NOV=37: LET NOL=46
6540 FOR I=1 TO NOV: READ X(I),Y(I),Z(I): NEXT I
6550 FOR I=1 TO NOL: READ L(1,I),L(2,I): NEXT I
6560 RETURN
7000 REM DRAWIT/TWO HALFES OF THE JET
7001 REM IN - NOV,NOL,R(4,4),X(NOV),Y(NOV),Z(NOV),L(2,NOL)
7010 LET IS=1
7019 REM LOOP THROUGH TWO HALFES OF THE JET.
7020 FOR J=1 TO 2
7030 FOR I=1 TO NOV
7040 LET XX=IS*X(I): LET YY=Y(I): LET ZZ=Z(I)
7049 REM PUT VERTICES IN OBSERVED POSITION.
7050 LET V(I)=XX*R(1,1)+YY*R(1,2)+ZZ*R(1,3)+R(1,4)
7060 LET W(I)=XX*R(2,1)+YY*R(2,2)+ZZ*R(2,3)+R(2,4)
7070 NEXT I
7080 FOR I=1 TO NOL
7090 LET L1=L(1,I): LET L2=L(2,I)
7100 LET XPT=V(L1): LET YPT=W(L1): GO SUB MOVETO
7110 LET XPT=V(L2): LET YPT=W(L2): GO SUB LINETO
7120 NEXT I
7130 LET IS=-1
7140 NEXT J
7150 RETURN
```

LISTING 9.10

```
6000 REM SCENE3/SPHEROID
6010 DIM X(32): DIM Y(32)
6020 DIM A(4,4): DIM B(4,4): DIM R(4,4)
6030 DIM S(16): DIM T(16)
6040 LET REVBOD=6500
6050 INPUT "NUMBER OF HORIZONTAL LINES=";NUMH
6060 INPUT "NUMBER OF VERTICAL LINES=";NUMV
6070 INPUT "ANGLE PHI=";PHI
6080 LET THETA=PI/2: LET TD=PI/NUMV
6089 REM GENERATE DEFINITION SET
6090 FOR I=1 TO NUMV+1
6100 LET S(I)=COS THETA: LET T(I)=SIN THETA
6110 LET THETA=THETA+TD
6120 NEXT I
6130 GO SUB IDR3: GO SUB LOOK3
6140 GO SUB REVBOD
6150 RETURN
```

Упражнение 9.4

Поэкспериментируйте с этой техникой, для этой цели подходит любая последовательность отрезков. Попробуйте начертить эллипсоид вращения: он чертится так же, как и сфера, за исключением того, что вместо половины окружности берётся половина эллипса. Нет необходимости ограничиваться только случаем выпуклых многоугольников: прямые могут пересекать друг друга, пересекать ось Y и допускается также колебание значения координаты X.

LISTING 9.11

```
6500 REM REVBOD/BODY OF REVOLUTION
6501 REM IN - PHI,NUMH,NUMV,S(NUMV+1),T(NUMV+1),R(4,4)
6509 REM CREATE FIRST VERTICAL SET.
6510 LET THETA=PHI: LET TD=PI*2/NUMH
6520 LET N1=NUMV+1: LET C=COS PHI: LET S=SIN PHI
6530 FOR I=1 TO N1
6540 LET XX=S(I)*C: LET YY=T(I): LET ZZ=S(I)*S
6550 LET X(I)=XX*R(1,1)+YY*R(1,2)+ZZ*R(1,3)+R(1,4)
6560 LET Y(I)=XX*R(2,1)+YY*R(2,2)+ZZ*R(2,3)+R(2,4)
6570 NEXT I
6579 REM LOOP THROUGH SECOND VERTICAL SET.
6580 FOR J=1 TO NUMH
6590 LET THETA=THETA+TD: LET C=COS THETA: LET S=SIN THETA
6600 FOR I=1 TO N1
6610 LET XX=S(I)*C: LET YY=T(I): LET ZZ=S(I)*S
6620 LET X(I+N1)=XX*R(1,1)+YY*R(1,2)+ZZ*R(1,3)+R(1,4)
6630 LET Y(I+N1)=XX*R(2,1)+YY*R(2,2)+ZZ*R(2,3)+R(2,4)
6640 NEXT I
6649 REM DRAW LINES IN FIRST VERTICAL SET.
6650 LET XPT=X(1): LET YPT=Y(1): GO SUB MOVETO
6660 FOR I=2 TO N1
6670 LET XPT=X(I): LET YPT=Y(I): GO SUB LINETO
6680 NEXT I
6689 REM JOIN CORRESPONDING HORIZONTAL ON THE TWO VERTICAL SETS.
6690 FOR I=1 TO N1
6700 LET XPT=X(I): LET YPT=Y(I): GO SUB MOVETO
6710 LET XPT=X(I+N1): LET YPT=Y(I+N1): GO SUB LINETO
6719 REM COPY SECOND VERTICAL SET INTO FIRST REPIAT PROCESS.
6720 LET X(I)=XPT: LET Y(I)=YPT
6730 NEXT I
6740 NEXT J
6750 RETURN
```

Можно предложить следующее обобщение нашего метода: по мере вращения вокруг центральной оси Y-координаты точек, не остаются постоянными, а изменяются по определенному закону; например, уменьшаются на некоторую величину при каждом вращении на 2PI/NUMH. Теперь имеет смысл позволить точке совершать более одного оборота вокруг оси.

Список программ

С настоящего момента мы будем именовать листинги 3.4 ("ANGLE"), 8.1 ("MULT3" и "IDR3"), 8.2 ("TRAN3"), 8.3 ("SCALE3"), 8.4 ГКОТ3"), 9.1 ("LOOK3") и 9.2 ("MAIN PROGRAM") как "LIB3".

1. "LIB1", "LIB3" и листинги 9.3 ("SCENE3") и 9.4 ("CUBE"). Необходимые данные: HORIZ, VERT, (EX,EY,EZ) и (DX,DY,DZ). Попробуйте задать значения 9, 6, (1,2,3), (-1,0,1).

2. "LIB1", "LIB3" и листинги 9.5 ("SCENES") и 9.4 ("CUBE"). Необходимые данные: HORIZ, VERT, (EX,EY,EZ) и (DX,DY,DZ). Попробуйте значения 9, 6, (1,2,3), (-1,0,1). Последовательно изменяйте один из параметров при фиксированных остальных параметрах.

3. "LIB1", "LIB3" и листинг 9.6 ("SCENES"), 9.7 ("CUBE") и 9.8 ("DRAWIT"). Необходимая информация: HORIZ, VERT, и затем повторный ввод величин (EX,EY,EZ), (DX,DY,DZ). Попробуйте ввести 9, 6, затем (1,2,3), (-1,0,1), (3,2,1), (0,0,1). Проведите систематическое изменение величин параметров.

4. "LIB1", "LIB3" и листинги 9.10 ("SCENES") и 9.11 ("REVBOD"). Необходимая информация: HORIZ, VERT, NUMV (<=15), PHI, (EX,EY,EZ) и (DX,DY,DZ). Попробуйте ввести 3.2, 2.2, 10, 10, 1, (1,2,3), (0,0,0); (3,2,1), (0,0,0).

ГЛАВА 10
Простые алгоритмы удаления невидимых линий и поверхностей

После того, как мы нарисовали куб и ещё несколько каркасных фигур, мы видим, что основным недостатком этих фигур является то, что они не воспроизводят реальных весомых тел. Нам бы хотелось рисовать реальные непрозрачные объекты, в которых передние грани делают невидимыми расположенные за ними грани и ребра. Для этой цели мы должны ввести алгоритмы удаления невидимых линий и поверхностей. Таких алгоритмов существует чрезвычайно много - некоторые из них элементарны и разработаны для специальных случаев, другие очень хитроумны и предназначены для работы со сложными объектами. Ограничения по времени, накладываемые использованием микрокомпьютера, не позволяет нам рассматривать очень сложные алгоритмы.

В случае SPECTRUMa накладывается дополнительное ограничение, запрещающее использовать в одном символьном блоке более двух цветов. Тем не менее, ограничив число типов и количество объектов в одном рисунке, можно добиться вполне удовлетворительных результатов. В главе 12 мы обсуждаем относительно сложный алгоритм, но здесь мы ограничимся двумя специальными типами сцен - мы используем свойства, явно заложенные в природе фигур, составляющих сцену, что облегчает задачу нахождения скрытых поверхностей и ребер. Ниже в этой главе мы приводим простой метод вычерчивания некоторых поверхностей в трёхмерном пространстве.

При определении алгоритма скрытых линий мы считаем, что информация о NOV вершинах объекта хранится в массивах X, Y, Z (в наблюдаемом положении). Однако, мы теперь будем использовать информацию о гранях, а не о ребрах. Информация о NOF-гранях хранится в массиве F (нам также необходим массив H, в котором хранится число вершин каждой многоугольной грани), а для сокращения объёма памяти ограничиваемся случаем, когда грань образуется не более, чем шестью ребрами.

Если требуются грани, содержащие большее число ребер, то такие грани следует разбивать на меньшие грани. Для того чтобы упростить алгоритм удаления невидимых линий, мы накладываем ограничение на порядок записи вершин в массиве F. Вершины следуют в порядке обхода грани так, что если на грань смотреть извне, то этот обход совершается против часовой стрелки. Естественно, что если смотреть изнутри объекта, то этот обход будет совершаться по часовой стрелке. Мы также полагаем, что каждая прямая находится на пересечении двух граней. Отдельные прямые, не связанные с гранями, добавляются в качестве вырожденных граней, ограниченных двумя ребрами.

Ориентация треугольника в трехмерном пространстве

После определения объекта в терминах вершин и граней возникает вопрос: как проверить, что вершины действительно расположены в порядке обхода граней против часовой стрелки? Следует просто написать программу! Так как ориентация выпуклого многоугольника может быть определена по трём его последовательным точкам, то нам достаточно ограничиться случаем треугольника. В главе 7 мы уже познакомились со способом определения ориентации треугольника, расположенного на плоскости. Проблема будет решена, если нам удастся свести задачу к случаю двух измерений.

Для упрощения вычислений мы предполагаем, что объекты находятся в исходном положении и содержат начало координат. Мы предполагаем, что плоскости, в которых лежат грани, не проходят через начало координат. Совершим такое вращение пространства, чтобы одна из вершин рассматриваемого треугольника лежала на отрицательной половине Z (сравните с программой "LOOK3", приведенной на листинге 9.1). Так как мы предполагаем, что начало координат находится внутри объекта, то достаточно его спроецировать на плоскость X/Y (что просто означает отбрасывание значения Z-координаты) и рассматривать этот треугольник лежащим в этой плоскости (одна из его вершин будет в точке (0,0)). На листинге 10.1 приведено решение этой задачи.

LISTING 10.1

```
100 REM ORIENTATION OF A 3-D TRIANGLE
110 DIM X(3): DIM Y(3): DIM Z(3)
120 DIM A(4,4):DIM B(4,4): DIM R(4,4)
130 LET ROT3=8600: LET ANGLE=8800: LET MULT3=9100: LET IDR3=9300
140 LET J$="TYPE IN COORDINATES ": LET K$="OF TRIANGLE "
150 FOR I=1 TO 3
160 LET I$="VERTEX("+STR$ I+")=("
170 INPUT (J$+K$+I$);X(I);",";Y(I);",";Z(I);")"
180 PRINT AT 2+2*I,0;I$;X(I);",";Y(I);",";Z(I);")"
190 NEXT I
200 REM FIND MATRIX R THAT WILL PUT (X(1),Y(1),Z(1)) ON NEGATIVE Z-AXIS.
210 LET AX=X(1): LET AY=Y(1): GO SUB ANGLE
220 LET AXIS=3: LET THETA=-THETA: GO SUB ROT3: GO SUB MULT3
230 LET AX=Z(1): LET AY=SQR(X(1)*X(1)+Y(1)*Y(1)): GO SUB ANGLE
240 LET AXIS=2: LET THETA=PI-THETA: GO SUB ROT3: GO SUB MULT3
249 REM TRANSFORM TRIANGLE SO THAT ALL VERTICES TIE IN AN X/Y PLANE.
250 FOR I=1 TO 3
260 LET XX=X(I): LET YY=Y(I): LET ZZ=Z(I)
270 LET X(I)=XX*R(1,1)+YY*R(1,2)+ZZ*R(1,3)+R(1,4)
280 LET Y(I)=XX*R(2,1)+YY*R(2,2)+ZZ*R(2,3)+R(2,4)
```

```
290 NEXT I
299 REM CHECK IF THE NOW 2-D TRIANGLE IS CLOCKWIZE OR ANTI-CLOCKWIZE.
300 PRINT AT 11,0;"IF THE EYE AND THE ORIGIN ARE ON "
310 PRINT AT 13,0;"OPPOSITE SIDES OF THE FACET THEN"
320 PRINT AT 15,0;"THE TRIANGLE IS ";
330 LET DX1=X(2)-X(1): LET DY1=Y(2)-Y(1)
340 LET DX2=X(3)-X(2): LET DY2=Y(3)-Y(2)
350 IF DX1*DY2-DX2*DY1>0 THEN PRINT "ANTI-";
360 PRINT "CLOCKWISE"
370 STOP
```

Упражнение 10.1

Перепишите программы последней главы, предполагая, что информация задается с помощью вершин и ориентированных против часовой стрелки граней, а не с помощью линий. Проверяйте правильность исходных данных с помощью программы на листинге 10.1. Конечно же, информация о гранях включает явным образом информацию о рёбрах - каждое из них соединяет две последовательные вершины грани. Информация о каждом ребре содержится в массиве дважды - по одному разу в каждой из прилегающих к нему граней, а мы не хотим тратить время на то, чтобы чертить каждую линию дважды. Заметим, что, так как направление обхода всех граней против часовой стрелки, то если в одном случае ребро проходится от I до J, то при обходе другой грани оно пройдет от J до I, поэтому условимся чертить только те ребра, у которых индекс первой вершины меньше индекса второй вершины.

Алгоритм удаления скрытых поверхностей для случая тела, ограниченного замкнутой и выпуклой поверхностью

Ограниченным выпуклым называется тело, которое вместе с любыми двумя внутренними точками содержит также и соединяющий их отрезок. Нетрудно видеть, что это не что иное, как обобщение определения, данного для двумерного случая. Это тело ограничено замкнутой поверхностью, и поэтому невозможно проникнуть внутрь тела, не пересекая его поверхности.

Ортогонально спроецируем все вершины тела на визуальную плоскость, заметив, что проекция выпуклого N-угольника есть выпуклый N-угольник (или вырождается в отрезок) на визуальной плоскости. Обходя вершины полученных N-угольников в той же последовательности, что и в оригинале, мы обнаружим, что наши двумерные многоугольники либо ориентированы против хода часовой стрелки, что означает, что мы смотрим на грань с внешней стороны тела, либо по ходу часовой стрелки, что означает, что мы смотрим на грань изнутри.

Так как объект ограничен замкнутой поверхностью, то мы не можем наблюдать грани с внутренней стороны - они являются невидимыми, а мы должны изобразить только видимые грани - получился довольно простой алгоритм, который может применяться как в конструирующих подпрограммах, так и в рисующей подпрограмме.

В качестве примера на рис. 37 приведено изображение куба, полученное с помощью усовершенствованной программы "CUBE", приведенной на листинге 10.2. В этом случае мы не храним информацию о гранях куба, а получаем информацию с помощью оператора READ из DATA и сразу же рисуем видимые грани. Мы используем все подпрограммы предыдущей главы, которые использовались для вычерчивания на рис. 33, кроме, конечно же, конструкционной подпрограммы, устанавливающей данные о вершинах и гранях, а затем вычерчивающей их (то есть мы заменяем подпрограмму на листинге 9.4 в программе, рисующей фигуру на рис. 33). Естественно, что мы используем часть информации, использовавшейся для получения рисунка 33.

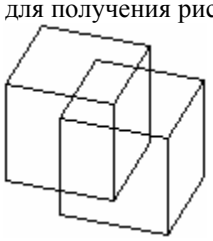


Рис. 37.

Упражнение 10.2

Измените, листинг 10.2 так, чтобы с помощью этой программы стало возможным вычерчивание изображения прямоугольного параллелепипеда длины LN, ширины BH и высоты HT, где LN, BH и HT - входные параметры подпрограммы. Затем нарисуйте тот же прямоугольник, но с удаленными невидимыми линиями. Нарисуйте тетраэдр, пирамиду, и т.д., удаляя при этом невидимые линии. Добавьте дополнительные параметры, с помощью которых можно искажать форму этого прямоугольного параллелепипеда, оставляя его при этом выпуклым.

LISTING 10.2

```
6500 REM CUBE/FACETS (NOT STORED) HIDDEN ELIMINATION
6501 REM IN - R(4,4)
6510 DATA 1,1,1,1,1,-1,1,-1,-1,1,-1,1,-1,1,1,-1,1,-1,-1,-1,-1,-1,-1,1
6520 DATA 1,2,3,4,5,8,7,6,1,5,6,2,2,6,7,3,7,8,4,1,4,8,5,1
6530 RESTORE CUBE
6539 REM PLACE VERTICES IN OBSERVED POSITION.
6540 FOR I=1 TO 8
6550 READ XX,YY,ZZ
6560 LET X(I)=XX*R(1,1)+YY*R(1,2)+ZZ*R(1,3)+R(1,4)
6570 LET Y(I)=XX*R(2,1)+YY*R(2,2)+ZZ*R(2,3)+R(2,4)
```

```
6580 NEXT I
6590 FOR I=1 TO 6
6599 REM Read facet information and draw it if oriented anti-clockwise.
6600 READ F1,F2,F3,F4
6610 LET DX1=X(F2)-X(F1): LET DY1=Y(F2)-Y(F1)
6620 LET DX2=X(F3)-X(F2): LET DY2=Y(F3)-Y(F2)
6630 IF DX1*DY2-DX2*DY1<0 THEN GO TO 6690
6640 LET XPT=X(F1): LET YPT=Y(F1): GO SUB MOVETO
6650 LET XPT=X(F2): LET YPT=Y(F2): GO SUB LINETO
6660 LET XPT=X(F3): LET YPT=Y(F3): GO SUB LINETO
6670 LET XPT=X(F4): LET YPT=Y(F4): GO SUB LINETO
6680 LET XPT=X(F1): LET YPT=Y(F1): GO SUB LINETO
6690 NEXT I
6700 RETURN
```

Тела вращения

Предложенный нами метод можно применять и для тел вращения, которые мы обсуждали в главе 9. На каждом из NUMH поворотов мы получаем NUMV граней. Если проследить за тем, чтобы при нумерации вершин этих граней соблюдалось правило обхода против часовой стрелки, то тогда применим метод предыдущего параграфа. На листинге 10.3 приведена такая программа; результат работы такой программы показан на рис. 38 - это та же фигура, что и на рис. 36, но с удалёнными невидимыми поверхностями. Опять-таки, в силу принятого нами модульного подхода все подпрограммы, необходимые для создания рис. 38, те же, что и в главе 9, за исключением процедуры "REVBOD". Теперь, однако, мы должны ограничиться только выпуклыми телами вращения.

Рис. 38

Программа, по мере вращения вокруг вертикальной оси, хранит в памяти вершины двух последовательных вертикальных наборов отрезков, которые образуют "дольку", состоящую из граней. Вершины этих граней немедленно преобразуются матрицей R, соответствующей преобразованию, переводящему объект из исходного положения в наблюдаемое положение, и записывает их в массивы X и Y. Первое множество вершин вертикальных отрезков нумеруется от 1 до NUMV+(=N1), авторов множество от N1+1 до 2*N1. Первая грань ограничена четырьмя отрезками, двумя вертикальными, соединяющими вершину I с I+1 и I+N1 с I+N1+1 и двумя вертикальными отрезками, соединяющими точки I и I+N1, I+1 с I+N1+1.

Если грань вырождается в треугольную, то в описанную выше процедуру следует внести следующие изменения. Выбранный нами порядок вершин таков, что он соответствует обходу вершин против часовой стрелки, если смотреть с внешней стороны тела. Это позволяет применить простой алгоритм скрытых линий.

Упражнение 10.3

Поэкспериментируйте с этой техникой. Для этой цели в качестве образующего набора годится любая выпуклая ломаная, оси которой лежат на оси Y.

LISTING 10.3

```
6500 REM REVBOD/CONVEX BODY OF REVOLUTION (HIDDEN LINE ELIMINATION)
6501 REM IN - PHI,NUMH,NUMV,S(NUMV+1),R(4,4),T(NUMV+1)
6510 LET THETA=PHI: LET TD=PI*2/NUMH
6520 LET N1=NUMV+1: LET C=COS PHI: LET S=SIN PHI
6529 REM CREATE FIRST VERTICAL SET.
6530 FOR I=1 TO N1
6540 LET XX=S(I)*C: LET YY=T(I): LET ZZ=S(1)*S
6550 LET X(I)=XX*R(1,1)+YY*R(1,2)+ZZ*R(1,3)+R(1,4)
6560 LET Y(I)=XX*R(2,1)+YY*R(2,2)+ZZ*R(2,3)+R(2,4)
6570 NEXT I
6579 REM LOOP THROUGH SECOND VERTICAL SET.
6580 FOR J=1 TO NUMH
6590 LET THETA=THETA+TD: LET C=COS THETA: LET S=SIN THETA
6600 FOR I=1 TO N1
6610 LET XX=S(I)*C: LET YY=T(I): LET ZZ=S(I)*S
6620 LET X(I+N1)=XX*R(1,1)+YY*R(1,2)+ZZ*R(1,3)+R(1,4)
6630 LET Y(I+N1)=XX*R(2,1)+YY*R(2,2)+ZZ*R(2,3)+R(2,4)
6640 NEXT I
```

```
6649 REM TAKE ANTICLOCKWIZE TRIANGLE FROM EACH FACET BETWEEN THE TWO SETS. IF
IT KEEPS ITS ORIENTATION ON PROJECTION THEN IT IS VISIBLE.
6650 FOR I=1 TO NUMV
6660 LET F1=I: LET F2=I+1: LET F3=F2+N1
6670 IF I=NUMV THEN LET F3=F3-1
6680 LET DX1=X(F2)-X(F1): LET DY1=Y(F2)-Y(F1)
6690 LET DX2=X(F3)-X(F2): LET DY2=Y(F3)-Y(F2)
6700 IF DX1*DY2-DX2*DY1<0 THEN GO TO 6770
6710 IF F3=F2+N1 THEN LET F4=F3-1
6720 LET XPT=X(F1): LET YPT=Y(F1): GO SUB MOVETO
6730 LET XPT=X(F2): LET YPT=Y(F2): GO SUB LINETO
6740 LET XPT=X(F3): LET YPT=Y(F3): GO SUB LINETO
6750 LET XPT=X(F4): LET YPT=Y(F4): GO SUB LINETO
6760 LET XPT=X(F1): LET YPT=Y(F1): GO SUB LINETO
6770 NEXT I
6779 REM COPY SECOND SET INTO FIRST AND REPEAT PROCESS.
6780 FOR I=1 TO N1
6790 LET X(I)=X(I+N1): LET Y(I)=Y(I+N1)
6800 NEXT I
6810 NEXT J
6820 RETURN
```

Вычерчивание поверхности специального вида, лежащей в трехмерном пространстве

В этом разделе мы приводим алгоритм, позволяющий начертить фигуру, не являющуюся выпуклой, для этого используются некоторые особенности этой фигуры. Мы ограничиваемся случаем поверхностей в трёхмерном пространстве, задаваемых с помощью однозначной функции "F", зависящей от X и Z координат; "F" вычисляется в отдельной подпрограмме; в программе, приведенной на листинге 10.4, эта функция равна Y=4*SIN(XZ)/XZ, где XZ=SQRT(X**2+Z**2); а изображение, задаваемой этой функцией поверхности, приведено на рис. 39. Необходимые входные данные: HORIZ=32, VERT=22, (EX,EY,EZ)=(3,2,1), (DX,DY,DZ)=(0,0,0), NX=NZ=16, XMIN=ZMIN=-10, XMAX=ZMAX=10.

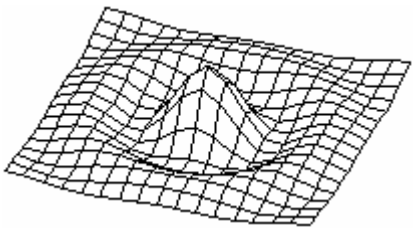


Рис. 39.

LISTING 10.4

```
6000 REM SCENE3
6010 DIM A(4,4): DIM B(4,4): DIM R(4,4)
6020 LET SURFACE=6500: LET DRAWLIN=7000: LET F=7200
6030 GO SUB IDR3: GO SUB LOOK3
6040 GO SUB SURFACE
6050 RETURN
6500 REM SURFACE
6501 REM IN - R(4,4)
6510 DIM D(256)
6520 INPUT "NX,XMIN,XMAX ";NX;",";XMIN;",";XMAX
6530 INPUT "NZ,ZMIN,ZMAX ";NZ;",";ZMIN;",";ZMAX
6540 LET XDIF=(XMAX-XMIN)/NX
6550 LET ZDIF=(ZMAX-ZMIN)/NZ
6559 REM DRAW ZERO'TH SET OF FIXED X-LINES.
6560 LET XX=XMAX: LET ZZ=ZMIN: GO SUB F
6570 LET XA=XX*R(1,1)+YY*R(1,2)+ZZ*R(1,3)+R(1,4)
6580 LET YA=XX*R(2,1)+YY*R(2,2)+ZZ*R(2,3)+R(2,4)
6590 FOR J=1 TO NZ
6600 LET ZZ=ZZ+ZDIF: GO SUB F
6610 LET XB=XX*R(1,1)+YY*R(1,2)+ZZ*R(1,3)+R(1,4)
6620 LET YB=XX*R(2,1)+YY*R(2,2)+ZZ*R(2,3)+R(2,4)
6630 GO SUB DRAWLIN
6640 LET XA=XB: LET YA=YB
6650 NEXT J
6659 REM DRAW ZERO'TH SET OF FIZED Z-LINES.
6660 FOR J=1 TO NX
6670 LET XX=XX-XDIF: GO SUB F
6680 LET XB=XX*R(1,1)+YY*R(1,2)+ZZ*R(1,3)+R(1,4)
```

```
      6690 LET YB=XX*R(2,1)+YY*R(2,2)+ZZ*R(2,3)+R(2,4)
      6700 GO SUB DRAWLIN
      6710 LET XA=XB: LET YA=YB
      6720 NEXT J
      6729 REM MOVE X VALUES BACK IN NX STEPS.
      6730 LET XS=XMAX
      6740 FOR I=1 TO NX
      6749 REM DRAW VISIBLE PARTS OF ONE FROM EACH OF THE FIXED Z-LINES. THE X VALUES
VARY FROM (I-1)'ST X-LINE TO THE I'TH.
      6750 LET ZS=ZMAX
      6760 FOR J=1 TO NZ
      6770 LET ZS=ZS-ZDIF
      6780 LET XX=XS: LET ZZ=ZS: GO SUB F
      6790 LET XA=XX*R(1,1)+YY*R(1,2)+ZZ*R(1,3)+R(1,4)
      6800 LET YA=XX*R(2,1)+YY*R(2,2)+ZZ*R(2,3)+R(2,4)
      6810 LET XX=XS-XDIF: GO SUB F
      6820 LET XB=XX*R(1,1)+YY*R(1,2)+ZZ*R(1,3)+R(1,4)
      6830 LET YB=XX*R(2,1)+YY*R(2,2)+ZZ*R(2,3)+R(2,4)
      6840 GO SUB DRAWLIN
      6850 NEXT J
      6859 REM DRAW VISIBLE PARTS OF THE FIXED X-LINES.
      6860 LET ZZ=ZMIN: GO SUB F
      6870 LET XA=XX*R(1,1)+YY*R(1,2)+ZZ*R(1,3)+R(1,4)
      6880 LET YA=XX*R(2,1)+YY*R(2,2)+ZZ*R(2,3)+R(2,4)
      6890 FOR J=1 TO NZ
      6900 LET ZZ=ZZ+ZDIF: GO SUB F
      6910 LET XB=XX*R(1,1)+YY*R(1,2)+ZZ*R(1,3)+R(1,4)
      6920 LET YB=XX*R(2,1)+YY*R(2,2)+ZZ*R(2,3)+R(2,4)
      6930 GO SUB DRAWLIN
      6940 LET XA=XB: LET YA=YB
      6950 NEXT J
      6960 LET XS=XS-XDIF
      6970 NEXT I
      6980 RETURN
      7000 REM DRAWLIN/DRAW LINE BETWEEN POINTS
      7001 REM IN - XA,YA,XB,YB,D(256)
      7009 REM VISIBLE PARTS OF LINE BETWEEN (XA,YA) AND (XB,YB). IA AND IB ARE THE
X-PIXEL POSITIONS OF THE TWO POINTS.
      7010 LET IA=FN X(XA):LET IB=FN X(XB)
      7020 LET Y=YA: LET YD=0
      7030 IF IA<>IB THEN GO TO 7080
      7040 PLOT IA,D(IA)
      7050 LET IY=FN Y(YB)
      7060 IF IY>D(IA) THEN DRAW 0,IY-D(IA): LET D(IA)=IY
      7070 RETURN
      7079 REM MOVE IN PIXEL COLUMNS FROM LEFT TO RIGHT.
      7080 LET YD=(YB-YA)/(IB-IA)
      7090 FOR K=IA TO IB
      7100 LET IY=FN Y(Y)
      7109 REM IF Y-PIXEL IS GREATER THAN D, THEN RESET D-VALUE.
      7110 IF D(K)<IY THEN LET D(K)=IY
      7120 LET Y=Y+YD
      7130 NEXT K
      7139 REM JOIN ALL THE POINTS (I,D(D) WHERE IA<=K<=IB.
      7140 PLOT IA,D(IA)
      7150 FOR K=IA+1 TO IB
      7160 DRAW 1,D(K)-D(K-1)
      7170 NEXT K
      7180 RETURN
      7200 REM F/FUNCTION TO BE DRAWN
      7201 REM IN - XX,ZZ
      7202 REM OUT - YY
      7210 LET YY=4: LET XZ=SQR(XX*XX+ZZ*ZZ)
      7220 IF XZ>0.000001 THEN LET YY=4*SIN(XZ)/XZ
      7230 RETURN
```

Так как невозможно нанести все точки поверхности, то мы должны выделить подмножество этих точек,

которые мы выбираем в узлах сетки в плоскости X/Y; другими словами, если смотреть на поверхность сверху, т.е. спроецировать её на ось X/Z, отбросив значения Y-координат, то эти точки образуют прямоугольную сетку. Эта сетка состоит из прямоугольников размера NX на NZ, а X-координаты вершин прямоугольников расположены в интервале между XB и XT (XB<XT), и Z-координаты расположены в диапазоне от ZB до ZT (ZB<ZT). Таким образом, сетка состоит из (NX+1) X (NZ+1) вершин (X,Z), которые могут быть отождествлены с парой целых чисел (I,J) при помощи формулы:

`X=XT+I*(XB-XT)/NX`, где 0<=I<=NX

`Z=ZT+J*(ZB-ZT)/NZ`, где 0<=J<=NZ

Точке на плоскости соответствует точка (X,Y,Z) на поверхности, где Y=F(X,Z). Каждая из (NX+1)×(NZ+1) точек сетки соединяется с каждым из своих четырёх непосредственных соседей по сетке (у этих точек одинаковые значения X или Z координат), за исключением точек, лежащих на границе сетки, которые соединяются с тремя точками, а в случае угловых точек - с двумя. Прямые, образующие сетку, можно считать состоящими из NX+1 отрезка, параллельного оси Y. Например, первый набор линий с фиксированным X состоит из NZ-прямых, соединяющих точки, соответствующие точкам от (I,J) до (I,J+1), где 0<=J<=NZ-1.

Поверхность может образовывать складки, так что не все точки поверхности будут видны с заданного положения глаза наблюдателя. Мы приводим простой алгоритм удаления невидимых линий, который просматривает точки поверхности от фронтальной части к тыльной.

Для упрощения алгоритма мы считаем, что глаз всегда находится в положительном квадранте (это означает, что EX>0, EY>0) и что взгляд направлен в начало координат (DX=DY=DZ-0). Если функция не симметрична, и мы хотим рассматривать её из другого квадранта, то для этого нам следует сменить знак Y, X или Z в выражении, определяющем функцию. После этого мы можем перевести поверхность в наблюдаемое положение. В этом положении мы сначала ортогонально проецируем на визуальную плоскость переднюю границу - две линии сетки с X=XZ и Z=ZT, соответствующие осям X и Y, после чего мы продвигаемся за NX шагов от переднего края к заднему. На I-ом шаге (1<=I<=NX) мы сначала рисуем видимые части отрезков, составляющих NZ Z-линий сетки. Другими словами, начиная с передней Z-линии сетки и продвигаясь назад, мы соединяем точки, лежащие на I-ой X-линии с точками, лежащими на J-ой X-линии сетки. После этого мы чертим видимые части набора отрезков, составляющих I-ую линию. Этот алгоритм реализован в программе "SURFACE".

До сих пор мы не объясняли, как чертить видимые части линий; эту работу выполняет подпрограмма "DRAWLIN". Мы определяем массив D, содержащий 256 элементов, каждый из которых соответствует столбцу элементов изображения на экране; затем чертим нулевую прямую с фиксированным X и нулевую прямую с фиксированным Z линиями. Для каждой точки этих линий заносим в массив D координаты элемента изображения, соответствующего этим точкам. Для каждого элемента изображения, лежащего на данном отрезке, мы вычисляем значение строки и столбца и наносим этот элемент, если только значение столбца больше, чем значение, записанное в массиве. Каждый раз, когда к рисунку добавляется новый элемент изображения, изменяется значение соответствующего элемента массива D.

В данном алгоритме существенно, в каком порядке мы рассматриваем прямые. Если мы начинаем с положительного квадранта или если поверхность выходит за границы экрана, то наш алгоритм непригоден.

Упражнение 10.4

Измените функцию "F", используемую в программе. Например, используйте F=4*SIN(T), где T=SQRT(X**2+Y**2).

Упражнение 10.5

Наша программа не позволяет изобразить на экране те куски поверхности, которые лежат ниже нулевых X и Z линий. Измените программу на листинге 10.4 с тем, чтобы исправить этот недостаток. Для этого введите массив E, который первоначально содержит информацию о ближайших краях, а когда значение элемента изображения оказывается меньше того, что содержится в массиве E, то этот элемент наносится на экран, и производится соответствующеё изменение в массиве E.

Список программ

1. "LIB3" и листинг 10.1. Необходимые данные: координаты X(I), Y(I), Z(I) вершин треугольников, где 1<I<3. Попробуйте ввести значения 1, 0, 1, 1, 1, 0, 0, 1, 0, а также эти вершины в другом порядке -1, 1, 0, 1, 0, 1 ,0, 1, 1.

2. "LIB1", "LIB3" и листинги 9.5 ("SCENE3") и 10.2 ("CUBE"). Необходимая информация: HORIZ, VERT, (EX,EY,EZ), (DX,DY,DZ). Попробуйте ввести значения 9, 6, (1,2,3), (0,0,-1).

3. "LIB1", "LIB3" и листинг 9.10 ("SCENE3") и 10.3 ("REVBOD"). Входные данные: HORIZ, VERT, NUMH, NUMV (<=15), PHI, (EX,EY,EZ), (DX,DY,DZ). Попробуйте ввести значения 3.2, 2.2, 10, 6, 1, (1,2,3), (0,0,-1).

4. "LIB1", "LIB3" и листинг 10.4 ("SCENES", "SURFACE", "DRAWLIN" и "F"). Входные данные: HORIZ, VERT, (EX,EY,EZ), (DX,DY,DZ), NX, XMIN, XMAX, NZ, ZMIN, ZMAX. Попробуйте ввести значения 30, 20, (1,2,3), (0,0,0), 16, -8, 8, 18, -8, 16.

ГЛАВА 11
Перспективная проекция

Мы знаем, что при ортогональном проецировании параллельность прямых сохраняется. Хотя изображения, полученные при таком проецировании, чрезвычайно полезны, выглядят они несколько необычно. Мы привыкли к перспективному изображению предметов, и когда мы видим ортогональную проекцию, то подсознательно интерпретируем её, как перспективное изображение некоего предмета. Например, кубы на рис. 33 и 37 кажутся деформированными.

По этой причине необходимо построить проекцию, учитывающую эффект перспективы, (который заключается в том, что параллельные прямые пересекаются на горизонте); предметы должны становиться меньше по мере их удаления от наблюдателя.

Разработанные художниками методы перспективы с помощью специальных планшетов для нас непригодны, однако методы трёхмерной геометрии и переход от фактического положения к наблюдаемому оказывается весьма полезным.

Что такое перспективное видение?

Для получения перспективного изображения мы дадим очень простое определение того, что мы подразумеваем под визуальным восприятием. Мы представляем, что каждая видимая точка испускает луч, который попадает в глаз. Глаз видит только те предметы, которые попадают в так называемый зрительный конус. Ось этого конуса называется направлением взгляда. Мы считаем, что пространство таково, что линия взгляда совпадает с осью.

Мы помещаем визуальную плоскость (которая в этом случае называется перспективной плоскостью) перпендикулярно оси зрительного конуса на расстоянии D от глаза. Для того чтобы построить перспективную проекцию, мы отмечаем точки пересечения лучей с этой плоскостью. Так как лучей бесконечно много, то практически эта задача невыполнима. Задача упрощается, если наносить на плоскость только те точки, которые соответствуют наиболее важным точкам тела: вершинам, концам отрезков, угловым точкам граней, ограниченных многоугольниками. Перспективное изображение получится, если мы соединим проекции этих точек таким же образом, каким были соединены сами точки, и отождествим визуальную плоскость с плоскостью экрана.

Построение перспективной проекции точки

Пусть визуальная плоскость расположена на расстоянии D от глаза (в дальнейшем в программах этой величине будет соответствовать переменная PPD). Рассмотрим точку P=(X,Y,Z), которая испускает луч, попадающий в глаз наблюдателя. Мы должны вычислить точку пересечения этого луча с визуальной плоскостью (с плоскостью Z=D); предположим, что это точка P'=(X',Y',Z'). Вычислим сначала значение Y'. Из подобия треугольников имеем Y'/D=Y/Z и, следовательно, Y'=Y*D/Z. Аналогичным образом получим X'=X*D/Z. Следовательно P'=/X*D/Z, Y*D/Z, D/. Так как визуальная плоскость отождествляется с плоскостью X/Y, то значение координаты Z, равное D, можно отбросить.

Пример 11.1

Постройте перспективную проекцию куба с восьмью вершинами (0,0,4)+(±1,±1,±1) на перспективную плоскость Z=4, с глазом, расположенным в начале координат, и направлением взгляда вдоль оси Z.

Точки в пространстве определены так, что объект находится в положении наблюдаемое. Пользуясь изложенным выше методом, мы можем вычислить проекции этих точек на визуальную плоскость. Так, например, точка (1,1,3) проецируется в точку (1*4/3,1*4/3,4), то есть в точки визуальной плоскости с координатами (4/3,4/3). Остальные точки проецируются:

(1,1,3)→(4/3,4/3);	(1,-1,3)→(4/3,4/3);	(-1,1,3)→(-4/3,4/3);
(-1,1,3)→(-4/3,4/3);	(1,-1,5)→(4/5,4/5);	(1,-1,5)→(4/5,4/5);
(-1,1,5)→(-4/5,4/5);	(-1,-1,5)→(4/5,4/5);	(-1,-1,5)→(4/5,4/5)

Свойства перспективной проекции

1. Перспективная проекция прямой линии (обозначим её Г3) есть прямая линия (обозначим её Г2). Это очевидно, т.к. глаз и прямая Г3 лежат в одной плоскости (обозначим её Л) и все лучи, исходящие из точек этой прямой и попадающие в глаз, лежат в этой плоскости (если прямая проходит через точку, в которой расположен глаз, то плоскость Л вырождается в прямую). Плоскость Л пересекает визуальную плоскость по прямой (которая может выродиться в точку) и теперь проекция точки, лежавшей на прямой Г3, лежит на прямой Г2. Важно помнить, что при перспективном проецировании прямые не искривляются.

2. Перспективная проекция грани (гранью называется плоская область, ограниченная замкнутой ломаной) есть грань на перспективной плоскости. Если грань ограничена ломаной, состоящей из N сегментов, то и перспективная проекция этой грани будет гранью, ограниченной ломаной, состоящей из такого же числа сегментов. Отметим, что и в этом случае не появляется никаких кривых.

3. Проекция выпуклой грани есть выпуклая грань. Предположим, что грань P1 проектируется в грань P2. Так как проекция замкнутой ломаной замкнута, то внутренние точки P1 проецируются во внутренние точки P2. Предположим, что P2 не выпукла: тогда существуют две точки P1 и P2, лежащие внутри P2, такие, что отрезок, соединяющий их, содержит точку P. не являющуюся внутренней точкой грани P2.

Если точки P1 и P2 являются проекциями точек Q1 и Q2, то точка P является проекцией некоторой точки Q, лежащей на отрезке, соединяющем точки Q1 и Q2. Так как P1 выпукла, то Q должна лежать внутри P1, а, следовательно, и её проекция P должна лежать внутри P2. Мы пришли к противоречию, что доказывает наше утверждение.

4. Если параллельные прямые продолжать до бесконечности, то они пересекутся в так называемой

бесконечно удаленной точке. Рассмотрим прямую с базовой точкой P и направляющим вектором H. Эта прямая принадлежит семейству прямых, параллельных H.

P+MU*H=(XP,YP,ZP)+MU*(XH,YH,ZH), где ZH>0,

тогда перспективная проекция произвольной точки, лежащей на этой прямой есть:

```
((XP+MU*XH)*D)/(ZP+MU*ZH),
((YP+MU*YH)*D)/(ZP+MU*ZH)
```

Это выражение можно переписать как

```
((XH+XP/MU)*D)/(ZH+ZP/MU),
((YH+YP/MU)*D)/(ZH+ZP/MU)
```

По мере того, как мы перемещаемся в направлении больших значений координаты (что эквивалентно MU → к бесконечности), проекция стремится к точке (D*XH/ZH,D*YH/ZH). Бесконечно удаленная точка не зависит от вектора P, базисного вектора прямой, и, следовательно, все прямые, параллельные H, пересекаются в одной бесконечно удаленной точке. Мы не рассматриваем случай, когда ZH<0, так как при MU → к бесконечности прямая выходит из визуального конуса.

На листинге 11.1 приведен пример конструирующей программы, а на листинге 11.2 - пример рисующей программы для построения перспективной проекции.

LISTING 11.1

```
6000 REM SCENE3/EXAMPLE2 (VARIETY OF VIEWS)
6010 DIM X(16): DIM Y(16): DIM Z(16)
6020 DIM V(16): DIM W(16): DIM L(2,24)
6030 DIM A(4,4): DIM B(4,4): DIM R(4,4)
6040 LET CUBE=6500: LET DRAWIT=7000
6050 LET PPD=3*VERT: LET NOV=0: LET NOL=0
6059 REM PLACE TWO CUBES IN ACTUAL POSITION.
6060 GO SUB IDR3
6070 GO SUB CUBE
6080 LET TX=3: LET TY=1.5: LET TZ=2: GO SUB TRAN3: GO SUB MULT3
6090 GO SUB CUBE
6099 REM LOOP THROUGH VARIETY OF VIEWS.
6100 GO SUB IDR3: GO SUB LOOK3
6110 CLS: GO SUB DRAWIT
6120 GO TO 6100
6130 RETURN
```

LISTING 11.2

```
7000 REM DRAWIT/PERSPECTIVE
7001 REM IN - PPD,NOV,NOL,R(4,4),X(NOV),Y(NOV),Z(NOV),L(2,NOL)
7009 REM PUT VERTICES IN OBSERVED POSITION.
7010 FOR I=1 TO NOV
7020 LET XX=X(I)*R(1,1)+Y(I)*R(1,2)+Z(I)*R(1,3)+R(1,4)
7030 LET YY=X(I)*R(2,1)+Y(I)*R(2,2)+Z(I)*R(2,3)+R(2,4)
7040 LET ZZ=X(I)*R(3,1)+Y(I)*R(3,2)+Z(I)*R(3,3)+R(3,4)
7049 REM PERSPECTIVE PROJECTION.
7050 LET V(I)=XX*PPD/ZZ
7060 LET W(I)=YY*PPD/ZZ
7070 NEXT I
7079 REM DRAW LINES.
7080 FOR I=1 TO NOL
7090 LET L1=L(1,I): LET L2=L(2,I)
7100 LET XPT=V(L1): LET YPT=W(L1): GO SUB MOVETO
7110 LET XPT=V(L2): LET YPT=W(L2): GO SUB LINETO
7120 NEXT I
7130 RETURN
```

Упражнение 11.1

Нарисуйте перспективное изображение тетраэдра или пирамиды.

Выбор визуальной плоскости

Единственным параметром, определяющим положение плоскости, который мы до сих пор не обсуждали, является величина PPD, равная расстоянию от визуальной плоскости до глаз наблюдателя. Мы видели, что этот параметр определяет величину проекции. Каково должно быть значение этой величины? Имеется ли какое-либо выделенное значение этой величины?

Когда наблюдатель сидит напротив экрана дисплея, то визуальную плоскость можно отождествить с плоскостью экрана. Как правило, расстояние от наблюдателя до экрана примерно в три раза больше высоты экрана. Возвращаясь к нашим обозначениям, это означает, что расстояние от глаза до визуальной плоскости равно 3*VERT. Если мы выбираем PPD большим этого значения, то мы получаем увеличенное изображение PPD, меньше 3*VERT, то мы получим уменьшенное изображение.

Кадрирование изображения

Теоретически объекты могут быть расположены в любом месте пространства, даже за глазом наблюдателя, однако мы рассматриваем только точки с положительными значениями Z - координаты в наблюдаемом положении. Даже в этом случае некоторые точки становятся невидимыми, так как не попадают в зрительный конус. На самом деле мы не можем отобразить весь зрительный конус на экране и ограничиваемся изображением части этого зрительного конуса - зрительной пирамиды. Таким образом, все точки, лежащие вне зрительной пирамиды (что означает, что их перспективное изображение лежит за пределами экрана), можно не принимать во внимание.

Мы отмечали, что SPECTRUM выдает сообщение об ошибке, как только мы пытаемся начертить с помощью оператора DRAW прямую, содержащую точки, лежащие за пределами экрана. Для того, чтобы избежать подобных ошибок, мы пользуемся вариантом программы "LINETO", использующей процедуру кадрирования (листинг 3.4). В дальнейшем мы ограничиваемся изображением только таких фигур, у которых Z-координаты всех вершин положительны; это означает, что все объекты должны находится перед наблюдателем.

Упражнение 11.2

Поупражняйтесь с перспективным изображением каркасных фигур различных типов; например, тел вращения, правильных тел.

Рассмотрите случай, когда объект создается конструирующей программой. Это означает, что значения V и W вычисляются в этих подпрограммах, а не в программе "DRAWIT". Измените программу, которая рисовала самолёт, так, чтобы получить её перспективное изображение; обратите внимание, что чем больше удален глаз от самолёта, тем меньшим кажется последний - эффект, который отсутствует при ортогональном проектировании.

Упражнение 11.3

Напишите алгоритм удаления невидимых линий, аналогичный приведенному в главе 10.

Обратите внимание, что, так как выпуклая грань проецируется в виде выпуклого многоугольника на визуальной плоскости, то всё, что нам надо сделать - это вычислить координаты проекций вершин граней, а затем проверить, обходится ли грань против часовой стрелки (в этом случае мы отображаем её на экране) или по часовой стрелке (в этом случае мы её не изображаем на экране).

Упражнение 11.4

Напишите программу, рисующую перспективное изображение поверхности функции, приведенной в главе 10. Эта программа совпадает с программой, приведенной на листинге 10.4, за тем исключением, что следует работать со значениями V и W, а не с массивами X и Y.

Эти алгоритмы дают хорошие результаты в случае одиночных объектов, однако нам теперь предстоит рассмотреть более общий случай, когда в пространстве расположены несколько предметов.

Список программ

1. "LIBI", "LIB3", и листинги 9.7 ("CUBE"), 11.1 ("SCENES") и 11.2 ("DRAWIT"). Входные параметры: HORIZ, VERT и множество значений (EX,EY,EZ) и (DX,DY,DZ). Попробуйте значения 9, 6, (5,15,10) и (0,0,0); (1,2,20) и (0,0,1).

ГЛАВА 12
Общий алгоритм удаления невидимых линий

Как и в предыдущих главах, мы полагаем, что объекты создаются программой "SCENE3", но теперь мы требуем, чтобы NOV вершин, принадлежащих изображаемым объектам, хранились в массивах X, Y, Z. Их перспективная проекция на визуальную плоскость хранится в массивах V и W. NOF граней хранятся в виде списка принадлежащих им вершин в массиве F (допускаются грани, содержащие не более шести вершин), в массиве H хранится число вершин каждой грани.

Мы считаем, что все объекты замкнуты. Не требуется выпуклость тел, однако мы предполагаем, что все их грани выпуклы, и они обходятся против часовой стрелки. При выполнении этих условий наблюдатель ни при каких условиях не может увидеть внутреннюю сторону грани, что означает, что при проекции на визуальную плоскость мы видим те грани, которые при проецировании сохраняют направление обхода вершин (против часовой стрелки). Строго говоря, это означает, что мы не можем рисовать плоские объекты. Если в какой-либо композиции требуется нарисовать такой объект, то каждую его грань следует записать дважды - один раз грань обходится по часовой стрелке, а другой раз - против, поэтому вне зависимости от положения глаза наблюдателя он видит один и только один из экземпляров каждой грани.

Мы также полагаем, что любой из отрезков, из которых состоит рисунок, получается в результате пересечения двух смежных граней: если требуется изобразить отдельную линию, то её следует представить, как вырожденную грань, ограниченную двумя отрезками. Эти ограничения введены для ускорения алгоритма удаления невидимых линий. Это очень важно, так как мы теперь приближаемся к предельным возможностям SPECTRUMa. Даже для такого простого рисунка, как два куба на рис. 40, требуется около пяти минут машинного времени, а для более сложных изображений ещё больше. SPECTRUM не был рассчитан на решение таких сложных задач, и то, что нам удается это сделать, следует считать показателем высокого качества работы проектировщиков SPECTRUMa.

Тем не менее, мы считаем, что важно изучить общий алгоритм удаления невидимых линий, это способствует повышению квалификации, так как читатель, желающий специализироваться в области машинной графики, обязан иметь представление о том, какие проблемы возникают при изображении трёхмерных объектов с невидимыми линиями. В программе, приведенной на листинге 12.1, реализован такой алгоритм удаления невидимых линий, который даёт отличные результаты на более мощных компьютерах. Если вы имеете доступ к более мощным компьютерам, то вам будет чрезвычайно полезно реализовать эту программу на таком компьютере.

Для того чтобы нарисовать изображение сцены, записанной в памяти в наблюдаемом положении, каждая линия, принадлежащая какому-либо из изображаемых объектов, должна сравниваться с каждой гранью. В силу принятых выше ограничений мы производим эти сравнения только для видимых граней, то есть тех, которые при проецировании сохраняют ориентацию против часовой стрелки.

Предположим, что типичная прямая Г3' в наблюдаемом положении соединяет две точки (X1',Y1',Z1') с (X2',Y2',Z2'), тогда координаты точки, лежащей на этой прямой, есть

(1-Ф)*(X',Y1',Z1')+Ф*(X2',Y2',Z2')

Предположим, что две концевые точки проецируются в точки (X1,Y1) и (X2,Y2) визуальной плоскости. Тогда Г3 проецируется на визуальную плоскость как прямая Г2, а координаты точек, лежащих на этой прямой есть

(1-MU)*(X1,Y1)+MU*(X2,Y2)

Обратите внимание, что ниоткуда не вытекает, что точка (1-Ф)*(X1',Y1',Z1')+ Ф*(X2',Y2',Z2') проецируется в точку (1-Ф)*(X1,Y1)+Ф*(X2,Y2), это означает, что Ф не равно MU.

Пусть грань общего положения ОМЕГА1 проецируется на область ОМЕГА2, лежащую в визуальной плоскости. Предположим, что H вершин проекции грани есть (XI,YI), где 1<=I<=H. Пусть 1-ое ребро ОМЕГА2 пересекает Г2 в точке (1- ЛI)*(XI,YI)+ЛI*(X(I+1),Y(I+1)). Если ЛI<0 или ЛI>1, то это означает, что Г2 пересекает продолжение I-го ребра в точке, лежащей за пределами области ОМЕГА2: если 0<=ЛI<=1, тогда Г2 пересекает область ОМЕГА2 в точке, лежащей на I-ом ребре. Так как проекция выпуклой грани выпукла, то число точек пересечения либо равно нулю, или равно двум (возможно совпадающим точкам).

Нам требуется рассмотреть только случай несовпадающих точек: предположим, что соответствующие этим точкам значения параметра MU есть MU(MIN) и MU(MAX), где MU(MIN)<MU(MAX). Тогда точки пересечения с Г2 есть

(1-MU(MIN))*(X1,Y1)+ MU(MIN)*(X2,Y2) и
(1-MU(MAX))*(X1,Y1)+ MU(MAX)*(X2,Y2).

Теперь необходимо установить, является ли отрезок, заключенный между этими точками, видимым или нет. Это проверяется нахождением середины этого отрезка - точки

(X(MID),Y(MID))=(1-MU(MID))*(X1,Y1)+MU(MID)*(X2,Y2),

где

MU(MID)=(MU(MIN)+MU(MAX))/2.

Затем мы находим точку (X,Y,Z), лежащую на Г3 такую, что перспективной проекцией этой точки будет (X(MIN),Y(MIN)). Отрезок линии между точками, со значениями параметра MU, равными MU(MIN) и MU(MAX) является невидимым, если и только если точка (X,Y,Z) и глаз наблюдателя расположены по разные стороны плоскости, в которой лежит ОМЕГА3. Уравнения этой плоскости можно найти с помощью методов главы 1, а затем применить для проверки приведенного выше условия функциональное представление этой плоскости.

Обратите внимание, что X*PPD/Z=X(MID), Y*PPD/Z=Y(MID) и точка (X,Y,Z) лежит на Г3. Поэтому для некоторого значения параметра Ф:

X=(1-Ф)*X1'+Ф*X2', Y=(1-Ф)*Y1'+Ф*Y2' и **Z=(1-Ф)*Z1'+Ф*Z2';**

так как

X(MID)=((X1'+Ф*(X2'—X1'))*PPD)/(Z1'+Ф*(Z2'—Z1')) и

Y(MID)=((Y1'+Ф*(Y2'—Y1'))*PPD)/(Z1'+Ф*(Z2'—Z1')),

то отсюда следует, что

Ф=(X(MID)*Z1'—X1'*PPD)/((X2'—X1')*PPD—X(MID)*(Z2'-Z1'))=
(Y(MID)*Z1'—Y1'*PPD)/((Y2'—Y1')*PPD—Y(MID)*(Z2'-Z1'))

Это позволяет нам вычислить Ф, а следовательно и точку (X,Y,Z), которая в свою очередь используется для проверки того, видима ли часть отрезка Г2 или нет.

В алгоритме, реализованном в программе "HIDDEN" на листинге 12.1, каждая линия, принадлежащая какому-либо объекту, сравнивается со всеми видимыми гранями (теми, проекции которых обходятся против часовой стрелки). Обратите внимание, что мы считаем, что все объекты непрозрачны, то есть невозможно увидеть внутреннюю поверхность ни одной из граней, т.е. увидеть грань, проекция которой обходится по часовой стрелке. Информация о ребрах содержится в информации о гранях, причём каждое ребро встречается дважды, один раз оно проходит от точки IV1 к точке IV2. Для того чтобы избежать повторений, мы рассматриваем только такие ребра, у которых IV1<IV2. Мы сопоставляем ребра только с видимыми гранями, так в силу наложенных на объекты ограничений, если ребро заслонено от наблюдателя невидимой гранью, то оно должно быть отделено от наблюдателя так же и видимой гранью.

Предположим, что на некотором этапе процедуры сравнения для прямой Г2, соединяющей вершины K-ой грани IV1 и IV2, мы нашли NRL видимых подотрезков данного отрезка; мы полагаем, что NRL<50. Значения параметра MU, соответствующего концевым точкам M-того видимого отрезка, заносятся в массив L как L(1,M) и L(2,M).

В начале процедуры мы полагаем NRL=1, L(1,1)=0 и L(2,1)=1. что означает, что мы считаем прямую видимой. Если на этой стадии обнаруживается новый невидимый отрезок, определяемый значениями MU(MIN) и MU(MAX) (переменные MAX и MIN), то эти значения заносятся в массив L, и NRL соответствующим образом изменяет своё значение.

Когда прямая сравнивается со всеми видимыми гранями, то в результате у нас NRL видимых отрезков, которые можно нанести на экран. Если в какой-нибудь момент NRL принимает значение 0, то это означает, что весь отрезок является невидимым, а следовательно можно прекратить процедуру сравнения для этого отрезка.

Пример 12.1

Теперь мы можем удалить невидимые линии на рис. 37; результат показан на рис. 40. Значение входных параметров: HORIZ=9, VERT=6, глаз наблюдателя расположен в точке (15,10,15), взгляд направлен в (0,0,0).

Рис. 40.

LISTING 12.1

```
      7000 REM HIDDEN/GENERAL HIDDEN LINE ALGORITM
      7001 REM IN - NOV,NOL,PPD,R(4,4),X(NOV),Y(NOV),Z(NOV),V(NOV),W(NOV),H(NOF),
F(6,NOF)
      7010 DIM L(2,50): DIM G(NOF): LET EPS=0.000001
      7019 REM CHECK ON I'TH PROJECTED FACET-ANTICLOCKWISE SET G(I)=1, CLOCKWISE OR
DEGENERATE SET G(I)=0.
      7020 FOR I=1 TO NOF
      7030 LET I1=F(1,I): LET X1=V(I1): LET Y1=W(I1)
      7040 LET I2=F(2,I): LET X2=V(I2): LET Y2=W(I2)
      7050 LET I3=F(3,I): LET X3=V(I3): LET Y3=W(I3)
      7060 LET DX1=X2-X1: LET DY1=Y2-Y1
      7070 LET DX2=X3-X2: LET DY2=Y3-Y2
      7080 LET G(I)=0
      7090 IF DX1*DY2-DX2*DY1>0 AND H(I)>2 THEN LET G(I)=1
      7100 NEXT I
      7109 REM FIND J'TH LINE ON THE EDGE OF I'TH FACET.
      7110 FOR I=1 TO NOF
      7120 LET HH=H(I): LET IV1=F(HH,I)
      7130 LET X1=V(IV1): LET Y1=W(IV1)
      7140 FOR J=1 TO HH
      7150 LET IV2=F(J,I)
      7160 LET X2=V(IV2): LET Y2=W(IV2)
      7170 IF IV1>IV2 THEN GO TO 8160
      7179 REM INITIALISE VARIABLES.
      7180 LET NRL=1: LET L(1,1)=0: LET L(2,1)=1
      7190 LET CA=X2-X1: LET CB=Y1-Y2
```

```
      7200 LET CC=-X1*CB-Y1*CA
      7209 REM COMPARE THIS LINE WITH THE K'TH FACET.
      7210 FOR K=1 TO NOF
      7220 IF G(K)=0 THEN GO TO 8040
      7230 IF K=1 THEN GO TO 8040
      7240 LET in=H(K)
      7249 REM LOOP TO FIND TWO POINTS OF INTERSECTION OF PROJECTED LINE WITH
PROJECTED FACET. THESE POINTS SPECIFIED BY MU VALUES MIN AND MAX.
      7250 LET MAX=-1: LET MIN=2
      7260 LET JV1=F(in,K)
      7270 LET VX1=V(JV1): LET WY1=W(JV1)
      7280 LET S1=SGN(CA*WY1+CB*VX1+CC)
      7290 FOR M=1 TO in
      7300 LET JV2=F(M,K)
      7310 LET VX2=V(JV2): LET WY2=W(JV2)
      7320 LET S2=SGN(CA*WY2+CB*VX2+CC)
      7330 IF S1=S2 THEN GO TO 7500
      7340 LET XE=VX1-VX2: LET YE=WY1-WY2
      7350 LET XF=VX1-X1: LET YF=WY1-Y1
      7360 LET DISC=CA*YE+CB*XE
      7369 REM IF LINE IS PARALLEL TO A LINE ON THE FACET THEN EXIT FACET LOOP.
      7370 IF ABS DISC>EPS THEN GO TO 7440
      7380 IF ABS CA>EPS THEN GO TO 7410
      7390 IF ABS XF<EPS THEN GO TO 8040
      7400 GO TO 7500
      7410 LET LAMBDA=XF/CA
      7420 IF ABS (XF+LAMBDA*CB)<EPS THEN GO TO 8040
      7430 GO TO 7500
      7440 LET LAMBDA=(CA*YF+CB*XF)/DISC
      7449 REM IF LINE MISSES K'TH FACET THEN GO TO NEXT FACET.
      7450 IF LAMBDA<-EPS THEN GO TO 7500
      7460 IF LAMBDA>1+EPS THEN GO TO 7500
      7470 LET MU=(YE*XF-XE*YF)/DISC
      7479 REM A TRUE INTERSECTION SO UPDATE MAX AND MIN.
      7480 IF MAX<MU THEN LET MAX=MU
      7490 IF MIN>MU THEN LET MIN=MU
      7500 LET S1=S2
      7510 LET VX1=VX2: LET WY1=WY2
      7520 NEXT M
      7529 REM CHECK IF INTERSECTIONS LIE BETWEEN SPECIFIED END POINTS OF LINE.
      7530 IF MIN>1 THEN GO TO 8040
      7540 IF MAX<0 THEN GO TO 8040
      7550 IF MAX>1 THEN LET MAX=1
      7560 IF MIN<0 THEN LET MIN=0
      7570 IF MAX-MIN<EPS THEN GO TO 8040
      7579 REM CALCULATE XMID AND YMID
      7580 LET MID=(MAX+MIN)*0.5: LET MUD=1-MID
      7590 LET XMID=MUD*X1+MID*X2
      7600 LET YMID=MUD*Y1+MID*Y2
      7610 LET DENOM=PPD*(X(IV2)-X(IV1))-XMID*(Z(IV2)-Z(IV1))
      7620 IF ABS DENOM<EPS THEN GO TO 7650
      7629 REM CALCULATE PHI AND HENCE XHAT,YHAT,ZHAT.
      7630 LET PHI=(XMID*Z(IV1)-PPD*X(IV1))/DENOM
      7640 GO TO 7670
      7650 LET DENOM=PPD*(Y(IV2)-Y(IV1))-YMID*(Z(IV2)-Z(IV1))
      7660 LET PHI=(YMID*Z(IV1)-PPD*Y(IV1))/DENOM
      7670 LET ZHAT=(1-PHI)*Z(IV1)+PHI*Z(IV2)
      7680 LET FACT=ZHAT/PPD
      7690 LET XHAT=XMID*FACT: LET YHAT=YMID*FACT
      7699 REM CALCULATE COEFFICIENTS OF PLANE CONTAINING FACET - A,B,C,D.
      7700 LET JV1=F(1,K): LET JV2=F(2,K): LET JV3=F(3,K)
      7710 LET DX1=X(JV1)-X(JV2)
      7720 LET DX3=X(JV3)-X(JV2)
      7730 LET DY1=Y(JV1)-Y(JV2)
      7740 LET DY3=Y(JV3)-Y(JV2)
      7750 LET DZ1=Z(JV1)-Z(JV2)
```

```
7760 LET DZ3=Z(JV3)-Z(JV2)
7770 LET A=DY1*DZ3-DY3*DZ1
7780 LET B=DZ1*DX3-DZ3*DX1
7790 LET C=DX1*DY3-DX3*DY1
7800 LET D=A*X(JV1)+B*Y(JV1)+C*Z(JV1)
7810 LET S1=A*XHAT+B*YHAT+C*ZHAT-D
7819 REM IF FACET HIDES PART OF LINE THEN CHANGE THE L ARRAY.
7820 IF ABS S1 <EPS THEN GO TO 8040
7830 IF ABS (SGN S1+SGN D)<2 THEN GO TO 8040
7840 LET MORE=NRL
7850 FOR M=1 TO NRL
7860 LET R1=L(1,M): LET R2=L(2,M)
7870 IF (R1>MAX) OR (R2<MIN) THEN GO TO 7960
7880 IF (R1>=MIN) AND (R2<=MAX) THEN GO TO 7950
7890 IF (R1<MIN) AND (R2>MAX) THEN GO TO 7920
7900 IF (R1<MIN) THEN GO TO 7940
7910 LET L(1,M)=MAX: GO TO 7960
7920 LET MORE=MORE+1
7930 LET L(1,MORE)=MAX: LET L(2,MORE)=R2
7940 LET L(2,M)=MIN: GO TO 7960
7950 LET L(1,M)=-1
7960 NEXT M
7969 REM TIDY UP THE L ARRAY.
7970 LET NRL=0
7980 FOR M=1 TO MORE
7990 IF L(1,M)<=EPS THEN GO TO 8020
8000 LET NRL=NRL+1
8010 LET L(1,NRL)=L(1,M): LET L(2,NRL)=L(2,M)
8020 NEXT M
8030 IF NRL=0 THEN GO TO 8160
8040 NEXT K
8049 REM DRAW VISIBLE PARTS OF LINE.
8050 FOR K=1 TO NRL
8060 LET R1=L(1,K): LET R2=1-R1
8070 LET XP1=X1*R2+X2*R1
8080 LET YP1=Y1*R2+Y2*R1
8090 LET R1=L(2,K): LET R2=1-R1
8100 LET XP2=X1*R2+X2*R1
8110 LET YP2=Y1*R2+Y2*R1
8120 IF (ABS(XP1-XP2)<EPS) AND (ABS(YP1-YP2)<EPS) THEN GO TO 8150
8130 LET XPT=XP1: LET YPT=YP1: GO SUB MOVETO
8140 LET XPT=XP2: LET YPT=YP2: GO SUB LINETO
8150 NEXT K
8160 LET IV1=IV2: LET X1=X2: LET Y1=Y2
8170 NEXT J
8180 NEXT I
8190 RETURN
```

Мы используем программу "HIDDEN", приведенную на листинге 12.1, "LIB1", "LIB3" и подпрограммы "SCENE3" и "CUBE", приведенные на листинге 12.2. В этой последней версии подпрограммы "CUBE" мы привлекли все методы построения объектов, использующие операции с массивами. Мы умышленно используем куб в наших программах, так как это чрезвычайно простой объект, на котором удается легко продемонстрировать общие идеи трёхмерной графики. Теперь настало время усложнить сами объекты: если вы понимаете ограничения, накладываемые алгоритмами на объекты, то все идеи, изложенные ранее, остаются в силе. Пользователи с объёмом памяти 16К обнаружат, что столь сложные программы не помещаются в памяти. В этом случае программы следует разбить на независимые сегменты, а информация об объектах и массивы должны записываться на пленку.

LISTING 12.2

```
6000 REM SCENES/EXAMPLES
6010 DIM X(16): DIM Y(16): DIM Z(16)
6020 DIM V(16): DIM W(16): DIM H(12): DIM F(4,12)
6030 DIM A(4,4): DIM B(4,4): DIM R(4,4): DIM Q(4,4)
6040 LET CUBE=6500: LET HIDDEN=7000
6050 LET PPD=3*VERT: LET NOV=0: LET NOF=0
6059 REM PUT FIRST CUBE IN OBSERVED POSITION-(ACTUAL-SETUP).
6060 GO SUB IDR3: GO SUB LOOK3
6070 GO SUB CUBE
6079 REM COPY ACTUAL TO OBSERVED MATRIX INTO Q.
```

```
      6080 FOR I=1 TO 4: FOR J=1 TO 4
      6090 LET Q(I,J)=R(I,J)
      6100 NEXT J: NEXT I: GO SUB IDR3
      6109 REM CALCULATE SETUP TO ACTUAL MATRIX.
      6110 LET TX=3: LET TY=1.5: LET TZ=2: GO SUB TRAN3: GO SUB MULT3
      6119 REM RECOVER ACTUAL TO OBSERVED MATRIX.
      6120 FOR I=1 TO 4: FOR J=1 TO 4
      6130 LET A(I,J)=Q(I,J)
      6140 NEXT J: NEXT I
      6149 REM CALCULATE SETUP TO OBSERVED MATRIX.
      6150 GO SUB MULT3
      6159 REM PLACE SECOND CUBE AND DRAW HIDDEN LINE VIEW OF SCENE.
      6160 GO SUB CUBE
      6170 GO SUB HIDDEN
      6180 RETURN
      6500 REM CUBE/VERTICES AND FACETS (STORED)
      6501 REM IN - PPD,NOV,VOF,R(4,4),(NOV),Y(NOV),Z(NOV),V(NOV),W(NOV),H(NOF),
F(4,NOF)
      6502 REM OUT - NOV,NOF,X(NOV),Y(NOV),Z(NOV),V(NOV),W(NOV),H(NOF),F(4,NOF)
      6510 DATA 1,1,1,1,1,-1,1,-1,-1,1,-1,1,-1,1,1,-1,1,-1,-1,-1,-1,-1,-1,1
      6520 DATA 1,2,3,4,5,8,7,6,1,5,6,2,2,6,7,3,3,7,8,4,4,8,5,1
      6530 RESTORE CUBE
      6539 REM EXTEND DATA BASE OF VERTICES IN OBSERVED POSITION.
      6540 LET NV=NOV
      6550 FOR I=1 TO 8
      6560 READ XX,YY,ZZ: LET NOV=NOV+1
      6570 LET X(NOV)=XX*R(1,1)+YY*R(1,2)+ZZ*R(1,3)+R(1,4)
      6580 LET Y(NOV)=XX*R(2,1)+YY*R(2,2)+ZZ*R(2,3)+R(2,4)
      6590 LET Z(NOV)=XX*R(3,1)+YY*R(3,2)+ZZ*R(3,3)+R(3,4)
      6599 REM PERSPECTIVE TRANSFORM.
      6600 LET V(NOV)=PPD*X(NOV)/Z(NOV)
      6610 LET W(NOV)=PPD*Y(NOV)/Z(NOV)
      6620 NEXT I
      6629 REM READ A*ND EXTEND DATA BASE OF FACETS.
      6630 FOR I=1 TO 6
      6640 READ F1,F2,F3,F4: LET NOF=NOF+1
      6650 LET H(NOF)=4
      6660 LET F(1,NOF)=F1+NV: LET F(2,NOF)=F2+NV
      6670 LET F(3,NOF)=F3+NV: LET F(4,NOF)=F4+NV
      6680 NEXT I
      6690 RETURN
```

LISTING 12.3

```
      6500 REM ICOSA/HEDRON
      6501 REM IN AND OUT-SAME AS CUBE ABOVE.
      6510 DATA 0,1,T,T,0,1,1,T,0,0,-1,T,T,0,-1,-1,T,0,0,1,-T,-T,0,1,1,-T,0,0,-1,-T,
-T,0,-1,-1,-T,0
      6520 DATA 1,3,2,1,2,4,1,4,8,1,8,6,1,6,3,2,3,5,2,9,4,4,12,8,8,11,6,3,6,7,2,5,9,
4,9,12,8,12,11,6,11,7,3,7,5,5,10,9,
      6530 RESTORE ICOSA: LET T=(1+SQR 5)/2
      6539 REM EXTEND DATA BASE OF VERTICES IN OBSERVED POSITION.
      6540 LET NV=NOV
      6550 FOR I=1 TO 12
      6560 READ XX,YY,ZZ: LET NOV=NOV+1
      6570 LET X(NOV)=XX*R(1,1)+YY*R(1,2)+ZZ*R(1,3)+R(1,4)
      6580 LET Y(NOV)=XX*R(2,1)+YY*R(2,2)+ZZ*R(2,3)+R(2,4)
      6590 LET Z(NOV)=XX*R(3,1)+YY*R(3,2)+ZZ*R(3,3)+R(3,4)
      6599 REM PERSPECTIVE TRANSFORM.
      6600 LET V(NOV)=X(NOV)*PPD/Z(NOV)
      6610 LET W(NOV)=Y(NOV)*PPD/Z(NOV)
      6620 NEXT I
      6629 REM READ AND EXTEND DATA BASE OF FACETS.
      6630 FOR I=1 TO 20
      6640 READ F1,F2,F3: LET NOF=NOF+1
      6650 LET H(NOF)=3
      6660 LET F(1,NOF)=F1+NV: LET F(2,NOF)=F2+NV: LET F(3,NOF)=F3+NV
      6670 NEXT I
```

```
6680 RETURN
```

LISTING 12.4

```
6700 REM CUBOCT/AHEDRON
6701 REM IN AND OUT - SAME AS CUBE ABOVE.
6710 DATA 0,1,1,1,0,1,1,1,0,0,-1,1,1,0,-1,-1,1,0,0,1,-1,-1,0,1,1,-1,0,0,-1,-1,-
1,0,-1,-1,-1,0
6720 DATA 1,2,4,8,1,6,7,3,2,3,5,9,4,9,10,12,5,7,11,10,6,8,12,11
6730 DATA 1,3,2,1,8,6,2,9,4,3,7,5,4,12,8,5,10,9,6,11,7,10,11,12
6740 RESTORE CUBOST
6750 LET NV=NOV
6760 FOR I=1 TO 12
6770 READ XX,YY,ZZ: LET NOV=NOV+1
6780 LET X(NOV)=XX*R(1,1)+YY*R(1,2)+ZZ*R(1,3)+R(1,4)
6790 LET Y(NOV)=XX*R(2,1)+YY*R(2,2)+ZZ*R(2,3)+R(2,4)
6800 LET Z(NOV)=XX*R(3,1)+YY*R(3,2)+ZZ*R(3,3)+R(3,4)
6810 LET V(NOV)=X(NOV)*PPD/Z(NOV)
6820 LET W(NOV)=Y(NOV)*PPD/Z(NOV)
6830 NEXT I
6840 FOR I=1 TO 6
6850 READ F1,F2,F3,F4: LET NOF=NOF+1
6860 LET H(NOF)=4
6870 LET F(1,NOF)=F1+NV: LET F(2,NOF)=F2+NV: LET F(3,NOF)=F3+NV: LET
F(4,NOF)=F4+NV
6880 NEXT I
6890 FOR I=1 TO 8
6900 READ F1,F2,F3: LET NOF=NOF+1
6910 LET H(NOF)=3
6920 LET F(1,NOF)=F1+NV: LET F(2,NOF)=F2+NV: LET F(3,NOF)=F3+NV
6930 NEXT I
6940 RETURN
```

Упражнение 12.1

Постройте композицию, состоящую из кубов, тетраэдров, пирамид, октаэдров, икосаэдров, кубических октаэдров. Для того, чтобы облегчить вашу задачу, на листингах 12.3 и 12.4 приведены конструирующие программы для кубооктаэдра и икосаэдра. Напишите конструирующую программу для октаэдра, и, пожалуй, ещё более сложную программу для ромбического додекаэдра.

LISTING 12.5

```
6000 REM SCENES/TWO STARS HIDDEN LINES REMOVED
6010 DIM X(22): DIM Y(22): DIM Z(22)
6020 DIM V(22): DIM W(22): DIM H(36): DIM F(3,36)
6030 DIM A(4,4): DIM B(4,4): DIM R(4,4): DIM Q(4,4)
6040 LET STAR1=6500: LET STAR2=6700: LET HIDDEN=7000
6050 LET PPD=3*VERT: LET NOV=0: LET NOF=0
6059 REM PLACE FIRST STAR.
6060 GO SUB IDR3: GO SUB LOOK3
6070 LET A=6: GO SUB STAR1
6080 FOR I=1 TO 4: FOR J=1 TO 4
6090 LET Q(I,J)=R(I,J)
6100 NEXT J: NEXT I: GO SUB IDR3
6109 REM PLACE SECOND STAR.
6110 LET TX=5: LET TY=5: LET TZ=5: GO SUB TRAN3: GO SUB MULT3
6120 FOR I=1 TO 4: FOR J=1 TO 4
6130 LET A(I,J)=Q(I,J)
6140 NEXT J: NEXT I
6150 GO SUB MULT3
6160 LET A=4: GO SUB STAR2
6170 GO SUB HIDDEN
6180 RETURN
```

LISTING 12.6

```
6500 REM STAR1
6501 REM IN AND OUT - SAME AS CUBE ABOVE.
6509 REM STAR BASED ON A CUBE.
6510 DATA 1,1,1,1,1,-1,1,-1,-1,1,-1,1,-1,1,1,-1,1,-1,-1,-1,-1,-1,1,A,0,0,-
A,0,0,0,A,0,0,-A,0,0,0,A,0,0,-A
6520 DATA
1,2,9,2,3,9,3,4,9,4,1,9,6,5,10,5,8,10,8,7,10,7,6,10,2,1,11,1,5,11,5,6,11,6,2,11,4,3,1
2,3,7,12,7,8,12,8,4,12,1,4,13,4,8,13,8,5,13,5,1,13,3,2,14,2,6,14,6,7,14,7,3,14
```

```
6530 RESTORE STAR1
6540 LET NV=NOV
6550 FOR I=1 TO 14
6560 READ XX,YY,ZZ: LET NOV=NOV+1
6570 LET X(NOV)=XX*R(1,1)+YY*R(1,2)+ZZ*R(1,3)+R(1,4)
6580 LET Y(NOV)=XX*R(2,1)+YY*R(2,2)+ZZ*R(2,3)+R(2,4)
6590 LET Z(NOV)=XX*R(3,1)+YY*R(3,2)+ZZ*R(3,3)+R(3,4)
6600 LET V(NOV)=PPD*X(NOV)/Z(NOV)
6610 LET W(NOV)=PPD*Y(NOV)/Z(NOV)
6620 NEXT I
6630 FOR I=1 TO 24
6640 READ F1,F2,F3: LET NOF=NOF+1
6650 LET H(NOF)=3
6660 LET F(1,NOF)=F1+NV: LET F(2,NOF)=F2+NV: LET F(3,NOF)=F3+NV
6670 NEXT I
6680 RETURN
LISTING 12.7
6700 REM STAR2
6701 REM IN AND OUT - SAME AS CUBE ABOVE.
6709 REM STAR BASED ON A TETRAHDRON.
6710 DATA 1,1,1,1,-1,-1,-1,1,-1,-1,-1,1,-A,-A,-A,-A,A,A,A,-A,A,A,A,-A
6720 DATA 2,1,8,3,2,8,1,3,8,1,2,7,4,1,7,2,4,7,2,3,5,4,2,5,3,4,5,3,1,6,4,3,6,1,
4,6
6730 RESTORE STAR2
6740 LET NV=NOV
6750 FOR I=1 TO 8
6760 READ XX,YY,ZZ: LET NOV=NOV+1
6770 LET X(NOV)=XX*R(1,1)+YY*R(1,2)+ZZ*R(1,3)+R(1,4)
6780 LET Y(NOV)=XX*R(2,1)+YY*R(2,2)+ZZ*R(2,3)+R(2,4)
6790 LET Z(NOV)=XX*R(3,1)+YY*R(3,2)+ZZ*R(3,3)+R(3,4)
6800 LET V(NOV)=PPD*X(NOV)/Z(NOV)
6810 LET W(NOV)=PPD*Y(NOV)/Z(NOV)
6820 NEXT I
6830 FOR I=1 TO 12
6840 READ F1,F2,F3: LET NOF=NOF+1
6850 LET H(NOF)=3
6860 LET F(1,NOF)=F1+NV: LET F(2,NOF)=F2+NV: LET F(3,NOF)=F3+NV
6870 NEXT I
6880 RETURN
```

Пример 12.2

Теперь, когда мы убедились, что программы, удаляющие невидимые линии, работают чрезвычайно долго, нам следует произвести ряд экспериментов с этими программами. Для генерации изображения двух кубов, показанных на рис. 40, требуется пять минут. Это означает, что мы весьма ограничены в выборе объектов для наших рисун-ков. Тем не менее полезно попытаться создать изображение сложных объектов с помощью нашей программы, и если у вас есть доступ к более мощному компьютеру, то вы можете проверить, что наш алгоритм пригоден и для более мощных компьютеров, причем рисунки создаются значительно быстрее.

На листинге 12.5 приведена программа "SCENE3", а на листингах 12.6 и 12.7 приведены программы, генерирующие два звездообразных трёхмерных объекта (при генерации каждого из которых следует задать параметр A, определяющий длину лучей). Обе эти программы при построении используют формы куба и октаэдра. Изображение двух звёзд (кстати, очень эффектное) может быть получено, если задать следующие значения параметров: HORIZ=48, VERT=32, точка, в которой расположен глаз - (35,20,25), а взгляд направлен в точку (0,0,0).

Упражнение 12.2

Программа на листинге 10.1 проверяет, что порядок обхода вершин треугольной грани - против часовой стрелки. Эта программа была написана для выпуклых тел, содержащих начало координат. Переделайте программу с учётом общего случая. Это означает, что задаются две точки, лежащие по разные стороны от плоскости, содержащей одну из граней объекта. В одной из этих точек находится глаз наблюдателя, а другая точка, не обязательно являющаяся началом координат, - внутренняя точка тела. Используйте эту программу для проверки объектов программы 12.6 и 12.7.

Напишите программы, создающие объекты на основе форм октаэдра, кубического октаэдра, икосаэдра или додекаэдра. Всегда проверяйте правильность обхода вершин граней создаваемых вами объектов. Звездчатые тела можно строить на основе тел вращения, кроме того, необязательно ограничиваться только симметричными телами. Для несимметричного случая вам потребуется модификация программы 10.1. Если соблюдены все условия, налагаемые на изображаемые объекты, то программа 12.1 начертит любой из них.

Упражнение 12.3

Введите дополнительную информацию о создаваемых объектах. Наряду с информацией о вершинах и гранях, введите ещё один массив L, значения элементов L(1,1) и L(2,1) которого есть индексы двух граней, которые содержат

I-ое ребро, 1<=I<=NOL. После того измените алгоритм удаления невидимых линий так, чтобы в нём не рассматривались прямые, являющиеся пересечением двух невидимых граней, и не сравнивалась грань с прямыми, лежащими на этой грани.

Теперь вы понимаете, что мы приблизились к пределу возможностей SPECTRUMa. Если вы хотите продолжить изучение машинной графики, то вам следует использовать для этой цели более мощные компьютеры. Вам так же потребуется изучить способы организации структур данных, более подходящие для машинной графики, чем массивы. Например, вся сцена может храниться в виде связного списка указателей, каждый из которых в свою очередь указывает на связный список информации об объекте. Каждая грань может быть представлена в виде связного списка. На первый взгляд, кажется, что эта конструкция слишком громоздка, но на самом деле она чрезвычайно облегчает программирование таких алгоритмов, как алгоритм удаления невидимых линий. При такой структуре данных легко извлечь отношение между объектами и гранями.

После того, как вы усвоите эти идеи, вы можете перейти к изучению более сложных графических алгоритмов, таких, как алгоритм раскраски, мультипликации и затенения объектов.

В следующей главе мы рассмотрим более сложные методы символьной графики и введём один метод мультипликации трёхмерной графики.

Список программ

1. "LIB 1", "LIB3", листинг 12.1 ("HIDDEN") и 12.2 ("SCENES" и "CUBE"). Входные данные: HORIZ, VERT, (EX,EY,EZ), (XD,DY,DZ). Введите, например, следующие значения:

а) 9,6, (15,10,5), (0,0,0);

б) 9,6, (10,10,-10), (0,1,0).

2. "LIB 1", "LIB3", листинги 12.1 ("HIDDEN") и 12.2 ("SCENES"), загрузите в память программы 12.3 ("ICOSA") и 12.4 ("CUBOCT") и измените подпрограмму "SCENE3" следующим образом:

```
6010 DIM X(24): DIM Y(24): DIM 7(24)
6020 DIM V(24): DIM W(24): DIM P(34): DIM H(34)
6040 LET CUBOCT=6700: LET ICOSA=6500: LET HIDDEN=7000
6070 GO SUB ICOSA
6160 GO SUB CUBOCT
```

Входные данные: HORIZ, VERT, (EX,EY,EZ), (DX,DY,DZ). Введите, например,

а) 9, 6, (15,10,5), (0,0,0);

б) 9, 6, (-10,10,10), (0,1,0).

3. "LIBГ, "LIB3", листинг 12.1 ("HIDDEN"), 12.5 ("TWOSTARS"), и 12.6 ("STAR1"), и 12.7 ("STAR2"). Входные данные: HORIZ, VERT, (EX,EY,EZ), (DX,DY,DZ). Попробуйте ввести, например,

а) 60, 40, (10,20,30), (0,0,0);

б) 90, 60, (-10-,20,-10), (0,1,0).

ГЛАВА 13
Белее сложные элементы программирования

Если вы хотите писать программы на профессиональном уровне, то вам следует позаботиться о том, чтобы они обладали свойствами "дружественности". Это один из немногих терминов, употребляемых в рекламе, обладающих вполне конкретным смыслом: программа называется дружественной, если ею удобно пользоваться не только создателю этой программы, но и любому пользователю. Мы все имели печальный опыт, когда, вернувшись к программам, написанным два-три месяца перед этим, убеждались, что они составлены и прокомментированы так плохо, что их невозможно понять.

Желательно, чтобы вы приучили себя комментировать тексты своих программ и организовывать ввод в форме диалога с программой. Облегчить работу с программой можно также, приложив к ней вводную программу-инструкцию, как это обычно делается в видеоиграх. Следите за тем, чтобы указания пользователю, выдаваемые программой во время исполнения, были ясными и непротиворечивыми.

В программах, где подпрограммы могут инициализироваться в любой последовательности, эффективно использовать меню (см. генератор символов в гл. 5). Это средство особенно эффективно для лиц, имеющих плохое представление о том, как устроена программа.

Для написания хороших комментариев вполне достаточно одного только здравого смысла. Избегайте классических нелепостей вроде "нажмите 1 для получения двойной информации и 2 для одиночной". Для управления движения курсором старайтесь использовать курсорные клавиши (см. 3-ю опцию программы генератор символов), что кажется вполне естественным для лиц, использующих SPECTRUM достаточно часто. На листинге 13.1 приведена программа, рисующая многоугольники (как в упражнении 1.3). Подпрограмма ввода "INPUT" использует курсорные клавиши и "EDIT", что облегчает работу лицам, имеющим навык работы со SPECTRUMом.

Управление выполнением программ

Команда INPUT может использоваться в случаях, когда по ходу выполнения программы необходимо выдать запрос и получить необходимую информацию. В большинстве случаев мы помещаем запрос в операторе INPUT в таком виде, в каком он отобразится на экране дисплея; запрос помещается в кавычки. В SPECTRUMe мы можем задавать любую символьную информацию, заключая необходимую информацию в скобки. В таком случае она эквивалентна запросу, заключенному в кавычки. Этот приём использован в вышеприведенной программе, но с его помощью можно создавать значительно более сложные запросы.

С помощью строк, содержащих контрольные коды, можно создавать сложные графические изображения в любом месте экрана. Значение переменной строчного типа может быть присвоено за несколько шагов, а саму переменную можно поместить в скобках в операторе CHR$, а численные переменные с помощью функций STR$. Эти приёмы использовались в программе "WORM CAME" (листинг 1.16). Строки с кодами, управляющими цветом символов, и позицией, с которой они печатаются, могут использоваться в игровых программах, когда возникает необходимость быстро создавать изображение в различных частях экрана. В качестве примера применения этих методов мы приводим на листинге 13.2 меню гиперболической программы обработки файлов.

Обратите внимание на естественное использование цветов для предупреждения о потенциально опасных операциях.

LISTING 13.1

```
10 DIM X(10): DIM Y(10)
20 LET INPUT=200: LET LIST=300
30 GO SUB INPUT
40 PLOT X(10),Y(10): LET DX=X(10): LET DY=Y(10)
50 FOR I=1 TO 10: DRAW X(I)-DX,Y(I)-DY: LET DX=X(I): LET DY=Y(I): NEXT I
60 PAUSE 250: GO TO 30
200 REM INPUT COORDS.
210 LET I=1
220 GO SUB LIST
230 LET I$=INKEY$: IF I$="" THEN GO TO 230
240 IF I$=CHR$ 10 AND I<11 THEN LET I=I+1: GO TO 220
250 IF I$=CHR$ 11 AND I>1 THEN LET I=I-1: GO TO 220
260 IF I$<>CHR$ 7 THEN GO TO 230
270 IF I=11 THEN INPUT "END ?  ";LINE I$: IF I$="Y" THEN CLS: RETURN
280 IF I=11 THEN GO TO 220
290 INPUT "XCOORD.=";X(I);" Y COORD.=";Y(I): GO TO 220
300 REM LIST DATA
310 CLS: FOR J=1 TO 10
320 PRINT AT J,1;"X COORD.=";X(J);"  Y COORD.=";Y(J)
330 NEXT J: PRINT AT 11,1;"END": PRINT AT I,0;">"
340 RETURN
```

Управляющие коды можно вводить с клавиатуры и вставлять их в символьные цепочки. Эти коды, которые во время выполнения программы игнорируются, можно включать в операторы программы для того, чтобы выделить те или иные участки текстов программ. Если на клавиатуре набрать код, соответствующий PAPER 6 (расширенное "6"), и сразу после этого нажать клавишу DELETE, то код для PAPER уничтожится и останется CODE 6, который устанавливает курсор на следующую границу строки или полустроки. CODE 6 можно использовать в программе для

пропуска строки перед началом нового раздела. Символ REM в начале комментария можно убрать, поместив код INK 0 (расширенное "0") после символов INK 7 (расширенное "7") перед REM.

LISTING 13.2

```
100 DIM G(6):DIM A$(6,24)
110 FOR I=1 TO 6: READ G(I): NEXT I
120 DATA 1000,1000,1000,1000,1000,1000
130 LET A$(1)=" EDIT FILE ": LET A$(2)=" PRINT FILE "
140 LET A$(3)=" SAVE FILE ": LET A$(4)=" LOAD FILE "
150 LET A$(5)=CHR$ 17+CHR$ 2+CHR$ 16+CHR$ 7+" DELETE FILE "+CHR$ 17+CHR$ 7
160 LET A$(6)=CHR$ 17+CHR$ 2+CHR$ 16+CHR$ 7+" ERASE ALL FILES "+CHR$ 17+CHR$ 7
500 REM MENU/SELECT OPTIONS.
510 CLS :PRINT AT 2,8;"N.O.N. FILE HANDLER "
520 FOR I=1 TO 6
530 PRINT AT I*2+4,8;I;A$(I)
540 NEXT I
550 INPUT "SELECT OPTION ? ";OP
560 IF OP<1 OR OP>6 THEN GO TO 550
570 IF OP<5 THEN GO TO 600
580 INPUT ("DO YOU REALLY WANT TO ";+CHR$ 6+A$(OP)+" ");Y$
590 IF Y$<>"Y" THEN GO TO 500
600 GO SUB G(OP): GO TO 500
1000 REM REMAINDER OF ROUTINES
1010 RETURN
```

Как правило, программы следует разбивать на модули. Выделение модулей облегчает чтение, исправление и модификацию программ. В качестве примера укажем часть программы листинга 13.4.

```
10 REM LOADER FOR MACHINE-CODE ROUTINE
20 CLEAR 63999
30 FOR I=0 TO 11: READ A
40 POKE 64000+I,A: NEXT I
100 REM DATA FOR MACHINE-CODE TRANSFER ROUTINE
110 DATA 17,0,64
120 DATA 33,0,80
130 DATA 1,0,8
140 DATA 237,176
150 DATA 201
200 REM - MAIN PROGRAM
210 CLS: FOR I=0 TO 21
220 FOR J=0 TO 31: PRINT AT I,J; CHR$(I+64): NEXT J
```

Структура дисплейного файла

Мы знаем, что бейсик позволяет корректировать и просматривать непосредственно дисплейный файл, но для этого следует иметь представление об устройстве дисплейного файла. Каждая горизонтальная строка дисплея состоит из 32 байт, по восемь бит каждый. Эти 32 байта расположены в последовательных ячейках файла, однако ячейки, в которых хранится информация об изображении строчкой ниже, находятся на том же месте на следующей странице памяти. Для восьмиразрядных процессоров таких, как установленный в SPECTRUMe Z-80, страница памяти состоит из 256 байтов; для обозначения этого числа мы используем идентификатор PAGE. Такая организация дисплейного файла позволяет эффективно создавать изображение. На каждой странице хранится восемь линий по 32 байта. На первой станице хранятся верхние линии первых восьми символьных строк, а на семи последующих страницах – оставшиеся семь линий. За этими страницами идут ещё два набора по восемь страниц, на которых хранится информация об оставшейся части экрана.

Теперь мы знаем, как вычислить номера ячеек, в которых хранится верхняя линия символа, и знаем, что номера ячеек, в которых хранится информация о других линиях, получаются прибавлением 256 (количество байт в странице). Для определения номера ячейки, в которой хранится верхняя строчка символа, необходимо знать, в каком из трёх наборов (по восемь страниц) хранится этот символ. Нам также надо будет вычислить, в какой из 256 страницы находится нужная информация. Мы вычисляем номер ячейки, в которой хранится информация о первой строке символа, находящегося в R-ом ряду в C-ой строке с помощью следующей функции:

```
DEF FN A(R,C)=16384+INT(R/8)*2048+(R-INT(R/8)*8)*32+C
```

В эту функцию входят четыре члена:

16384 - начало дисплейного файла

+ INT(R/8)*2048 - позиция строки в соответствующей трети экрана (0,1,2), умноженной на длину трети экрана (2048=8*PAGE)

+ (R-INT(R/8)*8)*32 - позиция строки в соответствующей трети (от 0 до 7), умноженная на 32 (количество столбцов в строке)

+ C - номер столбца (0-31) в этой строке.

В Z80 также имеются внутренние регистры, в которые можно поместить числа и использовать их в первых линиях необходимой трети экрана. Последние два элемента устанавливают, какие из 256 байтов страницы

соответствуют верхней линии символьного блока.

Адрес начала данных, относящихся к символьному блоку, можно узнать с помощью функции, использующей этот символ в качестве значения параметра; затем мы преобразуем эти данные в номера ячеек дисплейного файла.

```
DEF FN D(A$)=PEEK 23606+PEEK 23607*256+8*CODE A$
```

Эта функция использует системную переменную CHAR$ (которая хранится в ячейках 23606 и 23607, см. главу 5) для определения начала таблицы символов, а затем умножает на восемь код соответствующего символа и таким образом находит адрес начала данных. Заметьте, что две эти функции могут быть использованы для занесения информации в две нижние строки экрана дисплея, что, как правило, нельзя сделать обычными средствами.

LISTING 13.3

```
100 REM MAIN PROGRAM
110 LET PRINT=500
120 LET P$="9 FAKE STATEMENT, 120:1": LET ROW=23: LET COL=0
130 GO SUB PRINT
140 LET P$="PRINT THIS PNYWH ERE ON THE SCREEN ": LET ROW=7: LET COL=16
150 GO SUB PRINT
160 PAUSE 250: STOP
500 REM PRINT ROUTINE
510 DEF FN A(R,C)=16384+INT(R/8)*2048+(R-INT(R/8)*8)*32+C
520 DEF FN D(A$)=PEEK 23606+256*PEEK 23607+8*CODE A$
530 FOR I=1 TO LEN P$
540 LET ADDRESS=FN A(ROW,COL)
550 LET DATA1=FN D(P$(I))
560 FOR J=0 TO 7
570 POKE ADDRESS+J*256,PEEK(DATA1+J)
580 NEXT J
590 LET COL=COL+1: IF COL=32 THEN LET COL=0
600 LET ROW=ROW+1: IF ROW=24 THEN LET ROW=0
610 NEXT I
620 RETURN
```

Быстрое преобразование содержимого экрана

Если мы хотим быстро изменять содержимое дисплейного файла, то скоро обнаружим, что операторы бейсика не подходят для этой цели, так как выполняются слишком медленно.

SPECTRUM оснащен процессором Z-80, позволяющим быстро менять содержимое дисплейного файла, однако для этого приходится использовать машинные коды, а не языки высокого уровня, каким является бейсик. В машинном коде имеются команды, подобные командам бейсика POKE и PEEK, позволяющие изменять содержимое дисплейного файла.

В процессоре Z-80 имеются внутренние регистры, в которых могут хранится числа; эти регистры используются аналогично тому, как в бейсике используются переменные. Можно составить последовательность команд (иначе говоря, программу), которая бы помещала в один из этих регистров содержимое некоторой ячейки памяти, а затем помещала содержимое регистра в другую ячейку. При этом время выполнения одной команды составляет около одной миллионной доли секунды. К сожалению, машинный код слишком сложен для начинающих программистов, так как программы на нем выглядят, как последовательность несвязанных на первый взгляд чисел. Удобнее пользоваться языком ассемблера, в котором машинные коды заменены мнемоническими кодами - аббревиатурами слов, напоминающими о выполняемой кодами функции.

В SPECTRUMe команды машинного кода засылаются в операторы DATA, а затем заносятся в память операторами POKE. Операторы ассемблера удобно включать в строчки операторов DATA, используя инструкцию REM.

Трансляторы с языка ассемблера - это программы, переводящие программы, написанные на ассемблере, в программы, написанные в машинном коде.

Одной из самых мощных команд процессора Z80 является команда 237, 176, или, на языке ассемблера LDIR, позволяющая пересылать данные из одного участка памяти в другой. Перед тем, как воспользоваться этой командой, следует заполнить некоторые регистры данными, а именно:

DE - адресом, куда следует занести данные;

HL - адресом, где находятся данные в настоящий момент;

BC - количество байт, подлежащих пересылке.

На рис. 41 приведена программа, составленная в машинных кодах, а также её эквивалент на языках ассемблере и бейсике.

Помните, что язык ассемблера - это всего лишь способ сделать машинный код более понятным.

На листинге 13.4 приведена программа, заносящая оператором POKE машинный код в память, а затем вызывающая подпрограмму. Эта программа копирует нижнюю треть дисплейного файла в верхнюю.

```
                                300 REM BASIC DATA TRANSFER
17,0,64       LD   DE,16384     310 LET DE=16384
33,0,80       LD   HL,20480     320 LET HL=20480
1,0,8         LD   BC,2048      330 LET BC=2048
237,176       LDIR              340 LET A=PEEK HL: POKE DE,A
```

```
                                    350 LET DE=DE+1: LET HL=HL+1
                                    360 LET BC=BC-1
                                    370 IF BC<>0 THEN GO TO 340
201          RET                    380 RETURN
```

Рис. 41.

LISTING 13.4

```
10 REM LOADER FOR MACHINE-CODE ROUTINE
19 REM FOR 16K MACHINES USE CLEAR 31999.
20 CLEAR 63999
30 FOR I=0 TO 11: READ A
39 REM FOR 16K USE POKE 32000.
40 POKE 64000+I,A: NEXT I
100 REM DATA FOR MACHINE-CODE TRANSFER ROUTINE
110 DATA 17,0,64  : REM LD DE,16384
120 DATA 33,0,80  : REM LD HL,20480
130 DATA 1,0,8    : REM LD BC,2048
140 DATA 237,176  : REM LDIR
150 DATA 201      : REM RET
200 REM MAIN PROGRAM
210 CLS: FOR I=0 TO 21
220 FOR J=0 TO 31: PRINT AT I,J;CHR$ (I+64): NEXT J
230 NEXT I
240 PRINT AT 19,1; INVERSE 1;"PRESS ANY KEY";"TO START ROUTINE"
250 IF INKEY$="" THEN GO TO 250
259 REM FOR 16K CHANGE TO USR 32000.
260 LET A=USR 64000: STOP
```

После занесения оператором POKE команд машинного кода в память мы запускаем программу функцией USR, в которой указан адрес начала программы. Функция USR просто-напросто вызывает подпрограмму, написанную в машинном коде, чей адрес совпадает со значением параметра USR. В программе, написанной на бейсике, этот вызов осуществляется с помощью команды вроде LET A=USR 32000 (32000 - это адрес, в котором хранится 17, первый байт кода).

Мультипликация (только для машин с объемом памяти 48К)

Для того чтобы поместить всю информацию, необходимую для создания изображения и хранящуюся в произвольном месте в памяти, в дисплейный файл, мы используем простую подпрограмму, что позволяет быстро менять изображение на экране.

В SPECTRUMe с объёмом памяти 48К одновременно могут храниться пять различных изображений, называемых кадрами. С помощью программы на листинге 13.5 можно создать пять программ на машинном коде, расположенные в 30200, 30300, 30400 и 30500, которые будут пересылать эти данные на экран. Предположим, что на ленте записаны данные о пяти изображениях. Мы загружаем их в память в виде пяти альтернативных кадров, которые с помощью соответствующих программ могут быть занесены на экран. Этот приём можно использовать во время лекции или разговора с покупателем. После показа пяти изображений следует загрузить следующие пять, на что уходит около пяти минут, в то время как с диска они загружаются за 10 секунд.

Если использовать программу "SLIDESHOW", то для создания изображения на экране достаточно нажать одну из пяти клавиш "1"-"5".

LISTING 13.5

```
100 REM LOADER FOR MACHINE-CODE ROUTINES
110 CLEAR 29999: DIM F(5)
120 FOR I=1 TO 5: RESTORE
130 LET F(I)=120+27*(I-1)
140 FOR J=0 TO 11
150 READ A: POKE 30000+I*100+J,A
160 NEXT J
170 NEXT I
200 REM DATA TRANSFER FROM I'TH FRAME TO SCREEN
210 DATA 17,0,64   : REM LD DE,16384
220 DATA 33,0,F(I) : REM LD HL,FRAME(I)
230 DATA 1,0,27    : REM LD BC,27*256
240 DATA 237,176   : REM LDIR
250 DATA 201       : REM RET
300 REM LOAD FRAMES
310 FOR J=1 TO 10: BEEP 0.1,30: PAUSE 5: NEXT J
320 CLS: FOR I=0 TO 4
330 LOAD "" CODE (120+I*27)*256,27*256
340 NEXT I
350 FOR J=1 TO 10: BEEP 0.1,30: PAUSE 5: NEXT J
400 REM SLIDE SHOW
```

```
410 IF INKEY$<>"" THEN GO TO 410
420 LET A$=INKEY$: IF A$="" THEN GO TO 410
440 IF A$="A" THEN GO TO 500
450 IF A$>="1" AND A$<="5" THEN LET A=USR (30000+VAL A$*100)
460 GO TO 410
500 REM MOVIE SHOW
510 FOR I=30100 TO 30500 STEP 100
520 LET A=USR I
530 IF INKEY$="S" THEN GO TO 400
540 NEXT I: GO TO 510
```

Развивая эти идеи, можно получить изображение движущихся объектов. Рассмотрим, например, пять кадров, созданных с помощью программ трёхмерной графики, приведенных в главе 12. Эти программы генерируют изображения сферы; значения параметров HORIZ=3.2, VERT=2.2, NUMH=10, NUMV=8 и PHI=0.4*PI*I/NUMH, где 0<=I<=4, глаз находится в точке (1,2,3), взгляд направлен в начало координат. Изображение вращающегося сфероида может быть получено, если показывать эти кадры, быстро сменяющие друг друга.

После прогона программ "MOVIE" и "SLIDESHOW" следует набрать на клавиатуре CLEAR 65367, в противном случае, при попытке загрузить какую-либо программу на экране дисплея появится надпись "OUT OF MEMORY" (в памяти нет свободного места).

Упражнение 13.1

Создайте в первом кадре перспективное изображение каркасного куба, а во втором - то же изображение, но с удалёнными невидимыми линиями. Создайте из этих кадров мультипликацию. Вы увидите на экране изображение куба, в котором невидимые линии будут мигать. Создайте изображение вращающейся звезды из главы 12.

Свертка (16К и 48К)

Использование машинного кода для решения более сложных задач требует от программиста высокой квалификации. Здесь мы приводим ещё один пример программы в машинном коде (программа на листинге 13.6 "прокручивает" изображение на экране сверху вниз). Решаемая программой задача осложнена тем, что данные, записанные в дисплейном файле, разбиты по линиям, содержащим 32 байта. Для того, чтобы программа была эффективной, нам необходимо знать адрес ячейки, в которой хранится начало данных об этих линиях. Вычисление этого адреса в машинных кодах громоздко и неэффективно, поэтому мы используем так называемую адресную таблицу.

Для облегчения программирования в машинных кодах создается адрес, который хранится в шестнадцати битах, распределенных по двум ячейкам - нижней и верхней. Когда нам нужно узнать адрес какой-нибудь строчки, мы просто разыскиваем в таблице две половинки, соответствующие этим байтам. Для заполнения таблицы мы используем бейсик. Эти таблицы и сам машинный код могут быть записаны на ленту. На листинге 13.6 приведена та же программа, написанная на бейсике. Обратите внимание на то, что программа на бейсике предполагает существование той же таблицы, используемой программой, написанной в машинном коде.

Упражнение 13.2

Напишите подпрограмму в машинном коде, которая бы за один вызов прокручивала файл атрибутов на одну строку. Напишите программу на бейсике, которая бы прокручивала в памяти файл атрибутов каждый раз после того, как графическое поле прокручивается на восемь линий.

LISTING 13.6

```
10 REM LOADER FOR MACHINE-CODE ROUTINE
20 CLEAR 31999
30 FOR I=0 TO 74
40 READ A: POKE 32000+I,A
50 NEXT I
100 REM DOWRWARDS WRAP-AROUND SCROLL FOR GRARHICS AREA
110 DATA 1,175,0        : REM LD   BC,175
120 DATA 221,33,75,125  : REM LD   IX,LOBYTE
130 DATA 221,9          : REM ADD  IX,BC
140 DATA 221,110,0      : REM LD   L,(IX+0)
150 DATA 221,33,251,125 : REM LD   IX,HIBYTE
160 DATA 221,9          : REM ADD  IX,BC
170 DATA 221,102,0      : REM LD   H,(IX+0)
180 DATA 17,171,126     : REM LD   DE,TEMP
190 DATA 205,67,125     : REM CALL MOVE
200 DATA 221,33,75,125  : REM LD   IX,LOBYTE
210 DATA 221,9          : REM ADD  IX,BC
220 DATA 221,94,0       : REM LD   E,(IX+0)
230 DATA 221,110,-1     : REM LD   L,(IX-1)
240 DATA 221,33,251,125 : REM LD   IX,HIBYTE
250 DATA 221,9          : REM ADD  IX,BC
260 DATA 221,86,0       : REM LD   D,(IX+0)
270 DATA 221,102,-1     : REM LD   H,(IX-1)
280 DATA 205,67,125     : REM CALL MOVE
290 DATA 13             : REM DEC  C
```

```
300 DATA 32,226          : REM JR   NZ,LINE
310 DATA 17,0,64         : REM LD   DE,16384
320 DATA 33,171,126      : REM LD   HL,TEMP
330 DATA 205,67,125      : REM CALL MOVE
340 DATA 201             : REM RET
350 DATA 197             : REM PUSH BC
360 DATA 1,32,0          : REM LD   BC,32
370 DATA 237,176         : REM LDIR
380 DATA 193             : REM POP  BC
390 DATA 201             : REM RET
400 REM CONSTRUCT TABLES OF DISPLAY-FILE LINE ADDRESSES
410 FOR I=0 TO 21: FOR J=0 TO 7
420 LET A=16384+INT(I/8)*2048+(I-INT(I/8)*8)*32
430 LET H=INT(A/256): LET L=A-H*256
440 POKE 32075+I*8+J,L: POKE 32251+I*8+J,H+J
450 NEXT J: NEXT I
500 REM MAIN PROGRAM
510 LIST 380
520 IF INKEY$ <>"" THEN LET A=USR 32000
530 GO TO 520
```

Структура программ на бейсике (перенумерация и уничтожение)

При модульном подходе к созданию программ часто встречается ситуация, когда не хватает места для дополнительных строк. В таких ситуациях желательно перенумеровать строки или уничтожить некоторые части. Эти две операции оказываются полезными и при создании новых программ с помощью команды MERGE. Для выполнения этих операций следует детально изучить способ хранения в памяти программ, написанных на бейсике.

Программы, написанные на бейсике, хранятся построчно в ячейках, находящихся с номера 23755. Первым элементом данных для каждой строчки является номер строчки (16 бит), занимающий две последовательные ячейки - нижнюю и верхнюю. Во всех остальных местах значения хранятся в стандартном нижне-верхнем представлении. Для иллюстрации этих идей предположим, что мы вводим строку 10 REM. Если наберем команду

```
PRINT PEEK 23755, PEEK 23756,
```

то мы увидим, что в этих ячейках находятся значения 0,10. Если мы знаем 10-ю строчку и введем в начале программы строку с другим номером, то мы увидим, что теперь этот номер хранится в ячейках 23755 и 23756. В следующих двух ячейках хранится длина строки (эта информация может быть использована для поиска начала следующей строки). Затем идёт собственно строка программы на бейсике, причём каждый символ или ключевое слово занимает одну ячейку. Конец каждой строки отмечен кодом 13, соответствующим клавише ENTER. После каждого числа, записанного в десятичном представлении и хранящемся в символьном виде, идёт код 14, после которого идет это же число в двоичном представлении, занимающее пять байтов.

Используя эти сведения, можно написать программу, нумерующую тексты программ, включая собственный. Такой способ нумерации программ особенно полезен при работе с текстами программ, каждый оператор которых начинается с новой строки, или когда требуется вывод на дисплей контрольных кодов.

Для перенумерации строк программы нам следует изменить номера строк, записанные в памяти. Этого достаточно в тех случаях, когда при работе программы строки выполняются только по порядку их расположения в тексте.

LISTING 13.7

```
10 REM BASIC VERSION SCROLL FOR GRAPHICS AREA
20 CLEAR 31999
30 GO TO 400
100 REM DOWNWARDS WRAP-AROUND SCROLL FOR GRAPHICS AREA
110 LET C=175: LET B=0: LET BC=C+B*256
120 LET IXLO=75: LET IXHI=125: LET IX=IXLO+IXHI*256
130 LET IX=IX+BC
140 LET L=PEEK(IX+0)
150 LET IXLO=251: LET IXHI=125: LET IX=IXLO+IXHI*256
160 LET IX=IX+BC
170 LET H=PEEK(IX+0): LET HL=L+H*256
180 LET E=171: LET D=126: LET DE=IXLO+IXHI*256
190 GO SUB 350
200 LET IXLO=75: LET IXHI=125: LET IX=IXLO+IXHI*256
210 LET IX=IX+BC
220 LET E=PEEK(IX+0)
230 LET L=PEEK(IX-1)
240 LET IXLO=251: LET IXHI=125: LET IX=IXLO+IXHI*256
250 LET IX=IX+BC
260 LET D=PEEK(IX+0): LET DE=E+D*256
270 LET H=PEEK(IX-1): LET HL=L+H*256
280 GO SUB 350
```

```
290 LET C=C-1: LET BC=C+B*256
300 IF C<>0 THEN GO TO 200
310 LET E=0: LET D=64: LET DE=E+256*D
320 LET L=171: LET H=256: LET HL=L+H*256
330 GO SUB 350
340 RETURN
350 LET S=BC
360 LET BC=32
370 LET A=PEEK HL: POKE DE,A: LET DE=DE+1: LET HL=HL+1: LET BC=BC-1: IF BC<>0
THEN GO TO 370
380 LET BC=S
390 RETURN
400 REM CONSTRUCT TABLES OF DISPLAY-FILE LINE ADDRESSES
410 FOR I=0 TO 21: FOR J=0 TO 7
420 LET A=16384+INT(I/8)*2048+(I-INT(I/8)*8)*32
430 LET H=INT(A/256): LET L=A-H*256
440 POKE 32075+I*8+J,L: POKE 32251+I*8+J,H+J
450 NEXT J: NEXT I
500 REM MAIN PROGRAM
510 LIST 380
520 IF INKEY$<>"" THEN GO SUB 100
530 GO TO 520
```

К сожалению, этого недостаточно в тех случаях, когда в программе используются такие операторы, как GO TO или GO SUB, и т.д. Поэтому мы должны найти все ключевые слова GO SUB, GO TO, RESTORE и RUN и, если за ними следует целое число, проверить, был ли изменен этот номер строки.

LISTING 13.8

```
100 REM SELF-LISTING PROGRAM
110 LET I=23755: CLS
120 LET LINE=256*PEEK I+PEEK(I+1): IF LINE>9999.5 THEN STOP
130 PRINT " ";LINE;
140 LET I=I+2: PRINT PAPER 5;PEEK I;",";PEEK (I+1);" ";
150 LET LENGTH=PEEK I+256*PEEK(I+1)
160 LET NUM=0: LET I=I+1
170 FOR J=1 TO LENGTH: LET I=I+1
180 LET P=PEEK I: IF P>32 AND NUM<=0 THEN LET NUM=1
190 IF P=13 THEN PRINT PAPER 4;P: PRINT: GO TO 230
200 IF P=14 THEN LET NUM=6
210 IF NUM>0 THEN PRINT PAPER 6;P: PRINT: GO TO 230
220 PRINT CHR$ P;
230 LET NUM=NUM-1
240 NEXT J
250 LET I=I+1: GO TO 120
```

Если да, то должны быть изменены как символьное представление этого числа, так и двоичное. Все, что надо сделать для того, чтобы уничтожить последовательность строк, это увеличить длину строки, предшествующую этой последовательности так, чтобы она целиком покрывала эту последовательность. Правильная длина этой строки устанавливается при повторном редактировании и повторном вводе этой строки. Остаток программы передвигается в памяти в то место, где теперь кончается строка.

На листинге 13.9 приведены две программы: программа перенумерации запускается командой RUN 9900, а программа уничтожения - командой RUN 9970.

Упражнение 13.3

Напишите подпрограмму, которая бы просматривала участок памяти, отведенный для хранения текстов программ на бейсике и находила любую заданную последовательность кодов. Используйте метод, примененный в программе уничтожения строк, для присвоения системной переменной EPPC (эта переменная контролирует положение курсора редактора) значения номера строки, где найдена эта последовательность.

LISTING 13.9

```
9900 INPUT "RENUMBER LINES ";LOW;" TO ";UP,"STARTING AT ";NEW;" IN STEPS OF
";STEP
9901 LET X=9900: IF LOW>X OR UP>X OR LOW>UP THEN GO TO 9900
9902 IF NEW>X OR NEW<1 OR STEP>X OR STEP<1 THEN GO TO 9900
9903 PRINT AT 19,0;"RENUMBER LINES ";LOW;" TO ";UP,"STARTING AT ";NEW;" IN
STEPS OF ";STEP,"PLEASE WAIT"
9904 LET START=23755: LET SCREEN=16384: LET HI=256
9905 LET A=START: LET B=SCREEN
9906 LET H=PEEK A: LET L=PEEK(A+1)
9907 POKE B,H: POKE B+1,L
9908 LET A=A+2: LET B=B+2
```

```
      9909 LET LEN=PEEK A+PEEK(A+1)*HI: LET A=A+LEN+2
      9910 IF H*HI+L<X THEN GO TO 9906
      9911 LET A=START+A: LET B=SCREEN: LET W=NEW
      9912 LET H=PEEK A: LET L=PEEK(A+1)
      9913 LET N=HI*H+L: IF N<LOW OR N>UP THEN GO TO 9918
      9914 IF W>X THEN GO TO 9955
      9915 LET H=INT(W/HI): LET L=W-H*HI
      9916 POKE A,H: POKE A+1,L
      9917 LET W=W+STEP
      9918 LET A=A+2
      9919 LET LEN=PEEK A+PEEK(A+1)*HI: LET A=A+LEN+2
      9920 IF N<X THEN GO TO 9912
      9921 LET A=START: LET B=SCREEN
      9922 LET H=PEEK A: LET L=PEEK(A+1): LET NN=PEEK B*HI+PEEK(B+1)
      9923 LET N=H*HI+L: IF N=X THEN STOP
      9924 PRINT AT 21,0;"CHECKING ";NN;" = NEW LINE ";N
      9925 LET A=A+2: LET W=A+2
      9926 LET LEN=PEEK A+PEEK(A+1)*HI: LET A=A+LEN+2
      9927 LET T=PEEK W
      9928 IF T=CODE("GO TO") OR T=CODE("GO SUB") OR T=CODE("RUN") OR T=CODE
("RESTORE") THEN GO SUB 9932
      9929 IF T=14 THEN LET W=W+5
      9930 LET W=W+1: IF W<A THEN GO TO 9927
      9931 LET B=B+2: GO TO 9922
      9932 LET V=W+1: LET A$=""
      9933 LET S=PEEK V: IF S<>32 AND S<>14 AND NOT (S>=48 AND S<=57) THEN RETURN
      9934 IF S<>14 THEN LET V=V+1: LET A$=A$+CHR$ S: GO TO 9933
      9935 LET V=V+3: LET AT=PEEK V+PEEK(V+1)*HI
      9936 IF NAT<LOW OR NAT>UP THEN RETURN
      9937 GO SUB 9943: LET H=INT(NAT/HI): LET L=NAT-H*HI
      9938 LET B$=STR$ NAT: IF LEN A$<>LEN B$ THEN GO SUB 9951
      9939 POKE V,L: POKE V+1,H
      9940 FOR I=1 TO LEN B$
      9941 POKE W+I,CODE B$(I): NEXT I
      9942 RETURN
      9943 LET C=START: LET D=SCREEN
      9944 LET H=PEEK D: LET L=PEEK(D+1)
      9945 LET E=PEEK C*HI+PEEK(C+1)
      9946 IF E>=X THEN LET NAT=0: RETURN
      9947 IF H*HI+L=AT THEN LET NAT=E: RETURN
      9948 LET C=C+2: LET D=D+2
      9949 LET LEN=PEEK C+PEEK(C+1)*HI: LET C=C+LEN+2
      9950 GO TO 9944
      9951 LET DIFF=LEN A$-LEN B$
      9952 IF DIFF>0 THEN FOR I=1 TO DIFF: LET B$=D$+" ": NEXT I: RETURN
      9953 PRINT AT 16,0;"NO ROOM AT ";CHR$ T;AT;" IN LINE ";NN,"TYPE EDIT AND ADD
";-DIFF;" SPACE(S)","TO LABEL THEN RE-RUN PROGRAM."
      9954 LET H=INT(NN/HI): POKE 23626,H: POKE 23625,NN-H*HI
      9955 LET A=START: LET B=SCREEN
      9956 LET H=PEEK A: LET L=PEEK(A+1)
      9957 IF H*HI+L=X THEN PRINT AT 0,0;"RENUMBER ABORTED": STOP
      9958 POKE A,PEEK B: POKE A+1,PEEK(B+1)
      9959 LET A=A+2: LET B=B+2
      9960 LET LEN=PEEK A+PEEK(A+1)*HI: LET A=A+LEN+2
      9961 GO TO 9956
      9970 INPUT "DELETE LINES ";LOW;" TO ";UP: IF LOW<2 OR LOW>=UP THEN GO TO 9970
      9971 PRINT AT 20,0;"DELETE LINES ";LOW;" TO ";UP,"PLEASE WAIT"
      9972 LET START=23755: LET SCREEN=16384: LET HI=256
      9973 LET A=START: LET B=SCREEN
      9974 LET H=PEEK A: LET L=PEEK(A+1)
      9975 IF H*HI+L>=LOW THEN PRINT AT 0,0;"PLEASE ENTER LINE ";LOW-1;" REM","AND
RE-RUN PROGRAM": STOP
      9976 LET H=PEEK A: LET L=PEEK(A+1)
      9977 IF H*HI+L>=LOW THEN GO TO 9981
      9978 LET NN=H*HI+L: LET LAST=A: LET A=A+2
      9979 LET LEN=PEEK A+PEEK(A+1)*HI: LET A=A+LEN+2
```

```
9980 GO TO 9976
9981 LET LAST=LAST+2
9982 LET LONG=PEEK LAST+PEEK(LAST+1)*HI
9983 IF H*HI+L>UP THEN PRINT AT 0,0;"NO LINES IN RANGE": STOP
9984 LET H=PEEK A: LET L=PEEK(A+1): IF H*HI+L=UP THEN GO TO 9990
9985 LET A=A+2: LET LONG=LONG+2
9986 LET LEN=PEEK A+PEEK(A+1)*HI: LET A=A+LEN+2: LET LONG=LONG+LEN+2
9987 IF H*HI+L<>UP THEN GO TO 9984
9990 LET H=INT(LONG/HI): POKE LAST+1,H: POKE LAST,LONG-H*HI
9995 LET H=INT(NN/HI): POKE 23626,H: POKE 23625,NN-H*HI
9999 PRINT AT 0,0;"TYPE EDIT AND ENTER": STOP
```

Структура программы на бейсике (эффективные программы)

Когда мы программируем на бейсике, то знание того, как строки хранятся в памяти, помогает писать эффективные программы, быстро работающие и экономящие память. Каждая числовая константа сопровождается её двоичным кодом, так что нет необходимости перевода числа из символьного представления в двоичное каждый раз, когда выполняется эта строка программы. Если же мы присвоим это число в качестве значения переменной, то для этого потребуется количество байт памяти, равное числу символов в имени переменной. Для присвоения значения мы используем символы ":", "LET", "=", а также имя переменной. Если число используется часто, то это дает существенную экономию памяти. В качестве иллюстрации вышесказанного рассмотрим первые две строки программы

```
10 PRINT AT 1,1
20 LET A=1: PRINT AT A,A
30 PRINT AT 1,1
40 PRINT AT A,A
```

Строка с номером 10 занимает 17 байт памяти: "PRINT", "AT", "1", 14, 0, 0, 1, 0, 0, "1", 14, 0, 0, 1, 0, 0.

Строка с номером 20 занимает 16 байт памяти: "LET", "A", "=", "1", 14, 0, 0, 1, 0, 0, ":", "PRINT", "AT", "A", ",", "A".

Для кодирования второй строки требуется меньше символов, чем для первой; вторая строка использует символ A, который занимает в любом месте программы шесть байт. Если мы хотим повторить произведенные выше операции, как это сделано в 30-ой и 40-ой строках программы, то обнаружим, что 30-ая строка занимает 17 байт, а строка 40 - всего пять. Мы видим, что если число встречается в программе достаточно часто, то использование переменных дает существенную экономию памяти.

Экономию памяти также дает размещение на одной строке максимального числа операторов. Следует отметить, однако, что это снижает удобочитаемость программы. Каждая новая строка занимает пять байт памяти: в один записывается код 13, отмечающий конец предыдущей строки, в двух других байтах - номер строки и ещё два байта отводятся под длину строки. Для двоеточия между операторами требуется только один байт.

Рассмотрим, например, небольшую программу, состоящую из 90 операторов: если первоначально операторы записаны по одному в строке, а затем нам удается поместить на каждой строке по три оператора, то этим достигается экономия в 60*(5-1)=240 байт. В больших программах этот способ позволяет сэкономить более 1К памяти. Кроме того, использование меньшего числа строк сокращает время выполнения таких команд, как GO TO и GO SUB, гак как тратится меньше времени на поиск нужной строки.

Наиболее часто используемые подпрограммы следует располагать ближе к началу программы. Если же нет необходимости в экономии времени, то подпрограммы следует располагать в порядке логического следования.

Упражнение 13.4

Перепишите и перерасположите графические подпрограммы с тем, чтобы они занимали меньше места и тратили меньше времени на выполнение.

Синхронное изображение

Используя команду PAUSE, можно получить забавный узор на рамке, окаймляющей графическое поле. Изображение на телеэкране сменяется пятьдесят раз в секунду, и в команде PAUSE указывается число пятидесятых долей секунды.

Команда PAUSE 1 не всегда дает задержку в 1/50 долю секунды, так как эта команда выполняется до момента появления нового изображения. Этот факт делает возможным соотносить моменты запуска команд с моментами смены кадра на дисплее. Если во время заполнения экрана мы меняем командой BORDER цвет рамки, то для этого кадра верхняя часть рамки будет одного цвета, а нижняя - другого. Очевидно, что мы можем использовать команду PAUSE для того, чтобы подождать смены кадров и повторить операции с новым кадром. Если мы будем постоянно повторять эту операцию, то на экране возникнет неподвижное изображение.

Команды SAVE и LOAD используют для этой цели подпрограммы, написанные в машинных кодах, в результате чего на рамке появляются полосы.

Пусть программа начинается так:

```
1 GO TO 10
2 PAUSE 1: BORDER 7: BORDER 2: BORDER 6: BORDER 4: BORDER 5: BORDER 1: BORDER 3: BORDER 7: BORDER 7: GO TO 2
```

Подпрограмма заканчивается командой GO TO 2, в результате чего после построения рисунка на рамке появляется последовательность цветных полос. Следует отметить, что если эти строчки поместить не в начале

программы, то выполнение команды GO TO может занять слишком много времени.

Упражнение 13.5

Напишите программу, которая бы окрашивала рамку в красный и циановый цвета, не используя для этой цели оператора PAUSE. Для этого вставьте между операторами дополнительные двоеточия, пока время выполнения команды не составит в точности 1/50 с, что и даст неподвижное изображение (подсказка примите время выполнения команды BORDER за 10, GO TO за 13, двоеточия - за 1, и подберите такое количество операторов, чтобы в этих единицах время выполнения всей строки равнялось 160).

Список программ

1. Листинги 13.1. Входные данные: десять наборов координат X и Y. На экране появляется таблица этих значений, за которыми идет надпись "END". Установленный в начальный момент у первой строки курсор можно передвигать вниз нажатием клавиши прописное "6" и вверх нажатием клавиши прописное "7". После того как курсор установлен у нужной строки, следует нажать "ENTER" (прописное "I"), компьютер запрашивает координаты элемента изображения, соответствующего этой строке. После того, как вы ввели все данные, установите курсор у надписи E, нажмите EDIT (прописное "I") и "Y". Программа нарисует многоугольник, соединив 10 точек.
2. Листинг 13.2. Входные данные: число для файловой системы. Введите любое число от "1" до "6". Если вы введете "5" или "6", то компьютер потребует подтвердить, что вы дали верный ключ: если вы подтверждаете, то напечатайте "Y", если нет, то "". По команде BREAK программа прекращает работу.
3. Листинг 13.3. Входных данных нет, по команде BREAK программа прекращает работу.
4. Листинг 13.4. Наберите любую последовательность символов. По команде BREAK программа прекращает работу.
5. Листинг 13.5. После звукового сигнала следует загрузить с ленты пять кадров. Напечатайте "1", "2", "3", "4", "5" для воспроизведения соответствующего изображения, " " для получения эффекта мультипликации, " ", чтобы прекратить смену кадров.
6. Листинг 13.6. После заполнения экрана любая клавиша вызовет "свертку " изображения на экране, а команда BREAK остановит этот процесс.
7. Листинг 13.7. Таже программа, что и в пункте 6, но написанная на бейсике.
8. Листинг 13.8. Входных данных нет.
9. Листинг 13.7 и 13.9. Наберите RUN 9900 для перенумерации строк и RUN 9970 для уничтожения строки. Пример - редактирование листинга 13.7:

```
RUN 9900
RENUMBER LINES 10 TO 30
STARTING AT 601 STEPS OR 5
```

Теперь посмотрите, какие изменения произошли в программе.

```
RUN 9970
DELETE LINES 60 TO 99
```

Остановка: перед удаляемым куском должна быть хотя бы одна строка.

```
RUN 9970
DELETE LINES 65 TO 99
EDIT (прописная "I")
LIST
```

ГЛАВА 14
Действующий пример видеоигры

В этой главе мы изучаем возможности бейсика для программирования видеоигр. Мы приходим к выводу, что программы на бейсике дороги; если учесть, что, как правило, игроки довольно быстро теряют интерес к программам, то естественно, что лица, имеющие навыки программирования на бейсике, предпочитают не писать программы небольших игр самостоятельно, а покупать только программы сложных игр, написанные на машинном коде. На листинге 14.1 приведена программа, не использующая подпрограмм написанных на машинном коде. Для создания таких программ от программиста требуется довольно высокая квалификация. Игра называется ISLAND DEFENCE ("оборона острова") и относится к разряду игр "перестреляй их", однако методы, использованные при составлении программы, пригодны для создания игр типа "мяч-бита" (таких, как теннис), или тактических игр (таких, как PACMAN).

Описание игры

На экране изображен зелёный остров с небольшим холмом, тёмно-синее море и голубое небо с жёлтым солнцем и белой тучкой. На острове имеются три дерева и участок светло-голубого бетона, на котором нарисована палатка. Из палатки выходит человек, подходит к стрелку, отдаёт ему 20 снарядов и возвращается в палатку.

Вражеские аэропланы появляются в небе, перелетают через холм и бомбят лагерь, который защищает стрелок, пытаясь выстрелами сбить аэропланы. Во время атаки аэроплан либо сбивают, либо он бомбит лагерь.

После каждого попадания бомбы размер тента уменьшается. После 14 попаданий тент исчезает, и самолеты бомбят огневую точку, после уничтожения которой рамка графического поля становится жёлтой, и на экране появляется та же сцена. В конце каждого рейда самолёт пролетает через тучу над орудием и покидает экран. Боевые запасы пополняются после 20-ти налётов. Если орудие разбомблено третий раз, то экран становится красным, и игра начинается снова.

Нажатием клавиш от "1" до "7" орудие устанавливается в одно из семи положений; после нажатия клавиши X происходит выстрел. Из-за ограничений, накладываемых медленной работой программы, на экране могут одновременно находиться только один самолёт и снаряд.

Для лучшего понимания материала этой главы мы рекомендуем вам загрузить программу на листинге 14.1 и запустить её. Программа попросит вас сначала загрузить набор специальных символов, затем подпрограмму "GAMESET" (эта подпрограмма должна рисовать тент, человека и т.д.), подпрограмму "BACKGROUND" (должна рисовать пейзаж), после чего можно начать игру.

Текст программы хорошо прокомментирован, так что нет необходимости объяснять принцип работы программы. Мы ограничимся изложением общего подхода к созданию таких игр.

ВНИМАНИЕ! Листинги с программами "GAMESET" и "BACKGROUND" в настоящем издании не приведены по не зависящим от издательства причинам.

Передний план

Перед написанием программы следует тщательно продумать схему игры. Сначала набросайте грубый эскиз на клетчатой бумаге, на котором должны быть обозначены неподвижные объекты и области, которые будут пересекаться подвижными объектами (например, те символьные блоки, которые в какой-нибудь момент будут включены в число блоков, из которых состоит изображение самолёта). Вы должны убедиться, что игровое поле полностью умещается на экране и что ни при каких обстоятельствах в символьном блоке не используются одновременно более двух цветов. После того, как вы будете четко представлять, как выглядит игровое поле, вам следует приступить к созданию объектов на переднем плане (в нашем случае - взрывы, самолёты, люди), т.е. движущихся объектов. Самая быстрая из команд бейсика, с помощью которой можно разместить на экране большой объект - это команда PRINT, поэтому мы создаем изображение объектов с помощью программы "CHARACTER GENERATOR" из определенного нами набора символов, для того, чтобы получить в точности то изображение необходимой величины и повторно отредактировать не удовлетворяющие вас символы. Возможно, что вам придется добавить куски вашей программы к концу программы CHARACTER GENERATOR (опция 7, глава 5), что позволит вам увидеть объекты в действии до того, как найдена их окончательная форма.

Упражнение 14.1

Добавьте опцию к программе CHARACTER GENERATOR с тем, чтобы стало возможным редактировать один символ из набора символов. Эта опция должна помещать восемь двоичных значений редактируемого символа в заданные ячейки дисплейного файла; например, два блока, из которых состоит самолёт, когда он летит горизонтально и направо.

LISTING 14.1

```
9 REM INITIALISE ROUTINES TO ALLOW USE OF ALTERNATE CHARACTER SET.
10 CLEAR 62294: INK 0: PAPER 7: BORDER 7: FLASH 0
20 DIM S(6): FOR I=1 TO 6: READ S(I): NEXT I
30 DATA 15360,62039,62807,63575,64343,64848
40 LET SET=50: LET S=1: GO SUB SET: GO TO 200
50 REM SET/CHANGE TO SET S
51 REM IN : S
60 LET HI=INT(S(S)/256): LET LO=S(S)-HI*256
70 POKE 23606,LO: POKE 23607,HI: RETURN
79 REM LOAD IN GAME CHARACTERS FOR SET 2.
```

```
      80 REM CHARLOAD
      90 LET N$="GAMESET"
      100 LET S=2
      110 INPUT ("LOADING"+N$+CHR$ 6+"START TAPE, THEN PRESS ENTER.")
      120 LOAD N$ CODE (S(S)+256),768: RETURN
      200 REM MAIN PROGRAMM
      209 REM INITIALISE VARIABLES POINTING TO ROUTINES.
      210 LET CHARLOAD=80: LET LOAD=4600: LET CREATE=5000: LET CREDIT=5500: LET
CHAR=5600
      220 LET KEYBOARD=500: LET CAMP=700: LET STATUS=800: LET PLANE=980
      230 LET RELOAD=1500: LET AMMO=1900: LET MISSILE=2000: LET EXPLODE=2500
      240 LET BOMBSAMP=3500: LET BOMBGUN=3000: LET HISCORE=5700: LET HSC=0
      249 REM IF CHARACTER SET IS NOT IN PLASE LOAD IT.
      250 IF PEEK 62303<>110 THEN GO SUB CHARLOAD
      258 REM PRINT NAMES ON BOTTOW TWO LINES.
      260 GO SUB LOAD: GO SUB CREATE: GO SUB CREDIT
      269 REM MAKE A COPY OF COLOUR ATTRIBUTES FOR SCREEN.
      270 DIM A(704): FOR I=1 TO 704: LET A(I)=PEEK(22527+I): NEXT I
      279 REM PREPARE STRINGS FOR USE BY DISPLAY ROUTINES.
      280 LET S$="SCORE": LET B$="BASE"
      290 FOR I=1 TO 4: LET S$=S$+CHR$ 8: LET B$=B$+CHR$ 8: NEXT I
      300 DIM N$(9): LET N$="ABCDEFGH"
      309 REM START/RESTART FOR GAME.
      310 BRIGHT 1: PAPER 8: INK 8: OVER 0
      319 REM RESET SCORE AND NO. OF BASES, PRINT OUT SCORE LINE
      320 LET SC=0: LET BS=3: GO SUB STATUS
      329 REM REMOVE REMAINING AMMUNITION AND REMOVE FLASH FROM GUNNERS SGUARE.
      330 PRINT AT 17,5;" ": PRINT AT 18,5;" ": PRINT AT 16,3;" "
      339 REM RESTORE COLOUR ATTRIBUTES FROM COPY.
      340 FOR I=1 TO 704: POKE 22527,A(I): NEXT I
      349 REM POINT NEW TENT AND PUNNER UPDATE SCORE LINE, RELOAD, AMMUNITION.
      350 GO SUB CAMP: GO SUB STATUS: GO SUB RELOAD
      358 REM PREPARE FOR MAIN LOOP OF PROGRAM SET "P" TO TOP OF "PLANE" CASCADE.
      359 REM SET "M" TO "MISSILE" READY STATE, SET "K" TO KEYBOARD ROUTINE.
      360 LET P=PLANE: LET M=MISSILE: LET K=KEYBOARD
      369 REM START A NEW WAVE OF 20 AIRPLANES.
      370 LET AR=20
      379 REM START OF MAIN LOOP
      380 GO SUB P: OVER 1: GO SUB M: OVER 0: GO SUB K
      389 REM JUMP OUT OF LOOP.
      390 IF DEAD THEN GO TO 420
      399 REM IF MORE AIRPLANES TO COME KEEP LOOPING.
      400 IF AR>0 THEN GO TO 380
      409 REM IF END OF PLANES THEN RELOAD AND START NEW WAVE, CONTINUE LOOPING.
      410 GO SUB RELOAD: GO TO 370
      419 REM MAKE SURE PLANE IS GONE, IF YOU HAVE ANY BASES LEFT USE THE NEXT ONE.
      420 INK 0: PRINT AT 12,0;"         ": IF BS>1 THEN LET BS=BS-1: GO TO 350
      429 REM GAME OVER SO FLASH BORDER AND TURN SCREEN BLOOD RED.
      430 OVER 1: FOR I=1 TO 22: BEEP 0.005,RND*30-15: BORDER RND*7
      440 FOR J=1 TO 2: PLOT 0,(22-I)*8: DRAW BRIGHT 1;PAPER 2;INK 2;255,0
      450 NEXT J: NEXT I: BORDER 7
      460 OVER 0
      469 REM CHECK WHETHER YOU BEAT THE HISCORE THEN RESTART GAME.
      470 IF SC>HSC THEN GO SUB HISCORE
      480 PAUSE 200: GO TO 310
      500 REM KEYBOARD
      509 REM IF NO KEY PRESSED WHEN ROUTINE CHECKS THEN YOU MISSED YOUR CHANCE.
      510 LET A$=INKEY$: IF A$="" THEN RETURN
      518 REM THE FIRE KEY IS PRESSED
      519 REM IF YOU'RE NOT OUT OF MISSILES AND A MISSILE IS READY THEN FIRE.
      520 IF A$="X" OR A$="X" THEN IF M=MISSILE AND NOT OUT THEN LET M=F: LET M$=F$:
GO SUB AMMO: OVER 1: GO SUB M: OVER 0: GO TO 550
      529 REM IF THE KEY IS NOT "I" TO "7" THEN WRONG KEY PRESS, IGNORE IT.
      530 IF A$>"7" OR A$<"1" THEN RETURN
      538 REM CALCULATE WHICH ROUTINE IS USED IF MISSILE IS FIRED IN THIS DIRECTION.
      539 REM SET STRING WHICH IS USED FOR MISSILE AND CHANGE GUN DIRECTION.
```

```
      540 LET F=2000+10*VAL A$: LET F$=CHR$(83+VAL A$)
      550 PRINT AT 16,3;F$
      560 RETURN
      700 REM CAMP
      709 REM PRINT NEW GUNNER AND TENT, RESET BORDER TO WRITE.
      710 PRINT AT 17,3;"Z": BRIGHT 0: BORDER 7: PRINT AT 16,23;"VW": PRINT AT
17,23;"X,Y": BRIGHT 1
      718 REM SET HEIGHT OF TENT TO 14, RESET TRUTH-FLAGS,
      719 REM USE KEYBOARD ROTUINE TO INITIALISE GUN POSITION.
      720 LET YT=14: LET NIT=0: LET DEAD=0: LET A$="3": GO TO 540
      800 REM STATUS
      809 REM USE SET 1 TO PRINT OUT SCORE AND NO, OF BASES LEFT THEN BACK TO SET 2.
      810 LET S=1: GO SUB SET
      820 PRINT AT 21,0;S$;SC,B$;BS
      830 LET S=2: GO SUB SET
      840 RETURN
      980 REM PLANE
      989 REM ENSURE THAT "BOMBGUN" POINTER ALWAYS STARTS AT TOP OF CASCADE.
      990 PRINT AT 15,5;" ": PRINT AT 15,4;" ": PRINT AT 17,3;"Z": LET BOMBGUN=3000
      999 REM CHANGE POINTER, IF RESULT IS OVER 1000 THEN PLANE WILL BE LAUNCHE1.
      1000 LET P=991+INT(RND*11): RETURN
      1009 REM LAUNCH AIRCRAFT, START POINTER GOING DOWN CASCADE, LOAD BOMB.
      1010 LET AR=AR-1: LET P=1020: LET BOMB=1: RETURN
      1019 REM MOVE PLANE ONE PLACE TO RIGHT.
      1020 PRINT AT R,C;" IJ": LET C=C+1: IF C<10 THEN RETURN
      1029 REM IF PLANE IS OVER 10 COLUMNS ONTO SCREEN MOVE POINTER DOWN.
      1030 LET P=1040
      1039 REM DIAGONAL DIVE.
      1040 PRINT AT R-1,C;" ": PRINT AT R,C;" N": LET R=R+1: LET C=C+1: PRINT AT
R,C;"N": IF C<20 THEN RETURN
      1049 REM IF OVER 20 COLUMNS ACROSS THEN MOVE POINTER, REMOVE LEFT OVER PLANE.
      1050 LET P=1060: PRINT AT R-1,C;" "
      1059 REM CALL THE "BOMBCAMP" CASCADE EACH OF THE FIVE TIMES THIS SECTION IS
USED.
      1060 GO SUB BOMBCAMP: PRINT AT R,C;" IJ": LET C=C+1: IF C<25 THEN RETURN
      1069 REM MOVE POINTER DOWN AND START CURVE UP TOWARDS CLOUD.
      1070 LET P=1080
      1080 PRINT AT R,C;" ": LET R=R-1: LET C=C+1: PRINT AT R,C;"KL": IF C<28 THEN
RETURN
      1090 LET P=1100: PRINT AT R,C;" ": LET C=C+1
      1100 PRINT AT R+1,C;" ": PRINT AT R,C,"O": LET R=R-1: PRINT AT R,C;"O": IF R>7
THEN RETURN
      1110 LET P=1120: PRINT AT R+1,C;" "
      1120 PRINT AT R,C;" ": LET R=R-1: LET C=C-1: PRINT AT R,C;"KL": IF R>3 THEN
RETURN
      1130 LET P=1140
      1139 REM FLY INTO CLOUD, PLANE STILL MOVES NORMALLY BUT YOU CAN'T SEE IT.
      1140 PRINT AT R,C;"IJ ": LET C=C-1: IF C>11 THEN RETURN
      1150 LET P=1160: PRINT AT R,C;" "
      1159 REM START DIVE TO WARDS GUNNER.
      1160 PRINT AT R,C;" ": LET R=R+1: LET C=C-1: PRINT AT R,C;"M": PRINT AT
R+1,C;"M": IF R<6 THEN RETURN
      1170 LET P=1180
      1180 PRINT AT R,C;" ": LET R=R+1: PRINT AT R,C;"P": PRINT AT R+1,C;"P": IF R<8
THEN RETURN
      1189 REM THIRD SECTION OF DIVE CHECK AND ERASE POSSIBLE LEFT OVER MISSILE.
      1190 LET P=1200: IF X<>9 AND Y<>7 THEN PRINT AT 7,9;" "
      1200 PRINT AT R,C;" ": LET R=R+1: LET C=C-1: PRINT AT R,C;"M": PRINT AT
R+1,C;"M": IF R<11 THEN RETURN
      1210 LET P=1220: PRINT AT R,C;" ": LET R=R+1
      1219 REM LAST SECTION OF FLIGHT, CALL BOMBGUN CASCADE EACH TIME.
      1220 GO SUB BOMBGUN: LET C=C-1: PRINT AT R,C;"IJ ": IF C>0 THEN RETURN
      1229 REM LAST TWO PARTS OF CASCADE DEAL WITH PLANE GOING OF SCREEN.
      1230 LET P=1240: PRINT AT R,C;"J ": RETURN
      1239 REM RESET PLANE'S ROW TO 2 AND SET POINTER BACK TO TOP OF CASCADE.
      1240 PRINT AT R,C;" ": LET R=2: LET P=PLANE
```

```
      1250 RETURN
      1500 REM RELOAD
      1509 REM IT THE CAMP IS DESTROYED THEN DONT RELOAD AMMUNITION IF NIT THEN
RETURN
      1519 REM EXPLICIT INSTRUCTIONS FOR ANTIMATION OF FIGURE.
      1520 BRIGHT 0: LET R=18
      1529 REM WALK FROM TENT TO TREES
      1530 FOR C=24 TO 19 STEP -1: PRINT AT R,C;"S": PRINT AT R+1, C;"S": BEEP 0.01,-
15: PAUSE 5
      1540 PRINT AT R,C;"R":PRINT AT R+1,C;" ": PAUSE 3
      1550 PRINT AT R,C;" ":PRINT AT R+1,C;" ": NEXT C
      1559 REM USE OVERPRINTING SO THAT TREES AREN'T DAMAGED AS FIGURE GETS CLOSE.
      1560 OVER 1: PRINT AT R,18;"S": PRINT AT R+1,18;"S": BEEP 0.01,-15: PAUSE 5:
PRINT AT R,18;"S": PRINT AT R+1,18;"S"
      1570 PRINT AT R,18;"R": PRINT AT R+1,18;" ": PAUSE 3: PRINT AT R,18;"R": PRINT
AT R+1,18;"R"
      1579 REM MIKE NOISES AS THOGH FIGURE GOES BEHIND TREE.
      1580 FOR J=1 TO 2: BEEP 0.01,-15: PAUSE 11: NEXT J
      1589 REM SHOW TEET VWERGING FROM BEHIND TREE.
      1590 PRINT AT R+1,15;"S": BEEP 0.01,15
      1600 PRINT AT R+1,15;"R": PAUSE 3: PRINT AT R+1,15;"R"
      1609 REM SHOW WHOLE FIGURE COMING OUT OF TREE.
      1610 PRINT AT R,14;"S": PRINT AT R+1,14;"S":BEEP 0.01,-15: PAUSE 5: PRINT AT
R,14;"S": PRINT AT R+1,14;"S"
      1620 PRINT AT R,14;"R": PRINT AT R+1,14;"R": PAUSE 3: PRINT AT R,14;"R": PRINT
AT R+1,14;"R"
      1629 REM WALK FROM TREE TO END OF STRIP.ONTO GRASS AT SOLUMN 8 WITH BRIGHT ON.
      1630 OVER 0: FOR C=13 TO 8 STEP -1: BRIGHT (C=8)
      1640 PRINT AT R,C;"S": PRINT AT R+1,C;"S": BEEP 0.01,-15: PAUSE 5
      1650 PRINT AT R,C;"R": PRINT AT R+1,C;"R": PAUSE 3
      1660 PRINT AT R,C;" ": PRINT AT R+1,C;" ": NEXT C
      1669 REM PRINT FIGURE AT COLUMN 7 READY TO REFILL AMMO DUMP.
      1670 PRINT AT R,C;"R": PRINT AT R+1 ,C;"R"
      1679 REM REFILL AMMO DUMP, WITH SOUND EFFECTS.
      1680 FOR G=6 TO 5 STEP -1: FOR H=18 TO 17 STEP -1: FOR N=2 TO 9
      1690 PRINT AT H,G;INK 3;N$(N): BEEP 0.02,H+G-N
      1700 NEXT N: NEXT H: NEXT G
      1709 REM REMOVE FIGURE FROM COLUMN 7.
      1710 PRINT AT R,C;" ": PRINT AT R+1,C;" "
      1719 REM RESET AMMUNITION PRINT AND TRUTH FLAGS.
      1720 LET N=8: LET OUT=0: LET H=17: LET G=6
      1729 REM WALK FROM EN D OF STRIP TO TREE.BRIGH OFF AFTER COLUMN 8.
      1730 FOR C=8 TO 13: BRIGHT (C=8)
      1740 PRINT AT R,C;")": PRINT AT R+1,C;"+": BEEP 0.01,-15: PAUSE 5
      1750 PRINT AT R,C;"(": PRINT AT R+1,C;"*": PAUSE 3
      1760 PRINT AT R,C;" ": PRINT AT R+1,C;" ": NEXT C
      1769 REM OVER-PRINT AS FIGURE REACHES TREE.
      1770 OVER 1: PRINT AT R,14;")": PRINT AT R+1,14;"+": BEEP 0.01,-15: PAUSE 5:
PRINT AT R,14;")": PRINT AT R+1,14;"+"
      1780 PRINT AT R,14;"(": PRINT AT R+1,14;"*": PAUSE 3: PRINT AT R,14;"(": PRINT
AT R+1,14;"*"
      1789 REM OVERPRINT LEGS AS THEY GO INTO TREE.
      1790 PRINT AT R+1,15;"+": BEEP 0.01,-15
      1800 PRINT AT R+1,15;"*": PAUSE 3: PRINT AT R+1,15;"*"
      1809 REM MAKE NOISES AS THOUGH IS BEHING TREE.
      1810 FOR J=1 TO 2: BEEP 0.01,-15: PAUSE 11: NEXT J
      1819 REM OVERPRINT AS FIGURE EMERGES FROM TREE.
      1820 PRINT AT R,18;")": PRINT AT R+1,18;"+":BEEP 0.01,-15: PAUSE 5: PRINT AT
R,18;")": PRINT AT R+1,18;"+"
      1830 PRINT AT R,18;"(": PRINT AT R+1,18;"*": PAUSE 3: PRINT AT R,18;"(":PRINT
AT R+1,18;"*"
      1839 REM WALK FROM TREE TO TENT.
      1840 OVER 0: FOR C=19 TO 24: PRINT AT R,C;")": PRINT AT R+1,C;"+": BEEP 0.01,-
15: PAUSE 5
      1850 PRINT AT R,C;"(": PRINT AT R+1,C;"*": PAUSE 3
      1860 PRINT AT R,C;" ": PRINT AT R+1,C;" ": NEXT C
```

```
      1869 REM RESET ROM AND COLUMN FOR USE BY PLANE ROUTINE.
      1870 LET C=0: LET R=2: BRIGHT 1
      1880 RETURN
      1900 REM AMMO PRINT MISSILE DUMP
      1902 REM OUT: OUT
      1909 REM TAKE ONE OF AMUNITION DUMP AND CHEK WHETHER OUT OF AMMO.
      1910 PRINT AT H,G;N$(N): LET H=N-1
      1920 IF N=0 THEN LET N=8: LET H=H+1: IF H=19 THEN LET H=17: LET G=G-1: IF G=4
THEN LET OUT=1
      1930 RETURN
      2000 REM MISSILE DIRECTIONS
      2009 REM REMOVE OLD MISSILE BY OVERPRINTING ANDCALCULATE NEW POSITION.
      2010 PRINT AT Y,X;M$: LET Y=Y-3: LET X=X-2: GO TO 2100
      2020 PRINT AT Y,X;M$: LET Y=Y-3: LET X=X-1: GO TO 2100
      2030 PRINT AT Y,X;M$: LET Y=Y-3: GO TO 2100
      2040 PRINT AT Y,X;M$: LET Y=Y-3: LET X=X+1: GO TO 2100
      2050 PRINT AT Y,X;M$: LET Y=Y-3: LET X=X+2: GO TO 2100
      2060 PRINT AT Y,X;M$: LET Y=Y-2: LET X=X+2: GO TO 2100
      2070 PRINT AT Y,X;M$: LET Y=Y-2: LET X=X+3: GO TO 2100
      2098 REM IF MISSILE IS ON SAME COLUMN AS PLANE CHECK FOR BEING NEAR ROW OF
PLANE.
      2099 REM IF MISSILE IS CLOSE ENDUNG EXPLODE AND RESET MISSILE TO READY.
      2100 IF X=0 THEN IF ABS(Y-R)<2 THEN GO SUB EXPLODE: LET M=MISSILE: GO TO M
      2109 REM IF MISSILE IS OFF SCREEN THEN RESET POINTER "M" TO READY
      2110 IF X<0 OR Y<0 THEN LET M=MISSILE: GO TO M
      2119 REM PRINT MISSILE AT NEW POSITION.
      2120 PRINT AT Y,X;M$: RETURN
      2200 REM MISSILE/READY TO FIRE
      2210 LET X=3: LET Y=16
      2220 RETURN
      2500 REM EXPLOCE
      2508 REM MISSILE HAS HIT PLANE SO BLOW UP PLANE AND ADD TO SCORE.
      2509 REM CHECKWHETHER EXPLOSION HAS TAKEN PLACE IN CLOUD OR IN SKY.
      2510 OVER 0: LET SKY=(R<>3 OR C<16): IF NOT SKY THEN PRINT AT R,C-1;" "
      2519 REM CENTRAL FLASH AND LOW BUZZ, ADD TO SCORE FOR NITTING PLANE.
      2520 INK 2: IF SKY THEN PRINT AT R,C;"T"
      2530 FOR I=1 TO 5: BEEP 0.01,I-10: BEEP 0.01,-I-10: NEXT I: LET SC=SC+1
      2539 REM IF EXPLOSION IS SEEN THEN PRINT CLOUD OF DEBRIS.
      2540 IF SKY THEN PRINT AT R-1,C-1;"> <": PRINT AT R,C-1;"S!$": PRINT AT R+1,C-
1;":%;"
      2549 REM NIGH PITCHED WHIZZ.
      2550 FOR I=-4 TO 4: BEEP 0.002,(50+ABS I): NEXT I
      2559 REM REMOVE ALL TRACE OF EXPLOSION.
      2560 INK 0: IF SKY THEN PRINT AT R-1,C-1;" ": PRINT AT R,C-1;" ": PRINT AT
R+1,C-1;" "
      2569 REM PRINT OUT NEW SCORE LINE,RESET PLANE AND MISSILE POINTERS.
      2570 INK 8: GO SUB STATUS: LET M=MISSILE: LET P=PLANE
      2580 LET R=2: LET C=0
      2590 RETURN
      3000 REM BOMB GUN
      3009 REM IF CAMP ISN'T DESTROYED OR PLANE HASN'T GOT A BOMB LEFT THEN DON'T
DROP.
      3010 IF NOT HIT OR NOT BOMB THEN RETURN
      3019 REM PLANE BOMB ON SCREEN AND START POINTER DOWN CASCADE.
      3020 PRINT AT 13,5;".": LET BOMBGUN=3030: RETURN
      3030 PRINT AT 13,5;" ": PRINT AT 15,4;".": LET BOMBGUN=3040: RETURN
      3040 PRINT AT 13,4;" ": PRINT AT 17,3;".": LET BOMBGUN=3050: RETURN
      3050 PRINT AT 17,3; FLASH 1;" ": LET BOMBGUN=3060: RETURN
      3059 REM CLEAR PLANE AWSY, MAKE SURE ANY MISSILES SHOOT OFF THE SCREEN.
      3060 PRINT AT 16,0;" ": OVER 1: FOR I=1 TO 7: GO SUB M: NEXT I
      3069 REM FLASH BORDER AND MAKE EXPLODING NOISES.
      3070 FOR I=1 TO 50: BORDER RND*7: BEEP 0.01,RND*30-15: NEXT I
      3078 REM SET FLAG FOR END OF GUNNER.
      3079 REM SET INK TO SKY-BLUE SUBSEQUENT MOVEMENT OF PLANE IS INVISIBLE.
      3080 LET READ=1: OVER 0: INK 5
      3089 REM RESET CASCADE.
```

```
3090 LET BOMBGUN=3000: RETURN
3500 REM BOMB CAMP
3509 REM IF THE CAMP HAS GONE SAVE BOMB FOR THE GUNNER.
3510 IF HIT THEN RETURN
3519 REM MARK BOMB AS DROPPED AND START CASCADE.
3520 LET BOMB=0: PRINT AT 14,21;".": LET BOMBCAMP=3530:  RETURN
3530 PRINT AT 14,21;" ": PRINT AT 15,22;BRIGHT 0;".": LET BOMBCAMP=3540: RETURN
3540 BRIGHT 0: PRINT AT 15,22;" "
3549 REM EXPLODE TENT.
3550 PRINT AT 16,23;OVER 1;INK 2;"#S"
3560 PRINT AT 16,23;OVER 1;INK 2;"%$"
3570 LET BOMBCAMP=3580: BRIGHT 1: RETURN
3579 REM REMOVE EXPLOSION FROM TENT.
3580 BRIGHT 0: PRINT AT 16,23;OVER 1;INK 0;"#S"
3590 PRINT AT 17,23;OVER 1;INK 0;"%$"
3600 LET BOMBCAMP=3610: BRIGHT 1: RETURN
3609 REM REMOVE TOP REMAINING LINE OF TENT.
3610 PLOT INVERSE 1;184,33+YT: DRAW INVERSE 1;16,0
3619 REM CHECK WHETHER TENT IS GONE AND RESET BOMBCAMP CASCADE.
3620 LET BOMBCAMP=3500: LET YT=YT-1: LET HIT=(YT=0)
3630 RETURN
4600 REM LOAD
4609 REM LOAD PICTURE FROM TAPE.
4610 LET N$="BACKGROUND"
4620 INPUT (" LOADING"+N$+CHR$ 6+"START TAPE.THEN PRESS ENTER."); LINE X$
4630 LOAD (N$)SCREEN$
4640 RETURN
5000 REM CREATE/CHARACTERS
5009 REM MAKE CHARACTERS FOR AMMO DUMP IN USER DEFINED GRAPHICS SET.
5010 LET D=255
5020 FOR I=0 TO 6: FOR J=0 TO 7
5030 LET P=USR "G"-I*8+J
5040 IF J>I THEN POKE P,D
5070 NEXT J: NEXT I
5080 RETURN
5500 REM CREDIT
5509 REM POINT ON BOTTOW TWO LINES.
5510 DEF FN S(R,C)=16384+INT(R/8)*2048+(R-INT(R/8)*8)*32+C
5520 DEF FN C(A$)=15360+CODE A$*8
5530 LET R=22: LET A$="]/ISLAND DEFENCE/[ BY BJJ $ IOA"
5540 FOR J=1 TO LEN A$: LET C=J-1: GO SUB CHAR: NEXT J
5550 LET R=23: LET A$="HI-SCORE SCORED BY BJJ"
5560 FOR J=1 TO LEN A$: LET C=J-1: GO SUB CHAR: NEXT J
5570 RETURN
5600 REM CHAR
5609 REM PRINT J'TH CHARACTER OF A$ AT D,C, CHANGE ATTRIBUTE TO RED PAPER AND
WHITE INK.BEEP.
5610 LET AT=FN S(R,C): LET FROM=FN C(A$(J))
5620 FOR I=0 TO 7: POKE AT+I*256, PEEK(FROM+I): NEXT I
5630 POKE 22528+C+R*32,87: BEEP 0.03,CODE A$(J)-50
5640 RETURN
5700 REM HISCORE
5709 REM CHANGN HI-SCORE AND PRINT IT ON BOTTOM OF SCRN.
5710 LET HSC=SC: LET A$=STR$ HSC
5720 LET R=23: FOR J=1 TO LEN A$: LET C=10+J: GO SUB CHAR: NEXT J
5729 REM CHANGE NAME TO FLASHING LETTERS.
5730 FOR I=23291 TO 23293: POKE I,PEEK I+128: NEXT I
5739 REM GET THREE INITIALS FOR HISCORE FROM KEYBOARD.
5740 LET J=1: FOR C=27 TO 29
5750 IF INKEY$<>"" THEN GO TO 5750
5760 IF INKEY$="" THEN GO TO 5760
5770 LET A$=INKEY$: GO SUB CHAR: NEXT C
5780 RETURN
```

Фон

Фон создается с помощью программы CHARACTER GENERATOR и программы для создания диаграмм. Сначала с помощью подпрограммы "PAPER" наносятся прямоугольные области, аппроксимирующие большие

участки игрового поля (например, трава, небо, солнце, и т.д.). На это изображение накладываются более мелкие детали (например, деревья, мешки с песком, края травы, солнца, облаков и т.д.). Эта работа может быть выполнена с помощью программ для построения диаграмм ("POINT", "LINE"). Например, добавляя к рисункам взрывов точки и прямые, можно получить изображение деревьев.

Цветовую гамму можно расширить, используя команду BRIGHT. Особое внимание следует уделять блокам, через которые будут проходить движущиеся объекты. Эти блоки должны содержать только один фоновый цвет, так как в противном случае при прохождении через этот блок объекта может возникнуть недопустимая ситуация - наличие в одном символьном блоке более двух цветов. На нашем игровом поле есть два таких участка.

1. Во время полёта самолёты проходят поверх облака и за ним, так что граница между небом и белым облаком не должна проходить внутри символьных блоков, через которые проходят самолёты. Во время полёта самолёт исчезает за облаком, так что некоторые блоки должны иметь фон белого цвета и белый INK цвет. Когда самолёт пересекает этот блок, рисующий цвет INK становится тоже белого цвета, что и фон, и самолёт невозможно отличить от облака.

2. Хотя, на первый взгляд, кажется, что солнце имеет форму круга с гладкими краями, при более внимательном рассмотрении видно, что края зубчатые, проходящие по границам блоков. Те блоки, которые во время игры не меняются, могут содержать два цвета.

Каскадная мультипликация

После загрузки изображения игрового поля на ленту мы перейдём к созданию программ, помещающих и передвигающих объекты на переднем плане. Так как от этих программ требуется высокая скорость исполнения, то мы должны позаботиться об эффективности их алгоритмов с тем, чтобы минимизировать количество вычислений во время игры.

Здесь мы приводим описание одного такого метода - каскадного мультиплицирования, или метод подвижной входной точки. К подпрограммам, быстродействие которых особенно важно, можно обращаться с помощью переменного указателя (идентификатора в нижнем регистре). Эти подпрограммы оформлены в виде последовательности (каскада) подзадач, из которых при каждом входе выполняется только одна. Решение каждой подзадачи записывается в виде последовательности строк, после чего указатель обращения к этой программе изменяется и мы возвращаемся в исходную программу. При повторном входе в подпрограмму обычно выполняются действия по другому алгоритму, так как указатель, как правило, указывает на начало следующей в списке подзадачи, и так процесс повторяется до тех пор, пока мы не достигаем конца списка (в этом случае, как правило, указатель устанавливается на начало списка).

Вся игра состоит из фазы инициализации и последующих обращений к подпрограммам, каждая из которых представляет из себя каскад, выполняющий ту или иную задачу (например, передвигает самолёт, производит бомбежку цели, и т.д.).

Каскадные подпрограммы могут обращаться и к другим каскадам, что производит эффект действий, происходящих одновременно. Это делается для того, чтобы игроку казалось, что на экране происходит ряд независимых событий. В нашей игре мы стараемся создать у играющего впечатление, что самолёт, орудие и снаряд движутся независимо друг от друга.

В силу своей сложности, наша программа не может служить наглядным примером применения этой техники, поэтому мы приведём другую программу, значительно более простую (которая пригодна для компьютеров с объёмом памяти 16К).

Следующие две программы действуют независимо друг от друга. Программа на листинге 14.2 после нажатия клавиши стреляет точкой, а программа на листинге 14.3 передвигает по экрану по зигзагообразной траектории знак "+". Обе программы используют технику быстрой мультипликации, примененной в игре ISLAND DEFENCE, однако они настолько малы, что для замедления движения на экране приходится пользоваться командой PAUSE (такие проблемы возникают только с очень простыми и короткими программами). Из этих подпрограмм мы составляем две каскадные программы (листинг 14.4), так что кажется, что точка и крестик движутся одновременно.

Каскад, передвигающий точку, состоит из трёх частей:

А. Точка помещается в 10-ую строку, нулевой столбец;

Б. Если игрок нажал клавишу, то точка начинает свое движение;

В. Точка двигается на одну позицию, пока не достигнет правого края экрана.

Как только какой-либо из этих процессов завершен, то значение идентификатора DOT устанавливается на следующий раздел. После того, как точка достигает границы экрана, указатель DOT устанавливается на первый раздел.

Каскад, управляющий движением крестика, также состоит из трех частей:

А. Крестик помещается в 22-ой строке, 8-ом столбце;

Б. В нижней половине экрана крестик перемещается на одну строку вверх и один столбец вправо;

В. В верхней половине экрана крестик перемещается на одну строку вверх и один столбец влево, и если точка и крестик находятся в одном блоке, то выполняется команда SPLAT.

При помощи основной программы из этих программ составляется игра. Основная подпрограмма вызывает попеременно то программу "DOT", то "CROSS", а значения идентификаторов изменяются так, что они указывают на правильную точку входа.

LISTING 14.2

```
200 REM DOT
210 PRINT AT 11,0;"."
220 IF INKEY$="" THEN GO TO 220
```

```
230 LET D=0
240 PRINT AT 11,D;" ": LET D=D+1
250 IF D=32 THEN GO TO 210
260 PRINT AT 11,D;"."
270 PAUSE 1: GO TO 240
```

LISTING 14.3

```
300 REM CROSS
310 LET R=21: LET C=8
320 PRINT AT R,C;"+"
330 PRINT AT R,C;" ": LET R=R-1
340 IF R<0 THEN GO TO 310
350 IF R>11 THEN LET C=C+1
360 IF R<=11 THEN LET C=C-1
370 PRINT AT R,C;"+"
380 PAUSE 1: GO TO 330
```

LISTING 14.4

```
100 REM MAIN LOOP
110 LET DOT=220: LET CROSS=300
120 GO SUB DOT: GO SUB CROSS
130 GO TO 120
200 REM DOT CASCADE
210 PRINT AT 11,0;".": LET DOT=220: RETURN
220 IF INKEY$="" THEN RETURN
230 LET D=0: LET DOT=240: RETURN
240 PRINT AT 11,D;" ": LET D=D+1
250 IF D=32 THEN LET DOT=210: RETURN
260 PRINT AT 11,D;"."
270 RETURN
300 REM CROSS CASCADE
310 LET R=21: LET C=8
320 PRINT AT R,C;"+": LET CROSS=330: RETURN
330 PRINT AT R,C;" ": LET R=R-1
340 LET C=C+1
350 PRINT AT R,C;"+"
360 IF R>11 THEN RETURN
370 LET CROSS=388: RETURN
380 IF R=11 AND C=D THEN PRINT AT R,C;"SPLAT": STOP
390 PRINT AT R,C;" ": LET R=R-1
400 IF R<0 THEN LET CROSS=310: RETURN
410 LET C=C-1
420 PRINT AT R,C;"+"
430 RETURN
```

Упражнение 14.2

Добавьте к вызывающей программе одну строчку, чтобы после выполнения команды SPLAT указатели устанавливались на начало каскада, и на экране появлялся счёт (количество попаданий).

Напишите программу "стрельба по летящей мишени", в которой охотник, движением которого можно управлять с клавиатуры, перемещается слева направо. Он стреляет по мишени, которая может летать слева направо, сверху вниз по диагонали.

Другие методы мультипликации

На примере такой простой игры мы не можем показать насколько эффективны каскадные программы. Это становится ощутимым для больших игр, таких, как ISLAND DEFENCE. Заметьте, что для создания изображения самолёта используется восемь наборов символов.

Каскад, рисующий изображение самолёта, состоит из различных секций, каждая из которых помещает один тип самолёта в определенное место на экране. Перемещение самолёта происходит в два этапа: сначала он рисуется невидимыми чернилами на новом месте, а затем забивается старое изображение. Снаряды наносятся невидимыми чернилами поверх уже существующего изображения, а затем стираются со старой позиции повторным выводом. У этих методов есть свои достоинства и недостатки. Когда есть возможность одновременно удалять изображение в старом положении и рисовать объект в новом положении (см. раздел программы, в котором самолёт летит горизонтально), то зачастую на создание изображения тратится меньше времени; но при этом исчезают некоторые детали фона.

При одновременном использовании двух техник могут возникать дополнительные сложности: если некоторый символьный блок пересекут два движущихся объекта: например, что произойдет, если занятый снарядом блок пересечет самолёт? В игре это может произойти в седьмой строке, девятом ряду. В результате после повторной печати участок ракеты, подлежащий уничтожению, остается висеть в воздухе. Для исправления этой ошибки лучше дописать программу самым простым способом, чем усложнять алгоритм, так как это может привести к появлению других ошибок. Каскад, управляющий полётом самолёта, содержит лишний оператор (на 1990 строке), который

обеспечивает быструю ликвидацию ошибок.

В двух местах в программе самолёт исчезает за предметами фона. Для того, чтобы самолёт мог исчезнуть за облаком, мы просто сделали в тех блоках, где он должен исчезнуть, рисующий цвет белым. Что касается человека, проходящего за деревом, то здесь проблема сложнее. Мы хотим видеть дерево двухцветным, когда за ним проходит человек.

Используемый нами приём состоит в следующем: человек подходит к дереву, и надпечатывается, если при этом попадает в блоки, содержащие листву, а затем и вовсе перестает печататься, но звук шагов продолжает воспроизводиться, и через соответствующий интервал времени человек выходит из-за дерева: сначала появляется нога человека, на один блок раньше головы, причём это должно происходить как при движении человека вправо, так и при движении влево. Все детали следует тщательно рассчитать перед тем, как программировать алгоритм на бейсике.

Комбинация этих техник для перемещения объектов по экрану и приёмы, позволяющие этим объектам проходить друг через друга, позволит вам создавать высококачественные программы. Если вы хотите создавать сложные игры с быстроменяющейся обстановкой, то вам придётся программировать в машинных кодах. Даже в этом случае некоторые из подпрограмм могут быть написаны на бейсике. Подпрограммы пишутся целиком в машинном коде, только когда они предназначены для продажи.

По мере того, как программируемые вами игры будут более сложными, вам придется прикладывать все больше усилий для того, чтобы они имели простую структуру, для этого программируйте методом сверху-вниз, используйте осмысленные имена для переменных и комментируйте программы, начиная со стадии разработки.

Теперь мы оставляем вас наедине со SPECTRUMом, который показал себя надежным устройством, простым в общении. Мы уверены, что у вас впереди много приятных часов, проведенных с этим компьютером. Для того чтобы помочь вам начать, мы приводим в следующей главе несколько идей проектов. Всего вам хорошего и хороших программ.

Список программ

1. Листинг 14.1: игра ISLAND DEFENCE. Мы рекомендуем вам приобрести кассету, так как для создания набора специальных символов требуется довольно много времени. Однако рано или поздно вам придется создавать наборы символов и фон.
2. Листинг 14.2 (программа "DOT"). Нажмите любую клавишу.
3. Листинг 14.3 (программа "CROSS"): входных данных нет.
4. Листинг 14.4 (основная программа, "DOT CASCADE" и "CROSS CASCADE"). Впечатайте любой ключ.

ГЛАВА 15
Проекты

1. С помощью SPECTRUMa создайте цифровые часы. Для этой цели используйте встроенный в SPECTRUM датчик времени.

2. Напишите программу, которая бы проверяла квалификацию радистов. Входной информацией служит кусок текста, который программа переводит в последовательность точек и тире в соответствии с азбукой Морзе. Кодировка текста может сопровождаться звуковыми эквивалентами точек и тире. Должна быть предусмотрена возможность регулировки скорости поступления закодированной информации.

3. Нарисуйте набор дорожных знаков. Ваша программа должна рисовать фон (например, красные треугольники), после этого используйте свои собственные программы, или программы для вычерчивания переднего плана глав 5 и 6.

4. Постройте на экране поле для кроссворда. Каждый квадрат кроссворда имеет сторону в две единицы. Эти четыре блока могут быть либо чёрными (в этом случае они не содержат информации), либо белыми; в левом нижнем углу которых содержится буква решения, а верхние два блока - номер ключа (если таковой имеется). При таком размере блока на экране умещается головоломка размером 16 блоков на 11 блоков.

Так как на экране должны быть размещены и ключи к решениям, то места на экране будет явно недостаточно. Решить проблему можно с помощью метода "показа слайдов", описанного в главе 13. В первом кадре поместим поле кроссворда, а в остальных ключи. Ответы могут вставляться в кроссворд с помощью курсора или специальных команд. Например, буква "В" (по вертикали), или буква "Г" (по горизонтали), после которых идёт номер ключа, сопровождаемый вашим решением. Если ограничиться только небольшими кроссвордами, то можно использовать эти команды для вынесения соответствующих ключей в поле кроссворда.

5. Напишите свои собственные программы, использующие генератор символов и подпрограммы для генерации диаграмм (главы 5 и 6) для вычерчивания флагов каких-либо стран. Используйте технику главы 13, делающую доступным содержимое ячеек дисплейного файла, с тем, чтобы добавить новую опцию к программам генерации диаграмм. Эта опция позволяет скопировать группу символов (находящуюся на экране и определенную с помощью курсора) в другую группу символов (также находящуюся на экране и определенную с помощью курсора). Должна быть предусмотрена возможность вращать или зеркально отражать эту группу.

6. Напишите программу, которая бы рисовала ноты, которые воспроизводятся командой BEEP. Для этого вам нужно начертить нотный стан и создать специальные символы, изображающие нотные знаки. Применявшийся в старых бальных залах приём "скачущего мячика" для отбивания такта можно применять и здесь.

7. Используйте блочную графику для создания лабиринтов. Естественно, что программа должна создавать лабиринты, для которых имеется правильное решение. Ограничивайте поиск решения лимитом времени. Генерируйте лабиринты, меняющиеся в ходу игры. Добавляйте дополнительные сложности, вроде рыскающих по лабиринту людоедов, неожиданно открывающихся люков, переносов в другие участки лабиринта, если вы движетесь по нему слишком медленно.

8. Расширьте набор методов, предложенных в главе 7. Нарисуйте гистограммы, графики и круговые диаграммы придуманного вами типа. Сделайте их подвижными (либо методом, использованным в программе "MOVIE", либо методом, использованным в программе "WORM GAME", главы 1 или его модификацией в главе 14).

Генерируйте множества специальных символов. Создавайте трёхмерные гистограммы, окрашивая каждый столбик тремя цветами. Высота передней части каждого столбика должна выражаться в целых числах и иметь ширину в два блока. Ширина боковой стенки столбца - один блок, боковая стенка оканчивается треугольной шапочкой, заполняющей правую нижнюю половину символьного блока. Третья часть столбца лежит поверх первой и имеет форму ромба, и касается двух первых частей. Такое строение столбика гарантирует, что в одном блоке используется не более двух цветов, а, кроме того, это изображение похоже на ортогональную проекцию прямоугольного столбца.

9. Создание орнаментов. Используйте команду OVER для нанесения большого числа случайных линий на экран, или нарисуйте большое число прямых, расположенных регулярным образом. Например, проведите прямые, соединяющие точки (0,I) с точками (255,175-I), где 0<=I<=175 и точки (0,I) с точками (255-I,175), где 0<=I<=255. Обобщите метод программы из главы 5, которая генерирует узоры, для создания сложных симметричных узоров - любой учебник кристаллографии может натолкнуть вас на хорошие идеи.

10. В книгах по кристаллографии имеется много интересных трёхмерных объектов. Перейдём в пространство четырёх измерений - вершины теперь - это вектора с четырьмя координатами, для преобразования которых требуется матрицы 5x5. При ортогональном проецировании мы просто отбрасываем значение двух координат, что является трансляцией, масштабированием и поворотом в четырёхмерном пространстве.

11. Имеется множество игр, где SPECTRUM можно использовать в качестве игровой доски, или в качестве судьи, или в качестве противника.

12. Используйте специальные символы для построения колоды карт. Эта программа может быть включена в программу игры в карты, где SPECTRUM выполняет роль банкомета или противника.

13. При помощи SPECTRUMa можно создавать головоломки на экране дисплея. Допустим, например, что имеются девять квадратов, каждый из которых поделен диагоналями на четыре части. Каждая четверть окрашена в свой цвет (голубой - "1", красный - "2", жёлтый - "3", зелёный - "4") и имеет свой знак ("+" или "-").

Каждый квадрат задается последовательностью четырёх чисел, обозначающих вершины, обходимые против часовой стрелки, например, (-1,-2,1,4), (-1,3,4,2), (1,-4,-2,3), (1,2,-3,-2), (2,-3,-4,3), (1,4,-3,-2) и два раза (-1,-4,3,2).

Проблема заключается в том, чтобы составить из квадратов квадрат 3x3, так что если два маленьких квадрата касаются друг друга, то они должны быть одного цвета, но с противоположными знаками. С помощью SPECTRUMa

вы можете поместить эти квадраты в левой части экрана, а в правой части экрана поместить сетку с ячейками величиной с малый квадрат. Квадратики можно вставлять в ячейки, либо переносить обратно на левую половину.

Напишите программу, решающую эту головоломку. Время выполнения этой программы около 10 минут, и она должна находить два различных решения.

14. Составьте пакет программ графики среднего разрешения, оперирующей с блоками две четверти на две четверти. Этот пакет должен быть аналогичен пакету главы 5. Экран состоит из 64 блоков по горизонтали и 44 блоков по вертикали.

Для каждого символьного блока выясните, какая окраска в нем преобладает - светлая или тёмная, и окрасьте блок соответствующим образом. На первый взгляд это кажется довольно трудоёмкой операцией, однако если учесть, что большинство блоков в изображении служат частью светлого фона, то используют черный рисующий цвет и белый фон, окажется, что для большинства блоков никаких изменений вносить не надо.

15. Напишите видеоигру типа PACMAN. Для этого потребуется создать на экране пять движущихся объектов. Для того чтобы программа создавалась быстрее, ограничьте число движущихся за один шаг привидений двумя.

В режиме нападения привидения выбирают кратчайшую дорогу к игроку, а в режиме бегства - оптимальный путь для бегства. Из-за сложной формы игрового поля найти этот оптимальный путь не так-то просто и в качестве простого приближения можно принять, что привидение движется по направлению к (или от игрока) игроку, если между ними нет стены. Найдите эффектный алгоритм движения привидений, так как эта часть программы занимает большую часть машинного времени.

Хорошие программы видеоигр отличает высокая скорость выполнения. Для перемещения привидений по экрану можно использовать машинный код.

16. Напишите программу, которая бы высвечивала в левой половине экрана меню, состоящее из специальных символов вроде стилизованных изображений элементов электронных цепей (конденсаторы, сопротивления, и т.д.). Элементы с помощью курсора можно помещать в любое место экрана. Кроме того, должна быть предусмотрена возможность проводить соединительные линии, делать надписи (например, для обозначения "ом") и сдвигать или удалять символы. Дополнительные опции могут позволять заносить изображение в память и на ленту, а также удалять меню из построенной схемы.

17. В наших программах мы предполагали, что все объекты расположены перед наблюдателем. Переделайте программы с учётом того, что объекты могут располагаться за наблюдателем.

ПРИЛОЖЕНИЕ А
SPECTRUM с объёмом памяти 16К

Более 7 К памяти отводится под дисплейный файл и системные переменные, что оставляет около 8.8 К памяти для программ и данных. Многие из программ, приведенных в этой книге, занимают больше места в памяти, однако все же большинство программ пригодны и для SPECTRUMa с объёмом памяти 16К, если соблюдать следующие правила.

Перед загрузкой программ из них следует удалить все комментарии. Используйте рекомендации главы 13 для оптимизации кода программы. Например, программа, генерирующая изображение самолёта, умещается в памяти, если удалены подпрограммы "SCALE", "PLOT" и все комментарии.

Возможно, что четыре набора символов и набор графических символов, определяемых пользователем, занимают слишком много места в памяти. При наличии места таблица адресов содержит значения 15360, 29271, 30039, 30807, 31575 и 32080. Возможно, что наборы с большими номерами мешают размещению в памяти программ или данных, а иногда даже приводят к ошибкам в работе компьютера. Оператор CLEAR 62294, находящийся в начале программ, использующих альтернативные на боры символов, следует заменить на CLEAR 255+адрес начала наиболее удаленного из наборов.

Вы можете убедиться, что программе CHARACTER GENERATOR не хватает места для второго набора символов, и в этом случае оператор есть CLEAR 255+30039. Если ваша программа использует второй набор, то запишите его на ленту и загружайте при необходимости.

Вы можете разбить программы на независимые куски и записывать результаты работы этих программ на ленту. Эти файлы перезагружаются в память вместе с другими программами; рассмотрим, например, программу построения диаграмм из главы 6. Вам следует загрузить оператором LOAD подпрограмму "LIBDIAG" и загрузить оператором MERGE одну из подпрограмм для построения гистограмм, круговых диаграмм или редактирующих подпрограмм. При запуске программ следует помнить, что они не должны вызывать подпрограммы, которые не загружены в память. Для записи изображения на экране на ленту используются команды SAVE и LOAD с опцией SCREENS, а для записи массивов используется опция DATA. После этого вы можете загрузить редактирующие программы для завершения оформления диаграмм.

К сожалению, некоторые программы не помещаются в памяти, несмотря ни на какие ухищрения. К таким программам относятся программы удаления невидимых линий нетривиальных объектов, программа мультиплицирования и программа игры ISLAND DEFENCE.

Если вы собираетесь серьёзно заниматься машинной графикой, то мы настоятельно рекомендуем вам приобрести дополнительную память.

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ 5
ВВЕДЕНИЕ 6
ГЛАВА 1 Графические команды ZX SPECTRUM 8
Команды PAPER и INK ("бумага" и "чернила", фон и тон) 8
Дисплейный файл (отобразить на экране содержимое файла) 8
Команда BORDER (граница, рамка экрана) 8
Символьные блоки 9
Команды PLOT и DRAW ("нанести точку" и "начертить линию") 9
Команды PRINT и LIST 9
Команды INVERSE и OVER 10
Команды FLASH и BRIGHT (мерцание и яркость) 12
Атрибуты 12
Простейшее движущееся изображение 14
Команды CIRCLE и DRAW (окружность и чертить) 15
Цвета в символьном блоке 15
Простая игра 16
ГЛАВА 2 От реальных координат к пикселям 19
Примитивы, обеспечивающие карту пространства на графическом экране 19
Список программ 27
ГЛАВА 3 Геометрия декартовой плоскости 28
Кадрирование 31
Задание кривых с помощью функциональной зависимости 34
Список программ 36
ГЛАВА 4 Матричное представление преобразований двумерного пространства 37
Композиция преобразований 38
Перенос 39
Растяжение 39
Вращение относительно начала координат 39
Обратное преобразование 40
Размещение объекта 40
Рассматривание объекта 40
Вычерчивание объекта 41
Сложные изображения – метод построения изображения "по кирпичикам" 41
Эффективное использование матриц 44
Хранение информации об изображении 46
Список программ 48
ГЛАВА 5 Символьная графика ZX SPECTRUM 49
Стандартный набор символов 49
Графический режим работы 50
Блочная графика 50
Графика, определяемая пользователем 51
Альтернативные множества символов 52
Список программ 55
ГЛАВА 6 Диаграммы и графики 56
Курсоры 56
Атрибуты 57
Точки и линии 57
Запись и загрузка 57
Надписи 58
Данные для графиков 58
Специальные символы 59
Определенные пользователем символы для использования в гистограммах 60
Гистограммы 60
Круговые диаграммы 63
Штриховка секторов 63
Графики 65
Список программ 67
ГЛАВА 7 Трёхмерная декартова геометрия 69
Определение прямой 69
Угол между двумя направляющими векторами 70
Определение плоскости 70
Точка пересечения прямой и плоскости 71
Расстояние от точки до прямой 71

Точка пересечения двух прямых....71
Плоскость, проходящая через три неколлинеарные точки....73
Точка пересечения трёх плоскостей....73
Линия пересечения двух плоскостей....75
Функциональное представление поверхности....76
Лежит ли точка по ту же сторону от плоскости, что и начало координат?....77
Каково направление обхода двумерного многоугольника, заданного последовательностью своих вершин?....77
Список программ....78
ГЛАВА 8 Матричное представление преобразования трехмерного пространства....79
Параллельный перенос....79
Растяжение....80
Вращение относительно координатной оси....80
(А) Вращение на угол TETA относительно оси X....81
(Б) Вращение на угол TETA относительно оси Y....81
(В) Вращение на угол TETA относительно оси Z....81
Обратное преобразование....82
Поворот на угол NU относительно произвольной оси P+MU*Q....82
Список программ....85
ГЛАВА 9 Ортогональные проекции....86
(1) Исходное положение....86
(2) Фактическое положение....86
(3) Наблюдаемое положение....86
Ортогональная проекция....86
Вычисление матрицы Q, переводящей объект из фактического положения в наблюдаемое....87
Как задавать объект....89
Конструирующие подпрограммы и метод построения блоков....89
Тела вращения....93
Список программ....95
ГЛАВА 10 Простые алгоритмы удаления невидимых линий и поверхностей....97
Ориентация треугольника в трехмерном пространстве....97
Алгоритм удаления скрытых поверхностей для случая тела, ограниченного замкнутой и выпуклой поверхностью....98
Тела вращения....99
Вычерчивание поверхности специального вида, лежащей в трехмерном пространстве....100
Список программ....102
ГЛАВА 11 Перспективная проекция....103
Что такое перспективное видение?....103
Построение перспективной проекции точки....103
Свойства перспективной проекции....103
Выбор визуальной плоскости....104
Кадрирование изображения....105
Список программ....105
ГЛАВА 12 Общий алгоритм удаления невидимых линий....106
Список программ....113
ГЛАВА 13 Белее сложные элементы программирования....114
Управление выполнением программ....114
Структура дисплейного файла....115
Быстрое преобразование содержимого экрана....116
Мультипликация (только для машин с объемом памяти 48К)....117
Свертка (16К и 48К)....118
Структура программ на бейсике (перенумерация и уничтожение)....119
Структура программы на бейсике (эффективные программы)....122
Синхронное изображение....122
Список программ....123
ГЛАВА 14 Действующий пример видеоигры....124
Описание игры....124
Передний план....124
Фон....129
Каскадная мультипликация....130
Другие методы мультипликации....131
Список программ....132
ГЛАВА 15 Проекты....133
ПРИЛОЖЕНИЕ А SPECTRUM с объёмом памяти 16К....135

Книга представляет собой пособие по разработке программ на языке Бейсик, предназначенных для обработки графической информации и воспроизведения изображения на экране дисплея и печатающем устройстве персональных компьютеров ZX SPECTRUM.

Для широкого круга пользователей персональных компьютеров ZX SPECTRUM и совместимых с ними.

**Ваши отзывы о данном издании,
предложения о сотрудничестве
просим присылать по адресу:**

127362, Москва, а/я 26

Графика ZX SPECTRUM

Ответственный за выпуск: *Васятко В.Н.*

Технический редактор: *Шелепова Е.Н.*
Панушкина О.В.
Константинова С.С.

Компьютерная верстка: *Тарасов С.Ю.*
Мельниченко А.Н.

Лицензия ЛР № 06721 от 23 октября 1992г.
Подписано в печать 19.01.94. Формат 60x88 1/16.
Бумага офсетная №1. Печать офсетная. Печ. л. 12,5.
Тираж 20000 экз. Зак. 6821.

ТОО "ВА ПРИНТ". 127322, Москва, ул. Яблочкова,
д. 37, к. В, кв. 210.

Отпечатано с оригинал-макета в филиале
ГМП "Первая Образцовая типография".
113114, Москва, Шлюзовая наб., 10.