



А. Капальцевич, И. Капальцевич, А. Евдокимов

## Как написать игру для ZX Spectrum

*издание 2-е, исправленное*

*Серия «Все о ZX Spectrum»*

*Книга седьмая*



«ПИТЕР»

Санкт-Петербург

1995

Главный редактор	В. Усманов
Художественный редактор	Н. Вавулин
Художники	А. Андрейчук, М. Кирпа
Корректор	П. Чекер
Оригинал-макет подготовил	А. Денисов

Второе издание этой популярнейшей книги выпускается по многочисленным просьбам читателей. Она представляет собой увлекательный учебник для тех, кто имеет компьютер ZX Spectrum и хочет научиться самостоятельно писать для него игровые программы.

Читатель научится составлять игровые программы на языке Бейсик, рисовать заставку к программе и делать многоуровневое меню, узнает, как создать и заставить двигаться по экрану спрайты, как ввести в программу звуковые эффекты. Он сможет использовать для создания программ диалект Laser Basic и процедуры в кодах из пакетов Supercode и New Supercode, генератор игр Games Designer.

Книга рассчитана в основном на начинающих программистов, но будет полезна всем, кто всерьез интересуется этой темой.

© Издательство «Питер» (Piter Ltd.), 1994  
ISBN 5-7190-0052-6  
Издательство «Питер»,  
194044, С.-Петербург, Выборгская наб., 17.  
Лицензия ЛР №060557 от 17.01.92.  
Подписано к печати 10.01.95. Формат 84x108 1/32.  
Усл. п. л. 14,9. Тираж 20 000. Заказ № 842.  
Отпечатано в типографии им. Володарского.  
191023, С.-Петербург, наб. р. Фонтанки, 57.

Scan ,OCR & spellcheck by NUK

## ВВЕДЕНИЕ

Обладатели домашнего компьютера ZX Spectrum, число которых стремительно растет, потратив массу времени на различные игры и накопив солидную библиотеку программ, начинают в конце концов задавать себе вопрос: «А нельзя ли самому научиться писать нечто подобное?» Если и вы всерьез заинтересовались этой проблемой, мы попробуем вам помочь.

Будем считать, что вы знакомы с клавиатурой ZX Spectrum, умеете записывать программы на магнитную ленту (или дискету) и загружать их в память компьютера, а самое главное - горите желанием создавать что-то свое, новое, крутое и неповторимое.

Прежде чем начать писать игровые программы, необходимо научиться хорошо представлять себе две вещи: как строятся такие программы, то есть из каких основных частей они состоят, и как они реализуются на языке программирования Бейсик.

О Бейсике написано огромное количество разнообразных книг и статей. Для первого знакомства с «идеологией» этого языка, его основными операторами и функциями, можно воспользоваться любой из них, но лучше сразу обратиться к книге «Диалекты Бейсика для ZX Spectrum» [2].

Приступая к написанию компьютерных программ на Бейсике, нужно всегда иметь в виду, что работать они будут медленно. Поэтому для первой пробы сил в программировании больше подойдут спокойные игры, похожие на шашки, нарды, морской бой и т. п. Позже, накопив опыт и знания, вы сможете ускорить движение объектов на экране, применяя различные программы-компиляторы типа Tobos, Softek или MCoder 2. Отличные результаты дает также один из диалектов Бейсика - Laser Basic, ориентированный на работу с графическими изображениями.

Ну, а тот, кто рвется писать игровые программы на профессиональном уровне, но еще не владеет языком ассемблера, узнает, как использовать при создании игр готовые процедуры в кодах из пакетов Supercode и New Supercode.

Для написания даже не очень сложных игровых программ нужны какие-то знания. Надеемся, что вы уже разобрались с такими понятиями, как «оператор», «функция», «программная строка» и т. п., научились пользоваться копировщиками и умеете переключать режимы курсора клавиатуры ZX Spectrum. Если нет - советуем внимательно ознакомиться с инструкцией по эксплуатации компьютера или попросить друзей ввести вас в курс дела.

Книга составлена таким образом, чтобы те, кто до этого не имел ни малейшего представления о Бейсике, смогли научиться программировать на нем. Но даже если вы что-то знаете об этом языке, не спешите сразу смотреть на последние страницы: ведь в первых главах описаны простенькие программы, которые используются в дальнейшем в качестве элементов более сложных программ.

Теперь несколько слов о структуре книги.

В первой главе рассматриваются блоки, из которых строятся игровые программы: заставка, игровое пространство, блок взаимодействия с играющим и блок оценки игровой ситуации. Дается краткая характеристика каждой части с примерами и рисунками.

Из второй главы вы узнаете, как сделать простенькую заставку к игре, пользуясь языком Бейсик, «защитым» в памяти-ZX Spectrum (Spectrum-Бейсик). На примере нескольких программ вы начнете знакомиться с операторами и функциями Бейсика. Каждая строка приведенных программ подробно

комментируется. В этой главе также описано, как воспользоваться графическим редактором Art Studio, чтобы нарисовать картинку вполне профессионального уровня.

В третьей главе мы расскажем, как построить игровое пространство, то есть создать те подвижные и неподвижные объекты, которые вы видите на экране во время игры. Особое внимание уделено подвижным объектам (в программировании они называются «спрайта»), показано, как можно нарисовать их различными способами.

В следующей, четвертой главе речь идет о приемах программирования, делающих игровую программу более совершенной. Даны рекомендации по составлению блок-схемы игры, без которой построить сложную программу почти невозможно. Показано, как сделать многоуровневую заставку, включающую в себя меню, и как ввести в программу разнообразное звуковое сопровождение.

Пятая глава посвящена взаимодействию игрока и программы. Она рассказывает о том, как можно управлять событиями, происходящими на экране - заставить самолет повернуть при соответствующем наклоне ручки джойстика, вызвать на экране взрыв при попадании снаряда в цель и т. п.

В шестой главе мы обсудим вопрос о том, как оценивать действия играющего во время игры: начислять ему очки, следить за количеством оставшегося топлива или снарядов, числом уничтоженных «врагов».

Седьмая глава касается самого сокровенного - операционной системы компьютера. Вы убедитесь в неисчерпаемости возможностей Spessу и ограниченности средств Бейсика. Возможно, эта глава побудит вас глубже разобраться в программировании и освоить для начала другие разновидности Бейсика, а затем и другие языки.

С одним из диалектов Бейсика - Laser Basic, дающим в руки программиста мощные средства для работы с графикой, - вы сможете познакомиться в следующей, восьмой главе. Этот язык дополняет стандартный Spectrum-Бейсик сотней новых операторов и функций, с помощью которых можно выполнять самые сложные действия со спрайтами: перемещать их по экрану с большой скоростью во всех направлениях, инвертировать, поворачивать и т. д. С помощью Laser Basic мы сделаем мультипликационную картинку и напишем игру БЕГА МЫШЕЙ.

Девятая глава перекидывает мостик от Бейсик-программ к программам в машинных кодах. В описанных примерах используются кодовые подпрограммы, взятые из специальных пакетов процедур Supercode и New Supercode. С помощью этих кодовых блоков генерируются всевозможные звуковые сигналы, мгновенно выводятся на экран изображения больших размеров, быстро перемещаются спрайты, в общем создается то, без чего нельзя представить хорошую игровую программу.

Впрочем, можно почувствовать себя настоящим программистом, даже не имея ни малейшего представления о бейсиках и ассемблерах. Генератор игр Games Designer, описанный в последней, десятой главе, позволяет собрать игру из составных частей, как из кубиков, - пусть всего лишь примитивную «стрелялку», но зато в ней можно заказать на свой вкус все: от цвета экрана до сценария игры. Вы можете задавать число, внешний вид и даже «степень агрессивности» противников, количество очков, получаемое за их уничтожение, менять звуковое оформление игры, траекторию и скорость движения объектов и т. д. и т. п.

В Приложении 1 приводятся листинги нескольких игровых программ. Здесь же описываются правила игры, а также даются краткие пояснения к листингам программ на тот случай, если вам захочется изучить их подробнее, а может быть,

внести ряд собственных дополнений и улучшений.

Приложения 2, 3 и 4 носят чисто справочный характер и содержат краткие сведения об операторах и функциях Бейсика (Приложение 2), графическом редакторе Art Studio (Приложение 3) и музыкальном редакторе Wham (Приложение 4).

Творите, осваивайте и ждите появления следующей книги серии «Как написать игру для ZX Spectrum», в которой мы расскажем о применении языка ассемблера для создания компьютерных игр.

Все программы этой книги написаны и отлажены И. Капутьевичем. Главы 7 и 10, посвященные системным переменным и программе Games Designer, написаны А. Евдокимовым.

## 1. СТРУКТУРА ИГРОВОЙ ПРОГРАММЫ

Перед каждым, кто впервые приступает к созданию игровой программы, встает непростая проблема: с чего же начать? Чтобы облегчить ее решение, попробуем условно разделить программу на несколько частей, каждую из которых можно было бы разрабатывать и отлаживать независимо от других. Если повнимательнее присмотреться к хорошим компьютерным играм, то таких частей можно выделить четыре (см. [6]):

- заставка;
- игровое пространство;
- блок взаимодействия с играющим;
- блок оценки игровой ситуации.

Давайте поподробнее познакомимся с каждой из этих частей и для начала в самых общих чертах решим, как их создавать.

### ***Заставка***

С этой частью программы мы сталкиваемся каждый раз, когда загружаем фирменные игры. Поскольку процесс загрузки с магнитофона занимает обычно несколько минут, то разработчики программ придумали первым делом загружать на экран красочную картинку, рекламирующую предстоящую игру (рис. 1.1).

Таким образом, пока идет загрузка, вы любуетесь заставкой и готовите себя к очередному компьютерному приключению. Надо сказать, что картинка на экране - это еще не вся заставка, а лишь первая ее часть, и в дальнейшем мы будем называть ее картинкой-заставкой, или просто картинкой.



Рис. 1.1. Заставка игры R-TYPE.

После того как игровая программа загрузилась, на экране последовательно появляются остальные части заставки, информирующие игрока о некоторых деталях предстоящей игры

Посмотрим, какие данные выводятся в заставке и как они распределяются между ее отдельными частями, которые мы будем называть кадрами.

На первой, обычно самой красочной картинке, сопровождающей загрузку, помещают название игры и фирмы-изготовителя, а также имена авторов программы (здесь вы можете увековечить свое имя, когда напишете хорошую игру).



Рис. 1.2. Меню программы NATE.

Это не только «визитная карточка» программы, но еще и неплохая реклама.

Далее в большинстве игр в качестве одного из кадров заставки выводится меню - список действий, которые необходимо выполнить для настройки игры. Пример меню приведен на рис. 1.2.

В меню вы можете выбрать способ управления игрой: один из типов джойстика (KEMPSTON, SINCLAIR и др.) или клавиатуру (KEYBOARD). Здесь же можно «заказать» удобные для игрока клавиши управления (REDEFINE KEYS), просмотреть таблицу рекордов (HI-SCORE), иногда - выбрать уровень игры (LEVEL). Обычно из меню осуществляется запуск игры (START GAME). Ввод имен играющих (ENTER YOUR NAME), переназначение клавиш и просмотр таблицы рекордов очень часто выполняются, как отдельные кадры. По окончании игры вы снова возвращаетесь в меню.

Обычно все части заставки красиво оформлены, а их демонстрация чаще всего сопровождается звуковыми эффектами или музыкой.

### ***Игровое пространство***

Игровое пространство - это совокупность всех подвижных и неподвижных объектов, которые появляются на экране во время игры.

В распространенной игре Рас-ман (упаковщик) игровое пространство представляет собой лабиринт с прожорливыми «колобками», одним из которых управляете вы, а несколько «врагов» подчиняются программе и пытаются причинить вам массу неприятностей. В программе Chess (Шахматы) игровое пространство - шахматная доска с расположенными на ней фигурами, которые в соответствии с вашими указаниями могут передвигаться по полю, не нарушая при этом шахматных правил. Наконец, в игре Tetris игровое пространство представляет собой стакан с падающими разноцветными геометрическими фигурками, которые вы должны плотно укладывать в нижней его части (рис. 1.3).



Рис. 1.3. Игровое пространство программы TETRIS.

Используя эти примеры, попробуйте теперь сами определить, из чего состоит

игровое пространство в знакомых вам играх.

Посмотрим на игровое пространство с другой стороны, а именно со стороны разработчика программ. Начнем с того, что любое сложное изображение на экране можно представить в виде некоторого количества небольших по размерам, но вполне законченных объектов. Часть из этих объектов передвигается, а основная масса составляет фоновый рисунок (пейзаж), на котором развивается действие игры.

Все ползающие, летающие, стреляющие и прочиедвигающиеся изображения в компьютерных играх носят название спрайты (от английского слова «sprite» - эльф). Более точно спрайт можно определить как перемещаемый графический объект с неизменными рисунком и размером (см. [2]). Трудно себе представить хорошую игру без таких «объектов», поэтому, чтобы научиться писать приличные игровые программы, нужно прежде всего освоить технику создания спрайтов.

Информация о спрайтах, представленная в виде блоков кодов, обычно хранится в спрайт-файле, который загружается в память компьютера вместе с игровой программой.

Теперь вы знаете, что шахматные фигуры в игре Chess - это спрайты, и геометрические фигурки в Tetris'e - тоже; даже снаряд, вылетевший из ствола пушки в программе Tanx - это тоже спрайт. Есть игры, которые буквально напичканы спрайтами, отчего игровое пространство, как правило, становится сложнее и богаче, а игра - интереснее. Например, в программе River Raid можно насчитать семь спрайтов-противников, шныряющих у вас под самым носом, один спрайт-самолет, принадлежащий вам, а также спрайты-снаряды, летящие со всех сторон.

О том, как делать спрайты, мы подробно расскажем дальше, здесь же только перечислим способы их создания. Самый простой из них состоит в том, чтобы нарисовать спрайт на бумаге, а затем вручную закодировать его изображение. Другой способ - создать изображение спрайта в любом графическом редакторе (например, Art Studio или The Artist II), а затем автоматически закодировать его с помощью специальной программы - генератора спрайтов. Некоторые генераторы спрайтов (например, из пакета программ Laser Basic) позволяют сразу и рисовать и кодировать спрайты, то есть включают в себя графический редактор (правда, довольно примитивный).

При помощи графических редакторов можно создавать не только спрайты, но и картинки-заставки, а также рисовать пейзажи (которые, впрочем, тоже можно назвать спрайтами). В нашей книге приводятся примеры использования для перечисленных целей редактора Art Studio.

В игровом пространстве возможны любые, самые фантастические действия: объекты на экране могут менять цвета, возникать и исчезать, двигаться с невероятной скоростью, превращаться в другие объекты и т. д. и т. п. Чтобы ваша игра была привлекательной и интересной, не жалейте времени на проработку всех объектов игрового пространства.

### ***Блок взаимодействия с играющим***

Пожалуй, одно из самых привлекательных свойств компьютерных игр заключается в том, что играющий может влиять на события, развивающиеся на экране - в игровом пространстве. Например, в программе Pac-man вы можете менять направление движения вашего «колобка» по лабиринту: вверх, вниз, вправо и влево. А в игре Flying Shark, помимо управления движением самолета, пролетающего над объектами противника, можно еще стрелять по целям, нажимая

кнопку джойстика Огонь, и сбрасывать бомбу, наклоняя джойстик на себя и одновременно нажимая Огонь. Есть игры с еще более сложным управлением.

Все эти перемещения по экрану и стрельба возможны благодаря той части игровой программы, которая носит название блок взаимодействия с играющим. Попробуем в общих чертах представить, как он работает.

Как известно, в ZX Spectrum существует два способа управления компьютером: с помощью клавиатуры и с помощью джойстика. Когда вы нажимаете клавишу или наклоняете ручку джойстика, программа переходит к выполнению той или иной своей части. Поворачивая, например, ручку джойстика вправо, вы «включаете» ту часть программы, которая перемещает спрайт-самолет вправо. Нажимаете кнопку Огонь - и начинает работать та часть программы, которая выводит на экран спрайт-ракету и перемещает ее в заданном направлении.

### ***Блок оценки игровой ситуации***

Во время игры на экране появляются не только предметы и персонажи, задействованные в сюжете, но и выводится различная дополнительная информация для играющего. Так, в Tetris'e на экран выводятся набранные играющим очки. В программе Into The Eagles Nest сообщается о количестве патронов, ключей от помещений, числе ранений и числе набранных очков. В River Raid программа информирует о количестве оставшегося топлива в самолете, числе жизней и набранных очках.

Несмотря на большое разнообразие игр, есть нечто общее, что объединяет выводимую на экран информацию: она представляет собой оценку игровой ситуации. Следовательно, в программе должна быть часть, которая оценивает действия играющего на всех этапах игры и выводит на экран результаты этой оценки. В большинстве игр ведется подсчет числа очков, что позволяет легко выяснить, как со временем растет мастерство играющего или сравнить силу нескольких игроков.

Мы попытались выделять наиболее важные части игровой программы для того, чтобы лучше понять, как она работает. А это, в свою очередь, поможет более четко и осмысленно подходить к созданию собственных игровых программ. В следующих главах мы подробно поговорим обо всех перечисленных частях программы, то есть от вопроса о том, что делает каждая из них, перейдем к вопросу о том, как сделать эти части и соединить их между собой.

## 2. ЗАСТАВКА

Заставок разного рода можно придумать великое множество. Все зависит от вашего опыта и фантазии, а также от того, черно-белый или цветной монитор (телевизор) имеется в вашем распоряжении. Можно выделить заставки, состоящие из одного кадра и заставки с двумя и более кадрами. В последнем случае возможна как автоматическая их смена по мере выполнения программы, так и смена после нажатия какой-либо клавиши.

Прежде чем начать конкретный разговор, необходимо сказать несколько слов о строении экрана, формируемого вашим Спрессу (рис. 2.1). Вы, конечно, уже заметили, что не вся площадь экрана используется для вывода различных надписей и картинок. Особенно хорошо это заметно во время загрузки программ с магнитофона. Картинка появляется во внутренней области, называемой рабочим экраном (или просто экраном), а внешнее поле, по которому во время загрузки бегут разноцветные полосы, именуется бордюром. С бордюром дела обстоят достаточно просто: единственное, что можно с ним делать - это менять его цвет (полосы получаются при очень быстрой смене цветов). Поэтому основное внимание уделим рабочему экрану.

Если ваш монитор позволяет получить достаточно резкое изображение, то вы можете заметить, что оно строится из отдельных точек, больше похожих на маленькие квадратики. Эти точки программисты называют пикселями. Символы, которые вы вводите с клавиатуры, вписываются в стандартные по размерам и положению площадки - знакоместа, причем размер каждого знакоместа - 8x8 пикселей. При разбивке рабочего экрана на знакоместа получается 24 строки и 32 столбца. Но в основном при написании программ на Бейсике используются только 22 верхние строки, называемые основным экраном. Оставшиеся две нижние строки имеют особое назначение: они используются для набора и редактирования строк программы, а иногда в них появляются различные сообщения. Эти две строки носят название служебное окно.

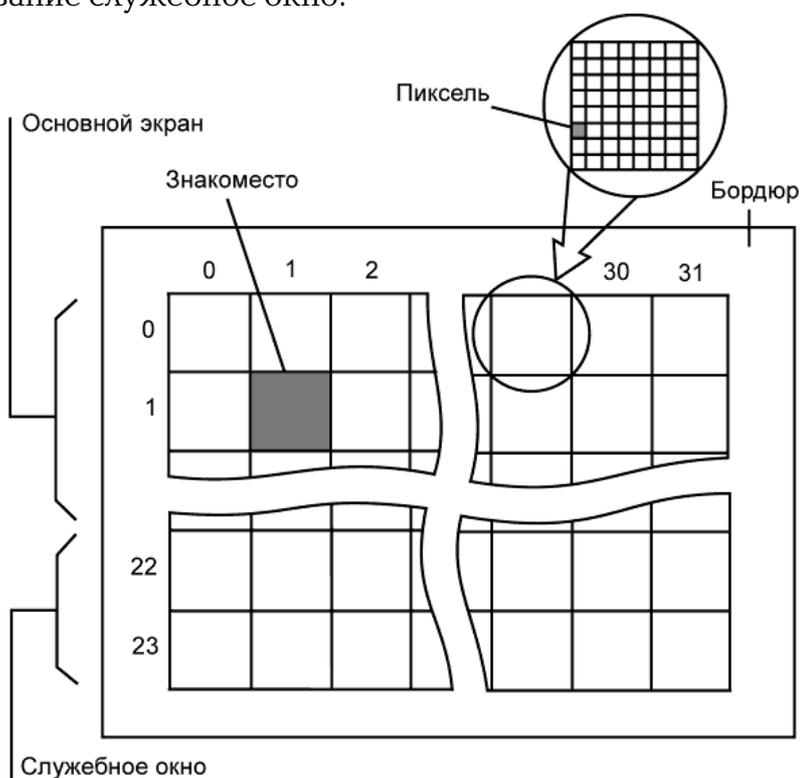


Рис. 2.1. Структура экрана ZX Spectrum

Вернемся теперь к нашим баранам и рассмотрим вначале простейшие заставки.

### **Простейшие заставки**

Прежде чем включать компьютер и начинать писать программу, есть смысл взять лист клетчатой бумаги, очертить на нем поле размером 32 x 24 клетки и попытаться изобразить на этом поле все то, что вам хочется увидеть на экране, вписывая буквы в клетки. Тем самым вы сэкономите массу времени, так как взаимное расположение слов и букв, размеры рамок и других элементов заставки с листа легко перевести на язык цифр, а затем и в строки программы.

Пусть предполагаемая заставка имеет вид, показанный на рис. 2.2.

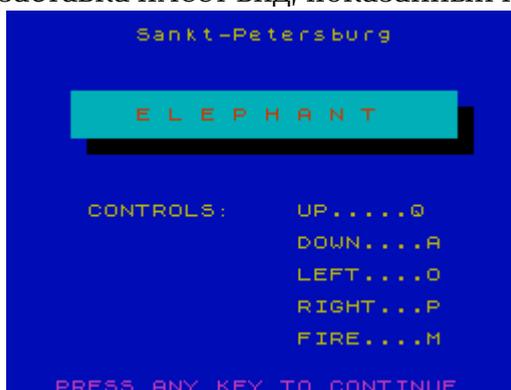


Рис. 2.2. Пример заставки 1.

Для воплощения этой картинки на экране нужно записать координаты (номера строк Y и номера столбцов X) первых букв слов или предложений. Например, слово SANKT-PETERSBURG должно быть напечатано на заставке в первой строке, начиная с восьмой позиции по горизонтали, а слово CONTROLS - в 12-й строке, начиная с 5-й позиции по горизонтали. Прделав эту работу до конца, можно выбрать цвет каждой надписи, цвет фона экрана и цвет бордюра (будем считать, что вы работаете с цветным монитором).

Теперь можно переходить к написанию программы, которая выводит на экран задуманное изображение. Программу желательно писать таким образом, чтобы строки шли в той же последовательности, что и строки текста заставки на экране. Такой способ построения позволяет при возникновении какой-либо ошибки легче ее обнаружить и исправить. Начнем работу с того, что попытаемся перечислить те действия, которые должен выполнить компьютер для осуществления нашей задумки.

1. Окрасить бордюр в синий цвет, установить синий фон экрана и желтый цвет надписей на нем.
2. Напечатать в 1-й строке экрана, начиная с 8-й позиции: SANKT-PETERSBURG.
3. Напечатать 24 пробела в 5-й строке экрана, начиная с позиции 4, закрасив их голубым цветом.
4. Напечатать в строке 6 с 4-й позиции красными буквами на голубом фоне:  
E L E P H A N T  
и один пробел черным цветом для создания эффекта тени.
5. В строке 7, начиная с позиции 4, напечатать 24 пробела голубым цветом и один - черным.
6. Вывести нижнюю границу тени: 24 пробела черным цветом в 8й строке, начиная с 5-й позиции.
7. Напечатать в строке 12 с позиции 5:  
CONTROLS: UP...Q
8. В строке 14 с 18-й позиции напечатать:

DOWN...A

9. В строке 16 с 18-й позиции -

LEFT...O

10. В строке 18 с 18-й позиции -

RIGHT...P

11. В строке 20 с 18-й позиции -

FIRE...M

12. Напечатать в служебном окне (в самой нижней строке экрана) фиолетовым цветом фразу

PRESS ANY KEY TO CONTINUE

(нажмите любую клавишу для продолжения),

отступив от левого края экрана три знакоместа.

13. Дождаться нажатия любой клавиши для перехода от заставки к следующей части игровой программы.

14. Очистить экран.

В принципе, это уже и есть программа, остается только перевести ее на язык Бейсик, то есть заменить слова соответствующими операторами. И если вы знакомы с английским языком, то задача не покажется вам сложной. В любом случае, по мере изучения операторов Бейсика полезно составлять таблицу, кратко поясняя для себя действие каждого оператора. Такую справочную таблицу мы поместили в Приложении 2.

Наша программа начинается предложением «окрасить бордюр в синий цвет» Бордюр по-английски - BORDER, а цвет, как вы узнаете позже, гораздо удобнее задавать числом, нежели определять словом. Приведем таблицу кодировки цветов, принятую в ZX Spectrum:

0 - черный,	4 - зеленый,
1 - синий,	5 - голубой,
2 - красный,	6 - желтый,
3 - фиолетовый,	7 - белый.

Таким образом, синий бордюр можно получить, введя в программу оператор

BORDER 1

Поэкспериментируйте с этим оператором, вводя его с клавиатуры с различными кодами цвета и нажимая клавишу Enter. Посмотрите, как будет меняться цвет бордюра.

Цвет фона (или «бумаги») определяется цветом «выключенных» пикселей, а «включенные» пиксели задают цвет тона - «чернил». Предложение «установить синий фон экрана» переводится на Бейсик как

PAPER 1

а «установить желтый цвет надписей» - как

INK 6

Однако нужно помнить, что эти операторы, в отличие от BORDER, не вызывают немедленного изменения цвета экрана, они лишь задают цвета или иначе - атрибуты) для дальнейших действий. Изменение цвета станет заметно, например, после очистки экрана. Команда «очистить экран» на английский язык переводится как CLEAR SCREEN, а Бейсик понимает это предложение в сокращенном виде: CLS.

Итак, введем первую строку нашей программы, закончив ввод нажатием клавиши Enter. Строка будет выглядеть так:

10 BORDER 1: PAPER 1: INK 6: CLS

Обратите внимание на то, что в каждой строке Бейсика может быть несколько операторов, но в этом случае они должны отделяться друг от друга двоеточием. Выбор номеров строк не имеет принципиального значения, но лучше начинать с





```

10 BORDER 1: PAPER 1: INK 6: CLS
20 PRINT AT 1,8;"Sankt-Petersburg"
30 PRINT AT 5,4; PAPER 5;"XXXXXXXXXXXXXXXXXXXXXXXXXXXX"
40 PRINT AT 6,4; PAPER 5; INK 2;
"PPPE L E P H A N TPPPP"; PAPER 0;" "
50 PRINT AT 7,4; PAPER 5; "XXXXXXXXXXXXXXXXXXXXXXXXXXXX";
PAPER 0;" "
60 PRINT AT 8,5; PAPER 0;"XXXXXXXXXXXXXXXXXXXXXXXXXXXX"
70 PRINT AT 12,5; "CONTROLS:PPPPUP....Q"
80 PRINT AT 14,18;"DOWN...A"
90 PRINT AT 16,18;"LEFT...O"
100 PRINT AT 18,18;"RIGHT...P"
110 PRINT AT 20,18;"FIRE...M"
120 PRINT #0; INK 3;"PPPPRESS ANY KEY TO CONTINUE"
130 PAUSE 0
140 CLS

```

Запускается программа оператором RUN, который вводится непосредственно с клавиатуры, после чего нажимается клавиша Enter.

И вот работа над заставкой закончена. Вы смотрите на экран, не веря своим глазам: набор строк вдруг превратился в яркое красочное изображение. Если же в эту программу удастся внести что-то свое, ценность ее еще больше возрастет, и вам наверняка захочется записать ее на ленту. Наберите следующую строку:

```
SAVE "ELEPHANT"
```

На экране появится надпись

```
Start tape then press any key
```

напоминающая о том, что нужно включить магнитофон на запись и нажать любую клавишу.

После записи программы ее можно загрузить, как и любую другую, введя с клавиатуры оператор

```
LOAD ""
```

или

```
LOAD "ELEPHANT"
```

Если загрузка пройдет нормально, компьютер скажет

```
0:OK
```

но программа не запустится: нужно выполнить еще оператор RUN.

Впрочем, программу можно записать на ленту и таким способом, что после загрузки она автоматически запустится (этот способ называется «запись с автостартом»). Для этого нужно выполнять оператор SAVE с ключевым словом LINE:

```
SAVE "ELEPHANT" LINE 10
```

Тогда после загрузки программа автоматически запустится со строки 10 и выведет на экран заставку.

Прежде чем перейти к следующей заставке, давайте отвлекуемся от практики и немного поштудироваем теорию.

К наиболее важным средствам программирования относятся так называемые циклы. Циклы могут быть организованы по-разному, но чаще всего для этого пользуются конструкциями типа FOR...NEXT. Попробуйте, например, набрать и выполнить такую программку:

```

10 FOR N=1 TO 5 STEP 1
20 PRINT "ZX Spectrum"
30 NEXT N

```

Компьютер напечатает на экране слово «ZX Spectrum» пять раз.

Циклы применяются в игровых программах очень часто, например, для перемещения объектов, плавного изменения различных характеристик (скажем,

цвета), создания звуковых эффектов и т. д. Переменная, следующая за оператором FOR (в приведенном примере - переменная N), называется управляющей, а запись N=1 TO 5 STEP 1 указывает, в каких пределах эта переменная будет изменяться (запись 1 TO 5 означает, что переменная изменяется от 1 до 5 включительно) и с каким приращением, или иначе - шагом (STEP 1). Правда, в данном случае шаг можно было бы и не указывать, так как по умолчанию он принимается равным единице. Далее выполняются все операторы до оператора NEXT, который изменяет управляющую переменную на величину шага, и если она не превысила конечного значения (в нашем случае - 5), программа вновь переходит к выполнению оператора, следующего за FOR.

В приведенном примере переменная N указывает лишь на количество повторений, но ее можно использовать и внутри цикла:

```
10 FOR N=0 TO 21
20 PRINT AT N,0;"Line: ";N
30 NEXT N
```

Здесь управляющая переменная изменяет позицию печати при выводе на экран. Кроме того, этот пример демонстрирует способность оператора PRINT печатать на экране не только слова, но и числа: в начале каждой строки на экране появится слово Line: с номером этой строки.

Переменные вообще широко используются в программировании для описания любых изменяющихся величин (например, для определения координат перемещающихся по экрану объектов или подсчета очков, заработанных игроком), поэтому необходимо научиться правильно обращаться с ними. Прежде всего, переменную нужно определить, то есть присвоить ей какое-то значение. Словами это можно выразить так: «Пусть переменная A равна двум». В переводе на язык Бейсик это будет выглядеть как

```
LET A=2
```

Изменить значение переменной можно с помощью того же оператора LET: LET A=A+7

Это означает, что к переменной A (которая равна 2) прибавляется число 7, а полученный результат (число 9) присваивается той же переменной A. Убедиться в этом можно, выполнив оператор

```
PRINT A
```

На экране появится число 9.

В Бейсике используются не только числовые, но и символьные переменные, с помощью которых можно обрабатывать любые последовательности символов - строки. Внешне символьная переменная отличается от числовой значком \$, стоящим после ее имени, например, A\$. Значение символьным переменным присваивается также с помощью оператора LET:

```
LET A$="String"
```

В этом примере переменной A\$ присваивается значение «String». Это можно проверить, выполнив

```
PRINT A$
```

На экране появится надпись String

В Бейсике вы можете пользоваться не только отдельными переменными, но и целыми списками изменяемых величин. Такие списки называются массивами.

Предположим, в вашей игрушке участвуют десять персонажей. Для описания координат всех десяти объектов вам понадобилось бы завести двадцать переменных (по две переменные на объект): X1, Y1, X2, Y2, ...X10, Y10. Понятно, что это не слишком удобно. Гораздо проще завести два списка по десять элементов в каждом: X(10) и Y(10), а затем выбирать требуемое значение, задавая только индекс

(порядковый номер объекта). Но предварительно нужно объявить массив, иначе говоря, указать компьютеру размеры списка. Делает это оператор DIM:

```
DIM X(10): DIM Y(10)
```

Далее при помощи все того же оператора LET элементам массива присваиваются необходимые значения. Однако если писать

```
LET X(1)=10: LET Y(1)=18 LET X(2)=8: LET Y(2)=20
LET X(10)=3: LET Y(10)=3
```

то удобства, о которых только что говорилось, будут весьма сомнительны. Избежать длинной цепочки присваиваний позволяют операторы DATA и READ.

Сначала в операторе DATA через запятую перечисляются значения для присваивания (данные), а затем они считываются из памяти оператором READ. В этом случае можно использовать цикл:

```
10 DIM X(10): DIM Y(10)
20 FOR N=1 TO 10
30 READ X: READ Y
40 LET X(N)=X: LET Y(N)=Y
50 NEXT N
60 DATA 10, 18, 8, 20, 11, 7, 5, 12, 3, 0, 2, 9, 7, 1, 6, 15, 9, 5, 3, 3
```

В строке 30 считывается пара значений X и Y из списка, приведенного в строке 60 после оператора DATA, а затем в строке 40 эти значения присваиваются N-м элементам массивов X(10) и Y(10).

Можно поступить еще проще, заменив строки 30 и 40 на одну строку:

```
30 READ X(N): READ Y(N)
```

или даже

```
30 READ X(N), Y(N)
```

Нужно заметить, что строки с оператором DATA могут располагаться в любом месте программы. Этот оператор не выполняется (как и оператор REM) и служит лишь для хранения данных.

После заполнения массива его элементы можно использовать в программе. Для примера выведем все значения массива на экран в два столбца, добавив к предыдущей программе строки:

```
100 FOR N=1 TO 10
110 PRINT "X(";N;") = ";X(N), "Y(";N;") = ";Y(N)
120 NEXT N
```

Обратите внимание на запятую в строке 110. Если в операторе PRINT встречается запятая, то текущая позиция печати перемещается на сере-

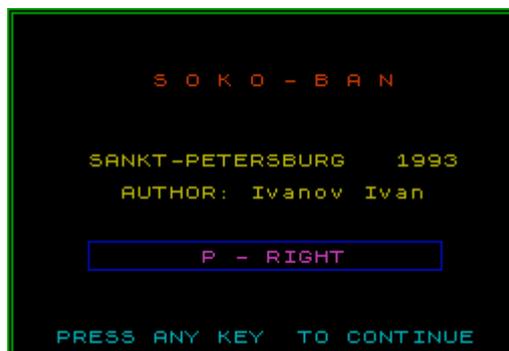


Рис. 2.3. Пример заставки 2.

дину экрана или в начало следующей строки, в зависимости от того, в первой или во второй половине экрана был выведен последний символ.

Говоря о массивах, можно еще добавить, во-первых, что они, так же как и переменные, могут быть не только числовыми, но и символьными, а во-вторых, - что они могут иметь несколько измерений. Примером двухмерного массива может

служить поле экрана, если мы зададим символьный массив DIM S\$(22,32), который описывает 22 строки по 32 символа в каждой.

Теперь, немного подучив теорию, закрепим ее практикой на примере создания еще одной, более сложной заставки. По ходу дела мы познакомимся еще с несколькими операторами Бейсика и новыми приемами программирования.

Предположим, что на экран необходимо вывести изображение, которое вы видите на рис. 2.3. Главное отличие этой заставки от предыдущей заключается в том, что информация о назначении клавиш последовательно появляется в специальной рамке. Время, отводимое на вывод сообщения о каждой клавише, ограничено, и мы видим постоянно сменяющиеся друг друга надписи:

```
Q - UP
A - DOWN
O - LEFT
P - RIGHT
M - FIRE
E - END
```

Такой прием удобен в тех случаях, когда большая часть заставки занята надписями и картинками, а количество клавиш управления, о которых необходимо сообщить, велико.

Начнем с того, что окрасим экран в черный цвет (фон) и выберем для надписей зеленые «чернила» (тон), как мы это делали в предыдущей программе:

```
10 BORDER 0: PAPER 0: INK 4: CLS
```

Затем обведем основной экран двойной рамкой, как показано на рисунке. Для изображения линий в Бейсике служит оператор DRAW (по-английски «чертить»). Попробуйте, например, выполнить оператор

```
DRAW 50,100
```

От нижнего левого угла экрана протянется линия, причем ее длина по горизонтали составит 50 пикселей, а по вертикали - 100. Теперь очистите экран командой CLS или просто нажав Enter и выполните строку:

```
DRAW 50,100: DRAW 50, 100
```

На экране появится рисунок, похожий на букву Л. Следовательно, оператор DRAW вычерчивает линию от последней поставленной точки. Поэтому прежде, чем рисовать первую линию на экране, нужно поставить точку, от которой протянется отрезок прямой. Точку ставит оператор PLOT, после которого через запятую указываются координаты экрана: сначала по горизонтали, затем по вертикали. Последующие линии можно рисовать, как показано в предыдущем примере, от конца уже проведенной.

Итак, началом координат для операторов PLOT и DRAW служит нижний левый угол основного экрана (две нижние строки служебного окна при графических построениях не используются). Точки и линии не привязываются к знакам и могут рисоваться в любом месте, поэтому измерение здесь ведется в пикселях.

В строке

```
20 PLOT 0,0: DRAW 0,175: DRAW 255,0: DRAW 0,-175: DRAW -255,0
```

сначала ставится начальная точка в левый нижний угол основного экрана (PLOT 0,0), затем от этой точки проводится вертикальная линия вверх до верхней границы экрана (DRAW 0,175), потом от конца предыдущей линии чертится горизонтальный отрезок до правой границы экрана (DRAW 255,0), а операторы DRAW 0,-175 и DRAW -255,0 замыкают рамку. Аналогично рисуется и внутренняя рамка.

```
30 PLOT 2,2: DRAW 0,171: DRAW 251,0: DRAW 0,-171: DRAW -251,0
```

После этого внутри рамки делаем надписи в соответствии с рисунком:

```
40 PRINT AT 4,9; INK 2;"S O K O - B A N"
50 INK 6: PRINT AT 9,5;"SANKT-PETERSBURG1993"
60 PRINT AT 11,7;"AUTHOR: Ivanov Ivan"
```

```
70 PRINT AT 20,3; INK 5;"PRESS ANY KEY TO CONTINUE"
```

Далее устанавливаем синие «чернила» и вычерчиваем маленькую рамку, в которой будут появляться надписи о назначении клавиш:

```
80 INK 1
```

```
90 PLOT 40,45: DRAW 0,14: DRAW 175,0: DRAW 0,-14: DRAW -175,0
```

Теперь начинается самое интересное. Зададим двухмерный символьный массив, состоящий из шести строк (по количеству выводимых надписей) по девять символов в каждой (максимальная длина надписей), и в цикле заполним его из списка DATA:

```
100 DIM P$(6,9) 110 FOR N=1 TO 6 120 READ P$(N) 130 NEXT N
```

```
110 DATA " Q - UP", "A - DOWN", "O - LEFT", "P - RIGHT", "M - FIRE", " E - END"
```

Затем начнем последовательно выводить в маленькой рамке надписи из созданного массива. Нам нужно сделать шесть надписей, поэтому организуем цикл, повторяющийся шесть раз, в котором управляющая переменная изменяется от 1 до 6, чтобы можно было последовательно обращаться к строкам массива с первой по шестую:

```
150 FOR N=1 TO 6
```

```
250 NEXT N
```

Что же у нас будет твориться внутри этого цикла? Можно поступить по-разному, но мы сделаем так: каждую надпись будем выводить новым цветом, и не сразу, а по одной букве. Причем и цвет, и очередную букву выберем случайным образом. Для получения случайного числа в Бейсике используется специальная функция RND. Она генерирует случайные числа в диапазоне от нуля до единицы (но всегда меньше 1). Попробуйте в качестве опыта выполнить такую строку:

```
FOR N=1 TO 20: PRINT RND: NEXT N
```

На экране появится столбик чисел, изменяющихся без видимой закономерности. Если вы еще раз выполните приведенную строку, числа окажутся совершенно другими. Чтобы получить случайное число из заданного диапазона, воспользуемся следующей формулой:

$$R = \min + \text{RND} * (\max - \min)$$

где min - нижняя, а max - верхняя граница диапазона. Например, случайное число от 5 до 12 можно получить с помощью строки:

```
LET R=5+RND*7
```

Однако функция RND далеко не всегда дает целые числа, а в нашей задаче числа должны быть именно целыми. Округлить полученное число до ближайшего меньшего целого поможет другая функция Бейсика - INT. В случае положительных чисел INT отбрасывает дробную часть числа, оставляя целую.

Теперь применим все вышесказанное к нашей заставке. Будем печатать надписи цветом от красного (код 2) до белого (код 7). Случайное число от 2 до 7 получится в результате вычисления такого выражения:

```
2+RND*(7-2)
```

Но нам еще нужно это число округлить. Если отбросить целую часть, то числа 7 мы никогда не получим, так как результат всегда будет хоть чуть-чуть, но меньше семи. Поэтому нужно написать так:

```
INT (2+RND*(8-2))
```

А применительно к цвету «чернил»:

```
160 INK INT (2+RND*6)
```

Эта строка, кроме всего прочего, демонстрирует преимущества числового, а не «словесного» задания цвета.

Покончив с цветом, перейдем теперь к надписям. Чтобы они появлялись постепенно, по одной букве, напишем еще один цикл и «вложим» его в уже описанный. Так как строки массива состоят из девяти символов, повторим цикл

девять раз:

```
170 FOR M=1 TO 9 210 NEXT M
```

Внутри этого цикла сначала определим порядковый номер очередной выводимой буквы и запомним его в переменной A:

```
180 LET A=INT (1+RND*9)
```

Вся надпись у нас должна располагаться в строке 15 экрана, начиная с 12-й позиции. Следовательно, буква с номером A должна быть напечатана в позиции 11+A:

```
200 PRINT AT 15,11+A; P$(N,A)
```

Запись P\$(N,A) указывает на букву с номером A из строки N (управляющая переменная внешнего цикла) массива P\$(6,9).

Поскольку число A случайное, то не обязательно все буквы строки P\$(N) будут напечатаны. Чтобы в конце концов фраза оказалась полной, после завершения внутреннего цикла на экран выводятся все символы строки:

```
220 PRINT AT 15,12; P$(N)
```

Теперь нужно остановить на время выполнение программы, чтобы успеть прочитать появившийся текст. Строка

```
230 PAUSE 100
```

даст задержку в две секунды (100:50 = 2). После этого сотрем надпись, напечатав поверх нее 9 пробелов:

```
240 PRINT AT 15,12; "ppppppppp"
```

После окончания цикла его нужно повторять снова и снова до тех пор, пока не будет нажата какая-нибудь клавиша. Перейти на начало цикла очень просто. Для этого допишем строку:

```
260 GO TO 150
```

которая переводится на русский язык как «перейти к строке 150», то есть к строке, с которой начинается цикл печати надписей в маленькой рамке.

Но в таком виде программа никогда не закончит свою работу. Чтобы иметь возможность выхода из нее при нажатии любой клавиши, вставим во внутренний цикл еще одну строку:

```
190 IF INKEY$<>"" THEN GO TO 300
```

которая означает: «если символ нажатой клавиши не равен пустой строке (то есть нажата любая клавиша), то перейти к строке 300». Более подробно оператор IF...THEN, называемый условным, а также функцию INKEY\$ мы рассмотрим позже, а пока ограничимся сказанным. Если же вам уже и так все понятно - тем лучше, значит вас как программиста ждет блестящее будущее. Теперь приведем «собранный» текст полученной программы:

## Программа 2. ЗАСТАВКА 2.

```
10 BORDER 0: PAPER 0: INK 4: CLS
20 PLOT 0,0: DRAW 0,175: DRAW 255,0: DRAW 0,-175: DRAW -255,0
30 PLOT 2,2: DRAW 0,171: DRAW 251,0: DRAW 0,-171: DRAW -251,0
40 PRINT AT 4,9; INK 2;"S O K O - B A N"
50 INK 6: PRINT AT 9,5;"SANKT-PETERSBURGpppp1993"
60 PRINT AT 11,7;"AUTHOR: Ivanov Ivan"
70 PRINT AT 20,3; INK 5;"PRESS ANY KEYpppTO CONTINUE"
80 INK 1
90 PLOT 40,45: DRAW 0,14: DRAW 175,0: DRAW 0,-14: DRAW -175,0
100 DIM P$(6,9)
110 FOR N=1 TO 6
120 READ P$(N)
130 NEXT N
140 DATA " Q - UP", "A - DOWN", "O - LEFT", "P - RIGHT", "M - FIRE", " E - END"
150 FOR N=1 TO 6
160 LET C=INT (2+RND*6)
```

```

170 FOR M=1 TO 9
180 LET A=INT (1+RND*8)
190 IF INKEY$<>" " THEN GO TO 300
200 PRINT AT 15,11+A; INK C;P$(N,A)
210 NEXT M
220 PRINT AT 15,12; INK C;P$(N)
230 PAUSE 100
240 PRINT AT 15,12;"ппппппппп"
250 NEXT N
260 GO TO 150
300 CLS

```

### **Бегущая строка**

Для придания заставке большей привлекательности используется ряд приемов, о которых должен знать программист.

Представим себе, что на экране слева направо «бежит» строка, содержащая информацию об игре (название, назначение клавиш, автор и т. п.), как это сделано, например, в программе R-TYPE.

Эффект бегущей строки можно получить, например, таким образом. Первым делом создадим символьную переменную, содержащую тот текст, который мы хотим видеть в «бегущей» строке. Затем последовательно будем выводить часть текста длиной 32 символа в одно и то же место экрана, например, в служебное окно. Первый фрагмент должен содержать символы с 1-го по 32-й, следующий - со 2-го по 33-й и так далее, пока не закончится текст. В результате каждая буква будет как бы перемещаться справа налево.

Ниже приведен листинг программы, реализующей этот принцип.

#### Программа 3. БЕГУЩАЯ СТРОКА.

```

10 BORDER 1: PAPER 1: CLS
20 REM --SCREEN---
100 LET B$="пппппппппппппппппппппппппппппппппппппп"
110 LET A$=B$+"PROGRAM: B.STROKA...пппп*** DEMO ***"
120 LET A$=A$+"ппппUP...Q, DOWN...A, LEFT...O, "
130 LET A$=A$+"RIGHT...P, FIRE...M, END...E"
140 LET A$=A$+"ппппппAUTHOR: Petrow Iwanпппп1993"
150 LET A$=A$+"ппппппппппPPRESS ANY KEY TO CONTINUE"
160 LET A$=A$+B$
200 FOR N=1 TO LEN A$-31
210 PAUSE 7: IF INKEY$<>" " THEN GO TO 300
220 LET B$=A$(N TO N+31)
230 PRINT #0;AT 1,0; INK 6;B$
240 NEXT N
250 GO TO 200
300 REM --CONTINUE---

```

Поскольку вы уже знакомы практически со всеми операторами этой программы, здесь и в дальнейшем мы будем использовать принцип построчного описания, рассматривая не отдельные операторы, а целые строки или даже группы строк программы. Таким образом, если вам понятны функции тех или иных строк, можете их пропускать и переходить к следующим.

10 - установка атрибутов экрана;

20 - в этом месте могут располагаться строки, рисующие на экране рамки, печатающие название игры и т. д.;

100...160 - в этих строках формируется символьная переменная, содержащая текст, который будет выведен впоследствии на экран. Обратите внимание, что итоговая фраза здесь получена сложением строк

символов. Сложение - это единственное «арифметическое» действие, применяемое для обработки символьных переменных. Здесь мы воспользовались этой возможностью исключительно для того, чтобы не писать длинную строку программы. Кроме того, мы сэкономили память компьютера, дважды используя переменную В\$, состоящую из 32 пробелов. Кстати, эти 32 пробела в начале и в конце текста нужны для того, чтобы первые буквы постепенно появлялись из-за правого края экрана, а последние целиком уходили за левый край;

200 - начало цикла: переменной N присваиваются начальное и конечное значения. Здесь использована новая функция - LEN (от английского слова Length - длина). Она возвращает длину символьной переменной, указанной в качестве аргумента этой функции. Здесь она введена для того, чтобы компьютер сам выяснял, до каких пор повторять цикл и избавил нас от необходимости пересчитывать это значение после внесения каких-либо изменений в текст. Конечное значение должно быть на 31 меньше длины текста для того, чтобы в конце цикла не выйти за пределы переменной A\$;

210 - вводится небольшая пауза, составляющая 7/50 секунды для того, чтобы строка «бежала» не слишком быстро и ее можно было успеть прочитать. В этой же строке проверяется, не была ли нажата какая-либо клавиша;

220 - основная операция программы. Она носит название сечение, а ее действие состоит в выделении из большой строки некоторой ее части - подстроки. Для выполнения этого после имени строки в скобках пишется ключевое слово TO и указывается, с какого и по какой символ требуется выделить в подстроку. Например, если выполнить оператор

```
PRINT "PRESS ANY KEY" (4 TO 11)
```

то на экране появится текст SS ANY K, то есть оператор выделил символы с четвертого по одиннадцатый (считая пробелы) и напечатал их. Если подстрока выделяется, начиная с первого символа, то можно писать: ( TO 6), а если подстрока выделяется до конца символьной строки, то допустима запись (4 TO );

230 - в начало строки 1 служебного окна (в самую нижнюю строку экрана) желтым цветом выводится подстрока В\$, состоящая из 32 символов (ранее В\$ была выделена из строковой переменной A\$). Так как N - величина переменная, то в каждом цикле на одном и том же месте печатается часть строки, со сдвигом на один символ вправо, что на " экране создает эффект движения всего текста влево.

Как видно, в этой строке программы мы применили директиву AT при выводе информации в служебное окно. В этом случае за начало отсчета принимается левый верхний угол не всего экрана, а служебного окна. Следует учитывать и то, что после очистки экрана позиция печати в служебном окне устанавливается в позицию с координатами (1,0), или относительно основного экрана - в начало строки 23, самой нижней строки экрана;

240 - конец цикла;

250 - оператор безусловного перехода на строку 200 заставляет повторить вывод всего текста с самого начала, и она будет повторяться до тех пор, пока вы не нажмете любую клавишу;

300 - отсюда может начинаться собственно игра.

### ***Печатающий квадрат***

Попробуем ввести в заставку еще один эффект, который может сделать ее привлекательнее: по экрану слева направо строка за строкой передвигается квадратик, за которым появляются буквы и символы какого-либо текста. Такой прием, напоминающий работу пишущей машинки, используется, скажем, в игре

## CHASE H.Q.

Осуществить задуманное можно, например, следующим образом. Зададим с помощью символьной переменной текст, а затем будем его выводить на экран по одной букве, причем впереди символов будем печатать пробел цветом, отличным от общего фона, то есть в виде квадратика-знакоместа. Каждая следующая буква печатается со сдвигом на одну позицию вправо, затирая при этом напечатанный ранее квадратик. Когда строка окажется целиком напечатана, уберем квадратик, поместив в соответствующую позицию пробел цвета фона.

Если понадобится сделать не одну, а несколько надписей таким способом, то придется несколько раз писать фрагменты программы, очень похожие один на другой. Но это только в том случае, если не знать об одном важном средстве структурного программирования - о подпрограммах.

Подпрограмма - это фрагмент основной программы, предназначенный для выполнения конкретных, обычно часто повторяющихся операций. Подпрограмма может быть исполнена в любой удобный момент и сколько угодно раз. После ее выполнения основная программа продолжает работу с прерванного места.

Вызываются подпрограммы оператором GO SUB с номером строки начала этой подпрограммы, а для возврата в основную программу используется оператор RETURN.

Ниже приведен листинг программы «Печатающий квадрат» и следом за ним - комментарии к тем строкам, которые могут оказаться не совсем понятными.

## Программа 4. ПЕЧАТАЮЩИЙ КВАДРАТ.

```

10 BORDER 0: PAPER 0: INK 6
20 CLS
30 LET A$="Sinclair ZX Spectrum"
40 LET line=0: GO SUB 100
50 LET A$="*****"
60 LET line=2: GO SUB 100
70 LET A$="@ 1982 Sinclair Research Ltd."
80 LET line=21: GO SUB 100
90 PAUSE 0
95 GO TO 200
100 LET S=LEN A$
120 FOR M=1 TO S
130 LET B$=A$(M)
140 PRINT AT line,M;B$; PAPER 3;" "
150 PAUSE 1: BEEP 0.002,20
160 NEXT M
170 PRINT AT line,M;" "
180 RETURN
200 CLS

```

30 - присваивание строковой переменной некоторого текстового значения;

40 - в переменной line задается номер строки экрана, куда будет выводиться текст, затем следует обращение к подпрограмме «печатающий квадрат». Оператор GO SUB вызывает подпрограмму, указывая номер строки, с которой она начинается. Оператор RETURN завершает подпрограмму и возвращает управление основной программе, на строку, следующую за GO SUB (при первом обращении - на строку 50);

50, 60 - эти строки работают аналогично строкам 30 и 40. Меняется лишь текст и номер строки экрана, куда этот текст выводится;

70, 80 - снова повторение строк 30 и 40; текст выводится в 21 строку;

90 - программа ждет нажатия любой клавиши;

95 - безусловный переход на строку 200, откуда начинается следующая часть

программы;

100 - начало подпрограммы. Как уже говорилось, в программировании принято выделять некоторые фрагменты программы в подпрограммы, тем более, если предполагается, что в процессе работы программы этот фрагмент будет вызываться несколько раз. В нашем случае в подпрограмму выделяется вывод на экран текстовой переменной. Заметим, что подпрограмма, в свою очередь, может содержать другие подпрограммы.

В строке 100 определяется количество символов (букв) в выводимом тексте и полученное значение присваивается переменной S;

120 - начало цикла, в котором осуществляется побуквенный вывод текста;

130 - выбор из текста одной буквы, имеющей порядковый номер M;

140 - печать выбранной буквы в знакоместо с координатами line и M, а затем печать фиолетовым цветом квадрата, который всегда будет расположен справа от выводимой буквы;

150 - небольшая пауза, за которой следует пока еще неизвестный оператор ВЕЕР. Этот оператор генерирует звуковой сигнал. Основные характеристики любого звука - длительность и высота, поэтому после оператора ВЕЕР должны указываться два параметра, определяющих эти характеристики. Длительность звука (первый параметр) измеряется в секундах, а высота (второе число, стоящее после запятой) - в полутонах. Нота «до» первой октавы передается числом 0. Более низкие звуки

задаются отрицательными числами, а более высокие - положительными. Например, оператор ВЕЕР 0.5,10 заставит компьютер воспроизвести звук, соответствующий ноте си-бемоль первой октавы, длительностью в половину секунды. Добавим к этому, что высота может задаваться не только целыми числами. Использование дробей позволит извлекать звуки любой высоты. Правда, в нашей программе оператор ВЕЕР используется не в музыкальных целях, а только лишь для имитации «стука» клавиш пишущей машинки;

160 - конец цикла;

170 - оператор убирает с экрана изображение последнего квадрата в конце строки с текстом перед переходом на новую строку;

180 - возврат в основную программу на оператор, который стоит сразу же после оператора GO SUB;

200 - очистка экрана перед началом следующей части программы.

### ***Прыгающие буквы***

Если предыдущую программу немного усложнить, то получится еще один эффектный прием, который можно использовать при написании заставки. В нем каждая строка текста получается из букв, как бы «выпрыгивающих» из-за правого края экрана, пока не заполнится строка. Затем все повторяется для второй и последующих строк, причем каждая строка отличается по цвету от предыдущей. После того, как вы хорошо разберетесь в программе, попробуйте и буквы окрасить в разные цвета, причем так, чтобы соседние цвета не повторялись.

```
20 BORDER 0: PAPER 0: CLS
40 LET x=5: LET y=2: LET c=6
50 LET a$="SINCLAIR ZX SPECTRUM"
60 GO SUB 500
70 LET x=4: LET y=3: LET c=3
80 LET a$="_____ "
90 GO SUB 500
100 LET x=7: LET y=6: LET c=4
110 LET a$="COMPUTER LITERACY"
```

```

120 GO SUB 500
130 LET x=5: LET y=8
140 LET a$="through applications"
150 GO SUB 500
400 PAUSE 300: GO TO 600
499 REM подпрограмма
500 LET d=LEN a$
505 INK c
510 FOR n=1 TO d
520 FOR m=30 TO n+x STEP -1
530 PRINT AT y,m;a$(n);" "
540 NEXT m
550 NEXT n
560 RETURN
600 CLS

```

20 - установка цветовых атрибутов экрана;

40 - операторы присваивания, которые задают координаты первой буквы в первой строке текста, а также цвет выводимых букв;

50 - в символьную переменную a\$ помещается текст первой выводимой на экран строки;

60 - обращение к подпрограмме «прыгающие буквы». После ее выполнения оператор RETURN передаст управление на строку 70;

70...90, 100...120, 130...150 - эти строки выполняют те же операции, что и строки 40...60;

400 - все изображение держится на экране  $300/50=6$  секунд, после чего оператор безусловного перехода передает управление в начало следующей части программы - на строку 600;

500 - с этой строки начинается подпрограмма, причем обращение к ней происходит четыре раза - по числу выводимых на экран строк текста. Сама строка 500 определяет количество букв в каждой фразе, и число, полученное в результате действия функции LEN, присваивается переменной d;

505 - установка цвета для очередной строки;

510 - начало внешнего цикла по переменной n, в котором устанавливаются номера букв, «выпрыгивающих» одна за другой на экран;

520 - начало внутреннего цикла по переменной m. Он задает координаты по горизонтали места на экране, куда будет выводиться буква при ее движении справа налево. Так как эта координата должна все время уменьшаться, то в операторе FOR...TO...NEXT необходимо задать отрицательный шаг. Как вы помните, шаг устанавливается ключевым словом STEP - число, стоящее за этим словом, задает величину приращения управляющей переменной m. Таким образом, в результате действия всех операторов строки, координата будет принимать значения 30, 29, 28,... n + X;

530 - вывод на экран буквы, имеющей в строке текста порядковый номер n. Так как m - величина переменная, то буква будет перемещаться справа налево. Пробел в кавычках в конце строки стирает букву на «старом» месте по горизонтали (например, на 28-м) перед тем как поставить ее на «новое» место (на 27-е);

540 - конец цикла по переменной m;

550 - конец цикла по переменной n;

560 - возврат из подпрограммы - переход к оператору, стоящему сразу за оператором вызова подпрограммы GO SUB. В данном случае переход происходит на строку 70, 100, 130 или 400;

600 - очистка экрана перед началом следующей части программы.

**Заставка из нескольких кадров**

В солидных игровых программах заставки обычно делаются из нескольких кадров, поскольку перед началом игры на экране должно появиться большое количество информации и «впихнуть» ее в один кадр практически невозможно.

Рассмотрим пример подобной многокадровой заставки. В ней три кадра: первый выводит на экран название игры (строки 10...470), во втором представлено меню (500...690), а в третьем (700...830) - таблица результатов. Кроме того, показано, как из меню можно попасть еще в три кадра, причем их число может быть увеличено по вашему желанию.

## Программа 6. МНОГОКАДРОВАЯ ЗАСТАВКА.

```

10 REM ** kadr 1 (zastawka) **
20 BORDER 0: PAPER 0: CLS
30 LET x=7: LET c=6
40 LET a$="SANKT - PETERBURG"
50 GO SUB 300
60 FOR n=0 TO 13
70 PRINT AT 13,n;" 19"
80 PRINT AT 13,29-n;"93 "
90 PAUSE 6
100 NEXT n
110 BEEP .005,0
120 INK 3
130 PRINT AT 20,4;"Press any key to continue"
140 LET c1=0
150 LET c=INT (RND*7+1)
160 IF c=c1 THEN GO TO 150
170 INK c
180 PRINT AT 1 0;"*****"
190 PRINT AT 2,0;"*nnnnn*nnnn*nnnn*nnn*nnn*nnn*"
200 PRINT AT 3,0;"*** ** * * * * * ** * * * * * **"
210 PRINT AT 4,0;"*** ** nnn* * * * * * ** * * * * * nnn**"
220 PRINT AT 5,0;"*** ** * * * * * ** nnn* * * * * * **"
230 PRINT AT 6,0;"*** ** * * * * * ** * * * * * ** * * * * * **"
240 PRINT AT 7,0;"*** ** nnnn* * * * * ** * * * * * nnn* * * * * **"
250 PRINT AT 8,0;"*****"
260 LET c1=c
270 PAUSE 10
280 IF INKEY$="" THEN GO TO 150
290 GO TO 500
300 REM --podprogramma 1--
310 LET b$="": LET c$=""
320 LET d=LEN a$
330 FOR n=1 TO d-1 STEP 2
340 LET b$=b$a$(n)+" "
350 LET c$=c$+" "+a$(n+1)
360 NEXT n
370 OVER 1: INK c
380 FOR n=1 TO 10
390 PRINT AT n,x; INK 0;b$
400 PRINT AT n+1,x;b$
410 PRINT AT 22-n,x; INK 0;c$ 420 PRINT AT 21-n,x;c$ 430 PAUSE 6 440 NEXT n 450 BEEP
.003,0 460 OVER 0 470 RETURN
500 REM ** kadr 2 (menu) **
510 BEEP .05,55
520 CLS
530 INK 5
540 PRINT AT 7,9;"1 - START GAME"
550 PRINT AT 9,9;"2 - REDEFINE KEYS"

```

```

560 PRINT AT 11,9;"3 - LEVEL"
570 PRINT AT 13,9;"4 - END GAME"
580 INK 4
590 PRINT AT 20,7;"Press keyпп(1 - 4)"
600 LET f=0
610 INK c
620 PRINT AT 3,13;"TETRIS"
630 LET c=c+1: IF c>7 THEN LET c=1
640 LET f=f+1: IF f>200 THEN GO TO 700
650 IF INKEY$="1" THEN GO TO 2000
660 IF INKEY$="2" THEN GO TO 1000
670 IF INKEY$="3" THEN GO TO 3000
680 IF INKEY$="4" THEN STOP
690 GO TO 610
700 REM ** kadr 3 (score) **
710 BEEP .05,55
720 CLS
730 INK 6
740 FOR n=1 TO 9
750 PRINT AT n*2+1,7;"N";n;" .....ппо"
760 NEXT n
770 LET f=0:
780 LET c=c+1: IF c>7 THEN LET c=1
790 LET f=f+1
800 IF f>200 OR INKEY$<>"" THEN GO TO 500
810 INK c
820 PRINT AT 1,11;"HI SCORE"
830 GO TO 780
999 REM
1000 REM ==== REDEFINE KEYS ====
1001 REM
2000 REM ==== START GAME ====
2001 REM
3000 REM ==== LEVEL ====
3001 REM

```

10 - начало фрагмента программы, описывающего первый кадр заставки. Мы дадим ему условное имя - НАЗВАНИЕ ИГРЫ;

20 - установка атрибутов экрана;

30 - задание переменных, определяющих позицию печати по горизонтали (x) и цвет букв (c);

40 - переменная a\$ содержит текст, выводимый на экран подпрограммой, начинающейся со строки 300;

50 - обращение к подпрограмме, разделяющей символы слова из переменной a\$ на две последовательности, в одну из которых входят знаки, занимающие в слове нечетные позиции, а в другую - символы с четными позициями, после чего две половинки слова двигаются сверху и снизу навстречу друг другу;

60...100 - слово «1993» также разбито на две части, которые двигаются в цикле слева и справа к средней части экрана. Чтобы за строками «19» и «93» не тянулся след, перед «19» и после «93» печатаем по одному пробелу, стирающему предыдущее изображение;

110 - звук, имитирующий удар при соприкосновении символов «19» и «93»;

120, 130 - установка фиолетовых «чернил» и печать в 20-й строке и 4-й позиции слов Press any key to continue (для продолжения нажмите любую клавишу);

140 - задание в переменной c1 цвета, который будем называть предшествующим текущему;

150 - вычисление кода цвета, в который будет окрашено название игры, как

случайного числа от 1 до 7 (черный цвет исключается) и присваивание полученного результата переменной *s*. Будем называть этот цвет текущим;

160 - здесь мы снова сталкиваемся с оператором IF.. THEN, который носит название условного и переводится как ЕСЛИ. . . ТОГДА. Проверяемое условие ставится сразу же за IF (например,  $X > 3$ ,  $A < 18$ ,  $R\$ = "AA"$ ). Если оно соответствует действительности, то выполняются операторы, стоящие за THEN, в противном случае управление передается на следующую строку программы. Содержание строки 160 можно перевести на русский язык как «если текущий цвет такой же, как и предшествующий, то перейти на строку 150». Такая строка введена для того, чтобы цвета не повторялись;

170 - установка текущего цвета в соответствии с переменной *s*;

180...250 - печать названия игры (TETRIS) с помощью звездочек;

260 - текущий цвет становится предшествующим и запоминается в переменной *s1*;

270 - пауза 1/5 секунды;

280 - оператор условного перехода. Если не нажата никакая клавиша, то на экране будет напечатано название новым цветом (переход на строку 150). Если же какая-нибудь клавиша окажется нажатой, то происходит переход к следующей строке программы (290);

290 - отсюда программа переходит на следующий кадр - его мы назовем МЕНЮ, - часть программы которого начинается со строки 500;

300 - начало подпрограммы вывода слов SANKT- PETERSBURG;

310 - двум строковым переменным присваивается значение пустой строки. В данном случае это сделано просто для инициализации переменных, чтобы впоследствии не появилось сообщение об ошибке Variable not found (переменная не определена);

320 - вычисляется количество букв в строковой переменной *a\$* и полученное значение присваивается переменной *d*;

330...360 - цикл, в котором исходная фраза, содержащаяся в переменной *a\$*, разбивается на две части, и буквы с нечетными позициями заносятся в *b\$*, а с четными - в *c\$*;

370 - устанавливается объединяющий режим наложения изображений на экране. Делается это для того, чтобы в момент сближения двух слов *b\$* и *c\$* пробелы одного не стерли буквы другого. Оператор OVER устанавливает режим наложения изображений на экране, причем если аргумент равен:

1 - изображение накладывается на уже существующее, не стирая

его; точки пересечения принимают окраску фона; 0 - новое изображение стирает уже имеющееся на экране;

380...440 - в цикле печатаются две половинки исходного слова, помещенного в переменную *a\$*, которые сходятся друг к другу сверху и снизу;

390, 410 - символы, помещенные в переменные *b\$* и *c\$* выводятся цветом фона (PAPER), что равнозначно их стиранию. При этом переменная *n* во всех строках цикла играет роль координаты *y*;

400, 420 - вывод символов, помещенных в переменные *b\$* и *c\$* на новое место. Цвет, которым будет выводиться текст, установлен ранее в строке 370;

430 - чтобы продвижение слов не было слишком быстрым, внутри цикла устанавливается пауза;

450 - звуковой сигнал;

460 - установка нормального режима наложения информации (без объединения);

470 - возврат из подпрограммы к строке, следующей после оператора GO SUB 300, то есть к строке 60;

500, 510 - начало нового кадра заставки - МЕНЮ. Его появлению предшествует звуковой сигнал;

520, 530 - очистка экрана, задание цвета букв в тексте;

540...570 - вывод на экран четырех пунктов меню;

580 - установка нового цвета символов;

590 - вывод на экран фразы, которая по-русски означает «нажмите клавиши 1-4»;

600 - присваивание переменной f, выполняющей в дальнейшем роль счетчика (смотрите строку 640), значения 0;

610 - установка текущего цвета символов;

620 - печать названия игры;

630 - увеличение номера текущего цвета на единицу, но так, чтобы он не превышал число 7. Если он станет больше 7, то переменная примет значение 1;

640 - в этой строке изменяется значение счетчика, контролирующего количество выполняемых циклов. Счетчик применяется в тех случаях,

когда по каким-либо причинам невозможно или затруднено использование оператора FOR...NEXT. Как только в счетчике окажется число, большее 200, произойдет переход на строку 700, начинающую вывод на экран третьего кадра, который мы назовем РЕЗУЛЬТАТЫ;

650..680 - очень важный блок операторов условного перехода, с помощью которого можно из меню попасть на другие кадры заставки или к разным частям игровой программы. Для этого необходимо нажать одну из клавиш 1, 2, 3 или 4;

650 - если нажать клавишу 1, то программа перейдет на строку 2000, с которой должна начинаться игровая часть программы;

660 - нажатие клавиши 2 вызывает переход на строку 1000, с которой предполагается начинать подпрограмму 4-го кадра, который назовем ПЕРЕНАЗНАЧЕНИЕ КЛАВИШ. Такая часть имеется в игре МАКСИТ, представленной в Приложении 1;

670 - при нажатии клавиши 3 начинается подпрограмма выбора уровня игры (строка 3000). Это может быть изменение каких-то переменных, либо создание нового лабиринта, как, к примеру, в игре СОКОБАН;

680 - если нажать клавишу 4, выполнится один из самых простых операторов Бейсика - STOP, который остановит выполнение программы. В служебном окне экрана появится сообщение

STOP statement;

690 - если будет нажата клавиша, отличная от 1...4, или не будет нажата никакая клавиша, то программа возвратится к строке 610 и, начиная с нее, повторится;

700 - начало нового кадра заставки - РЕЗУЛЬТАТЫ;

710 - звуковой сигнал;

720, 730 - очистка экрана и установка цвета символов;

740...760 - печать таблицы из 9 строк, в каждой из которых указан ее номер, затем точками отмечено место, куда потом будет заноситься имя игрока и наконец, набранные очки;

770 - еще один счетчик, необходимый для того, чтобы таблица «держалась» на экране определенное время. Оператор PAUSE ни здесь, ни в меню не может быть использован, так как надпись HI SCORE (строка 820), расположенная над таблицей, должна переливаться разными цветами;

780 - эта строка работает так же, как и строка 630;

790 - увеличение счетчика на единицу;

800 - если число в счетчике станет больше 200, или будет нажата какая-либо клавиша, то произойдет переход к строке 500. Таким образом, строки 640 и 800 выполняют автоматическое переключение с меню на таблицу результатов и наоборот.

Здесь нужно сказать вот о чем. Когда при выполнении одного из нескольких условий должны следовать одинаковые действия компьютера, то обычно строки с этими условиями записываются как одна, а сами условия соединяются ключевым словом OR - «или». Таким образом, два условия объединены в одной строке:

```
800 IF f>200 OR INKEY$<>"" THEN GO TO 500
```

Это позволяет не только сократить текст программы, но и несколько увеличить ее быстродействие. Надо добавить еще, что в операторе IF...THEN при необходимости подобным образом можно проверять не два, а сколько угодно условий;

810, 820 - установка текущего цвета и печать в строке 1 и позиции 11 слов HI SCORE (максимальный счет);

830 - безусловный переход на строку 780, где изменяется цвет надписи HI SCORE;

1000 - начало подпрограммы переназначения клавиш. Ее, если потребуется, вы сможете написать самостоятельно;

2000 - с этой строки должна начинаться игровая часть программы;

3000 - начало подпрограммы установки уровня игры.

### ***Делаем картинку-заставку с помощью Art Studio***

Выше мы рассмотрели программы-заставки, написанные на Бейсике, в которых использованы различные интересные эффекты. Но, как известно, первым кадром в фирменных заставках почти всегда идет яркая картинка, которую средствами Бейсика сделать практически невозможно. Для этого больше подходят специальные графические редакторы, с одним из которых - пожалуй, наиболее мощным - мы сейчас познакомимся.

Применение редактора Art Studio позволит вам создавать красочные картинки, которые по качеству не уступят фирменным. Предел совершенству может положить только отсутствие фантазии и трудолюбия. Подробное описание Art Studio имеется в книге [1], но не у всех она есть, поэтому мы сочли необходимым поместить в Приложении 3 краткую сводку команд редактора. Здесь же мы расскажем о том, как использовать эту программу для конкретных целей - создания картинки-заставки и как после записи экранного файла на ленту встраивать его в конкретную игру.

После загрузки Art Studio в компьютер, в верхней части экрана появится главное меню, а ниже - стрелка курсора. Управление стрелкой может осуществляться как джойстиком, так и с помощью клавиатуры: клавиши Q, A, O и P двигают стрелку соответственно вверх, вниз, влево и вправо. Если нажать сразу две клавиши, например, Q и O, то стрелка будет двигаться по диагонали. Ввод в действие выбранной функции выполняет клавиша M или кнопка Огонь джойстика.

Рассмотрим последовательность действий при создании экранной картинки.

Начнем работу с установки стрелки на функцию ATTRIBUTES (выбор палитры). Нажав клавишу M (или Огонь), вы увидите в прямоугольной рамке еще одно меню, в котором легко узнать встречавшиеся ранее слова INK, PAPER, BORDER и другие. Если же действие какой-то функции вам не совсем понятно - загляните в Приложение 3 или в книгу [1].

Если вы случайно попали не в то меню, достаточно отвести стрелку в область, не занятую меню, и нажать М.

Предположим, что выбрана функция SET INK (установка цвета тона - INK). На экране сразу появится цветовая палитра, в которой с помощью стрелки выбирается нужный цвет. То же самое можно проделать для PAPER и BORDER. А вообще, не бойтесь экспериментировать, проверьте действие всех функций этого меню и подберите комбинацию цветов для создания своей первой картинки.

Следующий шаг - выбор инструмента, с помощью которого вы будете создавать свое произведение. Для этого установите стрелку в главном меню на функцию PAINT (рисование кистью, пером, распылителем) и нажмите М. В появившемся новом меню выберите для начала PEN (рисование пером), а затем - одно из 18 перьев (в каждом из 18 появившихся квадратиков показано, какова будет толщина линий по горизонтали, вертикали и диагонали).

Техника рисования очень проста: нажатие клавиши М опускает перо на «бумагу», а клавиши Q, A, O и P перемещают его в разных направлениях. Выполните рисунок из линий, причем таким образом, чтобы в нем было несколько замкнутых областей (рис. 2.4), назначение которых станет понятно вам позднее.

Если в какой-то момент у вас дрогнула рука и линия получилась не такой ровной, как хотелось бы, установите стрелку на функцию UNDO главного меню и нажмите М. Экран примет тот вид, который он имел до выполнения последнего действия. Таким образом функция UNDO позволяет исправить практически любую ошибку

Попробуйте теперь улучшить свой рисунок при помощи распылителя. Для этого необходимо вернуться стрелкой к пункту PAINT и выбрать в нем функцию SPRAY CAN (рисование распылителем) Плотность распыления можно установить выбором одного из 8 квадратиков. Попрактикуйтесь немного в малярном ремесле (рис. 2.5), после чего можно переходить к работе кисточкой.

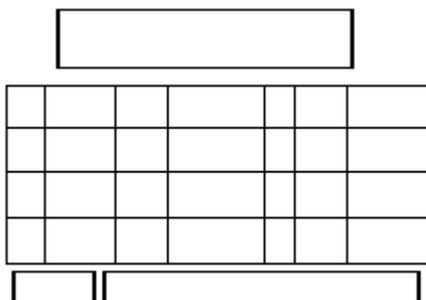


Рис. 2.4. Рисование пером в Art Studio.

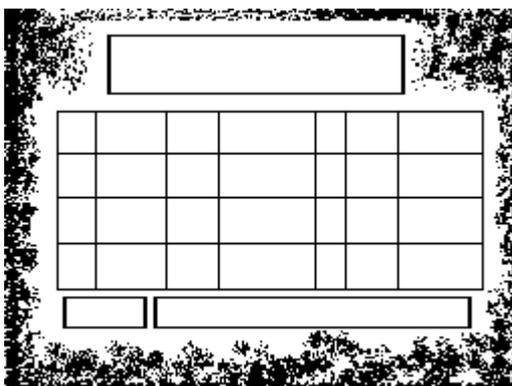


Рис. 2.5. Рисование распылителем.

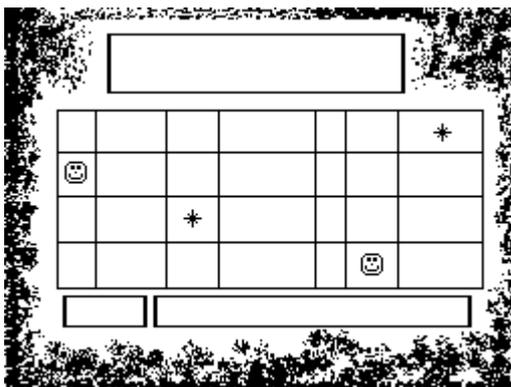


Рис. 2.6. Работа кистью.

Этот режим устанавливается в пункте BRUSH главного меню. Выберите одну из 16 появившихся кисточек, например, «рожицу», и проведите несколько линий (рис. 2.6). Думается, что это занятие доставит вам удовольствие.

Графический редактор Art Studio позволяет не только рисовать линии, но и закрашивать замкнутые области различными фактурами. Поставьте стрелку в главном меню на функцию FILL (закрашивание объекта)

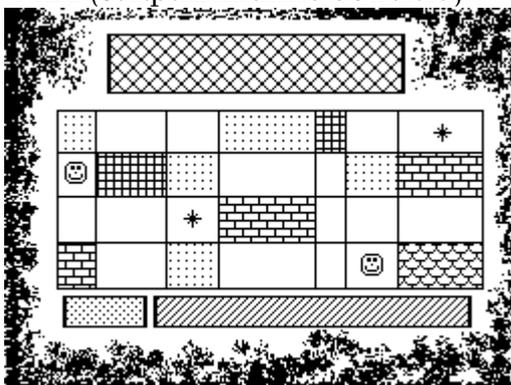


Рис. 2.7. Заполнение замкнутых областей фактурой.

и нажмите клавишу M. В появившемся новом меню выберите пункт SOLID FILL (полное закрашивание объекта) и передвиньте валик, заменивший в этом режиме стрелку-курсор, на одну из замкнутых областей вашего рисунка. После нажатия клавиши M эта область должна закраситься цветом, установленным ранее функцией SET INK пункта ATTRIBUTES. Если вам захочется закрасить новую область другим цветом, обратитесь к пункту ATTRIBUTES и после появления меню выберите SET INK и далее желаемый цвет. Теперь опять перейдите к пункту FILL и установите новый режим - TEXTURED FILL (заполнение объекта фактурой). Какой только фактуры нет в этих 32 квадратиках - на любой вкус! Выберите «кирпичи» и заполните ими одну область, затем подберите какой-нибудь «текстиль» и заполните другую, и так далее, для всех областей (рис. 2.7).

Если в качестве фактуры взять, к примеру, вертикальные полосы, а валик установить за пределами всех замкнутых областей, то после нажатия M вся внешняя часть рисунка заполнится этой фактурой, а рисунок тем самым выделится и станет еще лучше.

И, наконец, последний штрих в создании рисунка. Программа Art Studio работает не только с изображениями, но и с текстом. Установите стрелку на пункт ТЕХТ главного меню (работа с текстом) и задайте высоту (...HEIGHT) и ширину (...WIDTH) букв. Теперь выберите функцию LEFT TO RIGHT (печать текста слева направо). Стрелка заменится курсором, размер которого зависит от высоты и ширины букв. Поместите курсор в то место экрана, откуда должен начинаться

текст, нажмите М и набирайте текст (рис. 2.8). Для редактирования можно использовать

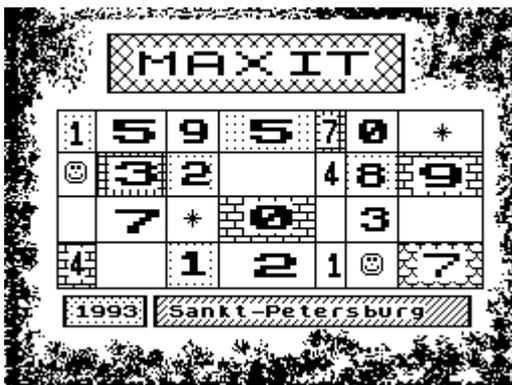


Рис. 2.8. Печать текстов в Art Studio.

курсорные клавиши и клавишу DELETE (Caps Shift и 0). Ввод надписи завершается нажатием клавиши Enter.

Для печати текста сверху вниз нужно выбрать в меню DOWNWARDS. О других способах вывода текста на экран можно узнать в Приложении 3.

С помощью графического редактора можно строить геометрические фигуры (функция SHAPES), редактировать мелкие детали изображений в увеличенных масштабах (функция MAGNIFY) и т. д. Некоторые из этих функций Art Studio мы рассмотрим в других главах.

Итак, вы нарисовали свою первую компьютерную картину, и теперь хорошо было бы записать ее на магнитную ленту. Для этого в редакторе предусмотрен пункт главного меню FILE, с помощью которого можно не только записать картинку на магнитный носитель, но загрузить и подправить рисунок от другой игровой программы, проверить правильность записи и многое другое. Поскольку в данный момент нас интересует запись, то выберите функцию SAVE FILE (запись экранного файла). На экране появится запрос: Filename? (имя файла?). Теперь с клавиатуры нужно ввести имя, под которым будет записан экранный файл, например, SCREEN\$. Нажав клавишу Enter, вы увидите на экране знакомое сообщение: Start tape, then press any key (включите магнитофон на запись и нажмите любую клавишу). Начнется запись на ленту экранного файла длиной 6912 байт. Когда файл будет записан на экране вновь появится главное меню.

В заключение покажем, как можно использовать получившийся экранный файл на конкретном примере - игровой программе МАКСИТ, представленной в Приложении 1. Запишите на ленту маленькую программу-загрузчик:

```
10 LOAD "SCREEN$"CODE 16384
20 LOAD "MAXIT"
```

В этой программе строка 10 загружает в экранную область памяти компьютера (с адреса 16384) созданный вами файл. А после того, как картинка полностью появится на экране, строка 20 обеспечит загрузку бейсик-программы игры МАКСИТ.

### 3. ПОСТРОЕНИЕ ИГРОВОГО ПРОСТРАНСТВА

Для того, чтобы написать хорошую игрушку, прежде всего нужно тщательно продумать сюжет, и только потом приступать к созданию спрайтов, необходимых для реализации ваших идей.

#### *Из чего состоит игровое пространство*

На смену одним играм приходят другие, количество сюжетов безгранично - но, как и все на свете, компьютерные игры поддаются классификации.

Большое количество компьютерных игр создано по сюжетам приключенческих фильмов и книг: Robocop, Rambo, Empire, Batman, Death Star и т. д. В основу других положены распространенные «настольные» игры - Chess, Poker, Draught. Компьютер позволяет смоделировать и логические игры типа Tetris, Petris, Sokoban, Bolder Dash и другие. Многие программы предлагают вам стать участником космических сражений (Elite, Star Raiders, R-Type, Star Fox), воздушных боев (MIG-29, Spitfire, F-16, Flying Shark, River Raid) или выполнить рискованные задания (Into The Eagles Nest, Saboteur, Bruce Lee, Flying Dragons). Получили распространение игры-имитаторы спортивных состязаний - Emlyn Huger Soccer, Basket Master.

В большинстве случаев игровое пространство может быть разбито на три существенно отличающихся друг от друга части. К первой отнесем все неподвижные элементы изображения, имитирующие небо, землю, воду, космическое пространство со звездами, и «мелкие» объекты - здания, корабли, деревья, мосты и т. д. В дальнейшем все это для краткости будем называть пейзажем.

Вторую часть составляют подвижные элементы изображения, на которые вы не можете оказывать управляющих воздействий. Как правило, к этой группе относятся изображения ваших противников, например, силуэт танка в игре Tanx или автомобиля в Wee Le Mans.

Наконец, к третьей части игрового пространства обычно относится одно изображение (или, реже, несколько), действиями которого вы можете управлять с помощью клавиатуры или джойстика. Так, в игре Капе это ковбой, преодолевающий различные препятствия и ведущий борьбу с многочисленными врагами. В игре Tetris - падающие на дно стакана геометрические фигурки, отличающиеся друг от друга цветом и формой. Наконец, в игре Cabal и многих других управляемым элементом является перекрестье прицела.

Итак, представим себе, что выбрана тема игры, тщательно проработан сюжет (мысленно определены все элементы, их цвет, размер и траектория движения), четко сформулирована цель и намечены способы ее достижения. Можно приступать к программированию.

#### *Создание графических объектов*

За исключением самых первых текстовых программ жанра Adventure, ни одна игра не обходится без множества подвижных графических объектов - спрайтов, поэтому давайте сделаем еще один шаг вперед по пути создания хороших игровых программ - научимся создавать такие объекты, а затем заставим их двигаться.

Каким образом получить изображение на экране? Во-первых, можно воспользоваться операторами PLOT и DRAW. Но при этом, как вы понимаете, программа выйдет достаточно громоздкой и «медлительной». Если не верите -

введите и выполните программку, рисующую на экране ракету:

```
10 PLOT 120,80: DRAW 3,-3
20 DRAW 0,-11: DRAW 4,-4
30 DRAW 0,-5: DRAW -4,4
40 DRAW -6,0: DRAW -4,-4
50 DRAW 0,5: DRAW 4;4
60 DRAW 0,11: DRAW 2,2
```

Нетрудно догадаться, что таким способом можно создавать только простейшие изображения, а для рисунков с обилием мелких деталей этот рецепт никуда не годится. Поэтому программисты обычно заранее заготавливают картинку, называемые спрайтами, а затем вводят их в память в виде блока данных (набора чисел). Нам остается выяснить, каким образом можно получить такой блок данных и как потом превратить его в изображение на экране.

Идея здесь довольно проста. Любое изображение на экране (спрайт) можно разбить на несколько маленьких спрайтиков. А любые символы (буквы, цифры) тоже, по сути, маленькие спрайтики, размером в одно знакоместо. И выводить их на экран проще простого - с помощью оператора PRINT. Значит, если нам удастся изменить некоторые символы таким образом, чтобы при выводе в совокупности они составили нужную картинку, то задачу можно будет считать решенной.

По счастью, разработчики ZX Spectrum предусмотрели возможность изменения набора символов. Проще всего изменить символы, специально предназначенные для этого. Они так и называются: символы, определяемые пользователем - User Defined Graphics (UDG). Они выводятся на экран в режиме курсора [G] при нажатии клавиш от A до U. Режим курсора [G] включается (и выключается) одновременным нажатием клавиш Caps Shift и 9.

Теперь перейдем к главному вопросу: как закодировать изображение? Вы знаете, что символы строго вписываются в знакоместа экрана с размерами 8x8 точек - пикселей. Начертим на листе в клетку поле 8x8 клеток и изобразим на нем какую-либо фигуру, например, «гномика» (рис. 3.1).

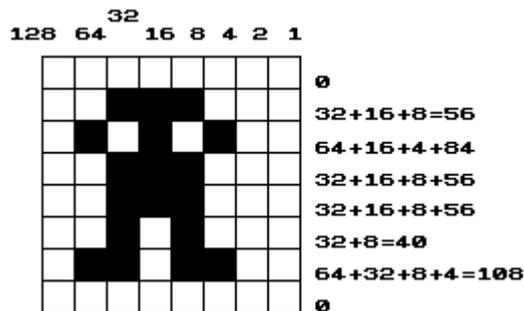


Рис. 3.1. Спрайт «Гномик».

Если заменить пустые клеточки нулями, а закрашенные - единицами, то мы получим последовательность чисел в двоичном представлении:

```
00000000
00111000
01010100
00111000
00111000
00101000
01101100
00000000
```

Для записи двоичных чисел в Бейсике используется ключевое слово BIN. Например,

```
PRINT BIN 10010110
```

или

```
LET a=BIN 10010110: PRINT a
```

Наконец, нам остается занести полученный ряд чисел в память. Как это сделать? Наверное, для вас не будет новостью, что память компьютера представляет собой последовательность ячеек, предназначенных для хранения чисел. При этом каждая ячейка имеет свой адрес, только не дом и не улицу, а номер от 0 до 65535. Чтобы занести в память какое-то число, нужно выполнить оператор РОКЕ, после которого сначала указывается адрес ячейки, а затем, после запятой, - само число. Помните только, что записывать в память можно не любые значения, а лишь целочисленные из диапазона от 0 до 255. Эти числа в дальнейшем мы будем называть байтами, а каждый отдельный разряд двоичного числа - битом. Таким образом, байт состоит из восьми битов.

По каким же адресам расположить числа, составляющие изображение нашего «гномика»? Узнать адреса размещения символов UDG позволяет функция USR, аргументом которой должен быть символ нужной клавиши. Например, PRINT USR "S" напечатает на экране адрес символа, определяемого пользователем, который соответствует клавише S<sup>1</sup>.

Теперь расположим, начиная с этого адреса, коды изображения «гномика» и затем выведем его на экран:

```
10 POKE USR "S", BIN 00000000
20 POKE USR "S"+1, BIN 00111000
30 POKE USR "S"+2, BIN 01010100
40 POKE USR "S"+3, BIN 00111000
50 POKE USR "S"+4, BIN 00111000
60 POKE USR "S"+5, BIN 00101000
70 POKE USR "S"+6, BIN 01101100
80 POKE USR "S"+7, BIN 00000000
90 PRINT "S"
```

Конечно же, вовсе не обязательно записывать в программе длинные блоки данных из двоичных чисел. Лучше заранее привести их к десятичному виду. Сделать это можно двумя способами: воспользовавшись ключевым словом BIN, либо подсчитать значения кодов вручную, просуммировав разряды числа. Например:

```
00111000 => 0S128 + 0S64 + 1S32 + 1S16 + 1S8 + 0S4 + 0S2 + 0S1 = 56
```

Теперь приведем пример программы, записывающей изображение нашего «гномика» в пользовательский символ, который соответствует клавише A.

```
10 FOR I=0 TO 7
20 READ N: POKE USR "A"+I, N
30 NEXT I
40 DATA 0, 56, 84, 56, 56, 40, 108, 0
```

10 - начало цикла, состоящего из восьми повторяющихся процедур (по числу запоминаемых байтов);

20 - функция USR "A" возвращает адрес символа UDG, соответствующего букве A, а оператор РОКЕ записывает в ячейку памяти с этим адресом значения переменной N, которые последовательно считываются оператором READ из списка данных, следующих за оператором DATA;

30 - конец цикла по переменной I;

40 - коды изображения «гномика», приведенные к десятичному виду.

После выполнения этой мини-программы в память компьютера будут введены коды «гномика». И если потребуется вывести на экран его изображение, например,

---

<sup>1</sup> В дальнейшем в текстах программ клавиши, которые нужно нажимать в режиме курсора [G], будут выделены подчеркнутым жирным шрифтом, например: A, B, C и т. д.

в знакоместо с координатами (10, 15), достаточно набрать в программе дополнительную строку

```
50 PRINT AT 10,15;"A"
```

причем клавишу А необходимо нажимать в режиме курсора [G].

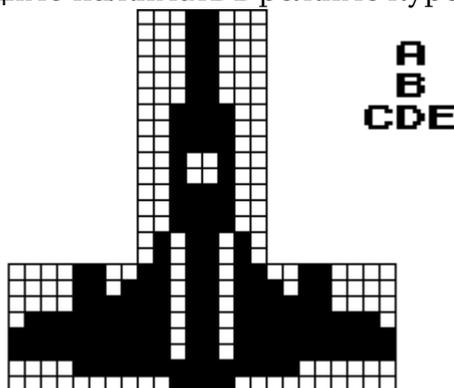


Рис. 3.2. Спрайт «Реактивный самолет».

Предположим теперь, что нам требуется закодировать, а затем вывести на экран более сложный рисунок, например, изображение реактивного самолета.

Рисунок состоит из пяти знакомест. Обозначим их буквами А, В, С, D и Е (рис. 3.2), а затем закодируем каждое способом, описанным в предыдущем примере.

Программа, осуществляющая запись изображения самолета в память, должна выглядеть примерно так:

Программа 7. Спрайт "реактивный самолет".

```
10 BORDER 1: PAPER 1: INK 6
20 CLS
30 FOR N=0 TO 39
40 READ S
50 POKE USR "A"+N,S
60 NEXT N
70 GO TO 300
100 DATA 24,24,24,24,24,24,60,60: REM A
110 DATA 60,36,36,60,60,60,90,90: REM B
120 DATA 12,13,15,127,255,255,15,0: REM C
130 DATA 219,219,219,219,219,219,255,60: REM D
140 DATA 48,176,240,254,255,255,240,0: REM E
300 PRINT AT 10,10;"A"
310 PRINT AT 11,10;"B"
320 PRINT AT 12,9;"CDE"
```

10 - установка атрибутов экрана;

20 - очистка экрана;

30 - начало цикла, в котором должно быть прочитано  $5 \times 8 = 40$  кодов (с 0 по 39 включительно);

40 - в процессе работы цикла оператор READ считывает в переменную S поочередно все данные из списка DATA;

50 - в этой строке происходит следующее. Число, присвоенное переменной S, теперь переписывается в ячейку памяти компьютера, имеющую адрес USR "A" + N. При  $N = 0$  этот адрес составляет 65368. Так как в цикле по N величина USR "A" + N все время увеличивается на единицу, то требуемые коды будут последовательно записаны в ячейки памяти с адресами 65368, 65369, 65370, ...

60 - конец цикла Теперь можно либо вывести изображение самолета на экран (как сделано у нас), либо обратиться к другим частям программы;

70 - переход на строку 300, с которой начинается вывод на экран изображения самолета;

100...140 - здесь хранятся коды, соответствующие пяти знакоместам

изображения самолета. Заметим, что мы могли бы записать все данные одним оператором DATA, но при этом пострадала бы «читабельность» программы;

300 - вывод на экран в знакоместо с координатами (10, 10) фрагмента А реактивного самолета. (Еще раз напомним, что при наборе строки клавишу А следует нажимать в режиме курсора [G]);

310 - вывод на экран фрагмента В;

320 - вывод на экран фрагментов С, D и E.

Итак, в нашем распоряжении появилась программа, которую можно использовать как фрагмент сложной компьютерной игры. Чтобы убедиться в этом, сделаем несколько заготовок с расчетом использовать их в будущем. Вначале создадим спрайт вертолета с вращающимся винтом (рис. 3.3). Здесь нам понадобятся две картинки, изображающие вертолет

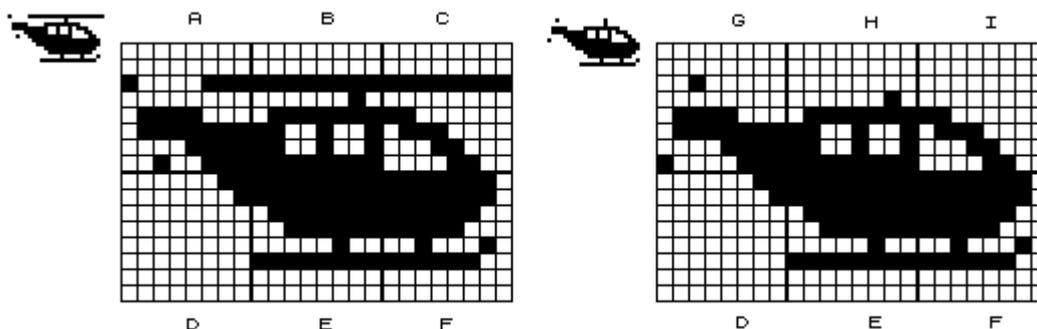


Рис. 3.3. Спрайт «Вертолет».

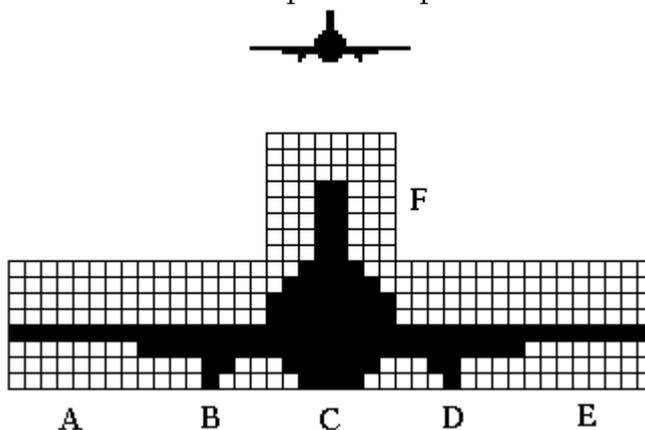


Рис. 3.4. Спрайт «Самолет».

с разным положением винта. Если быстро подменять одну другой на экране, то появится иллюзия движения - как в мультфильме.

Закодируем два этих изображения, заменив в предыдущей программе строки с оператором DATA на такие:

```
100 DATA 0, 0, 135, 0, 120, 137, 15, 39: REM A
110 DATA 0, 0, 255, 2, 127, 201, 201, 255: REM B
120 DATA 0, 0, 255, 0, 192, 112, 24, 12: REM C
130 DATA 3, 1, 0, 0, 0, 0, 0, 0: REM D
140 DATA 255, 255, 127, 63, 4, 255, 0, 0: REM E
150 DATA 254, 254, 252, 248, 34, 252, 0, 0: REM F
160 DATA 0, 0, 32, 0, 120, 127, 15, 135: REM G
170 DATA 0, 0, 7, 2, 127, 201, 201, 255: REM H
180 DATA 0, 0, 0, 0, 192, 112, 24, 12: REM !
```

Кроме того, максимальное значение переменной N в строке 30 программы нужно исправить на 71. Теперь можно выводить спрайт на экран, придерживаясь следующей последовательности символов: ABC - верхняя часть картинка, DEF - нижняя часть, GHI - верхняя часть, но с винтом, повернутым в другую сторону.

Нижняя часть вертолета не изменяется, поэтому во второй картинке используются те же символы DEF. Попробуйте написать программку, рисующую вертолет с вращающимся винтом, самостоятельно. Если же у вас возникнут проблемы, загляните в главу 5 и посмотрите, как это сделано в программе ВЕРТОЛЕТ

Теперь нарисуем спрайт с изображением самолета (рис. 3.4), который в пятой главе мы заставим летать по всему экрану, участвуя в воздушном бою. Операторы DATA здесь будут такими:

```
100 DATA 0,0,0,0,255,0,0,0: REM A
110 DATA 0,0,0,0,255,255,6,4: REM B
120 DATA 60,126,255,255,255,126,60: REM C
130 DATA 0,0,0,0,255,255,96,32: REM D
140 DATA 0,0,0,0,255,0,0,0: REM E
150 DATA 0,0,0,24,24,24,24,24: REM F
```

Управляющая переменная N принимает максимальное значение 55. Для вывода спрайта на экран оператором PRINT сначала рисуются крылья и фюзеляж, изображаемые рядом символов UDG, соответствующих буквам A, B, C, D и E, а затем точно над C ставится символ F, хранящий изображение хвоста самолета.

Ниже приведен листинг еще одной небольшой программы, создающей на экране подвижные спрайты.

#### Программа 8. ПОДВИЖНЫЕ СПРАЙТЫ.

```
10 BORDER 1: PAPER 1: INK 6: CLS
20 FOR I=0 TO 47
30 READ N: POKE USR "A"+I,N
40 NEXT I
55 DATA 0,2,3,7,15,19,83,126
56 DATA 0,64,64,224,240,200
57 DATA 202,126,127,115,120,31
58 DATA 15,4,4,28,254,206,30
59 DATA 248,240,32,56,0,127
60 DATA 127,120,27,15,4,28,0
61 DATA 254,254,30,216,240,32
62 DATA 32,56
100 PLOT INK 4;60,71: DRAW INK 4;135,0
105 LET C=3
110 PAUSE 50: IF INKEY$="" THEN GO TO 200
120 PRINT AT 10,14;"AB";AT 10,18; INK C;"ПП"
130 PRINT AT 11,14;"CD";AT 11,18; INK C;"AB"
140 PRINT AT 12,14;"ПП";AT 12,18; INK C;"EF"
150 PAUSE 50: IF INKEY$="" THEN GO TO 200
160 PRINT AT 10,14;"ПП";AT 10,18; INK C;"AB"
170 PRINT AT 11,14;"AB";AT 11,18; INK C;"CD"
180 PRINT AT 12,14;"EF";AT 12,18; INK C;"ПП"
190 GO TO 110
200 CLS
```

Поскольку все операторы этой программы рассматривались нами ранее, попробуйте самостоятельно разобраться в том, как она работает. Заметим лишь, что нажатие любой клавиши приводит к ускорению движения фигурок. Это связано с прекращением действия оператора PAUSE. А если нажать клавишу Space, то экран очистится и программа завершит свою работу.

### **Спрайт-генератор**

Кодирование изображения спрайта вручную - хотя и полезная, но довольно утомительная работа. У тех, кому приходится создавать много спрайтов, возникает естественный вопрос: нельзя ли сам компьютер заставить заниматься кодированием, причем в удобной для пользователя форме? Эта мысль была

реализована в программе генератора спрайтов. Поскольку эта программа представляет самостоятельный интерес как пример сервисной программы, облегчающей программисту жизнь, то

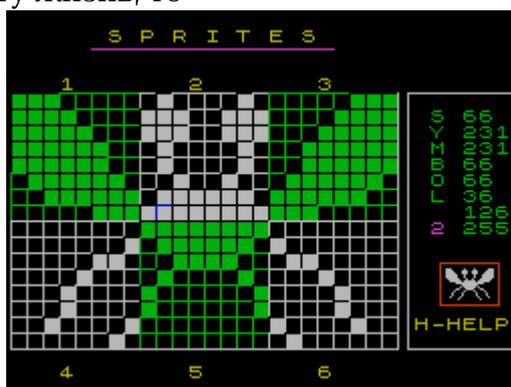


Рис. 3.5. Рабочее поле спрайт-генератора.

рассмотрим ее, как и все другие, - подробно. Однако предупредим вас заранее, что в этой программе встречается ряд операторов и приемов, о которых еще не говорилось и речь о которых пойдет в следующих главах. Поэтому начать досконально разбираться с некоторыми ее фрагментами лучше после изучения пятой главы книги.

После ввода программы и ее пуска на экране появится рабочее поле (рис. 3.5), каждая клеточка которого соответствует пикселю изображения спрайта. Поле разделено на шесть квадратов размером 8x8 клеток. Каждый квадрат изображает одно знакоместо спрайта. По клеткам поля может перемешаться курсор, который выполняет роль пера. Об управлении программой можно узнать, нажав клавишу H (HELP - подсказка). Курсор перемещается с помощью клавиш Q - вверх, A - вниз, O - влево и P - вправо. Точки ставятся (или удаляются) нажатием клавиши M. После нажатия клавиши N справа от поля по вертикали расположатся 8 чисел, соответствующих кодам знакоместа, номер которого виден тут же под словом SYMBOL. Эти числа затем можно записать в оператор DATA и использовать в нужном месте игровой программы подобно тому, как мы делали это для спрайтов «гномика», вертолета, реактивного самолета и т. д. Когда очередное знакоместо будет закодировано, нажмите клавишу Enter - и курсор переместится в следующий квадрат-знакоместо. При необходимости, квадрат, в котором находится курсор (текущее знакоместо), можно очистить от изображения, нажав клавишу C.

Прежде, чем привести текст программы, необходимо сказать вот о чем. В ней не предусмотрена возможность остановки и выхода в Бейсик. Но если вы уже набрали и запустили программу, а записать ее на магнитофон или диск забыли, то не хватайтесь за голову и не тянитесь к кнопке сброса. Разработчики ZX Spectrum позаботились о возможности остановки любой программы на Бейсике, если только она специально не защищена. Одновременное нажатие клавиш Caps Shift и Space (Break) позволяет прервать работу программы в любой момент, не дожидаясь ее завершения. При этом на экране появится сообщение BREAK into program.

Теперь рассмотрим саму программу.

#### Программа 9. СПРАЙТ-ГЕНЕРАТОР.

```
20 DIM g$(6,8,8): DIM c(8): GO SUB 6000
30 LET a$="SYMBOLПП"
40 GO SUB 8000
50 BORDER 0: PAPER 0: INK 6: CLS
60 PRINT AT 0,6;"S P R I T E S"
70 INK 3
80 PLOT 40,165: DRAW 120,0
90 INK 4
```

```

100 FOR n=1 TO 8
110 PRINT AT n+3,0;g$(1,n);
120 PRINT INK 7;g$(2,n);
130 PRINT INK 4;g$(3,n)
140 NEXT n
150 INK 7
160 FOR n=1 TO 8
170 PRINT AT n+11,0;g$(4,n);
180 PRINT INK 4;g$(5,n);
190 PRINT INK 7;g$(6,n)
200 NEXT n
210 INK 7
220 PLOT 0,15: DRAW 63,0
230 PLOT 128,15: DRAW 64,0: DRAW 0,128
240 PLOT 197,15: DRAW 0,128
250 DRAW 54,0: DRAW 0,-128
260 DRAW -54,0
270 PLOT 64,15: DRAW 63,0
280 INK 2
290 PLOT 213,37: DRAW 29,0: DRAW 0,21
300 DRAW -29,0: DRAW 0,-21
310 INK 6
320 PRINT AT 18,25;"H-HELP"
330 FOR i=0 TO 1
340 FOR j=0 TO 2
350 PRINT AT i*18+3,j*8+3;i*3+j+1
360 NEXT j
370 NEXT i
400 REM -----
410 LET x=1: LET y=1: LET s=1
420 LET px=-1: LET py=3
430 LET c=4: LET s1 = 1
450 REM -----
460 GO SUB 2000
470 LET x1=x: LET y1=y
480 LET k$=INKEY$
490 IF k$="P" OR k$="p" THEN LET x=x+1: GO TO 600
500 IF k$="O" OR k$="o" THEN LET x=x-1: GO TO 600
510 IF k$="Q" OR k$="q" THEN LET y=y-1: GO TO 600
520 IF k$="A" OR k$="a" THEN LET y=y+1: GO TO 600
530 IF (k$="M" OR k$="m") AND g$(s,y,x)="A" THEN LET g$(s,y,x)="B": PRINT AT
py+y,px+1; INK c;g$(s,y): GO SUB 7310: GO TO 600
540 IF (k$="M" OR k$="m") AND g$(s,y,x)="B" THEN LET g$(s,y,x)="A": PRINT AT
py+y,px+1; INK c;g$(s,y):GO SUB 7310: GO TO 600
550 IF CODE k$=13 THEN GO SUB 7210: LET s1=s1+1: IF s1=7 THEN LET s1=1
560 IF k$="N" OR k$="n" THEN GO SUB 2000: GO TO 600
570 IF k$="C" OR k$="c" THEN GO SUB 7200: GO TO 3000
580 IF k$="H" OR k$="h" THEN GO SUB 5000: GO TO 50
590 GO TO 620
600 REM -----
610 GO SUB 7110
620 IF x>8 OR x<1 OR y>8 OR y<1 THEN LET x=x1: LET y=y1
630 PRINT AT y+py,x+px; OVER 1; INK 8; PAPER 1;" "
640 IF k$<>" " THEN PRINT AT y1+py,x1+px; OVER 1; INK 8;PAPER 0;" "
650 LET s=s1
660 GO SUB 1010: GO TO 470
1000 REM -----
1010 IF s=1 THEN LET c=4: LET px=-1: LET py=3: RETURN
1020 IF s=2 THEN LET c=7: LET px=7: LET py=3: RETURN
1030 IF s=3 THEN LET c=4: LET px=15: LET py=3: RETURN

```

```
1040 IF s=4 THEN LET c=7: LET px=-1: LET py=11: RETURN
1050 IF s=5 THEN LET c=4: LET px=7: LET py=11: RETURN
1060 IF s=6 THEN LET px=15: LET py=11: LET c=7
1070 RETURN
2000 REM ---- CODE ----
2010 GO SUB 7000
2020 FOR n=1 TO 8
2030 LET o=0
2040 LET ad=(USR CHR$(66+s))*(n-1)
2050 FOR m=1 TO 8
2060 IF g$(s,n,m)="B" THEN LET o=o+c(m)
2070 NEXT m
2080 POKE ad,0
2090 INK 4
2100 PRINT AT n+4,26;a$(n);"nnnn"
2110 PRINT AT n+4,28;o
2120 NEXT n
2130 INK 7
2140 PRINT AT 15,27;"CDE"
2150 PRINT AT 16,27;"FGH"
2160 INK 3
2170 PRINT AT 12,26;s
2180 RETURN
3000 REM ---- CLEAR ----
3010 LET b=1
3020 PRINT #0;AT 1,2; INK 2; BRIGHT b;"CLEAR SYMBOL ";s;"?(Y/N)"
3030 IF INKEY$="N" OR INKEY$="n" THEN BEEP .01,20: GO SUB 3200: GO TO 600
3040 IF INKEY$="Y" OR INKEY$="y" THEN BEEP .01,0: GO SUB 3200: GO TO 3100
3050 LET b=NOT b
3060 GO TO 3020
3100 FOR m=1 TO 8
3110 LET g$(s,m)="AAAAAAAA"
3120 PRINT AT m+py,1+px; INK c;g$(s,m)
3130 NEXT m
3140 GO SUB 1000
3150 GO SUB 2000
3160 GO TO 600
3200 PRINT #0;AT 1,2;"oooooooooooooooooooooooooooo": REM 21 Space
3210 RETURN
5000 REM ---- HELP ----
5010 LET C=0
5020 BORDER 1: PAPER 1: INK 6
5030 INK 5: CLS
5040 PLOT 0,0: DRAW 0,175
5050 DRAW 255,0: DRAW 0,-175
5060 DRAW -255,0
5070 INK 7
5080 PRINT AT 8,2;"P - RIGHT";TAB 14;"C - CLEAR SYMBOL"
5090 PRINT AT 10,2;"O - LEFT";TAB 14;"N - CODE SYMBOL"
5100 PRINT AT 12,2;"Q - UP";TAB 14;"ENTER - SYMBOL+1"
5110 PRINT AT 14,2;"A - DOWN";TAB 14;"M - SET/RESET"
5120 PRINT AT 20,4; INK 6;"Press any key to continue"
5130 PRINT AT 2,12; INK C;"H E L P"
5140 LET C=C+1: IF C=8 THEN LET C=0
5150 IF INKEY$="" THEN GO TO 5130
5160 BEEP .002,15
5170 RETURN
6000 REM ---- DATA 1 ----
6010 FOR n=1 TO 8
6020 LET c(n)=2^(8-n)
```

```

6030 NEXT n
6040 RETURN
7000 REM ---- BEEP 1 ----
7010 BEEP .0004,30
7020 BEEP .0004,40
7030 BEEP .0004,50
7040 RETURN
7100 REM ---- BEEP 2 ----
7110 BEEP .0003,50
7120 BEEP .0003,30
7130 RETURN
7200 REM ---- BEEP 3 ----
7210 BEEP .005,15
7220 BEEP .004,10
7230 RETURN
7300 REM ---- BEEP 4 ----
7310 BEEP .003,25
7320 RETURN
8000 REM ---- GRAPHICS ----
8010 FOR n=0 TO 15
8020 READ s
8030 POKE USR "A"+n,s
8040 NEXT n
8050 FOR n=0 TO 48
8060 POKE USR "C"+n,0
8070 NEXT n
8080 FOR i=1 TO 6
8090 FOR n=1 TO 8
8100 LET g$(i,n)="AAAAAAA"
8110 NEXT n
8120 NEXT i
8130 RETURN
9000 DATA 255,128,128,128,128,128,128,128
9010 DATA 0,127,127,127,127,127,127,127

```

20 - массив g\$(6,8,8) описывает 6 знакомест спрайта площадью 8x8 точек, а с(8) содержит значения отдельных битов для кодировки изображения;

40 - обращение к подпрограмме, формирующей в определяемых пользователем символах, соответствующих клавишам А и В. графические элементы, изображающие выключенные (А) и включенные (В) пиксели:

50 - установка атрибутов экрана;

60...80 - вывод на экран названия программы и его подчеркивание;

90...140 - печать трех увеличенных знакомест спрайта, расположенных в верхней части поля;

150...200 - печать трех знакомест нижней части спрайта;

210...270 - дорисовывание нижней и боковой частей рабочего поля, а также рамки справа от него;

280...300 - вычерчивание рамки красного цвета, в которой после кодировки будет размещаться изображение полученного спрайта целиком и в реальном масштабе;

320 - печать надписи, подсказывающей, что для получения страницы помощи нужно нажать клавишу Н;

330...370 - вывод на экран номеров фрагмента спрайта;

410...430 - установка начальных значений переменных: x, y - текущие координаты курсора внутри любого фрагмента (знакоместа спрайта); s - номер фрагмента; rx, ry - приращения координат x и y. изменяющиеся в зависимости от номера выбранного фрагмента; c - цвет фрагмента;

460 - обращение к подпрограмме, которая выводит в рамке справа от рабочего поля коды и, кроме того, рисует спрайт в реальном масштабе;

470 - запоминание «старых» координат;

480...520 - стандартная схема управления курсором с помощью клавиш. С такой схемой мы еще встретимся и подробно ее обсудим в пятой главе;

530, 540 - эти строки управляют включением и выключением пикселя, на который указывает курсор. Если при нажатии клавиши M пиксель выключен, то он включится, и наоборот. Оператор AND в этих строках означает, что операторы, следующие за THEN, будут выполняться только в том случае, если истинно И то, И другое условие (and по-английски значит «и»);

550 - при нажатии клавиши Enter происходит перемещение курсора в следующее знакоместо спрайта;

560 - кодировка текущего знакоместа (того, где стоит курсор);

570 - очистка текущего знакоместа;

580 - вызов подсказки;

620 - проверка на предмет выхода курсора за пределы знакоместа;

630 - вывод на экран курсора синего цвета;

640 - удаление курсора на старом месте, но только в том случае, если была нажата какая-либо клавиша;

1000...1060 - в зависимости от номера текущего знакоместа определяется цвет и величины приращений координат курсора;

2000 - начало подпрограммы кодировки текущего знакоместа спрайта, печати полученных кодов и вывода изображения спрайта в реальном масштабе;

2020 - начало цикла кодирования 8 байт текущего знакоместа;

2030 - в переменной o будет подсчитываться значение каждого байта;

2040 - расчет адреса в области UDG, куда будет записываться полученный байт. Попробуем разобраться со сложным выражением после знака равенства. В переменной cod хранится так называемый ASCII-код символа, соответствующего номеру знакоместа (s).

Для начала выясним, что же такое ASCII-код. Возможно, вы уже задавались вопросом, каким образом компьютер обрабатывает символьные значения, ведь микропроцессор может работать только с числами и не знает, что такое буквы. Поэтому всем символам (и даже ключевым словам Бейсика) соответствуют специальные числа - ASCII-коды. Например, буква B кодируется числом 66, которое и используется в нашей программе. Так как переменная s принимает значения от 1 до 6, то переменная  $cod = 66 + s$  будет равна 67...72, что соответствует кодам букв C, D, E, F, G и H. Функция CHR\$ превращает код обратно в символ. Дальше все просто: функция USR возвращает адрес соответствующего символа UDG, к которому прибавляется номер кодируемого байта и полученный адрес записывается в переменную ad.

Теперь давайте вернемся немного назад и еще раз посмотрим на строку 550, Код символа перевода строки и клавиши Enter равен 13. Вы можете проверить это, введя строку

```
PAUSE 0: PRINT CODE INKEY$
```

и нажав клавишу Enter. Функция CODE - обратная по отношению к CHR\$ и возвращает код символа (в данном случае - символа нажатой клавиши). Таким образом, условие  $CODE k\$ = 13$  в строке 550 выполняется при нажатии клавиши Enter;

2050 - начало цикла подсчета значения байта, которое определяется состоянием 8 битов;

2060 - если бит установлен (графический символ В изображает включенный бит), то переменная *o* увеличивается на значение соответствующего бита;

2070 - конец цикла кодирования одного байта;

2080 - полученный байт заносится в область UDG по ранее рассчитанному адресу *ad*;

2100 - печать слова SYMBOL по вертикали и стирание ранее напечатанных кодов;

2110 - печать новых значений кодов;

2120 - конец цикла кодирования байтов;

2130...2150 - вывод изображения спрайта в реальном масштабе;

2170 - печать номера кодируемого знакоместа;

2180 - выход из подпрограммы;

3000...3140 - очистка того знакоместа спрайта, в котором находится курсор. Не будем описывать эти строки подробно. Объясним лишь смысл функции NOT в строке 3050. Эта функция возвращает одно из двух возможных значений: 1, если аргумент равен нулю, и 0, если аргумент ненулевой. Таким образом, короткая запись LET *b* = NOT *b* заменяет целых две строки. IF *b* = 0 THEN LET *b* = 1 и IF *b* <> 0 THEN LET *b* = 0,

Далее идут подпрограммы, смысл строк которых вам уже должен быть понятен без лишних подсказок:

5000...5170 - печать странички помощи HELP;

6000...6040 - заполнение массива *s*(8), определяющего значения отдельных битов, которые равны степеням числа 2:  $2^0 = 1$ ,  $2^1 = 2$ , ...,  $2^7 = 128$ . Этот массив включен в программу для ускорения кодирования изображения;

7000...7320 - четыре подпрограммы звукового оформления;

8000...9010 - подпрограмма кодирования используемых в программе образов включенного и выключенного пикселей увеличенного в 8 раз изображения спрайта, а также очистка шести символов UDG для записи в них готового спрайта.

### **Русификация программ**

Взгляните еще раз на последнюю программу - разве вам не хочется, чтобы страничка помощи (HELP) была написана на русском языке? И хотя ваш любимый Спрессу, скорее всего, не знает русских букв, научить его не так уж сложно.

Прежде всего обратите внимание на заглавные буквы английского алфавита. Многие из них совпадают по начертанию с русскими: А, В, Е, К, М, Н, О, Р, С, Т, Х. Русскую букву З можно заменить цифрой «три»

Остальные буквы можно создать точно так же, как мы до этого рисовали «гномика», то есть поместив русский шрифт в область определяемых пользователем символов (UDG). Таких символов потребуется 20, а именно: Б, Г, Д, Ж, И, Л, П, У, Ф, Ц, Ч, Ш, Щ, Ъ, Ы, Ь, Э, Ю, Я. И Программа, которая формирует эти недостающие буквы, выглядит так:

Программа 10. РУССКИЙ АЛФАВИТ.

```

9905 BORDER 0: PAPER 0: INK 5: CLS
9910 FOR N=1 TO 20
9920 READ A$
9930 FOR M=0 TO 7
9940 READ S
9950 POKE USR A$+M, S
9960 NEXT M
9970 NEXT N
9975 DATA "В", 0, 124, 64, 124, 66, 66, 124, 0
9976 DATA "Г", 0, 126, 64, 64, 64, 64, 0

```

```

9977 DATA "D", 0, 28, 36, 36, 36, 36, 126, 66
9978 DATA "J", 0, 73, 73, 62, 73, 73, 73, 0
9979 DATA "I", 0, 66, 70, 74, 82, 98, 66, 0
9980 DATA "L", 0, 30, 34, 34, 34, 34, 98, 0
9981 DATA "P", 0, 126, 66, 66, 66, 66, 66, 0
9982 DATA "O", 0, 66, 66, 36, 24, 16, 96, 0
9983 DATA "C", 0, 68, 68, 68, 68, 68, 126, 2
9984 DATA "H", 0, 66, 66, 66, 62, 2, 2, 0
9985 DATA "N", 0, 65, 73, 73, 73, 73, 127, 0
9986 DATA "M", 0, 65, 73, 73, 73, 73, 127, 1
9987 DATA "E", 0, 60, 66, 30, 2, 66, 60, 0
9988 DATA "U", 0, 76, 82, 114, 82, 82, 76, 0
9989 DATA "A", 0, 62, 66, 66, 62, 34, 66, 0
9990 DATA "S", 24, 66, 70, 74, 82, 98, 66, 0
9991 DATA "R", 0, 64, 64, 124, 66, 66, 124, 0
9992 DATA "T", 0, 192, 64, 124, 66, 66, 124, 0
9993 DATA "F", 0, 62, 73, 73, 73, 62, 8, 0
9994 DATA "Q", 0, 66, 66, 114, 74, 74, 114, 0

```

После выполнения программы вместо латинских букв, если вводить их в режиме курсора [G], мы увидим на экране следующие русские буквы:

```

В, G, D, J, I, L, P, O, F, C, H, N, M, T, Q, R, E, U, A, S
Б, Г, Д, Ж, И, Л, П, У, Ф, Ц, Ч, Ш, Щ, Ъ, Ы, Ь, Э, Ю, Я, Й

```

Если к предложенной программе добавить оператор 9995 RETURN, то ее можно будет использовать как подпрограмму. А теперь несколько слов о самой программе.

9910 - внешний цикл, в котором переменная N изменяется в пределах от 1 до 20 - по количеству вводимых букв.

9920 - ввод строковой переменной из оператора DATA. Во всех таких операторах на первом месте стоят латинские буквы, которые и будут в графическом режиме определять соответствующие русские буквы;

9930 - внутренний цикл, в котором осуществляется ввод кодов выбранной в предыдущем цикле буквы;

9940 - присваивание переменной S числа из списка DATA;

9950 - ввод кода в память компьютера по адресу USR A\$ + M;

9960, 9970 - операторы завершения циклов;

9975...9994 - блок данных, содержащий латинские буквы для обозначения символов UDG, и соответствующие коды для русских букв. Число операторов DATA равно количеству новых символов.

Приведенную программу не обязательно загружать и выполнять всякий раз, когда вам требуется русский шрифт. Достаточно сделать это один раз и записать сформированную область UDG-символов на магнитную ленту оператором

```
SAVE "RUS"CODE USR "A", 168
```

Если теперь вставить в любую программу оператор LOAD "RUS"CODE - то после загрузки этого короткого блока можно смело писать по-русски в соответствии с приведенной выше таблицей.

Но при такой русификации область UDG окажется занятой полностью и места для размещения спрайтов в ней уже не останется. Поэтому лучше воспользоваться графическим редактором Art Studio, с помощью которого можно изменять символы основного набора, оставляя UDG для прочих нужд.

Правда, работа с Art Studio, как вы уже заметили, требует определенных навыков. Если этих навыков у вас еще недостаточно, можете воспользоваться приведенной ниже программой, которая решает задачу по переводу всех символов стандартного (латинского) набора ZX Spectrum в совершенно новый - русский набор. При этом появляются как строчные, так и прописные буквы, а также сохраняются знаки препинания и дополнительные символы - всего 96. Эта

программа не затрагивает область UDG, которую можно использовать для создания спрайтов уже хорошо знакомыми вам способами одновременно с русским шрифтом.

После загрузки программы перед вами появится краткая инструкция по ее использованию. Так, для перехода к новому набору символов (символы русского алфавита) следует включить в программу оператор РОКЕ 23607,250 - а для возврата к стандартному (латинскому) набору РОКЕ 23607,60 В инструкции дана также таблица соответствия русских и латинских букв и знаков.

Вот эта программа:

#### Программа 11. НОВЫЕ СИМВОЛЫ.

```

10 BORDER 0: PAPER 0: INK 5: CLEAR 64255
20 POKE 23607,60
30 PRINT AT 10,10;"PLEASE WAIT"
40 LET s=0
50 FOR n=64256 TO 65023
60 READ a: LET s=s+a
70 POKE n, a
80 NEXT n
90 IF s<>64289 THEN PRINT AT 10,10;"ERROR IN DATA": STOP
100 REM -
110 POKE 23607,250
120 CLS
130 PRINT AT 1,6; INK 6;"Новый набор символов"
140 LET z$="-----"
150 PRINT AT 2,0;z$
160 PRINT AT 5,0;"РОКЕ 23607,250"; INK 4;"-Включение русских букв"; INK 5;"РОКЕ
23607,60"; INK 4;"-Включение латинских букв"
170 PRINT AT 12,6; INK 3;"Таблица соответствий:"
180 POKE 23607,60
190 LET a$="абвгдежзиклмнопрстуфхцщъьэюя"
200 PRINT INK 5;a$
210 POKE 23607,250
220 PRINT INK 1;z$
230 PRINT INK 4;a$
240 POKE 23607,60
250 LET a$="АБВГДЕЖЗИКЛМНОПРСТУФХЦЩЪЬЭЮЯ"
260 PRINT INK 5;a$
270 POKE 23607,250
280 PRINT INK 1;z$
290 PRINT INK 4;a$
300 REM -
310 POKE 23607,60
320 SAVE "RUS"CODE 64256,768
900 REM -
1000 DATA 0,0,0,0,0,0,0,0: REM Space
1010 DATA 0,24,24,24,24,0,24,0: REM !
1020 DATA 0,108,108,0,0,0,0,0: REM "
1030 DATA 36,36,255,36,36,255,36,36: REM #
1040 DATA 24,62,88,60,26,124,24,0: REM $
1050 DATA 96,102,12,24,48,102,70,0: REM %
1060 DATA 56,108,56,58,110,124,54,0: REM &
1070 DATA 0,12,24,48,0,0,0,0: REM '
1080 DATA 12,24,48,32,32,48,24,12: REM (
1090 DATA 48,24,12,4,4,12,24,48: REM )
1100 DATA 0,102,60,255,60,102,0,0: REM *
1110 DATA 0,24,24,126,126,24,24,0: REM +
1120 DATA 0,0,0,0,0,24,24,48: REM ,
1130 DATA 0,0,0,126,126,0,0,0: REM -

```

1140 DATA 0,0,0,0,0,24,24,0: REM .  
1150 DATA 2,6,12,24,48,96,64,0: REM /  
1160 DATA 124,198,206,214,230,198,124,0: REM 0  
1170 DATA 24,56,24,24,24,24,126,0: REM 1  
1180 DATA 60,102,6,60,96,102,126,0: REM 2  
1190 DATA 60,102,6,28,6,102,60,0: REM 3  
1200 DATA 28,60,108,204,254,12,30,0: REM 4  
1210 DATA 126,98,96,124,6,102,60,0: REM 5  
1220 DATA 60,102,96,124,102,102,60,0: REM 6  
1230 DATA 126,102,6,12,24,24,24,0: REM 7  
1240 DATA 60,102,102,60,102,102,60,0: REM 8  
1250 DATA 60,102,102,62,6,102,60,0: REM 9  
1260 DATA 0,24,24,0,24,24,0,0: REM :  
1270 DATA 0,24,24,0,24,24,48,0: REM ;  
1280 DATA 12,24,48,96,48,24,12,0: REM <  
1290 DATA 0,126,126,0,126,126,0,0: REM =  
1300 DATA 48,24,12,6,12,24,48,0: REM >  
1310 DATA 60,102,102,12,24,0,24,0: REM ?  
1320 DATA 204,214,214,246,214,214,204,0: REM @  
1330 DATA 30,54,102,102,126,102,102,0: REM A  
1340 DATA 254,96,124,102,102,102,252,0: REM B  
1350 DATA 204,204,204,204,204,206,254,6: REM C  
1360 DATA 28,44,108,108,108,108,254,0: REM D  
1370 DATA 254,98,104,120,104,98,254,0: REM E  
1380 DATA 124,214,214,214,124,16,56,0: REM F  
1390 DATA 254,102,96,96,96,96,240,0: REM G  
1400 DATA 198,108,56,56,108,198,198,0: REM H  
1410 DATA 198,198,206,222,246,230,198,0: REM I  
1420 DATA 118,182,206,222,246,230,198,0: REM J  
1430 DATA 230,102,108,120,108,102,230,0: REM K  
1440 DATA 30,54,102,102,102,102,102,0: REM L  
1450 DATA 198,238,254,214,198,198,198,0: REM M  
1460 DATA 102,102,102,126,102,102,102,0: REM N  
1470 DATA 124,198,198,198,198,198,124,0: REM O  
1480 DATA 126,102,102,102,102,102,102,0: REM P  
1490 DATA 126,198,198,126,54,102,198,0: REM Q  
1500 DATA 252,102,102,102,124,96,240,0: REM R  
1510 DATA 60,102,192,192,192,102,60,0: REM S  
1520 DATA 126,90,24,24,24,24,60,0: REM T  
1530 DATA 102,102,102,62,6,102,60,0: REM U  
1540 DATA 214,214,214,120,214,214,214,0: REM V  
1550 DATA 248,108,108,124,102,102,252,0: REM W  
1560 DATA 120,48,60,54,54,54,124,0: REM X  
1570 DATA 198,198,246,218,218,218,246,0: REM Y  
1580 DATA 60,102,6,28,6,102,60,0: REM Z  
1590 DATA 198,214,214,214,214,214,254,0: REM [  
1600 DATA 60,102,6,30,6,102,60,0: REM \  
1610 DATA 198,214,214,214,214,214,254,6: REM ]  
1620 DATA 102,102,102,102,62,6,6,0: REM  
1630 DATA 0,0,240,60,54,54,60,0: REM Z  
1640 DATA 0,0,204,214,246,214,204,0: REM  
1650 DATA 0,0,62,102,102,102,58,0: REM a  
1660 DATA 0,0,126,96,124,102,124,0: REM b  
1670 DATA 0,0,204,204,204,206,254,6: REM c  
1680 DATA 124,6,62,102,102,102,60,0: REM d  
1690 DATA 0,0,60,102,124,96,62,0: REM e  
1700 DATA 0,16,124,214,214,124,16,0: REM f  
1710 DATA 0,0,126,102,96,96,96,0: REM g  
1720 DATA 0,0,198,108,56,108,198,0: REM h  
1730 DATA 0,0,102,102,110,118,102,0: REM i

```

1740 DATA 24,24,102,102,110,118,102,0: REM j
1750 DATA 0,0,230,108,120,108,230,0: REM k
1760 DATA 0,0,30,54,102,102,102,0: REM l
1770 DATA 0,0,198,238,254,214,198,0: REM m
1780 DATA 0,0,102,102,126,102,102,0: REM n
1790 DATA 0,0,60,102,102,102,60,0: REM o
1800 DATA 0,0,126,102,102,102,102,0: REM p
1810 DATA 0,0,62,102,126,54,102,0: REM q
1820 DATA 0,0,252,102,102,124,96,240: REM r
1830 DATA 0,0,60,102,96,102,60,0: REM s
1840 DATA 0,0,126,24,24,24,24,0: REM t
1850 DATA 0,0,102,102,62,6,60,0: REM u
1860 DATA 0,0,214,214,124,214,214,0: REM v
1870 DATA 112,88,112,124,102,102,60,0: REM w
1880 DATA 0,0,96,120,108,108,120,0: REM x
1890 DATA 0,0,198,246,218,218,246,0: REM y
1900 DATA 0,0,60,102,28,102,60,0: REM z
1910 DATA 0,0,198,214,214,214,254,0: REM {
1920 DATA 0,0,60,6,30,6,60,0: REM I
1930 DATA 0,0,198,214,214,214,254,6: REM Y
1940 DATA 0,0,102,102,102,62,6,0: REM ~
1950 DATA 126,227,221,209,209,221,227,126: REM @

```

10 - подготовительные операции;

20 - объявление текущим стандартного набора символов на случай повторного запуска программы. При этом текст следующей строки будет выводиться по-английски;

30 - вывод на экран надписи PLEASE WAIT (пожалуйста, подождите), так как чтение кодов новых символов из операторов DATA занимает некоторое время;

40 - установка переменной s (счетчика контрольной суммы) в начальное состояние;

50...80 - цикл, в результате выполнения которого оператор READ из списка DATA считывает коды, которые затем оператором POKE записываются в ячейки памяти, начиная с адреса 64256 и по адрес 65023. Одновременно происходит суммирование всех кодов в переменной s;

90 - если сумма всех прочитанных кодов (контрольная сумма) не равна 64289, то счет останавливается, так как несовпадение контрольной суммы с числом 64289 указывает на ошибку в списке DATA. В этом случае необходимо тщательно проверить, что же вы ввели в компьютер на самом деле;

110 - включение режима новых символов;

120...290 - вывод на экран инструкции по применению этой программы;

310 - восстановление стандартного символьного набора;

320 - запись кодов русского шрифта на ленту. Если вы работаете с дисководом, то строка 320 должна выглядеть так:

```
320 RANDOMIZE USR 15619: REM: SAVE "RUS"CODE 64256,768
```

### ***Изготовление спрайтов с помощью Art Studio***

Мы познакомились с графическим редактором Art Studio, когда делали картинку-заставку, а теперь попробуем использовать его для создания спрайтов со сложным рисунком. При этом можно применять уже знакомые средства: рисование кистью, пером или распылителем, закрашивание замкнутых областей спрайтов всевозможными фактурами, и все это - с помощью богатой палитры цветов. Но в редакторе есть и еще несколько режимов, о которых мы раньше не говорили, а ведь они дают прекрасные результаты при создании спрайтов.

Изучив новые возможности редактора, мы вначале познакомим вас с наиболее

простым способом кодировки изображения, который назовем методом «окон», а затем расскажем о другом способе, более сложном, но позволяющем более экономно расходовать память.

### ***Редактирование увеличенного изображения***

Если вы уже пробовали создавать спрайты в генераторе спрайтов, то могли заметить, насколько проще переносить изображение с бумаги на экран и насколько точнее получается рисунок при использовании увеличенного масштаба. Редактор Art Studio также позволяет работать с увеличенным изображением, о чем, в основном, и пойдет разговор в этом разделе.

Загрузите Art Studio в компьютер и после появления главного меню установите стрелку на опцию MAGNIFY (редактирование увеличенного изображения). Art Studio дает возможность при тонком редактировании пользоваться одним из трех масштабов. Вы можете увеличить изображение в два, в четыре или в восемь раз. После нажатия клавиши M и появления списка функций выберите, например, MAGNIFYx8 - редактирование деталей изображения с увеличением в 8 раз). В результате этого курсор-стрелка заменится «увеличительным стеклом». Переместите его в нужное место и снова нажмите M - весь экран покроется сеткой, в которой каждая клеточка соответствует одному пикселю. Слева и сверху от рабочего поля будут находиться датчики положения увеличенного фрагмента по отношению к полному экрану (светлые прямоугольники); по краям этих датчиков - изображения стрелок, позволяющих передвигать лупу по экрану, не возвращаясь в основное меню. Пиктограмма с двумя прямоугольниками (слева вверху) дает возможность переместить лупу в верхний левый угол экрана.

Над рабочим полем появится новое меню, в котором вы можете, не выходя из режима, изменить масштаб увеличения (x2, x4 и x8), выбрать один из трех способов редактирования (SET - ставить точки,

RESET - стирать точки и TOGGLE - инвертировать точки). Опция ATTRS. полностью повторяет одноименный пункт в главном меню, а для выхода из режима увеличения используется слово MENU в правом верхнем углу экрана.

Чтобы получить первые навыки работы с лупой, изобразите в средней части экрана прямоугольник. Лучше всего начинать рисовать, установив режим TOGGLE (если он еще не установлен). Тогда вы сможете не только закрашивать «пустые» точки, но и стирать лишние или неверно поставленные. Техника рисования такая же, как и при использовании пера: нажмите клавишу M и, не отпуская ее, переместите перекрестье курсора в другую точку. Перемещаясь, курсор будет оставлять за собой след. После того как прямоугольник будет готов, установите стрелку на слово Menu в правом верхнем углу экрана и нажмите M. Перед вами снова появится весь экран, и на нем - маленький черный прямоугольник, тот самый, который вы только что изобразили.

Проделайте еще один опыт, но уже с цветовой палитрой. Опять войдите в режим восьмикратного увеличения, и теперь в функции ATTRS. меню подберите для INK желтый цвет, а для PAPER - красный. Попутно можете изменить и цвет бордюра. В любой части экрана нарисуйте новый прямоугольник, и вы увидите интересную картину: как только в клетке будет поставлена желтая точка, вокруг нее сразу же будет возникать фон красного цвета. Доведите прямоугольник до конца и вернитесь в главное меню. Теперь на экране появится желтый прямоугольник в окружении красного фона. Попробуйте еще несколько комбинаций для INK и PAPER, после чего можно очистить экран. Для этого в пункте ATTRS. восстановите стандартные атрибуты (строка STANDARD), а затем выберите пункт MISC. главного меню и выполните функцию CLEAR SCREEN -

экран очистится для новых упражнений.

Дадим еще несколько полезных советов перед тем, как вы начнете создание полноценного спрайта для своей игры. Работать будет намного удобнее, если на экране дополнительно выделить клетки размером 8x8 пикселей, то есть разбить поле экрана на знакоместа. С этой целью в главном меню необходимо выбрать пункт MISC. и установить режим BRIGHT GRID 1. Весь экран превратится в «шахматную доску», клетки которой отличаются друг от друга яркостью. Теперь снова можно переходить в режим MAGNIFY и создавать самые лучшие в мире спрайты.

Нарисуйте в увеличенном масштабе несколько самолетиков разных типов, ракету, танк, попробуйте свои силы в изображении взрыва. Как только очередной спрайт будет закончен, выбирайте пункт MENU, и сразу станет видно, как выглядит ваше детище в реальном масштабе. Может оказаться, что вы замахнулись на такую картинку, увеличенное изображение которой не помещается в окне. Не отчаивайтесь, вспомните о стрелках, которые дают возможность прокручивать экран во всех четырех направлениях. Когда же перемещение не помогает, помните, что в крайнем случае всегда можно изменить масштаб увеличения.

Если у вас стало рябить в глазах от мелкой сетки, то от нее легко избавиться. Для этого в опции MAGNIFY установите стрелку на позицию GRID и нажмите M. «Галочка» справа от слова GRID заменится крестиком, что говорит об удалении сетки в режиме «увеличительного стекла». Если нажать M еще раз - «галочка» и сетка вернутся на свое место. Чтобы убрать с экрана «шахматную доску», необходимо в пункте MISC. главного меню выбрать функцию REMOVE GRID.

И последнее. Рисовать спрайты обычно удобнее в черно-белом изображении. Если нужно их потом раскрасить, можно поступить следующим образом. Выберите режим редактирования TOGGLE, а затем подберите в пункте ATTRS. подходящие цвета и переместите курсор на то знакоместо, которое нужно раскрасить («шахматную» сетку при этом лучше установить). Теперь дважды нажмите клавишу M. Два раза нужно нажимать для того, чтобы рисунок не изменялся - при первом нажатии точка ставится (или удаляется), а при повторном нажатии - стирается (или появляется вновь). Аналогично раскрасьте и все остальные знакоместа спрайта.

#### ***Кодирование спрайта методом «окон» и запись его на ленту***

Итак, созданный вами спрайт уютно расположился на экране программы Art Studio и теперь хотелось бы превратить это изображение в блок данных, а затем записать его на ленту. Как это сделать?

В главном меню выберите пункт WINDOWS (работа с окнами). Как только вы нажмете клавишу M, появится новое меню, едва помещающееся на экране - так много в нем разных функций. С их помощью можно задавать окна произвольного размера, а со спрайтом, попавшим в окно, делать множество потрясающих вещей. Например, копировать его в разные места экрана (CUT&PASTE WINDOW), копировать с изменением масштаба (RE-SCALE WINDOW), зеркально отображать его относительно горизонтальной или вертикальной оси (FLIP HORIZONTAL и FLIP VERTICAL), поворачивать на 90, 180 и 270 градусов (ROTATE 1/4, 1/2, 3/4) и многое другое, о чем подробно можно узнать в книге [1] или кратко - в Приложении 3. Чтобы по достоинству оценить меню WINDOWS, попробуйте поэкспериментировать хотя бы с теми функциями, о которых мы упомянули выше. Вернемся, однако, к кодированию спрайта.

Установите стрелку на функцию DEFINE WINDOW (задание окна произвольного размера) и нажмите M. Стрелку заменит маленький квадратик.

Переместите его в левый верхний угол спрайта и нажмите M, после чего двигайте квадратик вниз и вправо - вы увидите, как за ним потянется прямоугольник из пунктирных линий. Спрайт должен полностью поместиться в этот прямоугольник, причем нужно постараться, чтобы границы спрайта и прямоугольника полностью совпали. Снова нажмите M - окно задано. Следующий шаг - установите курсор на пункт TEXT главного меню (работа с текстами). Здесь тоже обширное меню, но нас будет интересовать только последняя его строка - FONT EDITOR (редактор набора символов). FONT EDITOR позволяет редактировать и создавать новые символьные наборы, которые можно использовать затем в своих программах, о чем мы упоминали в разделе «Русификация программ». После перехода в редактор появляется новый экран, в нижней части которого размещаются все 96 возможных символов. Символ, отмеченный указателем, и два соседних с ним отображаются в центре экрана, увеличенные в восемь раз (так же, как и в режиме MAGNIFYx8). Их можно изменить. Для этого поставьте курсор на тот пиксель, который вы хотите включить или выключить и нажмите M. Нужные вам буква или знак из набора тоже выбираются с помощью стрелки и клавиши M.

Итак, выберем стрелкой букву A, начиная с которой будут кодироваться части изображения на экране, и перейдем к функции MISC. в меню FONT EDITOR. Оно содержит всего две строки, из которых нам потребуется CAPTURE FONT (копирование символов с экрана). Нажимаем клавишу M - спрайт закодировался. Теперь всем буквам, начиная с A, соответствуют графические символы, представляющие части вашего спрайта. Если вам что-то не понравилось в спрайте и вы решили его доработать, прежний набор символов легко восстановить, выбрав в режиме MISC. функцию COPY ROM (загрузка стандартного набора символов из ПЗУ). Поскольку все 96 символов набора можно использовать как графические, то можно получить довольно большие спрайты с размерами 6 x 16, 8.x 12 и т. д. Примеры спрайтов приводятся на рис. 3.6.



Для записи кодов, содержащих новый набор символов, необходимо, находясь в редакторе символов FONT EDITOR, выбрать пункт FILE (запись, чтение и проверка файлов), а в нем - функцию SAVE FILE... и нажать M. После этого на экране появится запрос:

Filename? (имя файла?)

Введите какое-нибудь имя, например, SYMBOL и нажмите клавишу Enter. Сразу же появится сообщение:

Start tape, then press any key

после чего необходимо включить магнитофон и нажать любую клавишу. Начнется запись блока данных нового набора символов длиной 768 байт. После ее

окончания можно проверить запись, выбрав в том же меню опцию VERIFY FILE.

### **Использование полученного спрайт-файла**

Теперь покажем, как использовать полученный кодовый блок в игровой программе. Сначала запишите на ленту программу-загрузчик:

```
10 CLEAR 64255
20 LOAD "SYMBOL"CODE 64256
30 LOAD ""
```

10 - удаляются все переменные и массивы, очищается экран, а также изменяется адрес, определяющий верхнюю границу области бейсик-программы. Тем самым выделяется место в памяти для загружаемых кодов;

20 - чтение кодов новых символов, содержащихся в файле SYMBOL, и загрузка их по адресу 64256 в память компьютера;

40 - загрузка игровой бейсик-программы, в которой будет использован созданный вами спрайт (спрайты).

А вот как можно использовать новый набор символов в самой бейсик-программе. Выведем на экран, например, спрайт 1 («катер»), изображенный на рис. 3.6.

```
100 POKE 23607,250
110 PRINT " !""#$%&`"
120 PRINT "( )*, - . /"
130 PRINT "01234567"
140 POKE 23607,60
```

100 - текущим устанавливается новый набор символов SYMBOL, который вы загрузили с ленты;

110...130 - вывод на экран закодированного изображения спрайта размером 8x3 знакоместа. Конечно, в вашей программе спрайты могут быть и других размеров, тогда изменятся и строки программы. Здесь важно то, что, в отличие от предыдущих примеров, буквы вводятся уже не в режиме курсора [G], а в обычном режиме. Хочется также обратить ваше внимание на способ печати оператором PRINT символа кавычек: их нужно вводить в тексте два раза подряд. А одиночные кавычки, как вы знаете, служат для ограничения печатного текста;

140 - выключение нового набора символов, после чего текущим опять становится стандартный набор символов компьютера.

### **Создание спрайтов с учетом их конфигурации**

При создании спрайтов достаточно сложной формы метод «окон» может оказаться весьма расточительным. Ведь окно в этом случае будет охватывать не только «полезное» пространство, но и некоторые пустые знакоместа, где изображение отсутствует. Поэтому иногда может оказаться полезным другой способ создания и кодировки спрайтов. Собственно, таким способом мы уже пользовались, когда делали спрайты самолетов (рис. 3.2 и 3.4), теперь же рассмотрим его более подробно и с применением редактора Art Studio.

После того, как спрайт будет нарисован на бумаге В клетку и разбит на знакоместа, выделим только те фрагменты, в которых есть включенные пиксели, и отбросим все пустые (рис. 3.7, а). Затем выделенным знакоместам поставим в соответствие различные символы (рис. 3.7, б). Закончив подготовительную работу, загрузим Art Studio и войдем в режим FONT EDITOR

Теперь предстоит довольно утомительная работа - перенос одного за другим кусочков «мозаики» на место выбранных символов. Для этого установите стрелку на букву или знак в нижней части экрана, с которого начнется кодирование, и после нажатия М этот символ и два соседних появятся в центре в восьмикратном увеличении, причем каждая клеточка будет соответствовать одному пикселю. Для очистки старого символа можете воспользоваться функцией CLEAR пункта

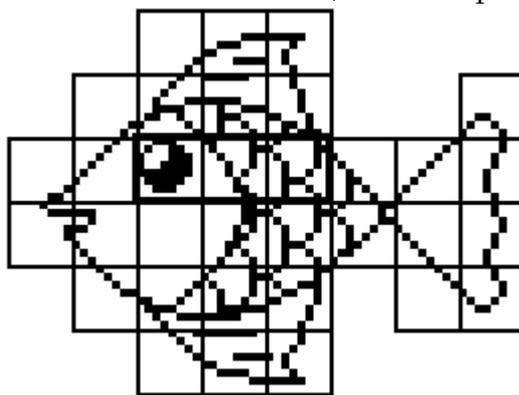
CHARACTER, а затем на чистом месте начинайте ставить точки в соответствии с вашим эскизом. Рисуйте так же, как и в режиме «увеличительного стекла» - наводите стрелку на пиксели, которые вы хотите включить или выключить, и нажимайте М. Покончив с одним символом, переходите к следующему. Таким способом можно закодировать один большой или несколько маленьких спрайтов; лишь бы общее число символов не превышало 96.

Итак, если сравнить этот метод с методом «окон», который мы рассмотрели ранее, то можно обнаружить, что способ создания спрайтов с учетом их конфигурации значительно экономнее с точки зрения использования памяти. С другой стороны, метод «окон» заметно проще - не нужно возиться с бумагой, линейкой, запоминать символы и т. д. Ну, а какой из этих двух способов выбрать, зависит от конкретной ситуации и вашего вкуса.

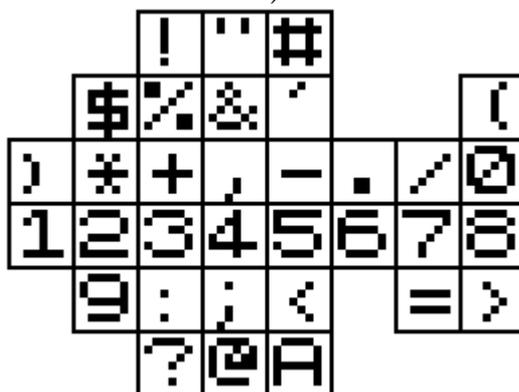
### ***Построение пейзажей***

Играя на компьютере в разнообразные игры, вы наверняка заметили, что, кроме спрайтов («своих» и «чужих»), игровое пространство заполнено объектами, о которых мы подробно еще не говорили. Речь идет о тех объектах, которые составляют фон игры и, будучи объединенными вместе, носят название пейзажа. Напомним, что ранее мы отнесли к пейзажам все неподвижные элементы изображения, такие, например, как небо, земля, вода, здания, деревья, мосты и т. д. Впрочем, неподвижность элементов относительна, во многих играх пейзаж все же движется (например, R-Туре) или немного меняется, как мы увидим ниже.

Назначение пейзажа различается от игры к игре. В одних программах (International Carate, Fist+) он играет чисто декоративную роль, демонстрируя безграничные возможности компьютерной графики. В других (Chess, Tetris, Saboteur) является неотъемлемым ее элементом, без которого



а)



б)

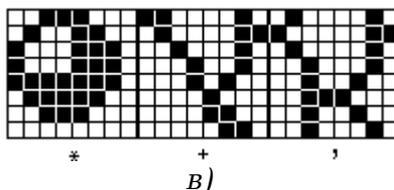


Рис.3.7. Подготовка спрайта к кодированию.

игра потеряла бы всякий смысл. Действительно, как можно передвигать фигуры в шахматах или шашках без доски, которая в данном случае составляет основу пейзажа! Таким образом, под пейзажем в компьютерных играх мы будем понимать нечто большее, чем «вид из окна», отнеся сюда еще и всевозможные лабиринты, стаканы, разрезы зданий, пульта управления самолетами и прочие неподвижные изображения. Теперь в качестве примеров рассмотрим несколько пейзажей, на основе которых вы сможете создавать свои, будем надеяться, более сложные и занимательные.

В простейшем случае пейзаж - это разделенный на две половинки экран, верхняя часть которого, синяя, имитирует небо, а нижняя, зеленая, соответствует поверхности земли. Без такого пейзажа не обойтись, когда создаются игры по сюжетам воздушных сражений. Программа состоит всего из четырех строк:

```
10 BORDER 0: PAPER 1: CLS
20 FOR N=14 TO 21
30 PRINT AT N,0; PAPER 4;TAB 31;" "
40 NEXT N
```

10 - устанавливается черный бордюр, и весь рабочий экран окрашивается в синий цвет;

20 - начало цикла. Переменная N соответствует номеру строки экрана;

30 - тело цикла, осуществляющего быстрое окрашивание строк экрана с 14 по 21 в зеленый цвет. Достигается это печатью в каждой строке 31 пробела плюс еще одного цветом PAPER. Если то же самое выполнить с помощью операторов PLOT и DRAW, то окрашивание будет происходить чрезвычайно медленно;

40 - конец цикла.

Следующая программа создает пейзаж в буквальном смысле этого слова. После ее ввода в компьютер и запуска, на экране вы увидите сверху голубое небо, снизу - синюю поверхность кристально чистой воды, а между ними хаотическое нагромождение гор (рис. 3.8).



Рис. 3.8. Горная гряда.

Программа 12. ГОРНАЯ ГРЯДА.

```
10 LET h=60
20 LET dir=1
30 BORDER 0: PAPER 5: INK 0: CLS
100 FOR n=0 TO 255
110 PLOT n,0: DRAW 0,h
120 IF RND>0.8 THEN LET dir=-dir
121 IF h>80 THEN LET dir=-1
```



небо с ярко вспыхивающими и гаснущими звездами, который вы можете использовать в какой-нибудь космической игре. Такой пейзаж, кстати, будет неплохо смотреться и в заставке. Для того, чтобы звездочка вспыхнула, программа должна сформировать и вывести один за другим 4 спрайта (рис. 3.9), а для того, чтобы погасла (причем не сразу, а постепенно), необходимо вывести эти же спрайты, но уже в обратном порядке. Рассмотрим программу, которая проделывает все эти действия.

#### Программа 13. ЗВЕЗДНОЕ НЕБО.

```

10 FOR n=0 TO 3
20 FOR i=0 TO 7
30 READ a: POKE (USR "a"+i+n*8),a
40 NEXT i
50 NEXT n
60 REM -----
100 INK 6: PAPER 0: BRIGHT 1: BORDER 0: CLS
110 LET star=30+INT (RND*20): DIM a(2,star)
120 FOR n=1 TO star
130 LET a(1,n)=INT (RND*22)
140 LET a(2,n)=INT (RND*32)
150 NEXT n
160 REM -----
200 FOR n=1 TO star
210 PRINT AT a(1,n),a(2,n); "A"
220 NEXT n
250 REM -----
300 FOR n=1 TO RND*20: PAUSE 3: NEXT n
310 LET z=INT (RND*star)+1
320 FOR n=0 TO 3
330 PRINT AT a(1,z),a(2,z);CHR$(CODE "A"+n)
340 PAUSE 3
350 NEXT n
360 FOR n=2 TO 0 STEP -1
370 PRINT AT a(1,z),a(2,z);CHR$(CODE "A"+n)
380 PAUSE 3
390 NEXT n
400 GO TO 300
1000 DATA 0,0,0,0,16,0,0,0
1010 DATA 0,0,0,16,56,16,0,0
1020 DATA 0,0,40,84,56,84,40,0
1030 DATA 0,16,84,56,254,56,84,16

```

10...50 - запись в символы, определяемые пользователем, изображений звезд. При этом переменная n «перебирает» строки с операторами DATA, а переменная i - числа в каждой строке DATA;

100 - установка атрибутов экрана;

110 - задание случайного количества звезд, но в пределах от 30 до 50. Резервирование памяти под массив a, в который будут заноситься координаты этих звезд;

120...150 - в цикле задаются случайные координаты положения звезд, причем отдельно для y (строка 130) и x (строка 140);

200...220 - вывод на экран звездочек, каждая из которых выглядит просто как точка на экране, то есть первую фазу на рис. 3.9;

300 - пауза случайной длительности;

310 - вычисляется случайный номер звезды, той самой, которая будет вспыхивать, а затем гаснуть;

320...350 - вывод одного за другим четырех спрайтов, причем в одно и то же

место экрана, что создает эффект вспышки;

360...390 - вывод одного за другим трех спрайтов в то же самое место экрана, но уже в обратном порядке. В результате звезда как бы гаснет;

400 - безусловный переход на строку 300, и теперь уже другая звездочка вспыхнет и погаснет;

1000...1030 - блок данных, который содержит информацию о четырех спрайтах, создающих на экране мультипликационный эффект.

О том, как сделать картинки более сложными и занимательными, мы расскажем в главе, посвященной Laser Basic.

Теперь рассмотрим метод, а вместе с ним и программу, которые помогут вам создавать очень сложные пейзажи, не уступающие по качеству фирменным. Коротко суть метода состоит в следующем. Вначале на листе бумаги, расчерченном под экран, выполняется эскиз будущей картинки, например, разрез здания, лабиринт, место будущего сражения и т. д. При этом желательно, чтобы элементы картинки как можно чаще повторялись (это могут быть кирпичи, бревна, деревья, камни). Затем каждый из элементов подвергается кодировке, например, с помощью программы «Генератор спрайтов», а полученные коды выписываются. Как использовать эти коды, лучше всего посмотреть на примере конкретной программы, которая создает пейзаж к игре с условным названием «Поиски клада» (рис. 3.10.).

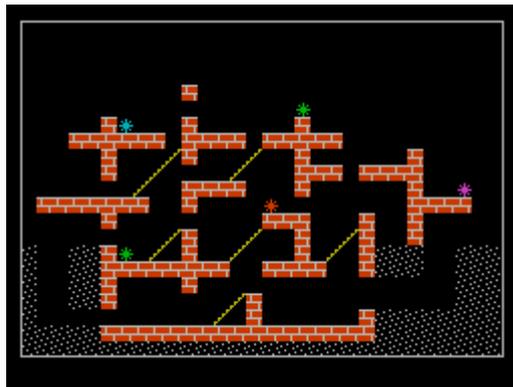


Рис. 3.10. Пейзаж «Дом с алмазами».

Пейзаж сделан таким образом, что в нем присутствует всего четыре элемента, каждый из которых занимает одно знакоместо. Это кирпич, фрагмент лестницы, грунт и алмаз. Конечно же, элементы могут быть и более сложными и занимать 2, 3, 6 знакомест, но тогда программу в части операторов DATA следует писать более внимательно, чтобы отдельные детали одного элемента не оказались в разных местах экрана. Перейдем теперь к рассмотрению программы и ее особенностей.

Программа 14. ПЕЙЗАЖ «ДОМ С АЛМАЗАМИ».

```

10 BORDER 0: PAPER 0: INK 7: CLS
15 REM *** Graphics ***
20 FOR n=0 TO 31
30 READ s
40 POKE USR "A"+n,s
50 NEXT n
60 REM *** Start ***
65 DIM c(20): DIM v(20): DIM h(20): LET m=1
70 FOR y=1 TO 20
80 FOR x=1 TO 30
90 READ s
100 IF s=0 THEN GO TO 150
110 IF s=1 THEN PRINT AT y,x: PAPER 2: INK 7;"A": GO TO 150
120 IF s=2 THEN PRINT AT y,x;"B": GO TO 150
130 IF s=3 THEN PRINT AT y,x: INK 6;"C": GO TO 150

```



данной строке, либо соответствует текущим атрибутам. После печати программа переходит на начало цикла;

140 - если в списке DATA записано число 4, то вначале устанавливается случайный номер цвета, затем запоминаются координаты x и y, где должен располагаться алмаз, и, наконец, на экране появляется его изображение;

150, 160 - конец циклов;

165 - установка белой рамки, ограничивающей размеры игрового поля;

170...196 - в цикле меняются цвета алмазов, они выглядят как драгоценные камни, переливающиеся разными цветами;

200...240 - коды графических символов;

250...350 - данные пейзажа, представленные в цифровой форме.

## 4. ПРИЕМЫ ПРОГРАММИРОВАНИЯ ИГР

Прием программирования - это четкий и эффективный способ выполнения некоторой задачи в программе. Минуты, израсходованные на планирование программы, помогут вам позднее сэкономить часы, которые были бы потеряны на ее отладку. При профессиональной разработке программных средств вначале составляется подробный план программы с учетом самых мелких деталей, а затем, при помощи языка программирования, этот план реализуется в программе. Примерно так же должны будете действовать и вы, когда начнете писать свою собственную игровую программу: перво-наперво придется составить «план действий компьютера» - краткий перечень тех основных задач, которые должен выполнить компьютер. Подобный план мы приводили во второй главе, описывая словами последовательность выполняемых компьютером операций.

Но существует еще и другой способ, который, возможно, покажется вам более удобным. Метод, о котором мы хотим рассказать, заключается в составлении так называемой блок-схемы. Она строится по очень простым правилам и представляет собой графическое изображение последовательности основных шагов выполнения программы. После составления такой схемы значительно проще писать и саму программу.

### *Составление блок-схемы программы*

При разработке программы очень важно, чтобы ее отдельные части следовали в правильном порядке. Например, еще до использования переменной необходимо присвоить ей какое-нибудь значение, а прежде чем напечатать какие-либо данные, их нужно каким-либо образом получить, например, ввести с клавиатуры. Для построения блок-схемы используют несколько видов геометрических фигур, причем каждой фигуре соответствуют вполне определенные типы действий. Чаще всего встречаются следующие элементы (рис. 4.1):



рис. 4.1, Элементы блок-схемы.

Овалы с надписями НАЧАЛО и КОНЕЦ показывают, где начинается и где заканчивается программа.

Параллелограммом обозначаются действия, связанные с вводом или выводом информации. Данные, поступающие в программу с клавиатуры, от джойстика или от магнитофона, называются входными. Выходными являются данные, идущие от компьютера. Они могут выводиться на экран с помощью оператора PRINT, записываться на магнитную ленту или распечатываться на бумаге.

Прямоугольник ПРОЦЕСС используется для обозначения операций внутренней обработки, производимой компьютером. Это может быть, например, присвоение переменной значения оператором LET или любая достаточно сложная задача, для решения которой необходимо использовать несколько операторов.

Мы уже говорили об операторе IF...THEN, который дает компьютеру задание

проанализировать некоторое условие, например, является ли истинным выражение  $S = 5$ . Дальнейшие действия компьютера зависят от того, выполняется или не выполняется конкретное условие. Возможные различные последовательности действий называются ветвями программы. Для обозначения оператора IF...THEN на блок-схемах используются ромбы, называемые блоками проверки условий. Вершины ромба соответствуют различным решениям, которые может принимать машина в зависимости от заданного условия.

Рассмотрим простой пример. Пусть компьютер генерирует 20 случайных чисел, каждое в пределах от 1 до 10. Задача программы - сосчитать количество чисел, превышающих пятерку. На рис. 4.2 показано, как выглядит блок-схема программы.

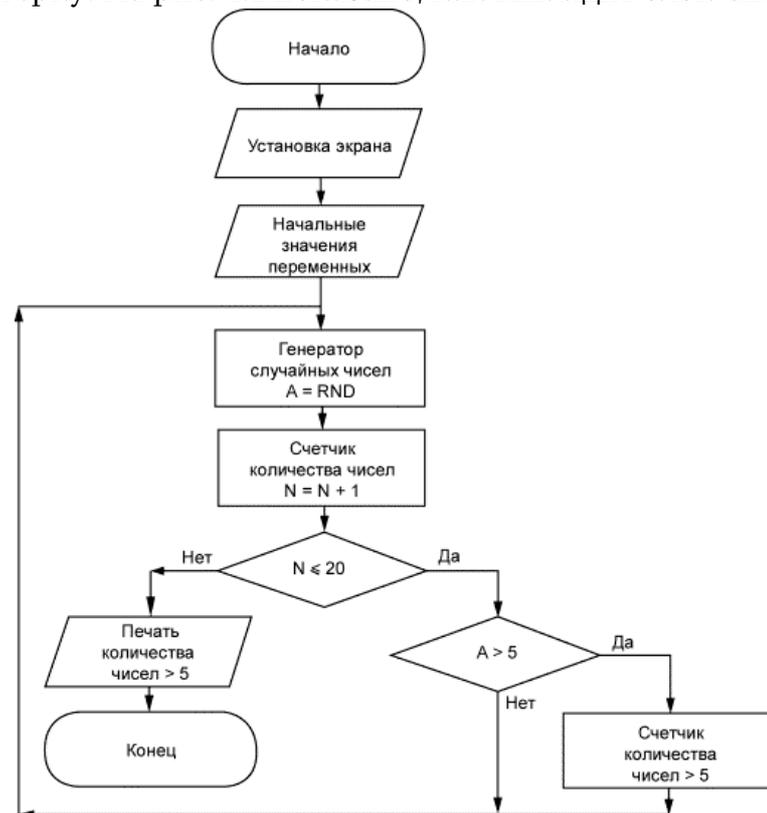


Рис. 4.2. Блок-схема сортировки случайных чисел.

Как видим, схема содержит блоки НАЧАЛО и КОНЕЦ, ряд прямоугольников, в которых выполняются операции, связанные с решением поставленной задачи, блоки проверки условий и печать результата, то есть все основные элементы, которые мы перечисляли. Заметим, что здесь необходимы два блока проверки условий: в одном из них проверяется, не превышает ли общее количество чисел 20, а в другом выносится суждение: «Полученное случайное число больше пяти?». Если ответ на вопрос, записанный в блоке, положительный (Да), то выполнение программы идет одним путем, а если отрицательный (Нет), то другим. Стрелки показывают последовательность решения задачи.

Далее приведена соответствующая этой блок-схеме программа.

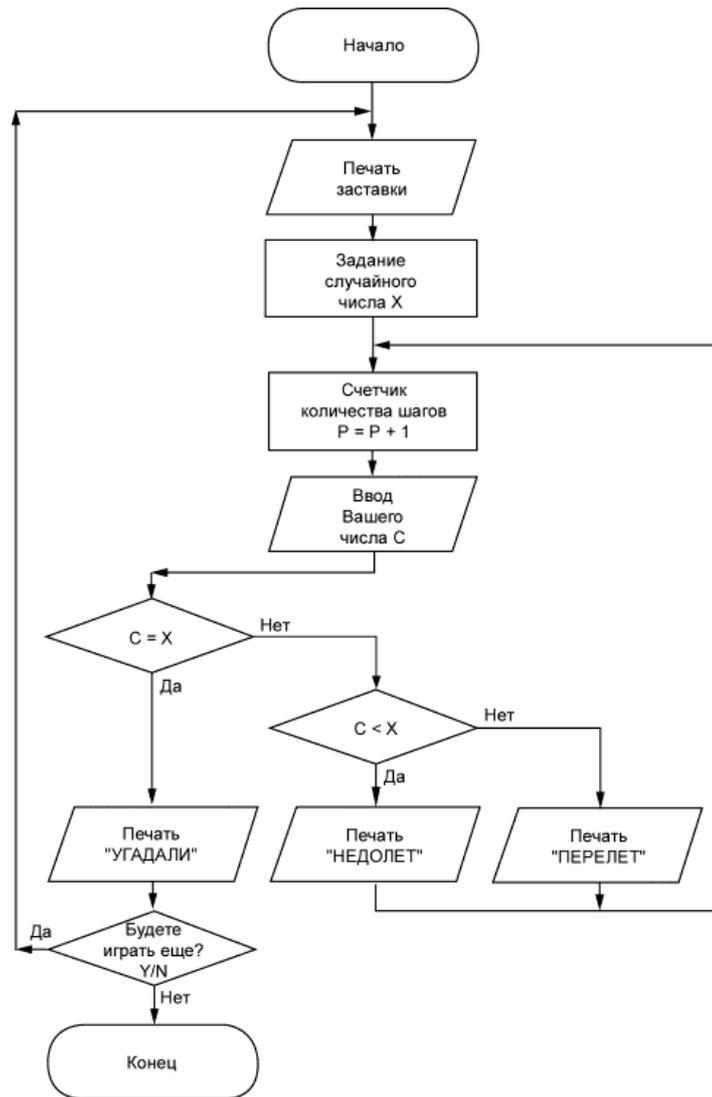


Рис. 4.3. Блок-схема игры ЧИСЛА.

Программа 15. RND.

```

10 BORDER 1: PAPER 1: INK 6
20 CLS
30 LET n=0: LET s=0
40 LET a=RND*10
50 LET n=n+1
60 IF n<=20 THEN GO TO 100
70 PRINT s
80 STOP
100 IF a>5 THEN LET s=s+1
110 GO TO 40
  
```

После того, как мы составили блок-схему, особых пояснений к программе уже не требуется.

Попробуем теперь составить блок-схему, а затем и саму программу для игры ЧИСЛА. Суть игры проста и состоит в следующем. Компьютер генерирует целое число в пределах от 1000 до 9999, которое вам неизвестно. Задача состоит в том, чтобы за минимальное количество шагов угадать это число. При этом программа на каждом шаге должна сравнивать ваше число с числом, «задуманным» компьютером и сообщать: меньше, больше или равно введенное вами число тому, что находится в памяти компьютера. На основании сюжета игры можно предложить такой вариант блок-схемы (рис. 4.3):

При положительном исходе игры должна быть предусмотрена возможность



кавычки вам мешают, их можно

убрать, включив в оператор INPUT непосредственно перед переменной ключевое слово LINE:

```
INPUT TAB 5; LINE a$
```

В этом случае курсор появится уже без кавычек, как и при вводе чисел.

Теперь прокомментируем остальные строки программы.

10...45 - вывод на экран названия игры и информации о диапазоне «задуманного» компьютером числа;

50 - переменной X присваивается случайное значение, которое нужно будет угадать;

60 - установка промежуточных переменных;

110 - счетчик числа шагов;

120 - с помощью оператора INPUT переменной С присваивается значение, введенное с клавиатуры;

130 - после полного заполнения экрана ответами, он очищается подпрограммой, расположенной со строки 600, освобождая место для последующих ответов;

140 - отображение на экране введенного вами числа;

150...170 - эти строки соответствуют двум блокам проверки условий;

200...220 - блок выдачи результата, если ваше число меньше «задуманного»;

300...320 - блок выдачи результата, если ваше число превышает «задуманное»;

400...440 - блок выдачи результата, если вы угадали число;

450...480 - проверка условия: желаете вы начать игру снова или хотите ее закончить;

600...640 - подпрограмма очистки части экрана, занятой ответами.

### ***Модульное построение программы***

Нетрудно убедиться, что при разработке сложных сюжетов игр и соответствующих программ их блок-схемы также усложняются. Если такая схема будет содержать слишком много блоков и ветвей, то ее проверка окажется затруднительной: невозможно будет выяснить, все ли блоки учтены. Поэтому при составлении сложных программ чаще всего используют модульный принцип их построения (см. [7]).

Модулем называется самостоятельная часть программы, обеспечивающая решение какой-нибудь определенной части задачи. Например, в программе могут существовать модули ввода данных, печати результата, очистки экрана (как в предыдущей программе), модуль заставки, отдельных картин игрового пространства и т. д. Программа, разделяемая на модули при разработке блок-схемы и на следующих стадиях проектирования, имеет модульную структуру (рис. 4.4). Про такую программу можно сказать, что она строится по принципу «сверху вниз», потому что ее создание начинается с разработки укрупненного модуля (определяющего главную задачу) и далее «вниз», к модулям, которые входят в укрупненный как составные части. Модульное проектирование позволяет



Рис. 4.4. Модульная структура программы

разделить сложную программу на фрагменты, для каждого из которых проще разрабатывать свою блок-схему, а затем соответствующую этой блок-схеме подпрограмму.

Для того, чтобы почувствовать эффективность такого подхода, рассмотрим развлекательную программу ЖИЗНЬ.

На экране вами создается некоторая первичная колония существ, располагающихся на поле из клеточек (рис. 4.5). Существа могут как размножаться, так и умирать, подчиняясь при этом двум законам. Первый: если вокруг пустой клетки есть ровно три особи (считая по горизонтали, вертикали и диагоналям), то в этой клетке появляется новая особь. Она переходит в следующее поколение, когда в ближайших

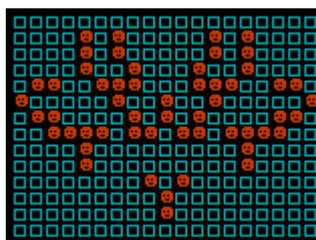


Рис. 4.5. Игровое пространство программы ЖИЗНЬ.

клетках вокруг нее есть 2 или 3 другие особи. В противном случае она умирает: если рядом есть только один организм, то этого недостаточно для воспроизводства, а 4, 5 и более создают перенаселение. С помощью такой программы можно наблюдать развитие своеобразной «компьютерной жизни». Это развитие зависит от формы заданной вами исходной колонии (популяции) и количества особей в ней.

Если взглянуть на текст программы, воплощающей эту идею, то может показаться, что она слишком велика и разобраться в ней почти невозможно. Однако программа состоит из семи модулей, которые реализованы в виде самостоятельных подпрограмм, что значительно упрощает понимание. Примечательно, что модули используются в основной программе 16 раз, в результате чего вся она становится вполне обозримой, существенно экономится память компьютера и место на магнитной ленте, куда эта программа записывается. Итак, перейдем к ее рассмотрению.

Программа 17. ЖИЗНЬ.

```

10 GO SUB 2000
20 GO SUB 1000
30 GO SUB 3000
40 BEEP .07,0
50 GO SUB 1000
60 DIM A$(30,20)
70 DIM B$(30,20)
  
```

```
80 LET S=0
90 PRINT AT 10,5; INK 5;"STANDARD COLONIA (Y/N)";INK 2;"?"
100 LET K$=INKEY$
110 IF K$="Y" OR K$="y" THEN BEEP .07,5: GO TO 200
120 IF K$="N" OR K$="n" THEN BEEP .07,5: GO TO 400
130 PAUSE 5
140 PRINT AT 10,27;" "
150 PAUSE 5: GO TO 90
200 REM -- standard---
210 LET A$(14,9)="1"
220 LET A$(14,10)="1"
230 LET A$(14,11)="1"
240 LET A$(15,11)="1"
250 LET A$(16,11)="1"
260 LET A$(16,10)="1"
270 LET A$(16,9)="1"
280 LET xmin=13: LET xmax=17
290 LET ymin=8: LET ymax=12
300 LET K=7
310 GO TO 600
400 REM -- NO standard ---
410 GO SUB 1000
420 INK 5
430 PRINT AT 21,0; INK 4;"KOL-WO OSOBEY:"
440 INPUT K
450 FOR I=1 TO K
460 PRINT AT 21,0; INK 6;" X,Y ";I;" OSOBI: "
470 INPUT INK 3;"X=";X, INK 3;"Y=";Y
480 IF X<2 OR X>29 OR Y<2 OR Y>19 THEN GO TO 470
490 LET A$(X,Y)="1"
500 IF I=1 THEN LET xmin=X-1: LET xmax=X+1: LET ymin=Y-1:LET ymax=Y+1
510 GO SUB 6000
520 NEXT I
600 GO SUB 1000
610 LET xmin1=xmin: LET xmax1=xmax
620 LET ymin1=ymin: LET ymax1=ymax
630 LET S=S+1
640 PRINT AT 0,6;"STEP:";S;" ПППKOL-WO:";K;" ППП"
650 BEEP .1,25
660 FOR Y=Ymin1 TO Ymax1
670 FOR X=Xmin1 TO Xmax1
680 IF A$(X,Y)=" " THEN PRINT AT Y,X; INK 5;"C":GO TO 710
690 PRINT AT Y,X; INK 2;"B"
700 GO SUB 6000: IF INKEY$<>" " THEN GO SUB 7000: GO TO 20
710 NEXT X
720 NEXT Y
730 IF K=0 THEN PAUSE 100: GO SUB 7000: GO TO 20
740 LET K=0
750 FOR Y=Ymin TO Ymax
760 FOR X=Xmin TO Xmax
770 LET J=0
780 IF INKEY$<>" " THEN GO SUB 7000: GO TO 20
790 FOR N=-1 TO 1
800 IF X+N<1 OR X+N>30 THEN GO TO 830
810 IF Y>1 THEN IF A$(X+N,Y-1)="1" THEN LET J=J+1
820 IF Y<20 THEN IF A$(X+N,Y+1)="1" THEN LET J=J+1
830 NEXT N
840 IF Y<1 OR Y>20 THEN GO TO 870
850 IF X>1 THEN IF A$(X-1,Y)="1" THEN LET J=J+1
860 IF X<30 THEN IF A$(X+1,Y)="1" THEN LET J=J+1
```

```

870 IF J=3 THEN LET K=K+1: LET B$(X,Y)="1": GO TO 900
880 IF J>1 AND J<4 AND A$(X,Y)="1" THEN LET K=K+1:LET B$(X,Y)="1": GO TO 900
890 LET B$(X,Y)=" "
900 NEXT X
910 NEXT Y
920 FOR Y=ymin TO ymax
930 FOR X=xmin TO xmax
940 LET A$(X,Y)=B$(X,Y)
950 NEXT X
960 NEXT Y
970 GO TO 610
1000 REM -----C L S-----
1010 BORDER 0: PAPER 0: INK 6
1020 FLASH 0: OVER 0: BRIGHT 0
1030 CLS
1040 RETURN
2000 REM -----Graphics-----
2010 FOR I=1 TO 3: READ S$
2020 FOR N=0 TO 7: READ S
2030 POKE USR S$+N,S
2040 NEXT N: NEXT I
2050 RETURN
2060 DATA "A",0,28,38,79,95,127,62,28
2070 DATA "B",0,60,126,86,126,102,60,0
2080 DATA "C",0,0,252,132,132,132,132,252
3000 REM -----SCREEN-----
3010 LET X=1: LET C=1
3020 FOR Y=1 TO 12
3030 PRINT AT Y-1,X;"oooooooooooooooooooooooooooooooooooo": REM 32 Space
3040 GO SUB 3150
3050 IF INKEY$<>"" THEN RETURN
3060 BEEP .01,20: PAUSE 5
3070 NEXT Y
3080 FOR Y=11 TO 6 STEP -1
3090 GO SUB 3150
3100 IF INKEY$<>"" THEN RETURN
3110 PRINT AT Y+10,x;"oooooooooooooooooooooooooooooooooooo": REM 32 Space
3120 BEEP .01,20: PAUSE 5
3130 NEXT Y
3140 GO TO 4000
3150 PRINT AT Y,X; INK C;
"oooooooooAAAAooAAAAAAooAAAAAAo
ooooooooooAAoooooooooooooooo
AAoooooooooAAoooooooooooooooo
AAoooooooooAAoooooooooooooooo
AAoooooooooAAooooAAAAoAAAAo
AAoooooooooAAooooAAoooooooooooo
AAoooooooooAAooooAAoooooooooooo
AAoooooooooAAooooAAoooooooooooo
AAoooooooooAAooooAAoooooooooooo
AAAAAAooAAAAooooooooooAAAAAA"
3160 RETURN
4000 LET S$="ProgramooLIFE": LET XN=10: LET YN=1: LET C=4: GO SUB 5000
4010 LET S$="Sankt-Peterburg 1993": LET XN=6: LET YN=3: LET C=6: GO SUB 5000
4020 LET S$="Press any key to continue": LET C=7: LET YN=20: LET XN=4:GO SUB 5000
4030 LET Y=6
4040 LET C=C+1: IF C>7 THEN LET C=1
4050 GO SUB 3150
4060 PAUSE 50
4070 IF INKEY$="" THEN GO TO 4030

```

```

4080 RETURN
5000 REM -----
5010 LET S=LEN S$
5020 FOR I=1 TO S
5030 LET L$=S$(I TO I)
5040 PRINT AT YN,XN+I-1; INK C;L$; PAPER 3;" "
5050 BEEP .01,25
5060 NEXT I
5070 PRINT AT YN,XN+S;" "
5080 RETURN
6000 REM -----1-----
6010 IF xmin>X-1 AND xmin>1 THEN LET xmin=X-1
6020 IF xmax<X+1 AND xmax<30 THEN LET xmax=X+1
6030 IF ymin>Y-1 AND ymin>1 THEN LET ymin=Y-1
6040 IF ymax<Y+1 AND ymax<20 THEN LET ymax=Y+1
6050 RETURN
7000 REM ----- end -----
7010 BORDER 1: PAPER 1: CLS
7020 LET z$="oooooooooooooooo": REM 15 Space
7030 FOR n=9 TO 11
7040 PRINT AT n,8; PAPER 4;z$; PAPER 0;(" " AND n>9)
7050 NEXT n
7060 PRINT AT 12,9; PAPER 0;z$
7070 PRINT AT 10,13; PAPER 8; INK 2;"E N D"
7080 FOR N=-20 TO -30 STEP -1
7090 BEEP .1,N
7100 NEXT N
7110 PAUSE 350
7120 BEEP .06,15
7130 RETURN

```

Опишем вначале все подпрограммы, каждая из которых соответствует определенному модулю.

1000...1040 - подпрограмма, которую мы назовем CLS. Она устанавливает цвет бордюра, задает атрибуты экрана, очищает его и окрашивает соответствующим цветом;

2000...2080 - подпрограмма GRAPHICS формирует все объекты игры, их требуется всего три. Первый - шарик, использующийся в заставке. Ему соответствует клавиша А в графическом режиме. Второй - голубой квадратик, составляет основу фона, на котором развивается та или иная популяция. В графическом режиме ему соответствует клавиша С. Наконец, третий - рожица красного цвета, имитирующая особь. В графическом режиме ей соответствует клавиша В;

3000...4080 - подпрограмма SCREEN формирует заставку - слово LIFE, перемещающееся по экрану и составленное из больших, переливающихся разными цветами букв;

3150...3160 - подпрограмма печати слова LIFE, которая расположена внутри подпрограммы SCREEN. Так как слово LIFE появляется на экране многократно, а его печать осуществляется в графическом режиме (см. буквы «А» в строке 3150), то вполне естественно, что эта операция выделена в отдельную подпрограмму;

5000...5080 - подпрограмма «печатающий квадрат». Она выводит на экран различные надписи в заставке после того, как слово LIFE займет свое конечное положение посередине экрана;

6000...6050 - подпрограмма MIN/MAX в процессе счета постоянно уточняет границы вновь возникшей колонии. Эта процедура позволяет существенно ускорить работу алгоритма;

7000...7130 - подпрограмма END. Осуществляет печать слова END в рамке, вокруг которой расположена тень, и включает звуковое сопровождение.

Группы операторов, записанные в строках 200...310, 400...520 и 4000...4020, не выделены в подпрограммы потому, что используются только один раз.

Рассмотрим теперь программу в целом и более подробно.

10 - вызов подпрограммы GRAPHICS. Заранее формируются все образы игрового пространства, которые затем по мере надобности будут извлекаться из памяти компьютера;

20 - вызов подпрограммы CLS;

30 - вывод на экран заставки, которая останется на экране до тех пор, пока не будет нажата какая-нибудь клавиша;

50 - очистка экрана;

60, 70 - резервирование памяти под двухмерные символьные массивы. В дальнейшем элементы массивов будут принимать значения «пробел» и «1», причем «пробел» будет означать, что особь в клеточке отсутствует, а «1» - что она здесь имеется. В массиве A\$(30,20) будет размещаться исходная популяция, а в B\$(30,20) - следующая;

80 - задание номера шага;

90...150 - здесь полагается либо выбрать заданную конфигурацию колонии, либо задать ее самому. Строки 140 и 150 необходимы для мерцания на экране символа ? в конце текста, который выводит на экран 90-я строка программы;

210...270 - в массив A\$(30,20) записывается стандартная популяция, которая на экране выглядит, как перевернутая буква «П»;

280, 290 - установка границ всей колонии;

300 - запись в переменную K количества особей;

310 - после выбора стандартной конфигурации осуществляется переход к алгоритму развития популяции;

400...520 - ввод вашего варианта конфигурации колонии посредством задания общего числа особей и ввода их координат. В строке 480 проводится проверка значения координат, которые не должны выходить за пределы игрового поля. Если хотя бы одна из координат не соответствует заданным условиям, ее ввод требуется повторить;

600 - очистка экрана;

610, 620 - запоминание промежуточных значений границ популяции, так как в процессе развития они меняются;

660...720 - здесь выводится на экран популяция из массива A\$(30,20), причем букве С в графическом режиме соответствует голубая клеточка, букве В - особь;

730 - если число особей стало равным нулю (популяция погибла), включается подпрограмма END, после чего осуществляется переход на начало игры;

740...910 - формирование новой популяции в массиве B\$(30,20) из старой A\$(30,20);

920...960 - массиву A\$(30,20) присваивается значение массива B\$(30,20), так как на печать выводится только массив A\$(30,20) (см. строки 680, 690);

970 - безусловный переход на строку 610, с которой начинается новый цикл размножения особей;

Начиная со строки 1000, следуют семь подпрограмм, о которых мы говорили выше. Поскольку их длины невелики, а с некоторыми текстами мы познакомились ранее, то нет особой необходимости разбирать каждую строку в отдельности. Тем не менее, рекомендуем самостоятельно изучить подпрограмму 3000...4080.

### **Составление программы меню**

Хорошая игровая программа всегда начинается с заставки. Познакомившись с модульным принципом построения программ, попробуем сделать заставку, состоящую из нескольких кадров-модулей. Поскольку во второй главе мы коротко рассмотрели принципы построения таких заставок, то здесь сосредоточим внимание на их основном элементе, который носит название меню.

Меню является связующим звеном между отдельными частями программы, чем-то вроде диспетчерского пункта: отсюда начинается игра, сюда же мы попадаем после ее окончания, а в некоторых играх - и в промежуточных ситуациях.

Что же делает меню? Вот типичный набор его пунктов:

```
НАЧАЛО ИГРЫ
ВЫБОР ВИДА УПРАВЛЕНИЯ
ПРАВИЛА ИГРЫ
ВЫВОД ТАБЛИЦЫ РЕЗУЛЬТАТОВ
ВЫБОР УРОВНЯ ИГРЫ
```

Что означает каждая из этих строк, думается, нет необходимости пояснять: они встречаются почти в каждой компьютерной игре. Как конкретно осуществляется выбор той или иной строки меню, мы рассмотрим на двух примерах.

Пусть первое меню выглядит так, как показано на рис. 4.6. Его работа может быть организована следующей программой:



Рис. 4.6. Меню с подвижным курсором.

Программа 18. МЕНЮ 1.

```
10 BORDER 0: PAPER 0: INK 1
20 CLS
22 PLOT 55,24: DRAW 0,120
24 DRAW 153,0: DRAW 0,-120
26 DRAW -153,0
28 INK 6
30 PRINT AT 2,13;"M E N U"
40 DIM a$(6,19)
50 LET a$(1)="      START    GAME    "
60 LET a$(2)="      CONTROL    "
70 LET a$(3)="      INSTRUCTION    "
80 LET a$(4)="      PRINT SCORE    "
90 LET a$(5)="      LEVEL    "
100 LET a$(6)="      QUIT PROGRAM  "
105 INK 5
110 FOR N=1 TO 6
120 PRINT AT N*2+4,7;a$(N)
130 NEXT N
140 INK 4
150 PRINT AT 20,9;"press 6, 7 or 0"
160 LET No=1
170 PRINT AT No*2+4,7; PAPER 1; INK 5;a$(No)
172 BEEP .07,20: PAUSE 30
```

```

175 LET Noo=No
180 LET b$=INKEY$
190 IF b$="6" THEN LET No=No+1: GO TO 220
200 IF b$="7" THEN LET No=No-1: GO TO 220
210 IF b$="0" THEN GO TO 300
215 GO TO 180
220 IF No>6 THEN LET No=1
230 IF No<1 THEN LET No=6
240 PRINT AT Noo*2+4,7; PAPER 0; INK 5;a$(Noo)
250 GO TO 170
299 REM -----
300 REM IF No=. THEN GOTO ...

```

Поясним действия входящих в программу строк.

10 - установка атрибутов экрана;

22...26 - формирование рамки синего цвета, в которой должны быть помещены пункты меню;

30 - вывод над рамкой слова MENU;

40 - задание массива строковых переменных;

50...100 - присваивание элементам массива текстовых значений - пунктов меню;

110...130 - вывод пунктов меню в рамке через пустую строку;

150 - информация о клавишах управления для выбора нужной строки, помещаемая под рамкой;

160 - основная переменная, указывающая номер надписи, на которую устанавливается создаваемый нами курсор;

170 - формирование курсора, представляющего собой горизонтальную полосу синего цвета, наложенную поверх текста надписи;

175 - здесь запоминается «старый» номер строки, когда положение курсора изменяется на один шаг;

180...210 - управление курсором вверх/вниз. Если нажать клавишу O, то из меню мы перейдем непосредственно к тому разделу программы, который выбран курсором;

220...230 - анализ номера строки с тем, чтобы он не вышел за пределы 1...6;

240 - когда курсор переходит на другую строку, на экран выводится старая надпись, но уже без курсора;

250 - передача управления группе строк, организующих переход курсора на другой пункт меню.

Еще одна программа меню, которую мы сейчас рассмотрим, отличается от предыдущей способом представления информации на экране и, соответственно, другим программным решением. Она выводит на экран следующие надписи:

```

НАЧАЛО ИГРЫ
ИНСТРУКЦИИ
ПЕЧАТЬ ТАБЛИЦЫ РЕЗУЛЬТАТОВ
КЛАВИАТУРА
ДЖОЙСТИК
УРОВЕНЬ ИГРЫ
ОБ АВТОРЕ
КОНЕЦ ПРОГРАММЫ

```

Особенность этого меню состоит в том, что эти надписи как бы нанесены на широкое бумажное кольцо, которое может «вращаться» вокруг своей оси, показывая в окне три любых строки из приведенного списка, расположенных подряд (рис. 4.7).



Рис. 4.7. Меню с подвижным текстом.

Процесс вращения управляется клавишами 6 (вниз) и 7 (вверх). Когда выбор пункта меню сделан, нажимается клавиша 0, и управление передается в один из блоков основной программы.

Программа 19. МЕНЮ 2.

```

10 BORDER 0: PAPER 0: INK 6
20 CLS
30 GO SUB 3000
40 DIM A$(8,19)
45 DIM B$(3,19)
50 LET A$(1)="      START    GAME"
55 LET A$(2)="      INSTRUCTION"
60 LET A$(3)="      PRINT SCORE"
65 LET A$(4)="      KEYBOARD"
70 LET A$(5)="      JOYSTICK"
75 LET A$(6)="      LEVEL"
80 LET A$(7)="      ABOUT    ME"
85 LET A$(8)="      QUIT    PROGRAM"
90 LET NO=1
92 PRINT AT 12,4: INK 2;"AAAAAAAAAAAAAAAAAAAAAAAAAAAA"
93 INK 1
94 PLOT 46,54: DRAW 154,0
95 DRAW 0,43: DRAW -154,0
96 DRAW 0,-43
97 INK 6
100 PRINT AT 1,8;"CS/1;4;CS/5 CS/4;3;2 CS/5 CS/5 CS/5 CS/5"
110 PRINT AT 2,8;"CS/5;2;CS/5 CS/4;2CS/4;CS/7;CS/5 CS/5 CS/5"
120 PRINT AT 3,8;"CS/5 CS/5 CS/1;CS/3;CS/7 CS/5;1;CS/5 6;CS/3;2"
122 PRINT AT 20,2: INK 3;"6...DOWN, 7...UP, 0...ENTER"
124 INK 4: PLOT 0,0: DRAW 0,175: DRAW 248,0: DRAW 0,-175: DRAW -248,0
130 LET NO=1
140 INK 5
150 LET N1=0
160 FOR N=NO-1 TO NO+1
165 LET NN=N
170 IF N<1 THEN LET NN=8
180 IF N>8 THEN LET NN=1
190 LET N1=N1+1
200 LET B$(N1)=A$(NN)
210 NEXT N
215 GO SUB 450
220 PRINT AT 10,6:B$(1)
230 PRINT AT 12,6: INK 4:B$(2)
240 PRINT AT 14,6:B$(3)
250 LET C$=INKEY$
260 IF C$="6" THEN GO SUB 300: GO TO 150
270 IF C$="7" THEN GO SUB 350: GO TO 150
280 IF C$="0" THEN GO TO 600

```

```

290 GO TO 250
300 REM -----
305 LET No=No-1
310 IF No<1 THEN LET NO=8
320 GO SUB 400
330 PRINT AT 11,6;B$(1)
334 PRINT AT 13,6;B$(2)
340 GO SUB 500: RETURN
350 REM -----
355 LET No=No+1
360 IF No>8 THEN LET No=1
365 GO SUB 400
370 PRINT AT 11,6;B$(2)
380 PRINT AT 13,6;B$(3)
385 GO SUB 500
390 RETURN
400 REM -----
410 PRINT AT 10,6;"oooooooooooooooooooo"
420 PRINT AT 12,6;"oooooooooooooooooooo"
430 PRINT AT 14,6;"oooooooooooooooooooo"
440 RETURN
450 REM -----
455 PRINT AT 11,6;"oooooooooooooooooooo"
465 PRINT AT 13,6;"oooooooooooooooooooo"
470 RETURN
500 REM -----
510 FOR M=1 TO 10: NEXT M
520 RETURN
600 REM -----
610 CLS
620 REM IF No=.. THEN GOTO ...
2999 STOP
3000 REM ---- U.D.G.-----
3010 FOR N=0 TO 15
3020 READ S
3030 POKE USR "A"+N,S
3040 NEXT N
3050 RETURN
3060 DATA 0,16,24,252,254,252,24,16
3070 DATA 0,16,48,126,254,126,48,16

```

Чтобы сделать программу удобной для анализа, в ней выделены шесть подпрограмм-модулей:

300...340 - подпрограмма сдвига всех надписей вниз;

350...390 - подпрограмма сдвига всех надписей вверх;

400...440 - подпрограмма, удаляющая три надписи в окне;

450...470 - подпрограмма, удаляющая две надписи в окне, которые занимают промежуточное положение по отношению к предыдущим трем надписям и служат для создания на экране эффекта перемещения строк сверху вниз и наоборот;

500...520 - подпрограмма паузы. Необходима для создания эффекта движения надписей вверх или вниз при нажатии нужных клавиш;

3000...3070 - запись в память компьютера изображений стрелок, которые должны располагаться за пределами окна и указывать на надпись, подобно курсору. Таким образом, в этой программе, в отличие от предыдущей, курсор неподвижен, а все надписи могут передвигаться вверх и вниз.

А теперь коротко об остальных строках программы.

10, 20 - установка атрибутов экрана;

30 - обращение к подпрограмме 3000...3070, которая создает две направленные

навстречу друг другу стрелки, выполняющие роль неподвижного курсора, расположенного посередине окна;

40 - резервирование памяти для 8 строк меню;

45 - резервирование памяти для 3 строк меню, выводимых на экран;

50...85 - запись 8 пунктов меню в массив A\$;

92 - вывод на экран стрелок курсора;

93...96 - вывод рамки, куда выводятся три слова из меню;

97...120 - вывод на экран слова МЕНЮ, причем здесь используются символы псевдографики, которые набираются в режиме курсора [G] нажатием цифровых клавиш отдельно или совместно с Caps Shift. В тексте программ мы будем обозначать такие символы подчеркиванием, например, 3 или CS/1, причем при отсутствии пробелов между ними будем ставить точку с запятой (;);

122 - вывод информации о клавишах управления;

124 - вывод внешней рамки меню;

150...210 - операторы этих строк записывают в массив B\$ подряд три строки меню, выбирая их из массива A\$;

220...240 - вывод на экран массива B\$;

250...290 - анализ нажатой клавиши управления и выполнение действий в соответствии с этой клавишей;

300...3000 - подпрограммы, действие которых описано выше, а построчный разбор предлагается сделать читателю.

### ***Звуковое сопровождение игр***

Говоря о создании игровых программ, невозможно пройти мимо скромного оператора BEEP, с помощью которого можно извлекать различные звуки, переключать на компьютерный язык известные мелодии и даже фрагменты классических произведений. Наверное, нет необходимости убеждать вас в том, насколько интереснее и богаче становится компьютерная игра, если в нее удачно вставлены различные музыкальные фразы и даже простые звуки, сопровождающие нажатия клавиш.

Мы уже использовали оператор BEEP для получения различных звуковых эффектов вроде щелчков, потрескиваний и простых звуковых сигналов. Сейчас же покажем, как с его помощью получают более сложные звуки и мелодии. Для этого чаще всего оператор BEEP включается в какой-либо цикл, где один из его параметров изменяется, например:

Программа 20. ЗВУК 1.

```
10 BORDER 0: PAPER 0: INK 6
20 CLS
30 FOR N=1 TO 40 STEP 2
40 BEEP 0.002,N
50 NEXT N
```

Ниже приведен листинг еще одной программы, создающей интересный звуковой эффект.

Программа 21. ЗВУК 2.

```
10 BORDER 0: PAPER 0: INK 6
20 CLS
30 FOR M=1 TO 2
40 FOR N=30 TO 1 STEP -5
50 BEEP 0.07,N
60 PAUSE 1
70 NEXT N
80 FOR N=1 TO 30 STEP 5
```

```

90 BEEP 0.07,N
100 PAUSE 1
110 NEXT N
120 NEXT M

```

Другой возможный вариант - использование в одном цикле нескольких операторов BEEP, «настроенных» на разные длительности и высоты:

Программа 22. ЗВУК 3.

```

10 BORDER 0: PAPER 0: INK 6
20 CLS
30 FOR N=0 TO 20
40 PAUSE 3
45 BEEP 0.007.N+7
50 BEEP 0.003,N
60 NEXT N

```

Входящий в цикл оператор PAUSE регулирует длительность паузы между отдельными звуками, но, вообще говоря, его можно исключить, и тогда переход от звука к звуку будет плавным.

Теперь приведем программу простейшего синтезатора, воспроизводящего короткую музыкальную фразу:

Программа 23. МЕЛОДИЯ.

```

10 BORDER 0: PAPER 0: INK 6
20 CLS
30 LET TEMP=0.05
40 FOR N=1 TO 9
50 READ DL,TON
60 BEEP DL*TEMP,TON
70 NEXT N
80 STOP
100 DATA 4,9,4,9,4,9,4,9
110 DATA 8,7,4,5,4,5
120 DATA 8,4,8,2

```

Здесь первый оператор READ DL считывает из списка данных DATA числа, стоящие на нечетных позициях и характеризующие длительности звуков, а второй оператор READ TON - числа, стоящие на четных местах и соответствующие высотам тона. Таким образом, при формировании списка DATA все числа необходимо группировать парами. Если эту программу запустить, то можно заметить, что мелодия очень напоминает песню «Во поле береза стояла». Чтобы с помощью этой программы воспроизвести другую мелодию, достаточно изменить данные в строках DATA и, если потребуется, верхний предел переменной N в строке 40.

### ***Музыкальный редактор Wham***

Итак, вы знаете, как извлекать из компьютера разные звуки, и даже сможете при желании написать программу, воспроизводящую простенькую мелодию. Однако на Бейсике далеко не уедешь, поэтому полезно знать, что существуют специальные программы, предназначенные для создания компьютерной музыки, называемые музыкальными редакторами. Наиболее известным из такого рода программ для ZX Spectrum является, пожалуй, редактор Wham, позволяющий писать двухголосные мелодии в диапазоне четырех октав, сдобренные различными шумовыми эффектами. Созданную или отредактированную мелодию можно сохранить на магнитной ленте или дискете в виде блока кодов и использовать затем в собственной игровой программе.

#### ***Сочинение новой мелодии***

После загрузки Wham с ленты или дискеты на экране появится главное меню.

Новая мелодия записывается в режиме редактирования, войти в который можно, нажав клавишу 6. При этом на экране появится изображение клавиатуры фортепиано, двух нотных станов (десять нотных линейек), указатели октавы OCTAVE и редактируемого голоса (рабочего канала) CHANNEL 1(2), а также счетчики шагов для каждого из каналов COUNTERS (CHN1, CHN2).

Для выбора функций используются два верхних ряда клавиш, а ввод и изменение мелодии осуществляются клавишами двух нижних рядов. Если вы всерьез хотите заняться сочинением компьютерных мелодий, вам придется досконально изучить функции всех клавиш (см. Приложение 4 или [1]). Краткую же сводку команд можно получить в режиме HELP PAGE, если нажать клавишу 7 в главном меню.

Давайте воспользуемся самым простым способом создания мелодии. Прежде всего следует найти какие-нибудь ноты. На первый раз сгодится все что угодно: учебник для первого класса музыкальной школы, альбом детских или народных песен - чем проще, тем лучше. Получите минимальную информацию о нотах (от специалистов или из учебника музыки) - и садитесь за компьютер, чтобы воплотить в звуках свой первый музыкальный шедевр.

Техника ввода целиком основана на сравнении, то есть сначала вы смотрите в книгу и запоминаете положение очередной ноты на линейках, а затем переносите ее на соответствующее место нотного стана на экране телевизора. Не забывайте при вводе нот переключать октавы клавишами 1...4 и внимательно следите за паузами. Если при вводе была допущена ошибка, нажмите клавишу 0, и все изображение возвратится назад на один шаг. После ввода десятка нот желательно послушать, что же получилось. Для этого прокрутите запись назад, нажав 9, а затем, удерживая клавишу P, слушайте мелодию.

Можно поступить и по-другому. Для возврата к началу мелодии нажмите клавишу R, а затем, чтобы прослушать мелодию - нажмите и отпустите Q. Остановить проигрывание можно нажатием любой клавиши.

Здесь хотелось бы дать один совет. Вводить ноты первого и второго голоса желательно параллельно, попеременно переключая каналы клавишей T, так как если вводить от начала до конца один, а затем другой голос, то ошибки просто неизбежны.

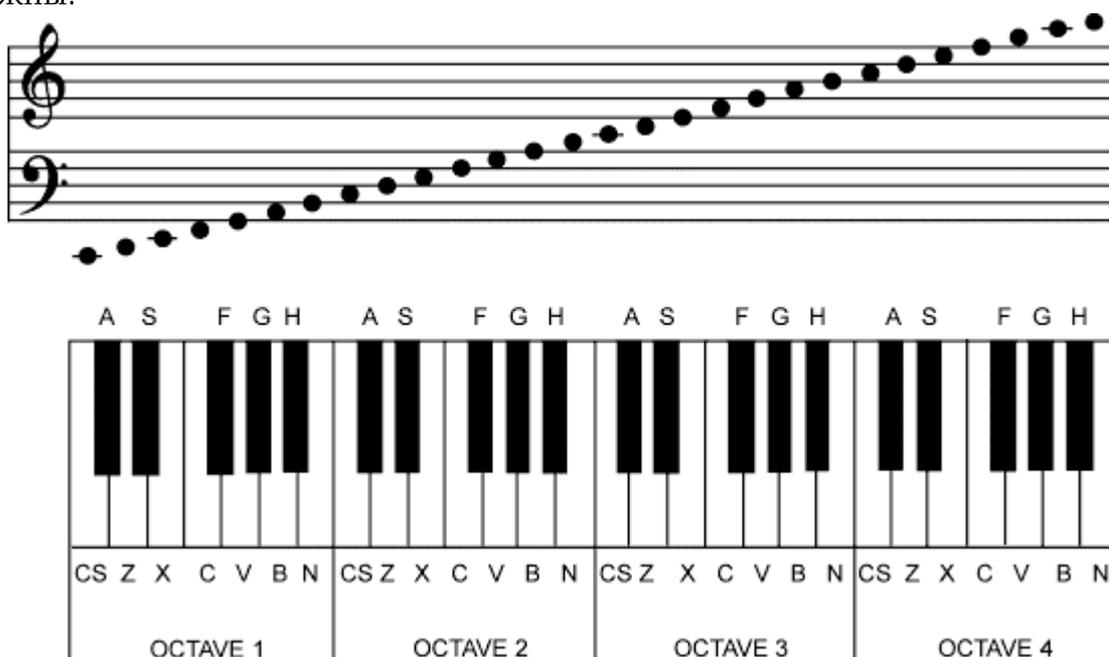


Рис. 4.8. Соответствие между нотами и клавишами компьютера.

Но ввести мелодию будет не так-то просто, если не установить, как связаны между собой ноты и клавиши вашего Спрессу. Сделать это поможет рис. 4.8. Внимательно рассмотрите его и попробуйте понажимать клавиши, сравнивая «выскакивающие» на экран ноты с теми, что находятся перед вами.

Может оказаться, что в памяти компьютера уже есть какая-то мелодия, и она мешает вашим упражнениям, занимая нотные линейки. Чтобы от нее избавиться, нажмите клавишу 7 и на вопрос ERASE CURRENT TUNE (Y/N)? отвечайте нажатием клавиши Y. После этого на экране появится начальный участок чистых нотных линеек, приглашающий вас к вводу музыкального произведения. Клавишей T установите 1-й голос (CHANNEL 1), а затем клавишами 1...4 - нужную октаву (OCTAVE), после чего можно вводить ноты. Пауза вводится клавишей Enter, а на нотных линейках (внизу) она отображается буквой R.

Выпишем для удобства значения всех клавиш в пределах одной октавы:

CS - ДО	F - ФА#
A - ДО#	V - СОЛЬ
Z - РЕ	G - СОЛЬ#
S - РЕ#	B - ЛЯ
X - МИ	H - ЛЯ#
C - ФА	N - СИ

Итак, начинаем вводить мелодию, которую назовем просто МЕЛОДИЯ 1 (рис. 4.9 и табл. 1):

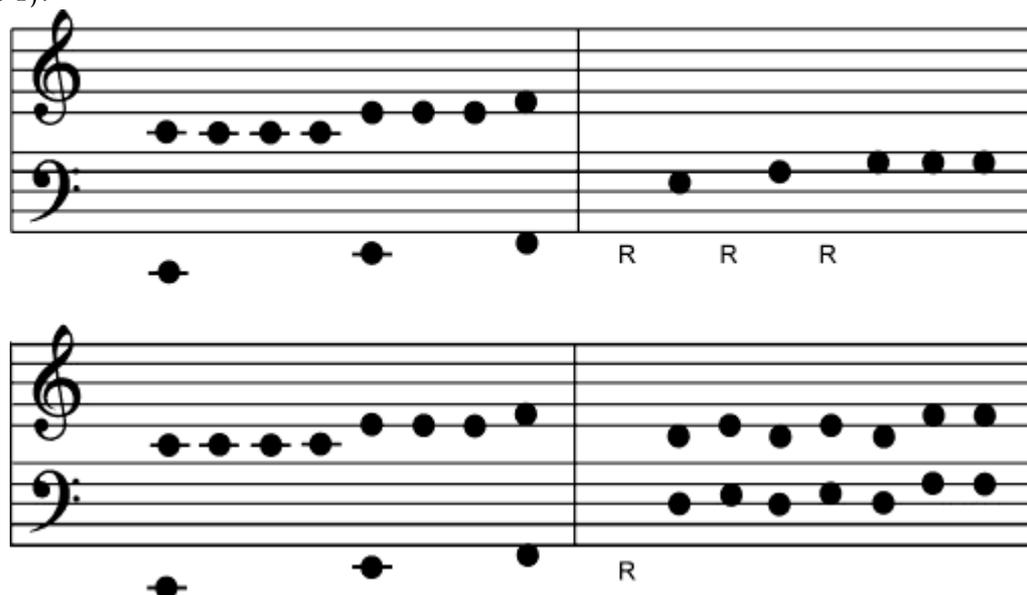


Рис. 4.9. Ноты МЕЛОДИИ 1.

Таблица 1

ГОЛОС 1	ГОЛОС 2	ГОЛОС 1	ГОЛОС 2
ОСТ.3 CS	ОСТ.1 CS	CS	ОСТ.1 CS
CS	ENT	CS	ENT
CS	ENT	CS	ENT
CS	ENT	CS	ENT
X	X	X	X
X	ENT	X	ENT
X	ENT	X	ENT
C	C	C	C
ENT	ENT	ENT	ENT
ENT	ОСТ. 2 X	ОСТ. 2 X	ОСТ.3 X

ENT	ENT	C	C
ENT	C	X	X
ENT	ENT	C	C
ENT	V	X	X
ENT	V	V	V
ENT	V	V	V
		ENT	ОСТ. 2 CS

Здесь ОСТ. - октава, CS - клавиша Caps Shift, ENT - клавиша Enter.) Этот музыкальный фрагмент лучше всего использовать как вступление к какой-нибудь игре.

На рис. 4.10 и в табл. 2 представлена хорошо знакомая всем мелодия «В траве сидел кузнечик».

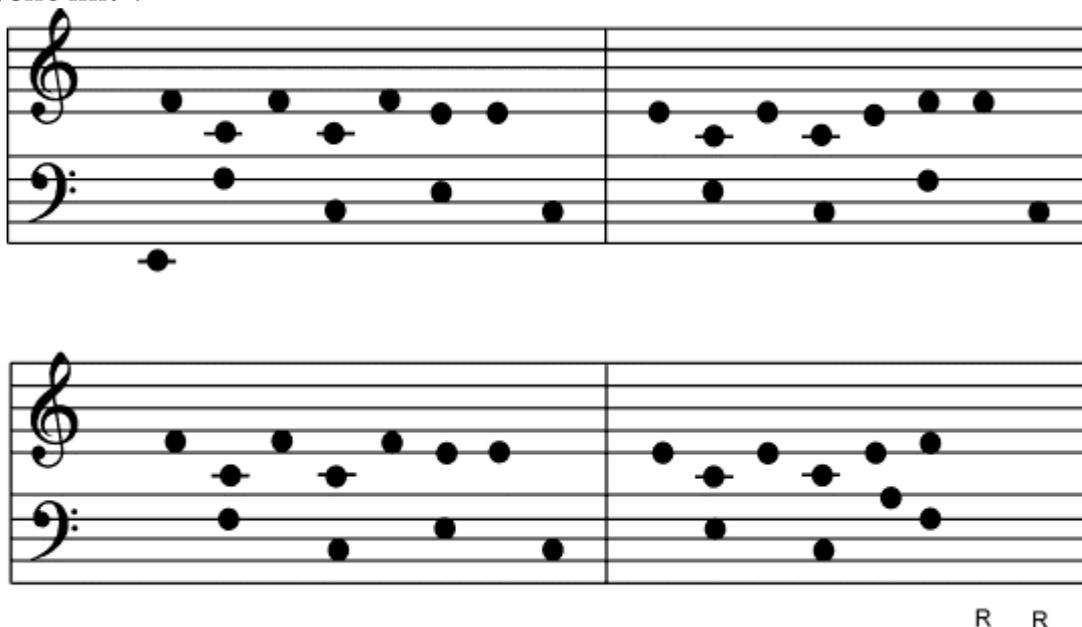


Рис. 4.10. Ноты мелодии «Кузнечик».

Таблица 2

Голос 1		Голос 2		Голос 1		Голос 2	
ОСТ.3	C	ОСТ. 2	ENT	C	ENT		
	CS		C	CS	C		
	C		ENT	C	ENT		
	CS		CS	CS	CS		
	C		ENT	C	ENT		
	X		X	X	X		
	X		ENT	X	ENT		
	ENT		CS	ENT	CS		
	X		ENT	X	ENT		
	CS		X	CS	X		
	X		ENT	X	ENT		
	CS		CS	CS	CS		
	X		ENT	X	ENT		
	C		C	C	C		
	C		ENT	ENT	ENT		
	ENT		CS	ENT	ENT		

Чтобы послушать введенную мелодию, необходимо перейти в главное меню, нажав клавишу 6, а затем выбрать в нем пункт 3. Может оказаться, что темп ее

исполнения вас не устраивает. Тогда в главном меню выберите режим изменения темпа (SET TEMPO) (клавиша 5) и с помощью клавиш 8 (ускорение) и 5 (замедление) добейтесь требуемого темпа воспроизведения мелодии. Для контроля скорости исполнения в нижней части экрана высвечивается фиолетовый прямоугольник. Прослушивать только что введенную мелодию лучше всего в медленном темпе, чтобы на слух контролировать правильность звучания.

Для придания создаваемой вами мелодии ритма можно использовать для ее аранжировки барабан и шумовые эффекты. Как это сделать, описано в Приложении 4.

#### **Компиляция и запись мелодии**

Если мелодия понравилась, можно переходить к следующему этапу. Поскольку вы хотите «подшить» эту мелодию к программе, написанной на Бейсике, необходимо записать мелодию в виде подпрограммы, работающей независимо от редактора Wham. Такая подпрограмма создается с помощью специального режима компиляции WHAMPILER. Однако перед компиляцией следует в обоих каналах выставить метки конца цикла, нажав клавишу W. При этом компьютер запросит:

```
SET LOOP HERE (Y/N)
```

- установить метку здесь? Если нажать клавишу Y, метки будут поставлены.

Нажав клавишу 4 в главном меню, войдите в режим компиляции. На запрос;

```
TUNENAME?
```

введите имя, под которым вы хотите записать файл, например, MEL 2. Далее программа попросит ввести адрес, с которого будет располагаться и запускаться этот файл:

```
ASSEMBLY ADDRESS?
```

Укажите какой-нибудь десятичный адрес, (но не слишком маленький, чтобы поместилась бейсик-программа), например, 60000. После этого компилятор выведет на экран информацию об условиях компиляции, которые влияют на исполняемый файл (подробно - в Приложении 4). Нажав, например, клавишу 1, выполним компиляцию, по окончании которой на экран будет выведена длина исполняемого файла и сообщение о нормальном завершении операции.

После ряда дополнительных сообщений, о которых подробно рассказано в Приложении 4, появится стандартная строка:

```
START TAPE, THEN PRESS ANY KEY
```

Включайте магнитофон на запись и нажимайте любую клавишу.

Осталось разобраться, как использовать «музыкальный блок» в игровой программе. Прежде всего в программу-загрузчик следует вставить дополнительную строку с оператором LOAD, загружающую с ленты музыкальный кодовый блок:

```
35 LOAD ""CODE 60000
```

А в самую игровую программу для исполнения мелодии нужно добавить строки:

```
10 CLEAR 59999
```

```
20 POKE 60035,230
```

```
30 POKE 60026,0
```

```
230 RANDOMISE USR 60000
```

10 - обнуляются все переменные и массивы, выполняется оператор CLS, изменяется адрес RAMTOP;

20 - в ячейку памяти с адресом 60035 записывается число в пределах от 230 до 255, что позволяет изменять темп проигрывания мелодии;

30 - в ячейку памяти с адресом 60026 записывается число в пределах от 0 до 7, которое во время исполнения мелодии устанавливает цвет бордюра;

230 - воспроизведение мелодии. Эту строку можно поставить в любом месте игровой программы, в зависимости от ее сюжета.

Если вы не до конца поняли, как вставить созданную и откомпилированную

---

мелодию в игровую программу, посмотрите, как это сделано в игре СОКОБАН, описанной в следующей главе.

## 5. БЛОК ВЗАИМОДЕЙСТВИЯ С ИГРАЮЩИМ

После того, как мы научились создавать спрайты самыми разными способами, настало время решить еще одну очень важную задачу - подчинить их своей воле и заставить двигаться по экрану. Вы уже хорошо знаете, что управлять движением объектов можно с помощью клавиатуры (keyboard) или используя различные типы джойстиков (Kempston, Sinclair, Cursor). Какой из этих способов выбрать, зависит в основном от сюжета игры и ваших пристрастий. При желании можно пользоваться тем и другим одновременно.

Познакомимся теперь со всеми перечисленными методами управления спрайтами и их особенностями с точки зрения программирования.

### *Управление спрайтами с помощью клавиатуры*

Рассмотрим программу, формирующую изображения вертолета и позволяющую это изображение перемещать по экрану (см. рис. 3.3).

Программа 24. ВЕРТОЛЕТ.

```

10 BORDER 1: PAPER 1: INK 6
20 CLS
30 FOR N=0 TO 71
40 READ S
50 POKE USR "A"+N,S
60 NEXT N
70 DATA 0,0,135,0,120,127,15,39
80 DATA 0,0,255,2,127,201,201,255
90 DATA 0,0,255,0,192,112,24,12
100 DATA 3,1,0,0,0,0,0,0
110 DATA 255,255,127,63,4,255,0,0
120 DATA 254,254,252,248,34,252,0,0
130 DATA 0,0,32,0,120,127,15,135
140 DATA 0,0,7,2,127,201,201,255
150 DATA 0,0,0,0,192,112,24,12
200 REM -----
210 LET X=10: LET Y=10: LET X1=X: LET Y1=Y
220 LET W=0
230 IF INKEY$="0" THEN LET X=X-1: GO TO 300
235 IF INKEY$="c" THEN LET X=X-1: GO TO 300
240 IF INKEY$="P" THEN LET X=X+1: GO TO 300
245 IF INKEY$="p" THEN LET X=X+1: GO TO 300
250 IF INKEY$="Q" THEN LET Y=Y-1: GO TO 300
255 IF INKEY$="q" THEN LET Y=Y-1: GO TO 300
260 IF INKEY$="A" THEN LET Y=Y+1: GO TO 300
265 IF INKEY$="a" THEN LET Y=Y+1: GO TO 300
270 GO TO 320
300 PRINT AT Y1,X1;"ППП"
310 PRINT AT Y1+1,X1;"ППП"
320 IF W<3 THEN PRINT AT Y,X;"ABC"; AT Y+1,X;"DEF":GO TO 340
330 PRINT AT Y,X;"GHI";AT Y+1,X;"DEF"
340 LET W=W+1: IF W=5 THEN LET W=0
350 LET X1=X: LET Y1=Y
360 BEEP .002,0
370 GO TO 230

```

10... 150 - работа этой части программы должна быть вам понятна, поэтому мы лишь отметим, что здесь используются два спрайта с изображением вертолета. Несколько спрайтов для одного и того же объекта рисуются всякий раз, когда

нужно создать эффект движения - как в мультфильмах. В нашем случае у вертолета должен вращаться винт, поэтому спрайты отличаются именно его положением. Каждая картинка состоит из шести полей. Для первой использованы графические символы, соответствующие клавишам A, B, C, D, E и F, а для другой - G, H, I, D, E и F. Нижняя часть вертолета одинакова в обоих спрайтах, поэтому три последних символа повторяются. Всего использовано 9 символов, следовательно, количество вводимых в программу кодов должно быть равно  $9 \times 8 = 72$ ;

210 - задание начальных координат вертолета. Координаты X и Y будем называть новыми, а X1 и Y1 - старыми. До начала движения новые и старые координаты совпадают;

220 - инициализация переменной W, которая в дальнейшем будет указывать компьютеру, какой из двух спрайтов вертолета выводить на экран.

Управляют положением вертолета на экране две части программы. Первая (строки 230...265) анализирует, какая клавиша нажата, а вторая (строки 300...350) строит изображение вертолета в новом месте экрана; 230...260 - в предыдущих главах мы уже упоминали функцию INKEY\$, которая возвращает символ, соответствующий клавише, нажатой в момент ее выполнения. Для перемещения вертолета задействуем клавиши: O - влево, P - вправо, Q - вверх и A вниз. Но, как вы знаете, с клавиатуры можно вводить и прописные, и строчные буквы. Все зависит от того, какой режим установлен клавишей Caps Lock. Поэтому в блоке опроса клавиатуры нужно предусмотреть все возможности и проверять как большие, так и маленькие буквы. Предположим, что нажата клавиша Q, тогда в строке 230 условие "Q" = "O" не выполняется. Также не выполняется и условие "Q" = "o". В результате программа переходит к строке 240. Но и здесь условия "Q" = "P" и "Q" = "p" не выполняются, поэтому проверка продолжается. Так программа доходит до строки 250, где условие "Q" = "O" истинно. В этом случае начинают выполняться операторы, стоящие после THEN данной строки, то есть из значения координаты Y вычитается единица (на экране вертолет должен подняться вверх) и программа переходит к строке 300. Возникает вопрос: что будет, если нажать какую-либо клавишу, отличную от перечисленных? Ответ можно найти в строке, следующей за блоком опроса клавиатуры;

270 - после блока условий в большинстве случаев ставится строка с безусловным переходом, которая после каждого «неправильного» нажатия возвращает управление на начало блока. Это продолжается до тех пор, пока не будет нажата одна из разрешенных клавиш. Другой вариант (именно так мы поступили в нашей программе) передать управление в ту часть программы, которая выводит на экран изображение вертолета в соответствии с текущими координатами, то есть на строку 320;

300, 310 - но вот нажата одна из нужных букв, то есть вертолету предписано куда-то передвинуться, и операторы строк 300 и 310 стирают изображение на старом месте (координаты X1 и Y1);

320, 330 - эти строки выводят на экран изображение в место, заданное новыми координатами. Причем в зависимости от значения переменной W на экран помещается тот или иной спрайт, что создает впечатление вращения винта;

340 - переменная W увеличивается на единицу, и если она становится равна 5, то обнуляется. Таким образом, если W равна 0, 1 или 2, то вертолет изображается одним спрайтом, а при W, равной 3, 4 или 5 - другим. Спрайты сменяются через три цикла для того, чтобы вращение винта было не чересчур быстрым;

350 - новые координаты запоминаются и становятся для очередного цикла движений старыми;

360 -- звуковой сигнал, имитирующий стрекотание вертолета;

370 - безусловный переход к блоку проверки нажатия клавиш для следующего изменения положения вертолета.

Конечно же, эта программа далеко не идеальна. В ней лишь показан принцип управления спрайтами с помощью клавиатуры. Надеемся, что этот принцип вам достаточно понятен. Теперь попробуем слегка улучшить блок опроса клавиатуры и привести его к тому виду, в котором его можно будет использовать в реальных программах.

Забегая вперед, расскажем об одной замечательной возможности, позволяющей заметно уменьшить блок опроса клавиатуры и, соответственно, увеличить скорость его работы. Проверять строчные и прописные буквы нужно только в том случае, если неизвестно заранее, какой режим клавиатуры установлен до запуска программы, или если он мог измениться во время работы (например, в операторе INPUT). Однако режим ввода символов можно изменить не только с клавиатуры, но и из программы. Не вдаваясь пока в подробности, скажем только, что установить режим курсора [C] (заглавные буквы) можно, включив в программу оператор

```
POKE 23658,8
```

вернуться к курсору [L] позволит оператор

```
POKE 23658,0
```

Вставьте в программу ВЕРТОЛЕТ строку

```
225 POKE 23658,8
```

и тогда строки 235, 245, 255 и 265 можно смело убрать.

### ***Управление спрайтами с помощью джойстика***

Объектами игры часто удобнее управлять не с клавиатуры, а с помощью джойстика, поэтому посмотрим, как это можно сделать. Если для опроса клавиатуры мы пользовались функцией INKEY\$, то узнать, в какую сторону повернута ручка Kempston-джойстика можно только с помощью функции IN. Эта функция возвращает числовое значение из порта, адрес которого записан в качестве ее аргумента. Возможно, вы еще не знаете, что собой представляют порты в компьютере, поэтому попробуем объяснить, что это такое.

В нашем мире порты существуют для сообщения с другими государствами, а для островных империй (вроде компьютера) это единственное средство связи с внешним миром. Но у маленького Спрессу портов гораздо больше, чем у самой великой державы - целых 65536. Правда, практическое применение имеют далеко не все из них, а остальные простаивают без дела. Любой ввод и вывод информации (кроме вывода на рабочий экран) происходит именно через порты. С ними связаны и клавиатура, и джойстик, и магнитофон, и принтер, и динамик, причем каждому устройству соответствует порт с определенным адресом (не путайте с адресами ячеек памяти). Интересующий нас в данный момент джойстик типа Kempston подключен к порту с адресом 31. Таким образом, узнать направление поворота ручки джойстика можно с помощью оператора

```
LET joy=IN 31
```

Зная коды, соответствующие различным направлениям наклона джойстика, нетрудно написать и блок управления.

Казалось бы, все очень просто, однако есть одно «но»: компьютеры, изготовленные в России, практически все «страдают» неполной совместимостью с фирменным Спрессу. В частности, значения, считываемые из порта 31, различны для разных моделей ZX Spectrum. На рис. 5.1 приведены два наиболее распространенных варианта кодов джойстика. Первый вариант (рис. 5.1, а, б)

характерен тем, что в нейтральном положении ручки из порта 31 считывается ноль (это соответствует фирменному Спрессу). В другом варианте (рис. 5.1, в, г) в нейтральном положении считывается число 224.

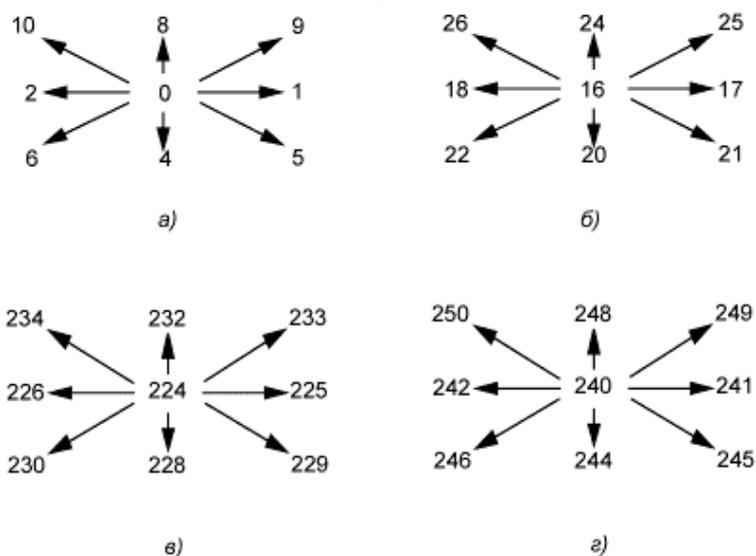


Рис. 5.1. Диаграмма кодов джойстика: а, в - для отжатой клавиши Огонь; б, г - для нажатой клавиши Огонь

Если ваш компьютер откажется реагировать на коды направлений джойстика, показанных на рисунке - ничего страшного. В этом случае можно самостоятельно выяснить необходимые коды с помощью маленькой программки:

```
10 PRINT AT 0,0;IN 31;" "
20 GO TO 10
```

Запустите ее оператором RUN и, поворачивая ручку джойстика в разные стороны, выпишите числа, которые будут появляться на экране.

Теперь давайте рассмотрим конкретный пример использования функции IN. Программу САМОЛЕТ уже вполне можно назвать мини-игрой, в которой имеется простейший пейзаж, состоящий из неба и земли,

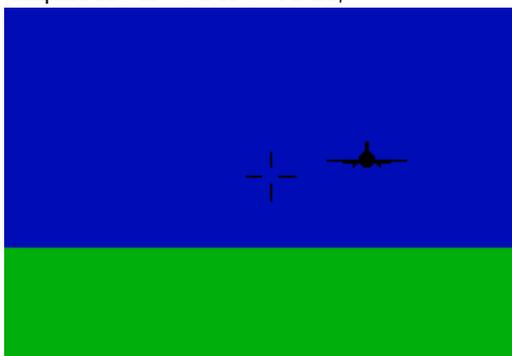


Рис. 5.2. Игровое пространство игры САМОЛЕТ.

самолет противника, передвигающийся по экрану случайным образом, и объект управления - прицел вашего самолета (рис. 5.2). Если самолет противника попал в прицел и вы успели нажать кнопку Огонь, то происходит взрыв, а самолет исчезает. Таков замысел этой игры.

Программа 25. САМОЛЕТ.

```
10 BORDER 0: PAPER 1: INK 0: CLS
20 FOR N=0 TO 55
30 READ S: POKE USR "A"+N,S
40 NEXT N
50 DATA 0,0,0,0,255,0,0,0
55 DATA 0,0,0,0,255,255,6,4
60 DATA 60,126,255,255,255,255,126,60
```

```

65 DATA 0,0,0,0,255,255,96,32
70 DATA 0,0,0,0,255,0,0,0
75 DATA 0,0,0,24,24,24,24,24
80 DATA 40,21,102,153,90,165,44,16
100 REM -----
110 LET XS=11: LET YS=10: LET XS1=XS: LET YS1=YS
120 FOR N=15 TO 21: PRINT AT N,0; PAPER 4;TAB 31;" ": NEXT N
140 REM -----
150 LET PRX=2*RND-1
160 LET PRY=2*RND-1
170 LET DLC=INT (RND*8+2)
180 LET CI=0
200 REM -----
210 LET CI=CI+1
220 IF XS<2 THEN LET XS=XS+1: GO TO 150
230 IF XS>25 THEN LET XS=XS-1: GO TO 150
240 IF YS>19 THEN LET YS=YS-1: GO TO 150
250 IF YS<3 THEN LET YS=YS+1: GO TO 150
260 IF CI>=DLC THEN GO TO 150
270 LET XS=XS+PRX: LET XSN=INT XS
280 LET YS=YS+PRY: LET YSN=INT YS
290 LET JO=IN 31
300 IF JO=226 OR JO=242 THEN LET XS=XS+1: GO TO 400
310 IF JO=225 OR JO=241 THEN LET XS=XS-1: GO TO 400
320 IF JO=232 OR JO=248 THEN LET YS=YS-1: GO TO 400
330 IF JO=228 OR JO=244 THEN LET YS=YS+1: GO TO 400
340 IF JO=240 THEN LET FIRE=1: GO TO 400
350 IF JO=224 THEN LET FIRE=0: GO TO 400
360 IF JO=234 OR JO=250 THEN LET XS=XS+1: LET YS=YS-1:GO TO 400
370 IF JO=233 OR JO=249 THEN LET XS=XS-1: LET YS=YS-1:GO TO 400
380 IF JO=229 OR JO=245 THEN LET XS=XS-1: LET YS=YS+1:GO TO 400
390 IF JO=230 OR JO=246 THEN LET XS=XS+1: LET YS=YS+1
400 REM -----
410 PRINT AT YS1,XS1; PAPER 8;"#####";AT YS1-1,XS1+2;" "
420 PRINT AT YSN,XSN; PAPER 8; INK 0;"ABCDE";AT YSN-1,XSN+2; INK 0;"E"
430 LET XS1=XSN: LET YS1=YSN
440 INK 0
450 PLOT 120,91: DRAW 7,0
460 PLOT 136,91: DRAW 8,0
470 PLOT 132,103: DRAW 0,-7
480 PLOT 132,79: DRAW 0,8
490 IF FIRE=0 THEN PRINT AT 10,16; PAPER 1;" ": GO TO 200
500 PRINT AT 10,16; INK 2; PAPER 1;"G"
510 IF YSN=10 AND XSN>=12 AND XSN<=16 THEN GO SUB 1000: CLS: GO TO 100
520 GO TO 210
1000 REM -- WZRYW --
1010 PRINT AT YSN-1,XSN+2; PAPER 8;" "
1020 BORDER 4: INPUT ""
1030 FOR I=1 TO 3
1040 PRINT AT YSN,XSN; INK 2;"ABCDE"
1050 PAUSE 3
1060 FOR N=0 TO 7
1070 PRINT AT YSN,XSN; INK N;"GGGGG"
1080 LET W2=0: LET BORD=INT (RND*8): IF RND<.5 THEN LET WZ=16
1090 OUT 254,WZ+BORD
1100 NEXT N
1110 NEXT I
1120 BORDER 0
1130 RETURN

```

Для удобства разбора программы ее функциональные части, как и во многих

предыдущих, отделены друг от друга операторами REM.

10...80 - формирование спрайта самолета, состоящего из семи полей 8x8 точек. Этот фрагмент аналогичен соответствующим частям в программах 7 и 8, поэтому повторяться не будем;

110 - установка начальных координат изображения самолета на экране. Переменные XS и YS задают новые координаты самолета, а XS1 и YS1 - старые;

120 - нижняя часть экрана окрашивается в зеленый цвет, имитируя землю. Синий цвет неба задается в строке 10;

150, 160 - здесь формируются приращения координат самолета по горизонтали и вертикали при изменении его положения в пространстве на один шаг. Эти числа могут быть как положительными, так и отрицательными в диапазоне от -1 до +1, что позволяет самолету перемещаться во всех четырех направлениях;

170 - устанавливается длина пролета самолета до момента изменения направления, измеряемая в шагах (длина должна быть целым числом). Таким образом, в строках 150...170 задается случайная траектория полета самолета по экрану;

210 - задается переменная CI - счетчик числа шагов;

220...250 - в этой группе строк проверяются текущие координаты самолета SX и SY, чтобы он не ушел за пределы игрового поля. Числа 2, 25, 19 и 3 взяты с учетом конфигурации самолета. Если одна из его координат вышла за пределы, то соответствующая строка увеличивает (уменьшает) нужную координату на единицу, и программа возвращается на строку 150 - к вычислению новых приращений и длины пролета;

260 - число уже выполненных шагов CI сравнивается с заданной длиной пролета DLC;

270, 280 - текущие координаты изображения самолета получают приращение, а затем округляются до меньшего целого - в строке 510 эти координаты (переменные XSN и YSN) будут использованы для проверки попадания вашего снаряда в самолет противника;

290 - переменной JO присваивается значение из порта джойстика;

300...390 - блок управления прицелом содержит 10 условных операторов. Предположим, что ручка отклонена вправо (на «восток»), тогда функция IN 31 возвратит код 225<sup>1</sup>, который присвоится переменной JO. В строках 300...390 анализируется значение этой переменной и в зависимости от результата будут изменяться значения координат самолета. Условие JO=225 проверяется в строке 310, следовательно, координата XS уменьшится на единицу, самолет отклонится влево, а прицел как бы передвинется на один шаг вправо. Аналогично действуют и операторы, управляющие перемещением прицела в другие стороны;

410 - стирается предыдущее изображение самолета. Включение оператора PAPER 8. позволяет не заботиться об изменении цвета фона при переходе самолета с синей части экрана на зеленую и обратно;

420 - изображение самолета появляется на новом месте с новыми координатами;

430 - новые координаты запоминаются в переменных XS1 и YS1, то есть становятся старыми;

440...480 - черным цветом рисуется перекрестье прицела;

---

<sup>1</sup> Как уже говорилось, с вашим компьютером дела могут обстоять несколько иначе. Приведенные значения кодов справедливы только в том случае, если функция IN 31 в нейтральном положении джойстика возвращает число 224.

490 - если кнопка Огонь не нажата, то в центре изображения прицела ставится пробел;

500 - при нажатой кнопке Огонь в перекрестии прицела появляется красный огонек;

510 - сравниваются координаты изображений прицела и самолета. Если они совпадают хотя бы в одной точке, программа с помощью оператора GO SUB 1000 обращается к подпрограмме WZRYU (взрыв). Затем экран очищается и начинается следующая игра;

520 - если ни одна из координат самолета не попадает в перекрестье прицела, программа переходит на строку 210, где счетчик шагов увеличивается на единицу, и можно вновь изменить положение ручки джойстика;

1000... 1130 - здесь располагается подпрограмма, имитирующая на экране взрыв;

1010 - в этой строке с экрана стирается хвост самолета, так как после взрыва он должен отвалиться;

1020 - оператор INPUT "" можно использовать, когда требуется просто очистить служебное окно без ввода данных. При этом цвет PAPER в нижних строках экрана станет таким же, как и цвет бордюра. Таким образом, в этой строке программы служебное окно закрашивается зеленым цветом;

1030 - начало цикла по переменной I. Цикл нужен для того, чтобы воспроизвести взрыв трижды;

1040 - окрашивание изображения самолета красным цветом;

1050 - небольшая пауза для фиксации изображения горящего самолета;

1060 - начало цикла по переменной N, которая управляет цветом осколков самолета;

1070 - вывод изображения осколков самолета;

1080 - переменная WZ служит для получения звука взрыва и может принимать значения 0 или 16, а переменная BORD задает случайный цвет бордюра;

1090 - оператор OUT производит действие, обратное функции IN, то есть записывает в указанный порт какое-то число. Порт с адресом 254 - один из наиболее важных в ZX Spectrum. К нему подключены клавиатура, магнитофон и динамик. Он же может управлять и цветом бордюра, если в него записать числа от 0 до 7. Если в него посылать числа 0 и 16, достаточно быстро чередуя их, то получится какой-нибудь звук. Сумма кодов бордюра и звука объединит оба эффекта и мы получим переливающийся всеми цветами бордюр в сопровождении треска из динамика;

1100...1110 - окончания циклов;

1120 - бордюру возвращается первоначальный черный цвет;

1130 - выход из подпрограммы.

Не правда ли, глядя на огромный размер блока управления, хочется как-то сократить и упростить его? К счастью, это вполне осуществимо. Но прежде, чем узнать, как это делается, необходимо понять, откуда берутся указанные числа. Ведь на первый взгляд в их расположении нет никакой видимой закономерности. Однако закономерность, очевидно, все же должна быть, и чтобы ее обнаружить, необходимо представить полученные коды в двоичном виде. Мы уже встречались с такими числами, когда кодировали изображение «гномика». Но тогда двоичные числа мы переводили в привычный десятичный формат, а теперь нам нужно поступить наоборот. О том, как это сделать, будет сказано чуть ниже, а пока посмотрим на коды джойстика в двоичном представлении:

Положение	Дес.	Двоич.
Нейтр.	224	11100000
Влево	225	11100001

Вправо	226	11100010
Вниз	228	11100100
Вверх	232	11101000
Огонь	240	11110000

Как видите, здесь уже наблюдается определенная закономерность. Напомним, что в программировании 8-разрядные двоичные числа называют байтами, а отдельные разряды - битами. Из приведенной таблицы видно, что каждому направлению джойстика соответствует свой бит, в то же время три старших бита не изменяются и никакого влияния на процесс управления не оказывают. Поэтому в дальнейшем мы их учитывать не будем. Кстати, различия между отдельными компьютерами в отношении кодов джойстика касаются именно этих трех битов.

Теперь посмотрим, как можно перевести числа из десятичной системы счисления в двоичную и «выловить» отдельные биты. К сожалению, в Spectrum-Бейсике не существует специальной функции, производящей такой перевод, поэтому придется написать небольшую программку:

Программа 26. Перевод чисел из десятичной системы счисления в двоичную.

```

10 DIM J(5)
20 LET JOY=IN 31
30 FOR N=1 TO 5
40 LET JOY=JOY/2
50 LET J(N)=(JOY<>INT JOY): LET JOY=INT JOY
60 NEXT N
70 PRINT AT 0,0;
80 FOR N=5 TO 1 STEP -1
90 PRINT J(N);
100 NEXT N
110 GO TO 20

```

Эта программка выполняет следующие действия:

10 - определение массива с размерностью, равной количеству интересующих нас битов;

20 - чтение числа из порта джойстика;

30..60 - в цикле проверяются отдельные биты, начиная с младшего и заносятся в элементы массива J(5). Если после деления числа на 2 результат получается дробным (то есть делимое было нечетным), то это говорит о том, что младший бит установлен. В этом случае нужно отбросить дробную часть результата, что и делает функция INT.

Возможно, у вас вызвала недоумение запись LET J(N)=(JOY<>INT JOY) в строке 50. Однако такой метод довольно часто применяется в программировании. Он позволяет обойтись без условного оператора IF... THEN, в результате чего программа не только сокращается, но и работает заметно быстрее. Если условие в скобках выполняется, то возвращается единица, иначе - ноль. Если вы хотите проверить этот метод, можете сначала выполнить оператор PRINT (2=2), а затем PRINT (5<>5). В первом случае на экране появится число 1, а во втором - 0;

70...100 - вывод двоичного числа на экран. В строке 70 текущая позиция печати устанавливается в левом верхнем углу экрана (не забудьте в конце оператора поставить точку с запятой, иначе позиция печати переместится). Затем в цикле выводятся значения отдельных битов от старшего к младшему;

110 - переход для считывания и обработки следующей порции информации.

После набора и запуска программы поверните ручку джойстика и понаблюдайте за реакцией. Обратите внимание на то, что если задавать диагональные направления, будут включаться одновременно несколько битов. В дальнейшем это даст возможность избежать лишних проверок и позволит включать в программу только пять строк (по количеству используемых битов) с

условными операторами вместо 18 (по количеству возможных комбинаций).

Теперь можно видоизменить программу САМОЛЕТ. Вставьте в нее строку

```
15 DIM J(5)
```

а затем удалите строки 300...390 и поставьте на их место новый блок управления:

```
300 FOR N=1 TO 5
310 LET J0=J0/2: LET J(N)=J0<>INT J0: LET J0=INT J0
320 NEXT N
330 IF J(1)1 THEN LET XS=XS+1
340 IF J(2) THEN LET XS=XS-1
350 IF J(3) THEN LET YS=YS+1
360 IF J(4) THEN LET YS=YS-1
370 LET FIRE=0
380 IF J(5) THEN LET FIRE=1
```

В таком виде программа уже должна нормально работать на всех компьютерах, претендующих на совместимость с фирменным ZX Spectrum.

### **Особенности управления клавиатурой**

С помощью функции IN, как уже говорилось, можно опрашивать и клавиатуру. Правда, у этого способа есть ряд существенных недостатков по сравнению с использованием функции INKEY\$, что и делает его крайне нераспространенным. К недостаткам, например, можно отнести то, что с помощью IN за один раз можно опрашивать не все клавиши, а только по одному полуряду клавиатуры, то есть по 5 клавиш, что заметно сказывается на быстродействии. Но иногда без обращения к порту бывает просто не обойтись. Например, если требуется получить диагональное движение, да при этом еще иметь возможность постреливать в противника. Поэтому игнорировать этот способ нельзя.

Для реализации его, так же как и для джойстика, нужно знать адреса портов. Как мы уже говорили, клавиатура связана с портом 254, но на практике этого недостаточно, так как в этом случае мы будем считывать значения со всех полурядов одновременно и при нажатии Caps Shift, A, Q, 1, 0, P, Enter и Space будем получать один и тот же код. Для выбора опрашиваемого полуряда нужно указать не только младший, но и старший байт адреса порта. Полный адрес может быть вычислен из выражения  $254 + 256 * (255 - 2 * N)$ , где N принимает значения от 0 до 7. Вот эти адреса и соответствие их полурядам клавиатуры:

Порт	Полуряд
65278	CS.....V
65022	A.....G
64510	Q.....T
63486	1.....5
61438	0.....A
57342	P.....Y
49150	ENTER....H
32766	SPACE....B

При ненажатых клавишах из всех портов считывается код 255 (то есть все биты установлены). Если нажать какую-нибудь клавишу, то один из битов «сбросится» в 0, причем младшим битам соответствуют крайние клавиши каждого полуряда, а старшим - те, что ближе к центру клавиатуры. Значит, чтобы определить, например, нажата ли клавиша O, нужно выполнить оператор

```
LET KEY=IN 57342
```

При нажатой клавише O полученный результат будет 1111101 в двоичном

<sup>1</sup> Такая запись равноценна следующей: IF J(1)<>0 THEN

виде, или 253 - в десятичном. Если окажутся «сброшены» и другие биты, это будет означать, что нажаты и другие клавиши. Рассмотрим небольшую программку, напоминающую графический редактор, в которой используется подобный способ управления.

#### Программа 27. ОПРОС КЛАВИАТУРЫ.

```

10 INK 6: PAPER 0: BRIGHT 1: BORDER 1: CLS
20 LET P=0
40 LET X=120: LET Y=80: LET X1=X: LET Y1=Y
50 GO SUB 1000
80 IF P THEN PLOT X-3,Y-2: DRAW 4,4: PLOT X-2,Y-2: DRAW 4,4
100 IF IN 57342=253 AND X>6 THEN LET X=X-2
110 IF IN 57342=254 AND X<248 THEN LET X=X+2
120 IF IN 64510=254 AND Y<168 THEN LET Y=Y+2
130 IF IN 65022=254 AND Y>6 THEN LET Y=Y-2
140 LET P=0
150 IF IN 32766=251 THEN LET P=1
160 GO SUB 1000
170 LET X1=X: LET Y1=Y: GO SUB 1000
180 IF INKEY$="N" OR INKEY$="n" THEN CLS : GO TO 50
190 GO TO 80
1000 OVER 1
1010 PLOT X1-2,Y1: DRAW -4,0: PLOT X1+2,Y1: DRAW 4,0
1020 PLOT X1,Y1-2: DRAW 0,-4: PLOT X1,Y1+2: DRAW 0,4
1030 OVER 0
1040 RETURN

```

После ввода и запуска программы экран окрасится в черный цвет, а синий бордюр ограничит рабочее поле. В центре экрана появится мигающий курсор желтого цвета в виде перекрестья прицела. Нажимая клавиши Q, A, O и P, курсор можно перемещать во всех направлениях. Если при этом нажать еще и клавишу M, то за курсором потянется след. Теперь прокомментируем вкратце строки этой программы:

10 - установка атрибутов экрана;

20 - переменная P указывает положение «пера»; 0 - «перо» поднято, 1 - опущено;

40 - установка начальных координат курсора;

50 - обращение к подпрограмме, выводящей изображение курсора на экран;

80 - если переменная P не равна нулю, то есть «перо» опущено, то рисуется его профиль;

100...190 - блок управления курсором и положением «пера». Если нажата клавиша O, то курсор (а вместе с ним и «перо») перемещается на один шаг, равный двум пикселям, влево. Но это только в том случае, если курсор не дошел до левого края экрана (если условие  $X > 6$  выполняется). В строках 110... 130 аналогично проверяется нажатие клавиш P, Q и A. Обратите внимание, что в конце этих строк отсутствует оператор GO TO, что позволяет при нажатии двух клавиш получить диагональное перемещение курсора, так как изменится не одна координата, а сразу обе;

140, 150 - управление положением «пера». Если нажата клавиша M, переменная P становится равной 1 и «перо» опускается на «бумагу»;

160 - курсор после обращения к подпрограмме 1000 стирается;

170 - изменение координат и вывод курсора в новое место экрана;

180 - если нажата клавиша N, то рисунок на экране стирается - можно начинать рисовать новый;

190 - переход для продолжения рисования;

1000...1040 - подпрограмма вывода изображения курсора на экран. Режим OVER 1 устанавливается для того, чтобы не портить полученный рисунок и стирать

курсор на старом месте. При первом обращении к подпрограмме курсор появляется, а при втором - исчезает.

Во многих программах вместо Kempston-джойстика используются Sinclair-джойстики. Для того чтобы приспособить рассмотренные программы к этому типу джойстиков, достаточно заменить в них опрос клавиш Q, A, O, P и M на цифровые клавиши 6, 7, 8, 9, 0 (правый Sinclair-джойстик) или 1, 2, 3, 4, 5 (левый Sinclair-джойстик).

### ***Взаимодействие играющего с программой***

Итак, вы научились при помощи клавиатуры или джойстика передвигать по экрану объекты, и может создаться впечатление, что проблема взаимодействия с играющим полностью решена. Однако это пока еще не так. В реальной игре, когда по экрану движется созданный вами человек, самолет или что-то еще, возникает масса проблем, которые надо решать. Прежде всего необходимо закрасивать фон позади движущегося объекта, причем цвет фона может изменяться в различных частях игрового поля экрана. А что делать, если движущийся объект, которым вы управляете, «натолкнулся» на какое-то препятствие - стенку, дерево или какой-то движущийся объект? Наконец, как поступить с противником, в которого попал снаряд выпущенный вашим самолетом или танком?

Рассмотрим на примере игры СОКОБАН, как решаются некоторые из перечисленных задач.

Суть игры состоит в следующем: в разных частях склада стоят ящики, и управляемый вами грузчик должен передвинуть их в строго определенное место. Заметим, что склад очень напоминает лабиринт, - здесь есть узкие проходы, тупики, а ящики поставлены таким образом, что их



Рис. 5.3. Игровое пространство игры СОКОБАН.

перемещение больше напоминает головоломку, чем работу грузчика. Причем при движении по складу нельзя одновременно толкать два ящика, стоящих вплотную друг к другу, и тянуть ящик на себя. Так что не задвигайте груз в угол, иначе он останется там навсегда (рис. 53).

Достоинством игры является возможность по своему усмотрению добавлять любое количество уровней, ничего не меняя в программе, а лишь дописывая строки в оператор DATA.

А сейчас обратимся к программе игры, коротко комментируя те строки, функции которых мы уже достаточно хорошо знаем. Подробно остановимся лишь на операторах, осуществляющих взаимодействие с играющим. Но прежде покажем, как можно использовать музыкальный кодовый блок MEL1 (см главу 4) для оформления игры. Сначала запишите на ленту следующую программу-загрузчик:

```
10 CLEAR 59999
20 LOAD ""CODE 60000
```



```

480 IF A$="C" THEN INK 6
490 IF A$="D" THEN INK 7: LET P$(I,J)="D"
500 PRINT AT YY,XX;A$
510 NEXT J
520 NEXT I
530 READ X,Y
540 POKE 23658,8
550 REM -----
560 BEEP .01,-10: PRINT AT Y+2,X+6; INK 7;"A"
570 REM -- INKEY$ -----
580 IF INKEY$="P" THEN LET X1=X+1: LET Y1=Y: GO TO 660
590 IF INKEY$="O" THEN LET X1=X-1: LET Y1=Y: GO TO 660
600 IF INKEY$="A" THEN LET Y1=Y+1: LET X1=X: GO TO 660
610 IF INKEY$="Q" THEN LET Y1=Y-1: LET X1=X: GO TO 660
620 IF INKEY$="E" THEN GO SUB 900: GO TO 110
630 IF INKEY$="L" THEN GO SUB 900: GO TO 410
640 GO TO 580
650 REM -----
660 IF U$(Y1,X1)<>" " AND U$(Y1,X1)<>"D" THEN GO TO 700
670 PRINT AT Y+2,X+6; INK 7;P$(Y,X)
680 LET X=X1: LET Y=Y1: GO TO 550
700 IF U$(Y1,X1)="B" THEN GO TO 550
710 LET X2=X1-X: LET Y2=Y1-Y: LET X2=X+2*X2: LET Y2=Y+2*Y2
720 IF U$(Y2,X2)="B" OR U$(Y2,X2)="C" OR U$(Y2,X2)="E" THEN GO TO 550
740 PRINT AT Y+2,X+6; INK 7;P$(Y,X)
750 LET X=X1: LET Y=Y1: LET U$(Y,X)=P$(Y,X)
760 IF P$(Y2,X2)="D" THEN LET U$(Y2,X2)="E": GO TO 780
770 LET U$(Y2,X2)="C"
780 IF U$(Y2,X2)="C" THEN PRINT AT Y2+2,X2+6; INK 6;"C":GO TO 550
790 IF U$(Y2,X2)="E" THEN PRINT AT Y2+2,X2+6; INK 7;"E"
800 PRINT AT Y+2,X+6; INK 7;"A"
810 FOR N=1 TO 17
820 IF U$(N)="oooooooooooooooooooo" THEN GO TO 860
830 FOR M=1 TO 19
840 IF U$(N,M)="C" THEN GO TO 550
850 NEXT M
860 NEXT N
870 FOR N=-10 TO 50 STEP 3: BEEP 0.01,N: NEXT N
880 FOR N=50 TO -5 STEP -5: BEEP 0.01,N: NEXT N
890 GO TO 310
900 REM -- BEEP -----
910 FOR N=30 TO 50 STEP 3
920 BEEP 0.01,N
930 NEXT N
940 RETURN
1000 REM -- LEVEL 1 -----
1001 DATA "oooooooooooooooooooo"
1002 DATA "oooooooooooooooooooo"
1003 DATA "oooooooooooooooooooo"
1004 DATA "ooooBBBBBoooooooooooo"
1005 DATA "oooooBooooBoooooooooooo"
1006 DATA "oooooBcnoBoooooooooooo"
1007 DATA "ooBBBBnoCBBooooooooooooo"
1008 DATA "ooBnoC C Boooooooooooo"
1009 DATA "BBB B BB BnoBBBBBB"
1010 DATA "BnooB BB BBBBnoDDB"
1011 DATA "B CnoCnoooooooooooooDDB"
1012 DATA "BBBBB BBB B BBnoDDB"
1013 DATA "oooooBnoooooBBBBBBBBBB"
1014 DATA "ooooBBBBBBBnooooooooooooo"

```

```

1015 DATA "aaaaaaaaaaaaaaaa"
1016 DATA "aaaaaaaaaaaaaaaa"
1017 DATA "aaaaaaaaaaaaaaaa"
1018 DATA 12,12
1100 REM -- LEVEL 2 -----
1200 REM -- LEVEL 3 -----
1300 REM -- LEVEL 4 -----
1400 REM -- LEVEL 5 -----

```

В этой программе присутствуют все те блоки, о которых шла речь в начале книги, то есть заставка, блок игрового пространства, блок взаимодействия с играющим, а также блок оценки игровой ситуации, о котором мы подробно расскажем в следующей главе.

10...90 - эти строки кодируют графические символы, необходимые для построения игрового пространства. Букве А в режиме курсора [G] соответствует грузчик, букве В - кирпичная стенка, С - ящик, D - место, куда этот ящик необходимо поставить и Е - ящик, поставленный на свое место;

100...220 - вывод на экран заставки. Сначала рисуется двойная рамка по периметру экрана, затем печатается название игры и информация о назначении клавиш;

230 - вызов кодовой подпрограммы, исполняющей музыкальный фрагмент MEL 1. Чтобы исполнение не было бесконечным, необходимо при выборе условий компиляции в программе Wham установить определенный режим. Выберем такой, когда мелодия прекращает звучать при нажатии любой клавиши (это соответствует условию 1. KEYPRESS).

Если вы хотите исключить музыкальный фрагмент из программы, строку 230 нужно заменить на такую:

```
230 PAUSE 0: CO SUB 900
```

250 - задаются два массива: в массив U\$(17,19) в строках 450...520 записывается картина игрового пространства, а массив P\$(17,19) служит для восстановления фона и содержит только те элементы картинки, которые могут стираться при прохождении по ним грузчика или ящиков, а именно обозначения мест установки ящиков.

260 - в переменной LE хранится номер уровня;

310 - увеличение номера уровня и проверка завершения последнего уровня. Если вы дополните программу своими лабиринтами, то понадобится изменить условие IF LE>1, вместо единицы подставив количество уровней в вашей версии программы;

320...390 - очистка экрана и печать рамки, в которую затем будет помещен склад-лабиринт. Рамка рисуется с помощью псевдографических символов;

400 - печать номера уровня;

410 - оператор RESTORE (восстановить) уже упоминался нами во второй главе. Здесь он необходим для указания начала данных для каждого уровня. В выражении, следующим за RESTORE вычисляется номер строки, с которой располагаются данные игрового пространства уровней. Так для первого уровня данные должны начинаться со строки 1000, а для каждого следующего уровня номера строк увеличиваются на 100;

420...520 - в этих строках осуществляется чтение оператором READ из списка DATA информации о конфигурации стен склада, расположении ящиков и тех местах, куда эти ящики необходимо передвинуть. Буквы в строках 1001...1017 набираются в режиме курсора [G] и соответствуют графическим символам, определенным в начале программы;

430 - здесь строка за строкой считываются данные из списка DATA и

записываются в двумерный массив  $U(17,19)$ , причем первая координата указывает на положение символа по вертикали, а вторая - на его положение по горизонтали;

440 - эта строка служит для ускорения построения экранного изображения. Если считанная строка символов целиком состоит из пробелов, то происходит переход на следующий «виток» цикла;

450...510 - во внутреннем цикле анализируется очередная строка массива  $U$  и формируется изображение на экране;

460 - перед тем, как поставить тот или иной символ на экран, его координаты (переменные  $Y$  и  $X$ ) изменяются таким образом, чтобы все изображение лабиринта оказалось посередине. В переменную  $A$  заносится анализируемый символ;

470, 480, 490 - в зависимости от выводимого символа устанавливается его цвет: стены (графический символ  $B$ ) будут иметь красный цвет, ящики ( $C$ ) окрасятся желтым цветом, а места для установки ящиков ( $D$ ) будут белыми. Кроме того, если в анализируемой строке массива встречается графический символ  $D$ , он записывается в соответствующее место другого массива -  $P$ ;

500 - печать очередного символа в место с заранее рассчитанными координатами;

530 - последними в операторах  $DATA$  читаются координаты грузчика (в отличие от ящиков, фрагментов стенки и т. д. у грузчика нет номера в массиве, а есть только координаты);

540 - оператором  $POKE 23658,8$  устанавливается режим ввода заглавных букв, чтобы в дальнейшем при опросе клавиатуры избежать лишних проверок. Более подробно об операторе  $POKE$  мы расскажем позже;

560 - изображение грузчика выводится на экран после короткого звукового сигнала;

Далее идут строки, которые образуют все вместе блок взаимодействия с играющим, представляющий в данный момент для нас наибольший интерес;

580...640 - блок, в котором осуществляется управление движением грузчика путем изменения его координат, повтор текущего уровня (клавиша  $L$ ) и выход на начальную заставку при нажатии клавиши  $E$ ;

660 - координаты  $X1, Y1$  показывают, куда перемещается грузчик, а  $X, Y$  - место на экране, где он находится сейчас. Здесь проверяется клетка, куда должен встать грузчик. Если она не фоновая (не пробел и не «ромбик»), то программа переходит на строку 700;

670 - в клетке, где ранее находился грузчик, нужно восстановить прежний фон, заполнив ее тем символом, который был здесь раньше. Требуемый символ может быть либо пробелом, либо «ромбиком», соответствующим графическому символу  $D$ . Этот символ мы берем из массива  $P$  и выводим на экран;

680 - переписывание новых координат в старые и переход на строку 550, где осуществляется вывод на экран изображения грузчика. После этого происходит новый цикл изменения координат, то есть перемещение грузчика на один шаг;

700 - проверяется условие: если клетка, куда должен встать грузчик, заполнена фрагментом стенки, то переход на строку 550, выводящую изображение грузчика. Координаты при этом не изменяются, так как идти сквозь стенки нельзя;

710 - все графические символы мы перебрали, кроме ящика, который может обозначаться символами  $C$  или  $E$ . В этой строке сначала вычисляются направление перемещения ящика, а затем координат клетки, расположенной впереди него ( $X2, Y2$ ), то есть того места, куда он перемещается;

720 - проверка содержимого клетки с координатами (X2, Y2). Если там стена, другой ящик или ящик, поставленный на свое место, то все остается по-старому (безусловный переход на 550);

740 - так же, как и в строке 670, восстанавливается фон клетки, где ранее находился грузчик;

750 - присваивание переменным, обозначающим координаты грузчика новых значений. Место в массиве U\$, откуда убран ящик, заполняется фоком из массива P\$;

760 - Если ящик встал на «ромбик», то в массиве U\$ на этом месте появляется символ E, обозначающий ящик, поставленный на свое место.

В противном случае (строка 770) ящик, как и прежде, обозначается символом C;

780 - если ящик не встал на свое место, то он закрашивается желтым цветом. Далее - безусловный переход на строку 550;

790 - если ящик установлен, куда нужно, то он закрашивается белым цветом;

800 - грузчик появляется на новом месте. Эта строка введена для того, чтобы изображение грузчика не пропадало на время последующих проверок;

810...860 - проверка: остались ли не поставленные на свои места ящики или нет. Строка 820 здесь введена для ускорения цикла. Если остался хотя бы один ящик, то происходит переход на строку 550, после чего весь цикл проверок повторяется;

870, 880 - если оказалось, что все ящики поставлены на свои места, то звучит трель, и мы попадаем на следующую строку;

890 - переход на новый уровень, то есть в новый лабиринт-склад, к новому расположению ящиков;

900 - подпрограмма звукового сопровождения начала игры (заменяет при необходимости кодовую подпрограмму MEL 1).

Со строки 1000 начинаются блоки данных, описывающих различные уровни игры, вам предоставляется возможность самостоятельно сформировать лабиринты уровней, начиная со второго. Впрочем, ничто не мешает вам изменить и первый уровень. Нужно помнить только, что каждая строка в кавычках должна состоять из 19 символов, а самих строк должно быть 17. В конце каждого блока необходимо указывать два числа, определяющих начальные координаты грузчика. И последний совет: количество ящиков в лабиринте (графический символ C) должно совпадать с количеством «ромбиков» (символ D).

## 6. ОЦЕНКА ИГРОВОЙ СИТУАЦИИ

«Оценка игровой ситуации» - довольно неоднозначное понятие. В него может входить как простой подсчет очков, числа попаданий или любой другой числовой величины, так и общий контроль ситуация в игровом пространстве, которую мы видим на экране. Разработчики игровых программ обычно стараются в равной мере использовать как количественную, так и качественную оценку, однако специфика игры часто диктует свои требования. Так, в известной игре Tetris нас больше интересует положение в «стакане», целиком захватывает процесс борьбы с фигурками, их правильная укладка, а об очках мы вспоминаем, когда игра уже закончена. Тем не менее, количество набранных очков в этой игре тоже играет важную роль, так как без них невозможно узнать, становится ли выше от раза к разу уровень нашего мастерства и каков он по сравнению с уровнем других игроков. Однако есть и такие популярные игры, как например, Tomahawk, где игровое пространство выглядит очень просто, зато количественная оценка ситуации изобилует всевозможной числовой и текстовой информацией.

Опираясь на эти и другие примеры игровых программ, можно сделать общие выводы относительно того, как формировать блок оценки игровой ситуации

Если создаваемая вами игра динамична, с быстро изменяющейся ситуацией на экране, с большим числом движущихся объектов, подобно Flying Shark, то не следует злоупотреблять числовыми данными, ограничив их одним-двумя. При этом располагать данные лучше где-то в дальней части экрана, делая изображения цифр небольшими и неброского цвета.

Если же вы разрабатываете спокойную игру с медленно развивающимся сюжетом типа Video Pool, то в этом случае все параметры игры могут характеризоваться числовыми величинами, и количество сообщений ограничивается лишь местом на экране. Возможен даже вариант, когда для демонстрации оценочных числовых характеристик отводится целый экран. Что касается оформления информации, то в рассматриваемом случае цифровые надписи желательно делать максимально отличными одна от другой, используя для этой цели символы различных форм и размеров, всевозможные рамки и цветовое оформление. Кроме того, будет нелишним сопроводить выдачу на экран особо ценной информации звуковыми эффектами.

А теперь от общих соображений перейдем к конкретным программам. Для начала рассмотрим те программы, которые использованы нами в предыдущих главах. В игре ЧИСЛА оценка результата игры чисто цифровая; нас интересует здесь даже не само неизвестное число, а количество шагов, которое делает играющий для его угадывания. Программа ЖИЗНЬ включает в себя оба вида оценок, то есть мы следим за очередной популяцией и оцениваем ее характер - идет размножение или угасание. Но для того чтобы уловить тонкости процесса развития, выведены и количественные оценки: номер поколения и количество особей в нем. Наконец, в игре СОКОБАН нет ни одной числовой характеристики, но от этого игра не становится менее интересной.

### ***Элементы оценок***

В качестве примеров будет полезно рассмотреть несколько фрагментов программ, в которых иллюстрируются разные виды оценок. Больше всего в играх распространены всевозможные подсчеты: числа очков, числа попаданий, количества оставшихся попыток (жизней), снарядов и т. д. Объясняется это прежде

всего простотой программной реализации в сочетании с высокой информативностью этих оценок. Действительно, если экран сигнализирует о том, что осталось всего 1-2 снаряда, то вы наверняка не броситесь сломя голову в гущу сражения, а будете изо всех сил нажимать на клавиши или ручку джойстика, стремясь уйти от выстрелов противника.

Предположим, что в некоторой игре требуется подсчитать количество оставшихся жизней, причем на момент начала игры их было 10. Тогда, если опустить «игровые» строки программы и, наоборот, выделить те, которые отвечают за выполнение поставленной задачи, то можно предложить следующий вариант:

```
50 LET KOL=10: LET POP=0: LET TIME=999 100 REM * START GAME *
480 LET KOL=KOL-1
485 PRINT AT 1,5; PAPER 0; INK 4;"LIVES: ";KOL
490 IF KOL>0 THEN GO TO 100
500 PRINT AT 10,11; PAPER 2; INK 7;"GAME OVER!"
```

50 - в этой строке задаются начальные значения всех переменных, которые будут использованы в программе в качестве счетчиков! Восклицательный знак - это не типографская опечатка, а попытка привлечь ваше внимание к важному элементу программы. Сколько сил, нервов и времени бывает потрачено программистами впустую только оттого, что где-то забыли задать начальные значения каких-то величин. Для будущих опытов в программировании опишем здесь переменные, которые нам еще понадобятся:

KOL - количество жизней;

POP - число попаданий;

TIME - оставшееся до конца игры время;

100 - с этой строки начинается часть программы, реализующая алгоритм игры;

480 - после того, как на экране завершится один цикл действий, обусловленных сюжетом игры, количество жизней уменьшается на единицу. Программа может попадать на эту строку, например, при поражении вашего корабля торпедой противника или после истечения времени, отведенного на одну попытку;

490 - в следующей после изменения счетчика строке чаще всего определяется, на какую строку перейти, если значение управляющей переменной (у нас - KOL) превысит или окажется меньше заданной величины. В нашем примере - если жизни еще остались (KOL не равна 0), то можно продолжить игру (управление передастся на строку 100), в противном случае (KOL=0) игра должна быть закончена;

500 - если условие в строке 490 не выполняется, то выводится сообщение о том, что игру пора начинать сначала или загружать другую.

Большое количество популярных игр связано с попаданием одних предметов в другие, что в переводе на язык программирования означает столкновение спрайтов, а если быть еще более точным и перейти на язык математики, - совпадение соответствующих координат. Чтобы реализовать в программе новую оценку - оценку факта попадания снаряда в цель, - достаточно вставить в нашу программку всего одну строку:

```
200 IF X=XS AND Y=YS THEN LET POP=POP+1:PRINT AT 1,25; PAPER 2; INK 5;"SCORE: ";POP:
GO SUB 1000
```

Здесь X и Y соответствуют координатам цели, а XS и YS - координатам снаряда. Если горизонтальные и вертикальные координаты одновременно совпадают, то счетчик POP увеличивается на единицу, после чего на экран выводится информация об общем количестве попаданий в цель. Поскольку каждое попадание желательно сопроводить аудио- и видеоэффектами, то на этот случай в строке предусмотрен вызов соответствующей подпрограммы, которую, как мы надеемся,

вы уже можете написать самостоятельно. Если хотя, бы одна из координат не совпадает, то операторы строки 200 после THEN выполняться не будут, переменная POP останется без изменений и программа перейдет на следующую строку.

Остановимся еще на одной величине, которая наиболее часто используется в игровых программах, - времени игры. Это может быть время, оставшееся до конца игры, или время, потраченное на игру с момента ее запуска. Вставим в приведенный выше фрагмент программы следующие строки:

```
300 LET TIME=TIME-1: PRINT AT 0,25; INK 7; PAPER 1; "TIME: ";TIME;" "
310 IF TIME=0 THEN PRINT AT 12,10; PAPER 4; INK1;"OUT OF TIME": GO TO 480
```

Как вы уже догадались, глядя на строку 300, здесь будет выводиться на экран время, оставшееся до конца игры (напомним, что начальная величина TIME = 999 установлена в строке 50). Одновременно со счетом происходит постоянный контроль переменной TIME в строке 310, и как только условие этой строки выполнится, на экране появится текст OUT OF TIME, который говорит о том, что время, отпущенное на выполнение задания в игре, закончилось. Затем программа переходит к строке 480, лишаящей игрока одной жизни.

Этими примерами, разумеется, не исчерпывается та информация, которую можно выводить на экран во время игры. Кроме числовых характеристик, хорошо смотрятся различные текстовые сообщения о ходе игры, особенно если они выполнены на русском языке, да еще хорошим шрифтом. От того, какая информация будет появляться в процессе игры и как она будет подана, зависит очень многое, поэтому блоку оценки всегда следует уделять должное внимание.

### ***Комплексная оценка игры***

Давайте обратимся к программе БОМБЫ (рис. 6.1), в которой присутствуют оба типа оценки игровой ситуации.

Несколько слов о том, что должно происходить на экране. После нажатия любой клавиши будет нарисован слой воды синего цвета, на дне возникнет зеленая растительность, и где-то посреди нее расположится красный люк. Слева направо начнет двигаться корабль



Рис. 6.1. Игровое пространство игры БОМБЫ.

белого цвета, причем скорость корабля в течение игры будет изменяться, чтобы нельзя было вычислить заранее время пуска бомбы. Нажимая любую клавишу, вы открываете бомбовый люк, и оттуда вырывается и устремляется к поверхности бомба, которая должна быть выпущена с таким расчетом, чтобы, дойдя до поверхности, столкнуться с кораблем. Если вы попали в корабль, то происходит взрыв, раздаются соответствующие звуки, и на табло появляется информация о попадании.

Программа 29. БОМБЫ.

```
30 FOR N=0 TO 71
40 READ S
```

```

50 POKE USR "A"+N,S
60 NEXT N
70 DATA 60,60,60,60,60,24,8,16
80 DATA 74,42,186,177,255,255,255,255
90 DATA 0,0,0,0,0,0,255,255
100 DATA 0,0,66,66,66,195,255,255
110 DATA 0,0,0,0,0,255,255,255
120 DATA 0,0,24,96,0,0,3,12
130 DATA 0,64,33,29,29,255,118,63
140 DATA 0,126,255,182,255,255,219,255
150 DATA 0,2,132,184,184,255,108,248
200 BORDER 1: PAPER 1: INK 6: CLS
205 PRINT AT 3,6; "*****"
210 PRINT AT 7,6; "*****"
220 FOR N=3 TO 7
230 PRINT AT N,6; "*"
240 PRINT AT N,25; "*"
250 NEXT N
260 PRINT AT 5,11; INK 3; "ВППППППППВ"
270 LET C=2: LET A$="PRESS ANY KEY TO CONTINUE"
280 FOR N=1 TO 25
290 PRINT AT 18,N+3; INK C;A$(N)
300 LET C=C+1: IF C>7 THEN LET C=2
310 NEXT N
320 IF INKEY$="" THEN GO TO 280
325 REM -----
330 BORDER 0: PAPER 0: INK 1: CLS
340 LET U=INT (RND*7+4)
350 LET KY=U-1
360 LET B=INT (RND*10+12)
370 LET B0=0: LET BX=B+1: LET BY=20
380 LET SC=0: LET BOM=10
400 FOR N=U TO 20
410 PRINT AT N,0; "FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF"
420 NEXT N
430 PRINT AT 21,0;INK 4;"BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB"
440 PRINT AT 21,8; INK 2;"CEC"
450 LET KX=0: LET KX1=KX: LET S=RND/5+.05
460 PRINT AT KY,KX-1; " ": PRINT AT KY,KX; INK 7;"GHI"
465 IF BOM=0 AND BY=20 THEN GO TO 700
470 IF INKEY$="" AND B0=0 THEN PRINT AT 21,B+1; INK 2;"E": GO TO 550
475 IF B0=0 THEN LET BOM=BOM-1
480 LET B0=1: PRINT AT 21,8+1; INK 2;"D"
490 LET BY=BY-.5: BEEP .003,BY
500 IF BY<U-1 AND BX>=KX AND BX<=KX+2 THEN LET BY=20:LET B0=0: LET SC=SC+1: GO SUB
1000: GO TO 450
510 IF BY<U-1 THEN LET B0=0: LET BY=20: PRINT AT U-1,BX; " ": GO TO 550
515 PRINT AT BY+1,BX; INK 1;"E"
520 PRINT AT BY,BX; INK 7;"A"
550 LET KX1=KX1+S: LET KX=INT KX1: IF KX1>29 THEN PRINT AT U-1,28;"NNN": GO TO 450
560 PRINT AT 0,5; INK 5;"SCORE:";SC;"NNNNNBOMB:";BOM;" "
600 GO TO 460
700 BORDER 1: PAPER 1: CLS
710 LET A$="WY POPALI "+STR$ SC+" RAZ IZ 10"
720 LET M=LEN A$
730 LET C=2
740 FOR N=1 TO M
750 PRINT AT 11,N+5; INK C;A$(N)
760 LET C=C+1: IF C>7 THEN LET C=2
770 NEXT N

```

```

780 IF INKEY$="" THEN GO TO 740
790 GO TO 200
1000 REM -- WZRYW ---
1005 FOR M=1 TO 2
1010 FOR N=2 TO 7
1015 BEEP .003,0: BEEP .003,0: BEEP .003,0
1020 PRINT AT KY,KX; INK 2;"GHI"
1030 BORDER N
1035 BEEP .003,0: BEEP .003,0: BEEP .003,0
1040 PRINT AT KY,KX; INK 3;"GHI"
1045 PAUSE 10
1050 NEXT N: NEXT M
1055 PRINT AT KY,KX;"ППП"
1060 BORDER 0
1070 RETURN

```

30...150 - кодирование и ввод в память компьютера графических символов, которым соответствуют следующие изображения:

- A - бомба,
- B - подводная растительность,
- C - фрагмент подводного люка,
- D - открывающаяся часть люка,
- E - крышка люка,
- F - вода,
- G - нос корабля,
- H - средняя часть корабля,
- I - корма корабля;

200...270 - вывод на экран заставки игры;

280...310 - последняя часть заставки выводится таким образом, чтобы буквы переливались разными цветами;

320 - вывод разноцветной фразы будет продолжаться до тех пор, пока игрок не нажмет какую-либо клавишу;

330...380 - переменным присваиваются первоначальные значения. U - уровень воды, KX - горизонтальная координата корабля, B - положение люка по горизонтали, B0=0 - бомба не выпущена, B0=1 - бомба из люка вышла, но до поверхности не дошла, BX - координата x бомбы, BY координата y бомбы, SC - набранные очки, BOM - количество имеющихся бомб, S - скорость движения корабля;

400...420 - на экране изображается толщина воды;

430 - вывод изображений зеленой растительности на дне моря;

440 - установка пускателя бомб красного цвета, символ E изображает закрытый люк;

450 - задание начальных координат и скорости корабля;

460 - вывод корабля;

465 - проверяется, остались ли в наличии бомбы;

470 - если никакая клавиша не нажата, и бомба не выпущена ( $B0=0$ ), то рисуется закрытая крышка люка, после чего происходит переход на строку 550;

480 - если клавиша нажата, то выводится изображение открытой крышки;

490 - изменение вертикальной координаты бомбы BY и подача звукового сигнала, свидетельствующего о ее движении вверх. Во время подъема бомбы высота сигнала меняется пропорционально переменной BY;

500 - проверка следующих условий: бомба дошла до поверхности и бомба соприкоснулась с кораблем. Если оба условия выполняются, то вызывается подпрограмма WZRYW, имитирующая взрыв, а затем появляется новый корабль, и

все начинается заново;

510 - проверка условия бомба дошла до поверхности, но не попала в корабль. Если это условие выполняется, то устанавливаются значения некоторых переменных, изображение бомбы на поверхности закрашивается, после чего происходит переход на строку 550;

515, 520 - изображение на экране бомбы, поднимающейся к поверхности из глубины;

550 - изменение горизонтальной координаты корабля, проверка - дошел он до края экрана (моря) или нет;

560 - вывод числовых характеристик игры - числа набранных очков и количества оставшихся бомб. Таким образом, результаты игры постоянно присутствуют в двух оценочных надписях: SCORE и BOMB;

600 - если бомбы еще есть, то переход на начало цикла игры;

700...780 - вывод на экран заключительной надписи о результатах игры. Каждая буква текста переливается разными цветами;

1000...1070 - подпрограмма со звуковыми эффектами, имитирующая взрыв корабля.

Подводя итог, можно сказать следующее.

В этой простой игре присутствуют обе оценки: качественная и количественная.

Качественную оценку можно соотнести со строками программы 450...550, то есть с теми операторами, которые формируют сложную динамическую (меняющуюся во времени и пространстве) картину, на которую играющий в состоянии оказывать определенное воздействие. Это воздействие влияет на результат игры и является чрезвычайно важным фактором.

Числовые характеристики игры, которые индицируются во время всего сражения, играют в этой игре вспомогательную роль. Их обеспечивают строки программы 560 и 700...770, причем тот набор данных, который мы видим, является минимально необходимым, то есть без этих результатов невозможно судить о ходе игры. Однако можно учитывать и другие параметры: скорость корабля, глубину цели, размеры цели и т. д. Это потребует внесения некоторых изменений в программу и преобразования игрового пространства. Попробуйте!

## 7. СИСТЕМНЫЕ ПЕРЕМЕННЫЕ

Наверное, ни один уважающий себя программист не ограничивается только теми средствами, которые предлагает стандартный Spectrum-Бейсик. Поэтому мы в нашей книге хотим сказать несколько слов о других возможностях, предоставляемых операционной системой компьютера. В первую очередь речь пойдет о так называемых системных переменных.

Системными переменными называют специальную область памяти компьютера, которую операционная система использует для хранения различных данных и промежуточных результатов при выполнении тех или иных операторов. Знание и умение пользоваться этой областью дает удивительные плоды и в одних случаях позволяет обогатить программу на Бейсике совершенно потрясающими эффектами, а в других - сократить ее или увеличить ее быстродействие.

С некоторыми из системных переменных мы уже встречались: вспомните, например, оператор РОКЕ 23658,8, который устанавливает режим ввода прописных букв. Все прочие системные переменные можно изменять точно так же, записывая в ячейки памяти с помощью оператора РОКЕ числа из определенного диапазона. Записываемые числа должны быть целыми и в пределах от 0 до 255.

Однако экспериментировать с системными переменными, так же, как мы это советовали для изучения операторов, не рекомендуется. Если вам неизвестен смысл той или иной ячейки памяти, лучше оставить ее в покое и не испытывать судьбу, потому что при изменении системных переменных мы вторгаемся в святая святых компьютера. При грубом и неумелом вмешательстве в операционную систему предугадать результат бывает совершенно невозможно.

Системные переменные были описаны в литературе достаточно подробно, поэтому мы не станем повторяться, а расскажем лишь о некоторых из них и о том, как их можно использовать в игровых программах. Если же вы захотите более подробно разобраться в этом вопросе, советуем обратиться к книге [1].

### ***Клавиатура***

Мы уже немало рассказали о способах управления программой и спрайтами, однако разговор этот можно продолжить. Ряд ячеек в области системных переменных имеет непосредственное отношение к клавиатуре, а значит и к теме нашей книги, поэтому рассказ о них не повредит. Вы прекрасно знаете, что если при наборе или редактировании строк программы понадобится ввести несколько одинаковых символов подряд (или удалить часть текста), то вовсе необязательно бешено колотить по одной и той же клавише, а достаточно слегка надавить на нее и подождать с полсекунды - нужный символ сам начнет быстро размножаться. В игровых программах также бывает полезно использовать принцип задержки перед повторением какого-либо действия: передвижения главного героя, перемещения курсора в меню и т. п. Для этого проще всего блок управления начинать оператором PAUSE 0, например:

```
100 PAUSE 0: LET K$=INKEY$  
110 IF K$="Q" THEN LET Y=Y-1  
120 GO TO 100
```

Правда, в этом случае придется отказаться от джойстика, так как действие оператора PAUSE прерывается только от клавиатуры, но тут уж ничего не поделаешь - нужно выбирать. При использовании данного способа управления у вас может возникнуть желание изменить время задержки перед повторением или

частоту самих повторений. Знание системных переменных позволит вам с легкостью осуществить это желание. Изменение времени задержки производится оператором

```
POKE 23561,N
```

где N - любое число от 0 до 255. Только не переборщите, иначе проще будет не дожидаться повторений, а упорно тыкать в клавиатуру. Полезно знать, что начальное значение, записанное в ячейке 23561, равно 35, а время задержки исчисляется в 50-х долях секунды. Частота повторений регулируется системной переменной, расположенной по адресу 23562. Здесь единица измерения времени такая же, как и для предыдущей переменной - 1/50 секунды, а начальное значение равно 5.

В длинных и продолжительных циклах, особенно если нельзя останавливать программу оператором PAUSE 0, бывает очень трудно определить нажатие какой-либо клавиши и возникает ситуация, когда жми, не жми - все впустую. Ведь компьютеру в этом случае нужно выполнить огромное количество разных операций, а когда очередь доходит до IF INKEY\$, оказывается, что клавиша давно уже отпущена. Бороться с подобными ситуациями поможет системная переменная LAST\_K, занимающая ячейку 23560. В этой ячейке хранится код последней нажатой клавиши, причем остается он там и при отпуске клавиши до тех пор, пока вы не нажмете другую или не выключите компьютер. Таким образом, для того чтобы определить, какая из клавиш была нажата, нужно сначала узнать код ячейки 23560, а затем перевести его в символ.

Читать значения ячеек памяти позволяет функция PEEK addr, где addr - адрес от 0 до 65535, а переводит числовые значения в символы функция CHR\$ kod. Посему, строки типа

```
LET K$=INKEY$
```

в описываемых случаях можно заменять на LET K\$=CHR\$ PEEK 23560

Помните только, что при таком способе опроса клавиатуры вам самим придется позаботиться о своевременном сбросе переменной LAST\_K в конце блока управления, вставив в программу оператор POKE 23560,0.

Говоря о клавиатуре, необходимо упомянуть и о системной переменной PIP (адрес 23609), изменяя которую, можно установить желаемую длительность звука, издаваемого компьютером при нажатии клавиш. В программах это бывает полезно, если используется оператор INPUT. Для разнообразия можно даже вводить различные данные с разным «звуковым сопровождением». Например, наберите и «послушайте» такую программку:

```
10 INPUT "0..255-->";P: LET P=INT P
20 IF P<0 OR P>255 THEN GO TO 10
30 POKE 23609,P
40 GO TO 10
```

Комментировать эту программку, наверное, нет особой нужды, скажем только, что после ввода целого числа от 0 до 255 оно записывается в системную переменную PIP и следующий ввод сопровождается уже другим звуком.

## **Экран**

Системные переменные охватывают все стороны жизнедеятельности компьютера, и уж конечно не обойден вниманием такой важный момент, как вывод информации на экран. В первую очередь поговорим о тех эффектах, которые невозможно получить только с помощью операторов Бейсика.

Как вы уже знаете, в Бейсике отсутствует возможность изменения цвета «чернил» в служебном окне - он устанавливается автоматически. Нельзя в нижних

строках также задать режим повышенной яркости или мерцания. Но это становится реальным, если отказаться от оператора BORDER и воспользоваться одноименной системной переменной BORDER, находящейся по адресу 23624. Небольшая трудность возникает при определении кода атрибутов, справиться с которой поможет формула:

$$ATTR=INK+PAPER \times 8+BRIGHT \times 64+FLASH \times 128$$

То есть если мы хотим делать надписи в служебном окне желтыми «чернилами» на синем фоне с повышенной яркостью, нам нужно в ячейку 23624 записать число  $6+1 \times 8+1 \times 64+0 \times 128=78$

выполнив оператор POKE 23624,78. Но помните, что выполнять эти действия надо после оператора BORDER, иначе все моментально вернется на круги своя.

Не лишне будет знать, что и с основным экраном можно поступать аналогичным образом, и вместо цепочки операторов INK, PAPER, BRIGHT и FLASH записывать код атрибутов, рассчитанный по приведенной формуле, в системную переменную ATTR\_P (адрес 23693). Конечно, текст программы при этом будет читаться труднее, но зато вы достигнете большего быстродействия и сократите объем самой программы.

Вы, наверное, еще помните трудности, связанные в выводе текстов в две нижние строки экрана. Но существование такого понятия как служебное окно вовсе не означает, что вы обречены пожизненно довольствоваться таким положением вещей. Оказывается, служебное окно при желании можно и вовсе убрать с экрана. Для этого достаточно записать в ячейку 23659 ноль (обычно там «сидит» двойка). Но будьте осторожны! Перед остановкой или завершением программы, а также при использовании оператора INPUT значение этой системной переменной обязательно нужно восстановить, включив в программу оператор POKE 23659,2. Если этого не сделать, то при попытке выдачи любого сообщения (даже если это сообщение ОК) компьютер, не обнаружив служебного окна, очень огорчится, и, скорее всего, огорчит и вас. Попробуйте ввести с клавиатуры POKE 23659,0 (только если в памяти нет какой-нибудь ценной программы, чтобы нечего было терять) и посмотрите на результат. Надо сказать, что этот оператор программисты часто включают в свои творения с целью защитить их от несанкционированного доступа к тексту программы. Но это между прочим, а в основном использование ячейки 23659 подразумевает именно изменение размеров служебного окна. Приведем такой пример:

```
10 POKE 23659,0
20 PRINT AT 22,5;"LINE 22"TAB 5;"LINE 23"
30 POKE 23659,2
40 PAUSE 0
```

Выполнение этой программки приведет к выключению служебного окна и появлению в строке 22 экрана, начиная с пятой позиции, сообщения LINE 22. Апостроф переместит позицию печати на следующую строку, в которой также в пятой позиции появится надпись LINE 23. Затем служебное окно восстановится и программа будет ожидать нажатия любой клавиши. Несмотря на то, что после удаления служебного окна основной экран займет все 24 строки, выполнять оператор PRINT AT 23,0 ни в коем случае нельзя, так как это приведет к выдаче сообщения Integer out of range со всеми вытекающими отсюда последствиями. Именно поэтому в строке 20 приведенного примера для перевода позиции печати в нижнюю строку экрана используется апостроф.

И еще об одной системной переменной, касающейся экрана. Представим себе такую ситуацию, когда нужно нарисовать что-нибудь (скажем, рамку) в режиме наложения (OVER 1) Напишем фрагмент программы, выполняющий это задание:

```

100 OVER 1
110 PLOT 80,88
120 DRAW 80,0: DRAW 0,40
130 DRAW -80,0: DRAW 0,-40
140 OVER 0

```

Казалось бы, все очень просто, о чем тут можно говорить? Однако это не так. Присмотритесь повнимательнее к тому, что получилось на экране. Нижний левый угол прямоугольника оказался ущербным: исчезла одна точка, та самая, которая ставится вначале оператором PLOT. Каким же образом исправить дефект? Можно, конечно, исхитриться и не доводить последнюю линию до конца, написав DRAW 0,-39 вместо DRAW 0,-40. Но это не всегда удобно, и тут-то вам на помощь придет системная переменная COORDS, хранящаяся в двух ячейках 23677 и 23678. Она содержит координаты последней поставленной точки. Замените в примере строку 110 на

```
110 POKE 23677,80: POKE 23678,88
```

- и дефект исчезнет: ведь точка на самом деле не ставится, а просто перемещается PLOT-позиция!

### **Наборы символов**

Не менее важны в игровых программах вопросы, касающиеся наборов символов, ибо именно в дополнительных наборах символов хранятся созданные вами спрайты. В третьей главе мы уже упоминали о системной переменной CHARS. Тогда для смены набора символов мы изменяли одну ячейку памяти 23607. На самом же деле эта переменная занимает два байта - 23606 и 23607, в которых хранится адрес, на 256 меньше адреса текущего набора. Вы можете создать не один, а несколько дополнительных наборов символов: один, к примеру, будет содержать русский шрифт, а другие - спрайты, причем для каждого уровня спрайты могут быть свои. Затем, меняя значение переменной CHARS, можно подключать к работе любой из дополнительных наборов. Если пользоваться только ячейкой 23607, то альтернативные фонты могут быть расположены по строго определенным адресам с шагом в 256 байт, использование же и второй половинки переменной CHARS позволит не заботиться о местоположении фонта и загружать его в любое удобное место памяти. Если, например, новый набор загрузить по адресу 60000, то включаться он будет таким фрагментом программы:

```

1000 LET AD=60000-256
1010 POKE 23607,INT (AD/256)
1020 POKE 23606,AD-256*PEEK 23607

```

Вернуться к стандартному знакогенератору можно с помощью строки:

```
9990 POKE 23606,0: POKE 23607,60
```

С помощью системной переменной CHARS могут быть получены довольно интересные эффекты. Рассмотрим один из них на примере такой программки:

```

10 INK 2: PAPER 0: BRIGHT 1: BORDER 0: CLEAR 59999
20 FOR N=0 TO 7: POKE 60000+N,2^N: POKE 60008+N,2^N: NEXT N
30 PRINT AT 10,11: INK 4: "G A M E"
40 POKE 23607,233
50 FOR I=0 TO 7
60 POKE 23606,96+I
70 PRINT AT 8,8: "aaaaaaaaaaaa";AT 12,8: "aaaaaaaaaaaa"
80 FOR N=9 TO 11: PRINT AT N,8: " ";AT N,20: " ": NEXT N
85 IF INKEY$<>" " THEN GO TO 100
90 NEXT I: GO TO 50
100 POKE 23606,0: POKE 23607,60

```

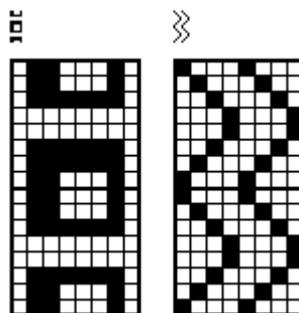


Рис. 7.1. Примеры фактур.

10 - установка атрибутов экрана и бордюра, а затем выделение свободной области памяти «над» бейсик-программой, начиная с адреса 60000. Кроме всего прочего оператор CLEAR попутно очищает экран, поэтому нет надобности в операторе CLS;

20 - в реальной программе эта строка может отсутствовать, если в памяти уже загружен дополнительный набор символов. Здесь же два соседних символа в цикле заполняются фактурой, плавно переходящей одна в другую. На самом деле можно использовать не два символа, а сколь угодно много; главное, чтобы каждый следующий продолжал предыдущий, а последний повторял первый, как показано на рис. 7.1;

30 - на экране появляется слово GAME зеленого цвета; 40 - заносим в старший байт системной переменной CHARS заранее рассчитанное значение:  $\text{INT}((60000-256)/256) = 233$

Далее в цикле (строки 50...90) адрес символьного набора постепенно смещается, и на экран вокруг слова GAME пробелами (которые в измененном виде выглядят как наклонные черточки) выводится рамка. Так как адрес набора смещается, то и символы начинают видоизменяться, что создает иллюзию прокручивания рамки;

60 - изменяется младший байт переменной CHARS. Поскольку в данном случае его значение не превышает 255, то старший байт остается без изменений;

70, 80 - рисуется рамка;

85 - если нажата любая клавиша, программа переходит на строку 100;

90 - конец цикла и после его завершения -- переход для повторения;

100 - восстанавливается стандартный символьный набор.

Операционная система ZX Spectrum позволяет менять не только адрес полного набора символов, но также и местоположение символов определяемых пользователем (UDG), начальный адрес которых хранится в ячейках 23675 и 23676. Известная вам функция USR "A" возвращает число, записанное именно в этих ячейках и равноценна выражению  $\text{PEEK } 23675 + 256 * \text{PEEK } 23676$ . Попробуйте самостоятельно изменить приведенную выше программку таким образом, чтобы задействовать в ней вместо полного набора символы UDG. Помните, что в этом случае адрес символов уменьшать на 256 уже не нужно.

### **Разное**

Системная переменная SEED, занимающая адреса 23670 и 23671, хранит значение, используемое функцией RND для расчета очередного псевдослучайного числа. В ней не было бы ничего примечательного, если бы не одна особенность, делающая ее в некотором смысле уникальной. В Spectrum-Бейсике оператор RANDOMIZE задает начальное значение для функции RND. Этот оператор выполняет одно-единственное действие - заносит в переменную SEED указанное число. В этом легко убедиться, если ввести строку:

```
RANDOMIZE 12345: PRINT PEEK 23670+256*PEEK 23671
```

В результате вы увидите на экране число 12345, то есть число, заданное оператором RANDOMIZE. Это обстоятельство можно с успехом использовать вместо расчетов младшего и старшего байтов адреса нового набора символов или для других целей. Во всяком случае, запись

```
RANDOMIZE AD: POKE 23606,PEEK 23670: POKE 23607,PEEK 23671
```

равнозначна записи

```
POKE 23607,INT(AD/256): POKE 23606,AD-256*PEEK 23607
```

но в первом случае мы обходимся без лишних расчетов. Правда, такой способ допустим только в тех случаях, когда число AD (или любое другое) заведомо не равно нулю, потому что оператор RANDOMIZE 0 или просто RANDOMIZE действует несколько по-иному. В этом случае в системную переменную SEED переписывается число из другой переменной - FRAMES которую можно назвать системным таймером. Располагается она в трех ячейках памяти: 23672, 23673 и 23674. После включения компьютера в ней содержится ноль, но как только система заработает, она начинает увеличиваться на единицу через каждые 1/50 секунды. Таким образом, прочитав значение из FRAMES можно узнать сколько времени прошло с момента включения компьютера. Это позволяет при желании встраивать с игровые программы часы, а также сопоставлять любые игровые моменты с реальным временем. Представим программку электронных часов, которую вполне можно использовать и в играх (см. программу МАКСИТ в приложении 1):

```
10 INPUT "h: ";h;"мм: ";m;"ппс: ";s
20 LET tim=(h*3600+m*60+s)*50
30 LET t3=INT (tim/65536): LET t2=INT ((tim-t3*65536)/256): LET t1=(tim-t2*256-
t3*65536)
40 POKE 23672,t1: POKE 23673,t2: POKE 23674,t3
100 LET tim=(PEEK 23672+256*PEEK 23673+65536*PEEK 23674)/50: LET h=INT (tim/3600):
LET m=INT ((tim-h*3600)/60):LET s=tim-h*3600-m*60
110 IF h>=24 THEN POKE 23672,0: POKE 23673,0: POKE 23674,0: GO TO 100
120 LET s1=INT s: LET s=s-s1: LET t$=":": IF s>.7 THEN LET t$=" "
130 PRINT AT 0,20;("0" AND h<10);h;t$;("0" AND m<10);m;t$;("0" AND s1<10);s1
140 IF INKEY$="" THEN GO TO 100
```

Прокомментируем строки этой программы:

10 - ввод текущего времени; h - часы, m - минуты и s - секунды;

20 - перевод введенного времени в 50-е доли секунды;

30 - переменная tim раскладывается на три байта для занесения ее в системную переменную FRAMES в строке 40;

100 - из системной переменной FRAMES берется значение текущего времени и переводится в часы, минуты и секунды;

110 - проверяется, не прошло ли 24 часа. Если да, то системный таймер обнуляется;

120 - значение секунд разделяется на целую (s1) и дробную (s) части. Если дробная часть не больше 0.7 секунды, то между цифрами часов, минут и секунд будет ставиться двоеточие, а если больше - пробел. В результате разделяющие двоеточия будут мигать в такт секундам, что «оживит» вид часов;

130 - вывод значений часов, минут и секунд, причем если какое-то значение получилось меньше 10 (то есть состоит из одной цифры), то перед ним допечатывается ноль, в результате чего числа всегда получаются двузначными;

140 - переход для расчета следующего момента времени.

Используя в своих программах системный таймер, нужно помнить также о том, что на момент извлечения звука с помощью оператора ВЕЕР (включая и писк клавиатуры при вводе строк программы или данных в операторе INPUT), а также при загрузке и сохранении программ на магнитной ленте время для компьютера

останавливается, и переменная `FRAMES` не изменяется. То же самое происходит и в некоторых других случаях, связанных, в основном, с обращением к подпрограммам в машинных кодах, речь о которых пойдет в следующем разделе.

### ***Подпрограммы ПЗУ***

Операционная система компьютера представляет собой большую программу, написанную в машинных кодах. И как в любой программе, в ней есть масса полезных подпрограмм. К великому сожалению, далеко не все из них можно вызывать непосредственно из Бейсика, но некоторые вполне успешно могут быть использованы.

Обращение к кодовой подпрограмме осуществляет известная вам (правда, в другом качестве) функция `USR`. Как вы знаете, любое выражение Бейсика должно начинаться с оператора, поэтому для вызова кодовых подпрограмм чаще всего используют конструкции типа `RANDOMIZE USR addr`, где `addr` - адрес подпрограммы. Например, `RANDOMIZE USR 3435` выполнит те же действия, что и оператор `CLS`.

Часто в меню игровых программ можно встретить пункт **КОНЕЦ ИГРЫ**, при выборе которого память очищается и на экране появляется надпись © 1982 Sinclair Research Ltd. Обычно это достигается включением в программу оператора `NEW`, который удаляет ее и обнуляет память вплоть до адреса, указанного в операторе `CLEAR`. Более кардинально очистить память можно с помощью оператора `RANDOMIZE USR 0`. В этом случае произойдет то же, что и при нажатии кнопки `Reset`.

В игре **САМОЛЕТ** у нас возникла необходимость очистить служебное окно экрана. Там мы воспользовались для этого оператором `INPUT`. Однако программисты часто поступают по-другому, вставляя в программу оператор `RANDOMIZE USR 3438`.

В заключение этой главы рассмотрим еще одну возможность, не имеющую аналогов среди операторов Бейсика.

При переходе от заставки к игре мы всегда очищали экран командой `CLS`, но в некоторых играх удаление заставки происходит другими способами. Пожалуй, самый распространенный из них - скроллинг. Заставка не сразу исчезает с экрана, а постепенно «уползает» в сторону.

Вертикальное движение изображения достигается использованием оператора `RANDOMIZE USR 3582`, который перемещает весь экран на одну строку вверх. Попробуйте в любую из приведенных ранее заставок включить строку

```
FOR N=1 TO 24: RANDOMIZE USR 3582: BEEP .01,N: NEXT N
```

и вам наверняка захочется использовать этот фрагмент и в своих собственных программах.

## 8. LASER BASIC В ИГРОВЫХ ПРОГРАММАХ

После того, как вы научились делать простые и не очень простые заставки и спрайты, нельзя пройти мимо пакета программ, специально предназначенного для манипулирования графическими изображениями - Laser Basic. Этот пакет значительно расширяет возможности Spectrum-Бейсика, вводя в действие более ста новых операторов и функций. Среди них вывод на экран готовых спрайтов, инвертирование, повороты, зеркальное отображение и перемещение этих спрайтов по всем направлениям. Картинки предварительно готовятся с помощью генератора спрайтов, или делаются одним из способов, о которых мы говорили раньше, например, с помощью редактора Art Studio. После отладки программы в интерпретаторе Laser Basic ее можно скомпилировать, что почти в два раза увеличит скорость ее работы и к тому же защитит текст от несанкционированного доступа.

### ***Введение в Laser Basic***

Подробное описание Laser Basic с примерами и пояснениями дано в книге [2], а мы ограничимся лишь теми сведениями, без которых невозможно разобраться в предлагаемых ниже программах. Основными объектами, с которыми придется иметь дело, будут окна и спрайты. С окнами мы уже встречались, когда учились записывать спрайты на магнитную ленту редактором Art Studio. В Laser Basic окно также выделяет прямоугольную область экрана, на которую распространяется действие операторов, выполняющих графические преобразования. Окно задается координатами левого верхнего угла, а также шириной и высотой (все параметры задаются в знакоместах). Кроме экранного окна, в которое можно поместить целиком весь спрайт, в Laser Basic есть еще одно полезное понятие - окно спрайта. Оно означает, что и в самом спрайте можно выделить определенную часть и вывести ее на экран, а затем преобразовать, передвинуть и т. д. Эти операции могут оказаться очень полезными при создании всевозможных зрительных эффектов.

Операторы Laser Basic задаются ключевыми словами, состоящими из пяти символов, первый из которых - точка, а четыре последующих - прописные буквы латинского алфавита. Например: .PTBL, .SETV, .CLSM и т. д. Набираются ключевые слова посимвольно, буква за буквой, а их параметры передаются интерпретатору через специальные графические переменные, которым перед выполнением оператора необходимо присвоить те или иные значения.

Имена графических переменных состоят из четырех символов: первый - точка, остальные - прописные буквы латинского алфавита. Всего графических переменных 10:

.ROW, .COL - вертикальная и горизонтальная координаты в знакоместах, причем их значения могут выходить далеко за пределы экрана (допустимые пределы - от -128 до +127).

.HGT, .LEN - высота и ширина графического объекта (окна экрана, спрайта), выражаемые в знакоместах (0...255).

.SRW, .SCL - вертикальная и горизонтальная координаты внутри спрайта в знакоместах (пределы - в зависимости от размеров спрайта).

.NPX - величина и направление вертикального скроллинга в пикселях (-128...127).

.SPN - номер спрайта в спрайт-файле (1...255).

.SP1, .SP2 - номера первого и второго спрайтов (1...255) для операторов,

работающих с двумя спрайтами.

Для присвоения значений графических переменных обычным числовым переменным используются специальные графические функции. Выглядят они так же как и графические переменные, но вместо точки в них стоит знак вопроса. Например, ?ROW, ?SPN. Тогда оператор присваивания имеет вид: LET R = ?ROW - где R - обычная числовая переменная.

Наиболее распространенная версия пакета Laser Basic поддерживает работу как с магнитофоном, так и с дисководом в системе TR-DOS. Если вы пользуетесь последним, то перед операторами LOAD, SAVE, MERGE и VERIFY следует вставлять: RANDOMISE USR 15619: REM:

Для работы с пакетом вначале следует загрузить программу-диспетчер LASER, после чего на экране появится меню, изображенное на рис. 8.1. (для магнитофонной версии опции 5 и 6 будут отличаться от изображенных на рисунке).

Выберите в меню опцию LOAD "SPRITE2B" SPRITE, нажав клавишу 4. Начнется загрузка фирменного спрайт-файла SPRITE2B. Этот файл используется в демонстрационной программе DEMO. Для разработки своих программ мы возьмем оттуда несколько готовых спрайтов, чтобы эти программы смогли потом посмотреть читатели. После окончания загрузки вновь появится главное меню Laser Basic.



Рис. 8.1. Главное меню программы Laser Basic.

Загрузим теперь интерпретатор Laser Basic, выбрав пункт 1 EXECUTE LASER BASIC. Бейсик-программа выгрузится, и в ОЗУ останется только интерпретатор. После этого можно выполнять операторы Laser Basic.

Однако прежде, чем писать программу заставки или какой-то игры, сделаем несколько простых опытов с новыми операторами и графическими переменными.

Введите программу:

```
10 BORDER 0: PAPER 0: INK 5
20 CLS: LET S=-2
30 FOR N=7 TO 10
40 .ROW=5: .COL=S: .SPN=N
50 .PTBL
60 PAUSE 5
70 NEXT N
80 LET S=S+2
90 IF S<32 THEN GO TO 30
```

После запуска программы оператором RUN вы увидите на экране изображение прыгающего слева направо забавного зверька, который, как в настоящем мультфильме, шевелит ушами и носом и двигает лапками во время прыжков.

Попробуем разобраться, каким образом достигается этот эффект.

10 - установка атрибутов экрана;

20 - очистка экрана и присваивание переменной S начального значения;

30 - начало цикла по переменной N, которая задает номер выводимого на экран спрайта. Сходство получающейся картинке с мультфильмом достигается благодаря тому, что на экран последовательно выводятся четыре спрайта с номерами 7, 8, 9 и 10, которые соответствуют разным фазам движения зверька;

40 - задание номера спрайта и его местоположения на экране. При этом координата расположения спрайта по вертикали остается постоянной, а горизонтальная координата все время увеличивается, в результате чего он перемещается слева направо;

50 - оператор вывода на экран спрайта с номером, заданным переменной .SPN в строке 30;

60 - оператор паузы;

70 - конец цикла;

80 - увеличение переменной S, определяющей положение спрайта по горизонтали;

90 - переход на начало цикла движения до тех пор пока спрайт не уйдет за пределы экрана.

Одним из достоинств Laser Basic является наличие в нем оператора перенумерации строк. То есть теперь можно не думать о том, какие номера ставить в начале каждой строки, а, написав программу, выполнить оператор .RNUM 10, 10, 10 - и строки вашей программы получат новые номера, кратные 10; при этом автоматически поменяются адреса в операторах GO SUB и GO TO. В операторе .RNUM первое число - это номер строки программы, с которой начнется перенумерация, второе число - новый номер этой строки и, наконец, третье - шаг нумерации строк программы.

Теперь полезно набрать и запустить маленькие программки, описанные в книге [2], иллюстрирующие простейшие действия с окнами и спрайтами. Они помогут познакомиться с новыми операторами и понятиями Laser Basic, которые пригодятся вам при написании серьезных программ.

### ***Заставка на Laser Basic***

Вы уже познакомились с большим количеством заставок и наверняка заметили, что подавляющее большинство из них представляют собой просто картинки - красивые, но статичные изображения. С помощью Laser Basic можно очень простыми средствами оживить не только саму игру, но и заставку. Имея желание и достаточно терпения вы вполне сможете создать даже заставку-мультфильм. Ниже приведен текст программы, на примере которой показано, как совместить уже известные вам операторы Spectrum-Бейсика с новыми операторами и графическими переменными Laser Basic.

Сначала несколько слов о том, что происходит при работе программы.

После ее ввода и запуска вы увидите в верхней части экрана движущийся автомобиль, к которому прицеплено название игры. Как только он скроется за пределами экрана, название появится вновь в средней части экрана. Затем вступит в действие другая часть программы и выведет на экран медленно перемещающееся поле из цветов, а вдалеке от него понесется локомотив с двумя пассажирскими вагонами. На то, что он не стоит на месте, укажут перемещающийся железнодорожный путь и бешено вращающиеся колеса (рис. 8.2). А теперь рассмотрим саму программу.



Рис. 8.2. Мультипликационная заставка «Паровоз».

Программа 30. ПАРОВОЗ.

```

10 LET K=1: LET P=1: LET C=1
20 RANDOMIZE USR 58841: GO TO 30
30 RANDOMIZE :.SET=0:.ROW=0:.COL=0:.HGT=24:.LEN=32
40 BORDER 0: PAPER 0: INK 4: BRIGHT 1
50.CLSV: .SETV
60 GO SUB 300
70.SET=0:.SPN=4
80 FOR N=1 TO 40:.ROW=INT (RND*8+14):.COL=INT (RND*31):.PTBL: NEXT N
90.ROW=13:.SPN=51: FOR N=0 TO 24 STEP 8:.COL=N:.PTBL:NEXT N
100.ROW=10:.COL=20:.SPN=34:.PTBL
110.ROW=10:.COL=10:.SPN=48:.PTBL
120.ROW=10:.COL=0:.SPN=48:.PTBL
130.SET=2:.ROW=12:.COL=0:.SPN=49:.PTBL
140.SET=3:.ROW=12:.COL=10:.SPN=49:.PTBL
150.SET=4:.ROW=13:.COL=0:.LEN=32:.HGT=12
160.SET=5:.ROW=12:.COL=20
170 FOR N=1 TO 4: IF N=4 THEN BEEP .005,0
180.SET=5:.SPN=N+34:.PTBL
190.SET=2:.SPN=K+48:.PTBL
200 LET K=K+1: If K=3 THEN LET K=1
210.SET=3:.SPN=K+48:.PTBL
220.SET=4: .WL1V:.WL1V:.WL1V
230 IF INKEY$<>" " THEN GO TO 500
240.SET=1: INK N:.SETV
250 NEXT N
260 BEEP .005,0: GO TO 170
300 REM -----
310 LET A$="SOUTHERN BELLE"
320.SET=1:.SPN=5
330.ROW=3:.COL=0:.HGT=2:.LEN=32
340 INK 4:.SETV
350.ATOF:.PTBL
360 FOR B=0 TO 13
380 FOR N=1 TO 8
390.SR1V
400 NEXT N
410 PRINT AT 4,0;A$(14-B)
420 NEXT B
430.ATON
440 FOR N=1 TO 260
450.SR1V
460 NEXT N
470 PRINT AT 4,10; INK 3;A$
480 PRINT AT 6,6; INK 5;"Sankt-Petersburg 1993"
490 RETURN
500 REM -----
510.SET=0:.ROW=0:.HGT=24:.LEN=16

```

```
520 FOR N=1 TO 16
530 .COL=0: .SL8V: .ATLV
540 .COL=16: .SR8V: .ATRV
550 BEEP .02, N
560 NEXT N
```

10 - присваивание переменным начальных значений;

20 - обязательный в Laser Basic оператор запуска интерпретатора. Следом за RANDOMIZE USR 58841 должен стоять оператор GO TO или RUN с номером строки. В противном случае возможны сбои в работе программы;

30 - восстановление счетчика случайных чисел, а также установка графических переменных набора 0. Обычно в Laser Basic различные объекты описываются разными наборами графических переменных, которые задаются специальной переменной .SET. Это позволяет изменять параметры одного объекта, не затрагивая остальных. Всего в программе может быть определено 16 наборов (от 0 до 15);

40 - установка атрибутов экрана;

50 - оператор ,CLSV очищает окно экрана (в данном случае в наборе .SET=0 задано окно во весь экран), а .SETV устанавливает в нем постоянные текущие атрибуты;

60 - обращение к подпрограмме, которая выводит на экран спрайт-автомобиль, перемещает его слева направо, постепенно добавляя к нему буквы названия игры. Подпрограмма заканчивается двумя строками текста;

70 - из спрайт-файла SPRITE2B программа выбирает 4-й спрайт («цветок»), после чего в цикле (строка 80) печатается 40 таких спрайтов. Координаты по вертикали задаются таким образом, чтобы все «цветочное поле» располагалось в нижней части экрана;

90 - после выполнения этой строки на экране появляются рельсы (спрайт 51);

100 - из спрайт-файла выбирается спрайт 34 - «локомотив» и выводится на экран на место с координатами (10,20);

110, 120 - вывод на экран двух вагонов (спрайт 48);

130, 140 - вывод на экран изображения колес вагонов (спрайт 49). Поскольку в дальнейшем координаты колес на экране будут использоваться для создания впечатления их вращения, графические переменные запоминаются в наборах 2 и 3;

150 - в 4-м наборе переменных задаются координаты, ширина и высота окна экрана, охватывающего рельсы и все цветочное поле. В дальнейшем горизонтальный скроллинг этого окна создаст иллюзию движения локомотива относительно пейзажа;

160 - в 5-м наборе переменных определяются координаты вывода на экран колес локомотива;

170 - здесь начинается цикл вращения колес, перемещения рельсов и цветов, причем N равное 35, 36, 37 и 38 соответствует спрайтам четырех положений колес локомотива. Если эти спрайты выводить на экран поочередно, то создается впечатление, что колеса вращаются. На последнем «витке» цикла (N = 4) раздается короткий звук, имитирующий стук колес;

180 - устанавливается набор номер 5 - колеса локомотива - и на экран выводится спрайт, соответствующий очередной фазе их вращения;

190...210 - вывод на экран спрайтов колес двух вагонов. В отличие от спрайтов колес локомотива, этих спрайтов только два, поэтому переменная K может принимать значения 1 или 2 (строка 200);

220 - циклическое смещение окна, содержащего изображения рельсов и цветочного поля (набор 4), на три пикселя влево (трижды выполняется оператор

.WL1W);

230 - нажатие любой клавиши вызовет переход к завершающей части программы;

240 - в наборе переменных номер 1 заданы параметры окна экрана, охватывающего название игры. Цвет тона изменяется с управляющей переменной N, и этим цветом окрашиваются буквы в окне. В результате надпись будто бы постоянно переливается;

250 - конец цикла;

260 - после завершения цикла еще раз повторяется звук, имитирующий стук колес, затем программа переходит на начало цикла и все повторяется;

300 - начало подпрограммы вывода на экран названия игры; 310 - присваивание строковой переменной названия игры; 320 - задание спрайта номер 5 - (автомобиля) в 1-м наборе; 330 - задание начальных координат автомобиля, а также высоты и ширины окна, в котором он будет перемещаться по экрану; 340 - установка зеленого цвета тона в заданном окне; 350 - запрет переноса атрибутов во всех операциях между экраном и спрайтами, затем вывод спрайта-автомобиля на экран;

360 - начало цикла по переменной B, в котором следом за автомобилем появляется название игры;

380...400 - заданное окно в цикле скроллируется на 8 пикселей (то есть на одно знакоместо) вправо (8 раз повторяется оператор .SR1W);

410 - в освободившееся после скроллинга окна знакоместо выводится очередная буква названия игры;

420 - конец цикла по переменной B;

430 - разрешение переноса атрибутов в операциях между экраном и спрайтами;

440...460 - скролинг окна с автомобилем и названием игры вправо на 1 пиксель. Этот скролинг повторяется 260 раз, в результате чего все изображение, пройдя по экрану слева направо, исчезает;

470, 480 - вывод неподвижных надписей в середину экрана; 490 - выход из подпрограммы вывода названия игры; 500 - завершающая часть программы;

510 - устанавливается набор 0, в котором задаются параметры окна экрана;

520...560 - в цикле две половинки экрана расходятся в разные стороны, как шторы;

530 - переменной .COL=0 задается окно, охватывающее всю левую половину экрана. Выполняется скролинг окна влево на одно знакоместо, и одновременно - скролинг атрибутов;

540 - заданное окно перемещается в правую половину экрана, и осуществляется скролинг на одно знакоместо вправо вместе с атрибутами;

550 - звуковое сопровождение цикла.

Увидев эту программу в действии, вы наверняка захотите в ней что-то изменить и даже улучшить. Отлично! Вместо изображения автомобиля можете использовать спрайт вертолета с вращающимся винтом, вместо поезда - бронетехнику и т. д. чтобы выбрать что-то подходящее, просмотрите содержание всего спрайт-файла SPRITES2B и запишите номера понравившихся спрайтов. Для этого придется ввести следующую небольшую программу:

```
10 .ROW=10:.COL=10
20 FOR N=1 TO 59
30 .SPN=N:.PTBL
40 PRINT AT 21,0;N
50 PAUSE 0
60 CLS
```

70 NEXT N

В цикле управляющая переменная N задает номер спрайта (строка 30), который выводится на экран оператором .PTVL в знакоместо с координатами (10,10), определенными в строке 10. Сам номер тоже печатается (строка 40). Внимательно рассмотрев спрайт и решив, подходит он вам или нет, переходите к следующему; для этого достаточно нажать любую клавишу.

### **Игровая программа**

Попробуем использовать операторы и графические переменные Laser Basic в простенькой игре, которую назовем БЕГА МЫШЕЙ. Как и ранее, спрайты возьмем из спрайт-файла SPRITES2B. Если же вам захочется создать собственный набор картинок для новой игры, следует сначала воспользоваться одним из генераторов спрайтов: SPTGEN или SPRITER. Обе программы предназначены для создания наборов собственных спрайтов которые можно будет использовать в Laser Basic. При этом, если SPTGEN ориентирован на создание изображений по точкам, то SPRITER позволяет использовать готовые картинки, нарисованные в любом графическом редакторе для ZX Spectrum (Art Studio, The Artist II и др.), или даже позаимствовать объекты из фирменных заставок. Последнее обстоятельство открывает богатейшие возможности для творчества. О том, как работать со спрайт-генераторами, подробно рассказано в книге [2].

А мы вернемся к нашей игре и для начала скажем несколько слов о ее сюжете.

Игровое пространство представляет собой горизонтальные цветные полосы, по которым слева направо двигаются 6 мышек (рис. 8.3). Скорость их движения задается генератором случайных чисел, поэтому предсказать заранее, какая из мышей первой придет к финишу, очень трудно. Но все же придется на какую-то из них сделать ставку, причем в полновесной валюте - таковы правила игры.



Рис. 8.3. Игровое пространство программы БЕГА МЫШЕЙ.

### Программа 31. БЕГА МЫШЕЙ.

```

10 RANDOMIZE USR 58841: GO TO 20
20 RANDOMIZE: LET D=300
30 REM -----
40 DIM X(6)
50 PAPER 0: INK 0: BORDER 0: CLS
60 PAPER 3: .HGT=3: .LEN=13
70 FOR N=0 TO 5
80 PAPER 7-N: .ROW=N: .COL=N*2: .CLSV: .SETV
90 PLOT N*16+1, 174-(N*8): DRAW 101, 0: DRAW 0, -21: DRAW -101, 0: DRAW 0, 21
100 PAUSE 5: NEXT N
110 PRINT AT 6, 11; "BEGA MYSHEJ"
120 PAPER 0: INK 3
130 PRINT AT 19, 0; INK 6; "U WAS "; D; "$"
140 PRINT AT 21, 0; INK 6; "SKOLKO $ POSTAVITE?"
150 INPUT "-->"; ST
160 IF ST<1 OR ST>D THEN GO TO 150

```

```

170 LET D=D-ST
180 BEEP .1,0
190 REM -----
200.ROW=15:.COL=0:.HGT=8:.LEN=32:.CLSV
210 PRINT AT 21,0; INK 6;"NA KAKUIU MYSH POSTAVITE?"
220 INPUT "No:";MY
230 IF MY<1 OR MY>6 THEN GO TO 220
240 BEEP .1,0
250 REM -----
260 CLS
270 PRINT AT 0,11;"BEGA MYSHEJ"
280 INK 0:.COL=0:.HGT=2:.LEN=32
290 FOR N=1 TO 6
300 PRINT AT 2+(N*2),31;N
310 PAPER N:.ROW=2+N*2:.SETV
320 NEXT N
330 PLOT 240,45: DRAW 0,100
340.SPN=2:.COL=0
350.ATOF
360 FOR N=1 TO 6
370.ROW=2+N*2:.PTBL
380 NEXT N
390.ATON
400 REM -----
410.ROW=0:.COL=10:.HGT=2:.LEN=13:.NPX=1:.WCRV
420.COL=0:.HGT=2:.LEN=30
430 LET R=INT (RND*6+1)
440 LET X(R)=X(R)+4
450.ROW=R*2+2:.SR4V
460 IF X(R)>205 THEN GO TO 500
470 GO TO 400
500 REM -----
510 INK 5: PAPER 0
520 PRINT AT 18,7;"POBEDILA MYSH No ";R
530 IF MY<>N THEN GO TO 590
540 FOR N=D TO D+ST*4
550 PRINT AT 20,0;"U WAS STALO: ";N;"$"
560 BEEP .002,50
570 NEXT N
580 LET D=D+ST*4
590 PAUSE 300
600 GO TO 40

```

10 - инициализация интерпретатора Laser Basic;

20 - восстановление генератора случайных чисел и задание начального капитала играющего;

40 - задание массива, определяющего горизонтальные координаты мышей;

50 - установка атрибутов экрана;

60 - задание размеров окна для вывода названия игры;

70...100 - на экране появляются 6 разноцветных окошек заданного размера, расположенные по диагонали как бы одно поверх другого;

110 - вывод в последнее появившееся окошко названия игры;

120...150 - информация о том, сколько у вас осталось денег и ввод суммы ставки;

160 - проверка размера ставки: она должна быть больше I, но не превышать размеров наличного капитала;

170 - после того, как ставка сделана, она вычитается из ваших денег;

180 - звуковой сигнал;

200...240 - ввод в программу новых данных;

- 200 - очистка нижней части экрана;
- 210 - вывод вопроса: «На какую мышшь поставите?»
- 220 - ход игрока: ввод номера дорожки, по которой побежит «ваша» мышшка;
- 230 - проверка условия, находится ли номер выбранной дорожки в пределах от 1 до 6;
- 240 - звуковой сигнал;
- 250 - начало формирования игрового пространства;
- 260 - очистка экрана;
- 270 - вывод названия игры;
- 280 - задание окна, охватывающего одну дорожку;
- 290...320 - вывод на экран шести дорожек разного цвета, в конце которых ставятся их номера;
- 330 - вычерчивание линии финиша;
- 340 - задается спрайт мыши (номер 2) и начальная позиция по горизонтали;
- 350 - запрет переноса атрибутов;
- 360...380 - вывод спрайтов мышек в начале каждой из шести дорожек;
- 390 - разрешение переноса атрибутов;
- 400 - начало «бегов»;
- 410 - задается окно, вмещающее название игры и осуществляется его вертикальный циклический скроллинг оператором `.WCRV`. Циклический - означает, что уходящее за пределы окна изображение появляется с другой его стороны. Величина скроллинга (в пикселях) задается графической переменной `.NPK`. При положительном значении этой переменной окно движется вверх, а при отрицательном - вниз;
- 420 - задание окна, охватывающего одну дорожку;
- 430 - генерация случайного числа в пределах от 1 до 6 и присваивание его значения переменной `R`. Предполагается, что мышшь с номером, выбранным в этой строке, будет в данный момент времени двигаться вперед;
- 440 - координата `X` выбранной случайным образом мыши увеличивается на 4, так как дальше ее изображение должно сдвигаться на 4 пикселя вправо;
- 450 - задание переменной `.ROW` в соответствии с положением на экране дорожки, по которой побежит выбранная мышшь, и сдвиг окна с мышью на 4 пикселя вправо;
- 460 - проверяется, не дошла ли мышшь до финишной линии. Если нет, то происходит переход на начало цикла «бегов». Если же мышшь достигла финиша, то программа переходит на строку 510, начиная с которой выводятся на экран результаты игры: «Победила мышшь N...», и в случае, если этот номер совпал с заданным вами, то ваш капитал увеличивается на 4 ставки (строки 540...580) и игра начинается заново.

То, о чем мы рассказали в этой главе, иллюстрирует лишь малую часть возможностей Laser Basic. Добавим, что перемещать спрайты можно пятью способами. Интересные результаты при разработке игр дает наложение спрайтов, различные виды их преобразований, а также операции с пейзажем. С помощью Laser Basic легко решается одна из непростых задач при создании динамических игр - определение факта «столкновения» спрайтов. Для этого достаточно вставить в вашу программу одну из функций `?SCV` или `?SCM`.

Будем надеяться, что мы сумели заинтересовать вас этим замечательным диалектом Бейсика, который может оказать неоценимую помощь в создании игровых программ.

## 9. ПЕРВЫЕ ШАГИ К КОДОВОЙ ПРОГРАММЕ

Тот, кто уже пробовал свои силы в написании программ на Бейсике, наверняка заметил, что спрайты передвигаются по экрану не слишком быстро. И если в логических играх типа ОХОТА НА ЛИС или МАКСИТ этот недостаток почти незаметен, то в более динамичных ситуациях он может довести игрока до белого каления. Что же делать?

В фирменных программах с медлительностью спрайтов борются довольно просто: на Бейсике пишется только загрузчик, а все остальное - в машинных кодах. Однако то, что хорошо для профессионалов, не всегда подходит любителям. Даже если вы разберетесь со всеми тонкостями работы и не растопчете ногами свой home computer в процессе отладки программы, то время, потраченное на написание программы в кодах, явно не будет соответствовать результату.

Есть и другие способы заставить спрайты двигаться побыстрее. Самый легкий из них - использование готовых кодовых подпрограмм из пакетов Supercode и New Supercode. С их помощью можно добиться различных видов перемещения (скроллинга) экрана, множества звуковых эффектов, быстрой смены экранов (причем самыми разными способами), изменения цветовых атрибутов экрана и многого, многого другого. Существует множество версий Supercode. Мы пользовались Supercode версии II (см. [5]).

### ***Программа Supercode***

После загрузки пакета на экране появится яркая заставка с названием, а также информация о том, что пакет разбит на две части, каждая из которых содержит 76 подпрограмм. Нажав клавишу 1 (или 2), загрузим

требуемую половину, после чего можно обратиться к ее меню с помощью клавиши I. Меню состоит из строк типа

42 LASER ZAP.....63950

состоящих из номера подпрограммы, ее названия и начального адреса, с которого подпрограмма должна запускаться.

В нижней части экрана выводится информация о действиях, которые вы можете выполнить.

D - DEMO - демонстрация действия подпрограмм;

P - PRINTER - вывод на принтер копии экрана;

I - INDEX - вызов на экран списка подпрограмм;

L - LOCATE - вывод на экран инструкции по перемещению подпрограмм;

J - JUMP PAGE # - переход к нужной странице списка;

C - CONTINUE - переход к следующей странице;

S - SAVE - вызов на экран инструкции по записи подпрограмм;

Q - QUIT - выход из программы Supercode, причем в ОЗУ компьютера она будет сохранена;

N - NUMBER - выбор номера подпрограммы.

В режиме меню нажатие любой цифровой клавиши устанавливает режим NUMBER, позволяющий набрать номер одной из имеющихся подпрограмм и познакомиться с подробной инструкцией о ее работе и размещении в памяти. Если вы захотите посмотреть подпрограмму в действии, нажмите клавишу E (EXAMPLE), а для записи ее на ленту или дискету - T (TAPE). Имя, под которым будет записана кодовая подпрограмма, можно установить самому или же использовать фирменное. В последнем случае достаточно нажать Enter.

Перейдем к описанию программы игры ИГРАЛЬНЫЙ АВТОМАТ, для работы которой следует переписать из пакета Supercode на ленту или дискету следующие кодовые подпрограммы:

```
1 - PIXEL UP-SCROLL 22 - PIXEL BOX LEFT- SCROLL
42 - LASER ZAP
43 - UNI-NOTE SOUND-GEN
44 - DIAL-NOTE
69 - BORDER EFFECTS
```

Заметим, что на ленте они должны располагаться после программы игры. Несколько слов о самой игре. После загрузки программы и всех подпрограмм, на экране появится игровое поле, включающее в себя следующие элементы (рис. 9.1):

- три окна, в которых будут появляться различные цифры, причем выигрышными считаются только те их комбинации, где совпадают две. или три цифры (всего используются семь цифр - 1, 2, ..., 7);

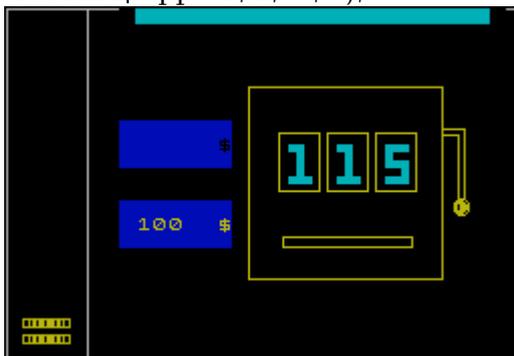


Рис. 9.1. Игра ИГРАЛЬНЫЙ АВТОМАТ.

продолжайте вводить цифры, пока не заполнится весь прямоугольник. Тогда число «сбросится», и можно будет вновь делать ставку;

- левый нижний прямоугольник синего цвета, где указывается размер вашего капитала в банке. Начальная сумма равна 100\$;

- слева все ваши доходы символически представлены в виде столбика золотых монет достоинством в 50\$ каждая. Если случится так, что вы все проиграете, программа издаст протяжный звук, и игра начнется вновь. Если же вам удастся набрать 20 монет (1000\$), то об этом радостно просигнализирует вначале изменение цвета бордюра, а затем звуковой сигнал, после чего игра закончится, и можно будет попытаться счастья еще раз;

- под тремя цифровыми окнами находится прорезь, откуда посыплются выигранные монеты по 50\$ в том случае, если ставка была удачной.

А теперь скажем о том, как устанавливается выигрыш. Если числовая комбинация содержит 2 единицы, 2 двойки, 2 тройки и т. д., то ваша ставка умножается соответственно на 1, 2, 3 и т. д. Если же эта комбинация состоит только из единиц, двоек, троек, то ставка умножается соответственно на 1, 2, 3, а затем еще на 2. В случае удачной игры, сумма выигрыша появляется в левом верхнем углу игрового поля одновременно с монетой и сопровождается звуковым сигналом.

Программа 32. ИГРАЛЬНЫЙ АВТОМАТ.

```
5 CLEAR 58570
10 BORDER 0: PAPER 0: INK 7
11 CLS
22 PRINT AT 9,10;"Please wait"
23 PRINT AT 11,8;"Loading program"
25 REM - Load code program --
30 FOR n=1 TO 6
40 INK 7: PRINT AT 13,12;"Code ";n
45 PRINT AT 0,0;
```

```
50 INK 0: LOAD ""CODE
60 NEXT n
61 CLS : PRINT AT 9,10;"Please wait!"
62 GO SUB 9000
63 GO SUB 1400
64 CLS
65 PRINT AT 13,12;"?????": REM 6 space
70 REM -----
90 DIM q(3): DIM c(3): DIM a$(3,16)
91 CLS
92 LET st1=2: LET st=2: LET no=0: LET rn=0
100 LET a$(1)="-GAME AUTOMAT-"
105 LET a$(2)="Sankt-Petersburg"
106 FOR n=0 TO 20
107 PRINT AT n,0; INK INT (RND*7+1);TAB 30
108 NEXT n
110 LET a$(3)="?????1993"
150 PRINT AT 8,8; INK 3;"??"; INK 5;"????????????";INK 3;""
155 PRINT AT 10,8; INK 6;"?????"; INK 2;" "; INK 6;"?????"
160 PRINT AT 12,14; INK 4;"????"
200 FOR n=1 TO 3
210 PRINT AT 21,8; INK 0;a$(n)
220 FOR m=1 TO 16
230 RANDOMIZE USR 64001
240 NEXT m
250 NEXT n
260 FOR n=1 TO 56
265 RANDOMIZE USR 64001
270 NEXT n
320 REM -----
330 POKE 63951,10
340 RANDOMIZE USR 63950
342 POKE 64648,7
343 POKE 64649,5
344 POKE 64649,150
345 POKE 64670,28
346 RANDOMIZE USR 64647
350 FOR n=1 TO 105
360 RANDOMIZE USR 64001
370 NEXT n
380 REM -----
400 CLS
402 FOR n=0 TO 20 STEP 3
403 FOR m=0 TO 30 STEP 3
404 INK 6: PRINT AT n,m;"DE"
405 PRINT AT n+1,m;"FG"
406 NEXT m
407 NEXT n
408 PAUSE 80: INK 7: CLS
410 PLOT 0,0: DRAW 0,175: DRAW 255,0: DRAW 0,-175:DRAW -255,0
425 PRINT AT 0,7; INK 0;" "; PAPER 5; INK 0;TAB 30;PAPER 0;" "
430 PLOT 40,0: DRAW 0,175
435 FOR n=19 TO 20
440 PRINT AT n,1; INK 6;"ABC"
445 NEXT n
450 PAPER 1: INK 6
455 FOR n=1 TO 3
460 PRINT AT 6+n,7;"????????"
465 PRINT AT 11+n,7;"????????"
470 NEXT n
```

```

475 PRINT AT 8,13;"$"
480 PRINT AT 13,8;"100nn$"
482 PAPER 0
485 PLOT 120,40: DRAW 0,96: DRAW 96,0: DRAW 0,-96: DRAW -96,0
490 PLOT 216,112: DRAW 8,0: DRAW 0,-32: DRAW 3,0: DRAW 0,35: DRAW -11,0
495 FOR n=4 TO 1 STEP -1
500 CIRCLE 225,76,n
505 NEXT n
515 PLOT 135,84: DRAW 0,29: DRAW 20,0: DRAW 0,-29: DRAW -20,0
520 PLOT 159,84: DRAW 0,29: DRAW 20,0: DRAW 0,-29: DRAW -20,0
525 PLOT 183,84: DRAW 0,29: DRAW 20,0: DRAW 0,-29: DRAW -20,0
530 PLOT 137,56: DRAW 64,0: DRAW 0,5: DRAW -64,0: DRAW 0,-5
532 INK 5
535 FOR n=17 TO 23 STEP 3
540 LET x=n: LET y=8: GO SUB 1300
545 NEXT n
550 LET d$="Programnn~GAME AUTOMAT~.nnnnAuthor Kapultsevich Igor,nnSaint-
Petersburg,nn1993.nnnstart-ENTER,nnnnEnd of game-E.nnnnnABC- 50$,nnDE- 50$.nnnnnnnnnn"
552 LET te=0: LET te1=0
553 LET st=2: LET st1=2: LET bank=100
554 REM -----
555 LET f$=""
556 PRINT AT 2,10;"nn";AT 3,10;"nn";AT 5,9;"nnnn"
557 IF bank=0 THEN POKE 64648,150: POKE 64649,70: POKE 64651,100: POKE 64670,28:
RANDOMIZE USR 64647: GO TO 400
558 IF bank>=1000 THEN POKE 60029,18: POKE 60020,1: POKE 60006,70: RANDOMIZE USR
60000: POKE 63951,10: RANDOMIZE USR 63950: GO TO 400
559 PRINT AT 8,8: PAPER 1;"nnnnnn"
560 PRINT AT 7,7: FLASH 1: PAPER 1: INK 0: "nnnnnnn"; AT 8,7;" ";AT 8,13;"$";AT
9,7;"nnnnnnn"
565 LET e$=INKEY$
566 GO SUB 1500
567 IF e$="e" OR e$="E" THEN GO TO 400
570 IF e$="" THEN GO TO 565
575 IF CODE e$=13 THEN GO TO 600
580 IF CODE e$<48 OR CODE e$>57 THEN GO TO 565
585 LET f$=f$+e$: PRINT AT 8,8: PAPER 1: INK 6;f$
586 BEEP .1,15
587 IF LEN f$>4 THEN GO TO 600
590 GO TO 565
600 IF f$="" THEN GO TO 558
605 LET d=VAL f$
610 IF d<1 OR d>bank THEN BEEP .3,5: BEEP .3,0: LET f$="": GO TO 559
612 LET bank=bank-d
613 PRINT AT 13,8: PAPER 1: INK 6;"nnnnn$";AT 13,8;bank
615 FOR n=0 TO 20 STEP 2: BEEP .01,n: NEXT n
620 PRINT AT 7,7: PAPER 1;"nnnnnnnn"
625 PRINT AT 9,7: PAPER 1;"nnnnnnnn"
630 PRINT AT 8,7: PAPER 1;" ";AT 8,13: INK 6;"$"
632 REM -----
640 FOR n=1 TO 12
641 FOR m=1 TO 3
642 LET rn=rn+1: IF rn>150 THEN LET rn=1
643 LET a=VAL r$(rn)
646 LET x=14+m*3: INK 5: LET y=8: GO SUB 950+a*50
647 BEEP .0005,55: BEEP .0004,40
648 IF n>7 THEN PAUSE n-6
649 NEXT m
650 NEXT n
652 REM -----

```

```
653 FOR n=1 TO 3
655 GO SUB 9900
660 LET a1=INT (rnd*141)
665 IF a1<36 THEN LET a=1: GO TO 700
667 IF a1>35 AND a1<66 THEN LET a=2: GO TO 700
669 IF a1>65 AND a1<91 THEN LET a=3: GO TO 700
670 IF a1>90 AND a1<111 THEN LET a=4: GO TO 700
672 IF a1>110 AND a1<126 THEN LET a=5: GO TO 700
674 IF a1>125 AND a1<136 THEN LET a=6: GO TO 700
676 IF a1>135 THEN LET a=7: GO TO 700
680 REM -----
700 LET x=14+n*3: LET y=8: GO SUB 950+a*50
705 BEEP .0005,55: BEEP .0004,40
707 GO SUB 1500
710 LET q(n)=a
720 NEXT n
725 GO SUB 1600
999 STOP
1000 REM ----1----
1010 PRINT AT y,x;"4:CS/7"
1015 PRINT AT y+1,x;"CS/5"
1020 PRINT AT y+2,x;"4:CS/1"
1025 RETURN
1050 REM ----2----
1055 PRINT AT y,x;"4:CS/3"
1060 PRINT AT y+1,x;"4:CS/2"
1065 PRINT AT y+2,x;"5:CS/3"
1070 RETURN
1100 REM ----3----
1110 PRINT AT y,x;"4:CS/3"
1115 PRINT AT y+1,x;"CS/2"
1120 PRINT AT y+2,x;"4:CS/2"
1130 RETURN
1150 REM ----4----
1160 PRINT AT y,x;"4:4"
1170 PRINT AT y+1,x;"5:CS/2"
1180 PRINT AT y+2,x;"5"
1190 RETURN
1200 REM ----5----
1210 PRINT AT y,x;"4:CS/3"
1220 PRINT AT y+1,x;"5:CS/3"
1230 PRINT AT y+2,x;"4:CS/2"
1240 RETURN
1250 REM ----6----
1260 PRINT AT y,x;"4:CS/3"
1270 PRINT AT y+1,x;"5:CS/3"
1280 PRINT AT y+2,x;"5:CS/2"
1290 RETURN
1300 REM ----7----
1310 PRINT AT y,x;"4:CS/3"
1320 PRINT AT y+1,x;"5"
1330 PRINT AT y+2,x;"5"
1340 RETURN
1400 REM -- Graphics --
1410 FOR n=0 TO 55
1420 READ S
1430 POKE USR "A"+n,s
1440 NEXT n
1450 RETURN
1460 DATA 0,0,0,255,150,150,150,255
```

```
1465 DATA 0,0,0,255,239,239,239,255
1470 DATA 0,0,0,255,105,105,105,255
1475 DATA 3,12,59,59,122,125,218,215
1480 DATA 224,88,188,86,238,215,183,127
1485 DATA 238,213,123,116,63,63,12,3
1490 DATA 191,189,185,122,242,196,24,224
1495 REM -----
1500 REM
1505 IF te1=8 THEN LET te1=0: LET te=te+1: PRINT AT 0,30;PAPER 0; INK 0;d$(te): IF
te>(LEN d$)-1 THEN LET te=1
1510 LET te1=te1+1
1520 POKE 58679,30: POKE 58680,64: POKE 58682,24:POKE 58681,8: RANDOMIZE USR 58571
1530 RETURN
1600 REM ---- PROIGRAL? ---
1605 IF q(1)=q(2) AND q(2)=q(3) THEN LET wy=q(1)*d*2: GO TO 1640
1610 IF q(1)=q(2) OR q(2)=q(3) THEN LET wy=q(2)*d:GO TO 1640
1620 IF q(1)=q(3) THEN LET wy=q(1)*d: GO TO 1640
1635 GO TO 2030
1640 REM -----
1645 POKE 64702,200: POKE 64693,30: POKE 64682,200:RANDOMIZE USR 64675
1650 LET ko5=INT (wy/50)
1656 PRINT AT 2,10; INK 6;"DE";AT 3,10;"EG";AT 5,9; INK 7;wy: PAUSE 10
1657 INK 6: OVER 1
1660 FOR n=1 TO ko5
1670 FOR m=14 TO 19
1680 PRINT AT m,20;"DE"
1685 PRINT AT m+1,20;"EG"
1690 PAUSE 5
1995 PRINT AT m,20;"DE"
1996 PRINT AT m+1,20;"EG"
2000 NEXT m
2005 BEEP .005,50: BEEP .005,45: BEEP .005,40: PAUSE 5
2010 NEXT n
2015 OVER 0
2020 LET bank=bank+wy
2022 PRINT AT 13,8; PAPER 1; INK 6; "nnnnn$"
2025 PRINT AT 13,8; PAPER 1; INK 6;bank
2030 REM -----
2035 LET st1=INT (bank/50)
2040 IF st1>st THEN LET s=1: GO TO 2047
2045 IF st1<st THEN LET s=-1: GO TO 2047
2046 GO TO 555
2047 OVER 1: INK 6
2048 LET g=20
2050 LET st=st+s
2052 IF s=1 THEN LET g=21
2055 PRINT AT g-st,1;"ABC": BEEP .1,20-(g-st): PAUSE 10
2060 IF st<>st1 THEN GO TO 2050
2070 OVER 0
2080 LET st=st1
2090 GO TO 555
8999 STOP
9000 REM -- RND ---
9005 LET t$="": LET r$=""
9010 FOR n=1 TO 200
9020 LET a=INT (RND*7+1)
9030 LET t$=STR$ a
9040 LET r$=r$+t$
9050 NEXT n
9060 RETURN
```

```
9899 REM ----RND generator----
9900 LET no=no+1.3415926: IF no>198 THEN LET no=no-194.768627
9902 LET n1=INT no
9905 LET rnd1=VAL (r$(n1)+r$(n1+1)+r$(n1+2))
9910 LET rnd=rnd1/783
9920 RETURN
```

5 - установка системной переменной RAMTOP. Адрес 58570 определяется самой нижней границей кодовых подпрограмм (см. строку 1520);

10 - установка атрибутов экрана;

11 - очистка экрана;

22, 23 - вывод текста, предупреждающего о том, что сейчас с ленты магнитофона будут загружаться кодовые блоки;

30...60 - этот фрагмент программы обеспечивает последовательную загрузку с ленты шести кодовых подпрограмм. При этом строка 40 информирует вас о том, какой именно блок читается в данный момент, а строка 45 устанавливает координаты слов Bytes:, которые появляются в процессе загрузки. Эти слова пишутся черным цветом по черному фону;

61 - экран очищается, выводится сообщение «Пожалуйста, подождите!»;

62 - обращение к подпрограмме, которая генерирует случайную последовательность из 200 чисел, необходимую для выработки комбинаций в окнах. Непривычный вид подпрограммы объясняется тем, что при загрузке кодовых подпрограмм нарушается нормальная работа RND-генератора в компьютере, но в то же время без случайных чисел обойтись нельзя;

63 - обращение к подпрограмме, формирующей графические символы: для монеты, стоящей на ребре - 4, для лежащей монеты - 3;

90 - задание рабочих массивов;

91...380 - эта часть программы выполняет функции заставки, обеспечивая медленное перемещение текста снизу вверх с использованием звуковых эффектов. Здесь применяются три кодовые подпрограммы, о которых будет сказано чуть позже;

106...108 - устанавливаются номера INK, которые начнут действовать при движении текста через определенные знакоместа, окрашивая его в переливающиеся тона;

200...250 - в нижней части экрана черным цветом вводятся три надписи, затем во внутреннем цикле они подвергаются вертикальному перемещению (скроллингу) на 16 пикселей. Строка 230 включает подпрограмму вертикального скроллинга;

260...270 - три надписи медленно перемещаются вверх на 56 пикселей, последовательно окрашиваясь в цвета, установленные ранее в строках 106...108;

340 - включение кодовой подпрограммы «трель»;

342...346 - подготовка и включение кодовой подпрограммы, обеспечивающей звуковой эффект «глиссандо»;

350...370 - после короткой остановки в центре экрана три надписи продолжают медленное перемещение (скролинг) вверх и в конце концов уходят за пределы экрана. Движение сопровождается окрашиванием строк текста (как в 260...270) разными цветами;

402...407 - заполнение всего экрана монетами (второй кадр заставки);

410...545 - на экран выводится изображение игрового поля, то есть устанавливаются окна, рисуются синие прямоугольники, прорезь для выпуска монет, а также выводится изображение двух монет «начального капитала» игрока в левой части поля. Подпрограмма 1300...1340 рисует три семерки в окнах, где затем будут появляться цифровые комбинации;

550 - задание текста, который будет выводиться в верхней части экрана в виде бегущей строки;

553 - задание начальных значений некоторых переменных;

557 - если в результате игры банк окажется пустым ( $bank = 0$ ), то включается кодовая подпрограмма «глиссандо», и программа переходит на начало игры;

558 - если действия игрока были удачными и в банке оказалось более 1000\$, то последовательно включатся кодовые подпрограммы «бордюр» и «трель», а затем произойдет переход на начало игры;

559 - если оба предыдущих условия не выполняются, то стирается надпись со ставкой;

560 - изображение синего прямоугольника, куда заносятся ставки, начинает «мигать»;

565...630 - эти строки обеспечивают ввод ставки, а подпрограмма 1500...1530 перемещает окно в верхней части экрана (вместе с текстом);

632...650 - вывод случайных комбинаций цифр, создающих иллюзию вращения дисков игрового автомата;

652...720 - эта часть программы фиксирует в окнах автомата последнюю комбинацию цифр, которая и определяет, выиграли вы или проиграли. Последовательность условий определяет, насколько часто могут появляться единицы, двойки, тройки, и т. д. до семерки. Предусмотрено, что единицы и двойки выпадают чаще, а семерки - реже;

725 - безусловный переход на строку 1600, где начинается подпрограмма проверки комбинации цифр для определения выигрыша или проигрыша;

1000...1340 - подпрограмма печати цифр 1, 2, ...7;

1400... 1490 - подпрограмма ввода графических символов, формирующих монеты;

1500... 1530 - подпрограмма бегущей строки, использующая кодовый блок скроллинга окна влево;

1600...1620 - проверка выпавшей числовой комбинации - есть выигрыш или нет. Проверяются все четыре возможных варианта совпадений:

- первой и второй цифр;

- первой и третьей;

- второй и третьей;

- первой, второй и третьей.

Например, выигрыши принесут 116, 242, 433 или 222 доллара, а если все цифры окажутся разными, произойдет переход к строке 2030, откуда начинается подпрограмма, уменьшающая количество монет в левом столбике;

1640... 1645 - формирование звукового сигнала;

1650...2010 - определяется, сколько монет должно быть выброшено в виде выигрыша. В левом верхнем углу показывается сумма выигрыша и рисуется монета;

2020 - к имеющейся у вас сумме добавляется выигрыш;

2030...2090 - изменение количества монет в левом столбике;

9000 - подпрограмма, вырабатывающая случайные числа для формирования комбинаций в окнах игрового автомата.

### ***Применение New Supercode***

Пакет New Supercode создан по образу и подобию Supercode и содержит 50 подпрограмм в машинных кодах, большинство из которых не имеет аналогов в Supercode.

После загрузки вы увидите список процедур (меню), из 50 строк, каждая из которых вызывает определенную подпрограмму и имеет вид:

```
33 ATTR CODE 48976,29
```

где слева направо расположены: номер кодовой подпрограммы, ее название, слово CODE, означающее принадлежность к кодовому типу, начальный адрес и длина программы в байтах. Нижняя часть меню даст информацию об управлении строками.

Выбор нужной строки осуществляется с помощью стрелки курсора, перемещаемой клавишами 6 (вниз) и 7 (вверх), а для того, чтобы воспользоваться процедурой, следует нажать Enter. При этом на экран будет выведена краткая инструкция о выбранной подпрограмме. Нажатие клавиши D (DEMO) позволит посмотреть, как действует подпрограмма, а S (SAVE) - записать ее на ленту или дискету. Возврат в меню происходит при нажатии клавиши T.

### ***Игровая программа «Звездная война»***

Покажем теперь, как можно использовать кодовые подпрограммы на примере игры «Звездная война». Но прежде, чем мы начнем говорить об игре, перепишем из комплекта NEW Supercode на магнитную ленту или дискету четыре следующих подпрограммы: LITERY, DZWIEK2, DZWIEK3 и EKРАН 2. О том, что они из себя представляют и какие функции выполняют в игре, мы расскажем при описании строк программы, а сейчас несколько слов о самой программе и о том, что произойдет после ее запуска.

Загрузив программу и выполнив оператор RUN, вы сразу увидите действие одной из кодовых подпрограмм: на экране появятся буквы необычных размеров и форм. Налюбовавшись названием игры, нажмите любую клавишу - раздастся характерный звук летящего снаряда и затем взрыва. Очень эффектно! Далее следует традиционная информация об управляющих клавишах и, наконец, возникает звездное небо, на фоне которого передвигается космический корабль противника. Здесь же окажется перекрестье прицела вашего лазера (рис. 9.2). Чтобы убедиться в его исправности, нажмите M, появятся два грозных луча, и вы услышите звук выстрела, созданный третьей кодовой подпрограммой.

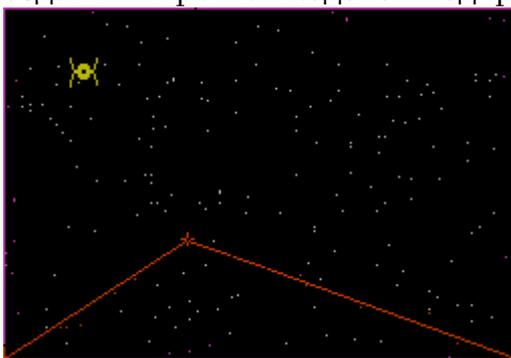


Рис. 9.2. Игра ЗВЕЗДНАЯ ВОЙНА.

Теперь возникает типичная игровая ситуация: вы должны, передвигая прицел, совместить его с вражеским кораблем и нажать M. При попадании раздастся грохот взрыва, а на экране возникнет хаотическая картинка, созданная кодовой подпрограммой EKРАН 2. После этого игра возвращается к своему началу.

Программа 33. ЗВЕЗДНАЯ ВОЙНА.

```
10 CLEAR 46945
12 POKE 23658,8
15 BORDER 0: PAPER 0: INK 7
20 CLS
30 GO SUB 7000
```

```

40 GO SUB 7100
50 CLS
60 LET y=0: LET xs=3: LET ys=20: LET d$="SPACE WAR":INK 2: BRIGHT 1: GO SUB 7200
70 LET y=160: LET xs=1: LET ys=2: LET d$="PRESS ANY KEY TO START": INK 6: GO SUB 7200
80 IF INKEY$="" THEN GO TO 80
90 BRIGHT 0
100 RANDOMIZE USR 59366
110 BORDER 0
120 REM -----
125 CLS
130 LET d$="O-LEFTПППР-RIGHT": LET y=10: LET xs=2:LET ys=5: GO SUB 7200
140 LET d$="Q-UPППППА-DOWN ": LET y=60: LET xs=2:LET ys=5: GO SUB 7200
150 LET d$="M-FIRE": INK 2: LET y=110: LET xs=4: LET ys=5:GO SUB 7200
160 IF INKEY$="" THEN GO TO 160
170 REM -----
172 LET s=0: LET sh=0
175 INK 7: CLS : OVER 1
176 FOR N=1 TO 200: PLOT RND*255,RND*175: NEXT N
177 PRINT AT 10,15;"E"
179 INK 3: PLOT 0,0: DRAW 0,175: DRAW 255,0: DRAW 0,-175: DRAW -255,0
180 LET zx=INT (RND*30): LET zy=INT (RND*20)
185 LET zx1=zx: LET zy1=zy
186 PRINT AT zy,zx;"AB": PRINT AT zy+1,zx;"CD"
190 LET px=15: LET py=10
195 LET px1=px: LET py1=py
197 LET f=0
200 IF INKEY$="P" THEN LET px=px+1: GO TO 300
210 IF INKEY$="O" THEN LET px=px-1: GO TO 300
220 IF INKEY$="Q" THEN LET py=py-1: GO TO 300
230 IF INKEY$="A" THEN LET py=py+1: GO TO 300
240 IF INKEY$="M" THEN LET f=1: GO TO 300
250 GO TO 400
300 REM -----
310 IF px<0 OR px>31 OR py<0 OR py>21 THEN LET px=px1:LET py=py1: GO TO 400
315 INK 7
320 PRINT AT py1,px1;"E"
330 INK 5: PRINT AT py,px;"E"
340 LET px1=px: LET py1=py
345 IF f=1 THEN INK 2: PLOT 0,0:DRAW px*8+4,172-(py*8): PLOT 255,0: DRAW -(253-
px*8),172-(py*8): RANDOMIZE USR 59342: INK 7: PLOT 0,0: DRAW px*8+4,172-(py*8): PLOT
255,0: DRAW -(253-px*8),172-(py*8)
350 IF f=1 AND ((zx=px AND zy=py) OR (px=zx+1 AND py=zy) OR (px=zx AND py=zy+1) OR
(px=zx+1 AND py=zy+1)) THEN RANDOMIZE USR 59366: GO TO 1000
360 IF f=1 THEN LET f=0
400 REM -----
410 IF s=sh THEN LET s=0: LET sh=INT (RND*5+4): LET prx=SGN (RND*2-1): LET pry=SGN
(RND*2-1)
420 LET s=s+1
422 LET zx=zx+prx: LET zy=zy+pry
425 IF zx<1 OR zx>29 OR zy<1 OR zy>19 THEN LET zx=zx1: LET zy=zy1: GO TO 200
440 INK 7
450 PRINT AT zy1,zx1;ПП"AB"
460 PRINT AT zy1+1,zx1;"CD"
470 INK 6: PRINT AT zy,zx; "AB"
480 PRINT AT zy+1,zx;"CD"
490 LET zx1=zx: LET zy1=zy
500 GO TO 200
1000 REM -----
1010 PAPER 0: BORDER 0: CLS
1015 FOR M=1 TO 10

```

```

1020 FOR N=1 TO 250 STEP 40: POKE 57399,N: RANDOMIZE USR 57392: NEXT N
1030 NEXT M
1040 GO TO 50
7000 REM -Graphics-
7010 FOR n=0 TO 39
7020 READ s
7030 POKE USR "A"+n,s
7040 NEXT n
7050 DATA 0,64,64,64,35,39,39,30
7060 DATA 0,2,2,2,196,228,228,120
7070 DATA 30,39,39,35,64,64,64,0
7080 DATA 120,228,228,196,2,2,2,0
7090 DATA 16,16,16,238,16,16,16,0
7095 RETURN
7100 REM -- LOAD code ---
7110 INK 0: PAPER 0: PRINT AT 0,0;
7120 LOAD ""CODE
7122 INK 7: LET d$="LOADING": LET y=50: LET xs=2: LET ys=10: GO SUB 7200
7123 LET d$="PLEASE WAIT": LET y=130: LET xs=2: LET ys=1:INK 3: GO SUB 7200
7130 INK 0: PRINT AT 0,0;: LOAD ""CODE
7140 PRINT AT 0,6;: LOAD ""CODE
7145 PRINT AT 0,0;: LOAD ""CODE
7150 RETURN
7200 REM -- Litery ---
7210 LET x=(256-xs*8*LEN d$)/2
7220 LET a=23306: POKE a,x
7230 POKE a+1,y: POKE a+2,xs
7240 POKE a+3,ys: POKE a+4,8
7250 LET a=a+4
7260 FOR i=1 TO LEN d$
7270 POKE a+i,CODE d$(i)
7280 NEXT i
7290 POKE a+i,255
7295 RANDOMIZE USR 59033
7296 RETURN
7300 REM ---

```

10. - установка RAMTOP;

12 - включение курсора больших букв;

30 - обращение к подпрограмме ввода графических символов: для корабля - 4, для прицела - 1;

40 - обращение к подпрограмме, загружающей с ленты четыре кодовые блока: LITERY, DZWIEK2, DZWIEK3, EKRAN 2;

60 - эта часть программы выводит на экран название игры, сформированное с помощью кодовой подпрограммы LITERY. Здесь y - верхняя граница выводимых букв, измеряемая в пикселях, начиная от верхнего края экрана; xs - их ширина в знаках; ys - высота букв в знаках. В символьной переменной d\$ задается текст, который требуется вывести;

70 - то же, что и в строке 60, но для текста

PRESS ANY KEY TO CONTINUE

80 - ожидание нажатия любой клавиши;

100 - с помощью кодовой подпрограммы DZWIEK3, вызываемой оператором  
RANDOMIZE USR 59366

создается звуковой эффект, имитирующий полет снаряда и его взрыв;

110 - восстановление цвета бордюра после действия предыдущей подпрограммы;

130...150 - вывод на экран текста, поясняющего функции клавиш;

160 - ожидание нажатия любой клавиши;

172 - установка начальных значений переменных: s - счетчик шагов перемещений корабля, sh - указатель количества этих шагов;

176 - создание на экране рисунка звездного неба. Оператор RANDOMIZE в начале строки восстанавливает счетчик случайных чисел после обращения к подпрограммам в машинных кодах;

177 - установка прицела в средней части экрана;

179 - вывод на экран рамки, ограничивающей игровое поле;

180 - вычисление случайным образом начальных координат корабля;

185 - ввод дополнительных координат корабля, в которых будут сохраняться предыдущие их значения;

186 - вывод на экран силуэта космического корабля;

190 - ввод начальных координат прицела, которые при его перемещении потребуются для операций стирания;

197 - вводится переменная f, соответствующая текущему положению кнопки Огонь, причем в данном случае кнопка считается не нажатой ( $f = 0$ );

200...240 - эта группа операторов обеспечивает управление прицелом;

310 - ограничения на перемещения прицела;

320 - закраска белым цветом «старого» прицела при его перемещении по экрану (напомним, что был установлен режим OVER 1);

330 - голубым цветом изображается «новый» прицел;

340 - запоминание координат, необходимое для того, чтобы их можно было восстановить, если они выйдут за пределы игрового поля;

345 - проверка условия, был ли сделан выстрел. Если да, то устанавливается красный цвет «чернил» и рисуются два лазерных луча, а с помощью кодовой подпрограммы DZWIEK2 воспроизводится звук выстрела. После выстрела устанавливаются белые «чернила» и на месте лазерных лучей рисуются две линии белого цвета;

350 - если была нажата клавиша Огонь, и есть попадание в одно из четырех знакомест, обозначающих корабль, то включается кодовая подпрограмма DZWIEK3, которая формирует взрыв, а затем осуществляется переход на подпрограмму 1000...1040, которая уничтожает все экранное изображение и создается процедурой EKРАН 2;

360 - если попадания нет, то переменной ОГОНЬ присваивается нулевое значение;

410 - если перемещение, которое сделал корабль, равно установленному ранее, то для следующего прохода S устанавливается равным 0, а переменной sh присваивается случайное значение в диапазоне от 4 до 8;

420 - увеличение счетчика на единицу;

422 - к текущим координатам корабля прибавляются приращения;

425 - проверка, не вышел ли корабль за границы игрового поля экрана;

440 - установка цвета вывода символов (белый);

450, 460 - вывод на экран изображения корабля, заданный старыми координатами;

470, 480 - установка цвета символов (желтый) и вывод на экран изображения корабля, заданного новыми координатами;

490 - изменение значений координат;

500 - безусловный переход к блоку управления прицелом;

1000... 1040 - подпрограмма, имитирующая уничтожение корабля противника. На экране мелькают хаотические пятна, созданные в двух циклах при участии

кодовой подпрограммы ЕКРАН 2, запускаемой оператором

RANDOMISE USR 57392

Перед ее использованием в ячейку 57399 заносится число в диапазоне от 0 до 255;

7000...7095 - подпрограмма определения графических символов, из которых четыре используются для изображения космического корабля, а один для создания перекрестья прицела;

7100...7150 - подпрограмма загрузки кодовых подпрограмм. Сначала грузится подпрограмма LITERY, затем на экран выводится фраза

LOADING, PLEASE WAIT

после чего загружаются DZWIEK3, DZWIEK2 и ЕКРАН 2;

7200...7296 - подпрограмма вывода на экран различных текстов, сформированных с помощью кодовой подпрограммы LITERY. При этом координату у необходимо задать заранее, а координату x определять заранее не нужно, так как подпрограмма сама выравнивает текст по центру экрана. Размеры букв по горизонтали - xs и вертикали ys задаются заранее. И, конечно же, необходимо ввести сам текст. Запуск программы осуществляется оператором

RANDOMISE USR 59033

### ***Компиляция игровых программ***

Несколько раз мы упоминали о существовании специальных программ-компиляторов, позволяющих значительно ускорить работу бейсик-программ. Теперь пришла пора выяснить, что же такое компиляторы и с чем их едят. Помимо увеличения быстродействия, с помощью компиляторов успешно решается ряд проблем, возникающих при создании «коммерческих» игрушек. Например, многие компиляторы исключают возможность остановки программы при нажатии клавиши Break. Кроме того, осложняется жизнь любителям ковыряться в чужих программах, что отчасти защищает такое достаточно эфемерное в нашей стране понятие, как «авторские права».

Большинство компиляторов переводит исходную программу на Бейсике в последовательность машинных кодов, понятных компьютеру. Тем самым заранее прodelывается работа интерпретатора, имеющая низкий КПД, за счет чего и достигается более высокое быстродействие. В некоторых случаях скомпилированная программа (или, как ее еще называют, объектный код) может работать почти в 100 раз быстрее, чем Бейсик. Однако не спешите радоваться. Как известно, у каждой палки есть два конца. Поэтому нужно сказать несколько слов и о недостатках применения компиляторов, чтобы вы могли реально оценить их возможности. Так, ни один из них не поддерживает полный набор инструкций интерпретатора, а откомпилированная программа, как правило, не может работать без присутствия в памяти самого компилятора или, по крайней мере, той его части, которая содержит рабочие процедуры. Чтобы не повторяться, рекомендуем обратиться к книге [3], где представлена полная информация о компиляторах, даются их сравнительные характеристики, а также приводится таблица совместимости каждого из них с бейсик-интерпретатором.

Итак, затратив немалое время, вы наконец завершили работу над игровой программой. При этом, еще на стадии ее отладки обнаружилось, что спрайты ползают по экрану со скоростью муравья, а ответной реакции на поворот ручки джойстика приходится дожидаться целую вечность. И вот в такой ситуации без услуг компиляторов просто невозможно обойтись.

Начинать знакомство с этим классом прикладных программ лучше всего с компилятора TOBOS FP. На сегодняшний день он, пожалуй, самый популярный,

хотя это и не означает, что он самый лучший, самый быстроедействующий, самый, самый... Все дело в том, что он самый простой в обращении и накладывает на исходную программу меньше всего ограничений. С его помощью можно обработать практически любую программу, приведенную в этой книге, правда, в них почти повсеместно придется вводить дополнительные задержки, иначе спрайты будут напоминать взбесившихся мустангов, и урезонить их окажется совсем непросто.

TOBOS состоит из двух файлов - бейсик-загрузчика и кодового блока длиной 12268 байт, загружаемого по адресу 53100. Вершина области бейсика (RAMTOP) устанавливается на адрес 39999, после чего выполняется оператор NEW. В результате может создаться впечатление, что компьютер «сбросился», но на самом деле все так и должно быть. Удаляется только уже ненужный бейсик-загрузчик, а сам компилятор остается в памяти. Теперь можете загрузить вашу собственную программу. После окончания загрузки для начала компиляции введите с клавиатуры команду

```
RANDOMIZE USR 53100
```

На экране появится фирменная заставка TOBOS, говорящая о том, что «процесс пошел». По окончании компиляции, если все в порядке и ошибок не обнаружено, на экране появятся еще две строчки вроде

```
RUN > RANDOMIZE USR 40000 SAVE > CODE 40000,7234
```

Числа, естественно, могут быть и другими. Важно то, что в первой строке компилятор сообщает команду для запуска объектного кода, а во второй - адрес и длину полученной программы для сохранения ее на внешнем носителе. То есть, для старта введите с клавиатуры оператор

```
RANDOMIZE USR 40000 а для записи на магнитофон -SAVE "name"CODE 40000,7234
```

Если потребуется, адрес размещения скомпилированной программы можно без труда изменить. Для этого достаточно ввести оператор

```
CLEAR (addr-1)
```

где addr - новый адрес.

Как уже говорилось, ни один компилятор не понимает весь набор операторов и функций Бейсика, и TOBOS в этом плане не является исключением. Если во время компиляции встретится недопустимая инструкция, появится сообщение

```
Nonsense in BASIC
```

с номером строки и оператора. Для TOBOS'а ограничения касаются, в основном, операций с внешней памятью, в остальном же он ведет себя вполне прилично.

Попробуйте обработать с помощью TOBOS, например, программу ЖИЗНЬ, и вы убедитесь в эффективности применения компиляторов. Разница окажется настолько очевидной, что вы, возможно, навсегда откажетесь от Бейсика «в чистом виде».

Теперь попробуйте оттранслировать какую-нибудь другую программу, например, СОКОБАН. После запуска объектного кода вы обнаружите, что, наряду с положительным эффектом (ускорение вывода лабиринта на экран), возникли определенные трудности: управлять грузчиком стало совершенно невозможно. Он так быстро стал бегать по складу, что даже массивные ящики оказались не способны его притормозить. А все оттого, что при написании программы изначально не было расчета на компиляцию. Поэтому прежде, чем приступить к работе над игрушкой, сразу стоит прикинуть, потребуется ли в дальнейшем компиляция, и если да, то это необходимо учесть еще на этапе разработки программы, чтобы в дальнейшем ничего не потребовалось переделывать. Но как же

поступить, если вы все-таки столкнулись с подобной ситуацией? В игре СОКОБАН исправить положение достаточно легко. Вставьте в строку 580 оператор PAUSE 0:

```
550 PAUSE 0: IF INKEY$="P" THEN LET X1=X+1: LET Y1=Y: GO TO 660
```

а в строку 660 для задержки включите «пустой» цикл:

```
660 FOR N=1 TO 100: NEXT N: IF U$(Y1,X1)<>" " AND U$(Y1,X1)<>"D" THEN GO TO 700
```

и все будет в полном порядке.

С другими программами, возможно, придется немного повозиться, но все же не настолько это сложно, чтобы вы не справились самостоятельно.

Для придания программе окончательного «коммерческого» вида необходимо собрать все ее части воедино. Как уже говорилось, скомпилированная программа не работает без присутствия компилятора, поэтому на ленте должен быть записан и кодовый блок TOBOS'a. Предположим, что программа оттранслирована с адреса 40000, тогда загрузчик может выглядеть примерно так:

```
10 INK 0: PAPER 0: BORDER 0: CLEAR 39999
20 LOAD ""SCREEN$: REM Картинка-заставка.
30 LOAD ""CODE 53100,12268: REM Кодовый блок TOBOS.
40 LOAD ""CODE 40000: REM Коды вашей скомпилированной программы.
50 RANDOMIZE USR 40000: REM Запуск программы.
```

Сохраняется программа-загрузчик уже известным вам способом: SAVE "name"  
LINE 10

который обеспечит последующий автостарт. Все блоки на ленте должны располагаться в порядке, указанном в загрузчике, то есть:

- сам загрузчик;
- экранная картинка (если она заранее заготовлена);
- кодовый блок компилятора;
- объектный код скомпилированной программы.

Может оказаться, что увеличения быстродействия, получаемого при использовании TOBOS'a недостаточно. Это может случиться при написании особо динамичных игр с большим количеством персонажей. Тогда на помощь приходят так называемые целочисленные компиляторы, то есть компиляторы, работающие только с целыми числами. К ним относятся программы ZX-Compiler, MCoder2, Softek IS. Однако задействовать все их возможности и получить блестящие результаты совсем непросто, так как они понимают лишь очень ограниченный набор операторов и функций Бейсика. Поэтому применять их следует только после досконального изучения их особенностей, а лучше всего писать программы специально ориентированные на эти компиляторы. В любом случае, советуем вам не оставлять их без внимания, тем более, что существует расширенная дисковая версия компилятора MCoder 2 - MC2b.v4 насколько нам известно, единственная, ибо дисковые версии прочих компиляторов по сути таковыми не являются, так как программа все равно может быть оттранслирована только из памяти. Кроме того, MC2b.v4, в отличие от прочих своих собратьев, создает совершенно независимый от компилятора объектный код, что несколько упрощает компоновку конечного продукта, а главное - освобождает дополнительную память для размещения фонтов, спрайтов, подпрограмм в машинных кодах и прочих нужд. Версия MC2b.v4 имеет еще ряд существенных улучшений и дополнений по сравнению со своим первоисточником, но описание всех этих подробностей выходит за рамки данной книги, поэтому ограничимся сказанным; отметим лишь, что программа эта написана одним из соавторов Александром Евдокимовым.

## 10. ГЕНЕРАТОР ИГР GAMES DESIGNER

Как вы уже, наверное, поняли, создание компьютерных игр - дело весьма долгое и непростое. Даже на Бейсике. К тому же, на Бейсике никак не удастся написать достаточно динамичную игру с большим количеством персонажей и предметов из-за невысокого быстродействия интерпретатора, встроенного в ZX Spectrum. Значит, нужно прибегать к помощи компиляторов или писать исключительно на ассемблере. Но это (особенно последнее) - еще более сложное дело. А ведь как хотелось бы сразу, не влезая глубоко в теорию программирования, произвести на свет собственное творение или, в крайнем случае, переделать чужую игрушку по собственному сценарию. Но ведь и это тоже не так-то просто. Глянешь в потроха какой-нибудь программы - сплошные нули, единицы, ничего не понять. Что поделаешь! Любишь кататься...

Впрочем, можно все-таки прокатиться за чужой счет, чтобы хоть узнать, насколько это интересное дело - создавать компьютерные миры. Ну, а узнав, уже решать, стоит ли тратить время на изучение программирования.

Существует целый ряд игровых программ, которые снабжены специальными редакторами, позволяющими изменять некоторые параметры игры (например, создавать свой вариант лабиринта). Но, как правило, этих изменяемых параметров очень мало - один-два.

Генератор игр Games Designer, созданный в 1983 году фирмой Software Studios, предоставляет гораздо большие возможности для творчества. Найдя правдами или неправдами эту программу, вы сможете создать игру, даже не имея ни малейшего представления о программировании.

Собственно говоря, Games Designer представляет собой обычную игру-«стрелялку», но в ней позволяется менять все (или почти все) - от характера звука и цвета экрана до сценария. При случае вы сможете потешить свое самолюбие и похвастаться собственным творением перед друзьями и знакомыми: все изменения, естественно, могут быть сохранены на ленте (или на диске, если вы располагаете версией, адаптированной к системе TR-DOS).

Однако хватит предисловий, перейдем к делу.

### **Главное меню**

После загрузки программы Games Designer на экране появляется главное меню (рис. 10.1). С помощью соответствующих цифровых клавиш выбирается один из восьми режимов работы. Вначале рассмотрим их все вкратце, а затем разберем некоторые из них более основательно.



Рис. 10.1. Главное меню программы Games Designer.

1. PLAY GAME (Запуск игры). Думается, что в особых комментариях этот режим

не нуждается..

2. SELECT NEW GAME (Выбор новой игры). Вы можете создать до восьми различных игр и затем быстро выбирать любую из них, не перезагружая Games Designer. Надоело играть в одну игрушку - тут же нажатием одной клавиши поменяли ее на другую! Быстро и удобно. После того, как вы нажмете клавишу 2 в главном меню, внизу экрана появится запрос

Select new game: [1-8] (Введите номер новой игры)

Нажмите цифровую клавишу, соответствующую номеру нужной игры, и можете начинать играть. Если нажать Enter, то вы возвратитесь в главное меню.

3. ALTER SPRITES (Изменение спрайтов). Напомним, что спрайтами называют движущиеся по экрану картинки. Таким образом, этот режим позволяет менять графику игры, вводить в нее новые персонажи. Более подробно режим ALTER SPRITES будет рассмотрен в разделе Изменение графики.

4. CONFIGURATION (Внешний вид игры). При выборе этого пункта на экране появляется дополнительное меню, позволяющее изменять раскраску экрана, звук, выбирать тип управления (клавиатура или джойстик) и сам сценарий игры. Но об этом несколько позже, в разделе Внешний вид игры.

5. MOVEMENT (Перемещения противника). Режим позволяет формировать траектории движения объектов, управляемых компьютером (в дальнейшем мы будем называть эти объекты «противником»). О том, как это делается, читайте в разделе Перемещения противника.

6. ATTACK WAVES (Уровни сложности). Games Designer предоставляет широкие возможности по формированию уровней сложности и «агрессивности» противника. Здесь вы можете задавать количество противников на каждом уровне и число очков, получаемое за попадание в цель, применять различные варианты движения и менять скорость нападающих объектов. Противники могут быть достаточно беззащитны, уничтожая вас на манер камикадзе, но могут, кроме этого, еще и подрывать вас торпедами. Конечно, если вы сами того пожелаете. Все подробности о работе с этой опцией будут описаны в разделе Уровни сложности.

7. LOAD FROM TAPE (Загрузить игру с ленты). Эта функция предназначена, как вы, очевидно, догадались, для загрузки вариантов игр, которые были созданы в Games Designer и записаны на ленту (см. режим SAVE TO TAPE). Отмотайте ленту на начало записанного ранее блока и включите магнитофон на воспроизведение. После окончания загрузки программа выйдет в главное меню. Если при загрузке возникнет ошибка, внизу экрана появится надпись: Tape loading error. Нажмите клавишу Enter для возврата в главное меню. Прервать операцию загрузки можно, как обычно, клавишей Space.

В дисковой версии эта функция отсутствует, а все изменения загружаются при старте Games Designer, если нажать клавишу Y на запрос Do you want load changes (Y/N). При нажатии любой другой клавиши программа запустится без загрузки изменений. Все восемь игр загружаются с диска одним махом, в ленточной же версии приходится загружать их по одной<sup>1</sup>.

8. SAVE TO TAPE или SAVE TO DISK (Записать игру на ленту или на диск)<sup>2</sup>. После выбора этого пункта главного меню внизу экрана появляется надпись Start tape then press Enter. Включите магнитофон на запись, затем нажмите клавишу Enter. По окончании записи компьютер попросит вас перемотать ленту на начало

---

<sup>1</sup> Не удивляйтесь, если в вашей дисковой версии дела обстоят несколько иначе. Здесь речь идет о версии, адаптированной автором этой главы.

<sup>2</sup> В дисковой версии это седьмой пункт меню.

для проверки, написав внизу экрана: Rewind tape for verification. Если вы вполне уверены в своем магнитофоне, можете нажать Space и продолжать работу, если нет - воспользуйтесь предложением: включите магнитофон на воспроизведение и проверьте, нет ли ошибок. При обнаружении ошибки компьютер известит вас надписью Tape verification has failed, при появлении которой нужно нажать клавишу Enter и повторить запись.

В дисковой версии запись производится аналогично, только отсутствует функция проверки записи. При работе с дисковой версией возможно появление сообщения File absent press Enter, говорящего о том, что вы вставили в дисковод не тот диск. Вообще диск может быть и другой, но на нем обязательно должен быть записан файл gd.chan. Если вы перепишите этот файл на другой диск, то сможете получить дополнительный набор игр.

### **Игра**

Вряд ли имеет смысл подробно описывать игру, которую можно изменять как угодно. Поэтому остановимся лишь на некоторых основополагающих моментах.

Игра, которую можно сконструировать в Games Designer, представляет собой, по сути, довольно примитивную «стрелялку». Однако это не делает ее менее увлекательной, особенно если учитывать возможность менять почти все ее параметры. На вас нападают неприятельские корабли в открытом космосе или на голову пикируют неизвестные науке чудовища, сбрасывающие бомбы на парашютах, а вам нужно отстреливаться от всей этой пакости, уворачиваться от лобовых столкновений... Ну, и так далее (рис. 10.2). Наверное, вы уже поняли основную идею, а воплощение этой идеи зависит от вас. Впрочем, если в данный момент у вас нет соответствующего творческого настроения, можете воспользоваться уже готовыми вариантами, любезно предоставленными авторами Games Designer.

Во время игры в верхней части экрана высвечивается счет (SCORE), количество дополнительных «жизней» (LIVES) и рекордное количество очков, полученное в предыдущих играх (HIGH). Вам дается три дополнительные «жизни», то есть всего у вас имеется четыре попытки. Увеличить их количество в программе нельзя, но, наверное, это и не столь важно, поскольку облегчить себе жизнь можно и другими способами.

Игра может состоять из нескольких уровней сложности. Переход на следующий уровень происходит после того, как вы уничтожили всех

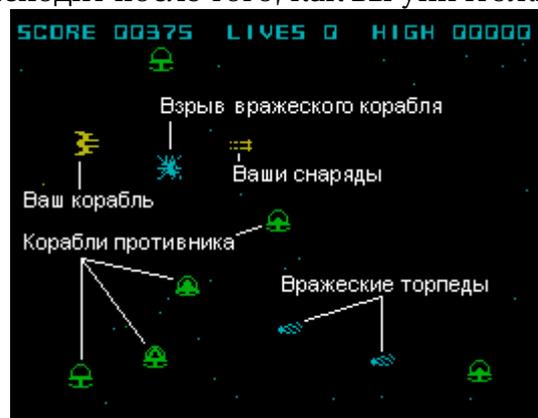


Рис. 10.2. Один из моментов игры.

противников на предыдущем. За каждое попадание вы получаете установленное количество очков.

Игра заканчивается, когда израсходована последняя «жизнь». Если у вас появится желание прервать игру досрочно, нажмите клавишу Enter. Если вы все же

доиграете до конца, компьютер попросит назвать ваше имя для внесения результата в таблицу рекордов - конечно, если количество набранных вами очков не слишком мало. Впрочем, на первых порах только нулевой результат не позволит вам стать рекордсменом. Используя клавиши **6** и **7**, подведите курсор к первой букве своего имени и нажмите клавишу **0**. Аналогично напечатайте и остальные буквы. Постарайтесь при этом не наделать ошибок: стереть неправильно введенную букву уже не удастся. Квадратик в конце алфавита означает пробел, а значок **#** используется для завершения ввода. Надо заметить, что зачислять себя в список рекордсменов вовсе не обязательно. Если вы не желаете этого делать, нажмите клавишу **S**, и игра начнется заново.

И в завершение этого раздела еще один важный момент, незаметный для «невооруженного глаза». Это возможность переопределения управляющих клавиш. Для этого игру нужно начинать, не просто нажав, находясь в главном меню, клавишу **1**, а удерживать при этом **Caps Shift**. Если у вашего компьютера расширенная клавиатура, то можете воспользоваться одной клавишей **Edit**. На экране появится надпись **Select new control keys** (выберите новые управляющие клавиши), вслед за которой необходимо последовательно нажать клавиши: Вверх (на запрос **Which key for up**), Вниз (**Which key for down**), Влево (**Which key for left**), Вправо (**Which key for right**) и, наконец, Огонь (**Which key for fire**).

### ***Порядок проектирования игры***

Если вы не любите, когда кто-либо навязывает вам свое мнение, можете смело пропустить это вступление без ущерба для дальнейшего восприятия. Всем же прочим осмелимся дать несколько практических советов по последовательности проектирования игр в **Games Designer**.

Начинать разработку лучше всего с выбора типа игры: определения места действия, направлений движения объектов, способа управления (клавиатура, джойстик) и т. д. Далее задаются цвета фона и программируется звуковое оформление. Все необходимое для этого вы найдете в режиме **CONFIGURATION** главного меню. Затем задайте траектории движения противника с помощью режима **MOVEMENT**, потом переходите к **ATTACK WAVES** для определения характеристик каждого из восьми возможных уровней сложности, и только в самом конце приступайте к рисованию графических объектов.

Такой порядок удобен тем, что до выбора типа игры и определения характеристик уровней трудно предсказать, сколько спрайтов вам понадобится и под какими номерами они будут фигурировать. В этом порядке мы и будем вести дальнейшее изложение.

### ***Внешний вид игры***

При выборе пункта **CONFIGURATION** на экране появляется дополнительное меню. В ответ на запрос **Select item: [1-8]** нужно нажать цифровую клавишу, соответствующую требуемому пункту. Параметры, введенные при выборе пунктов **1...4**, влияют на внешнее оформление игры. При этом внизу экрана появляется соответствующая подсказка, а номер пункта начинает мерцать. Если вы по ошибке выбрали не тот пункт, нажмите **Enter**, и все останется без изменений. Пункты **5...8** служат для редактирования звуковых эффектов: выстрелов, взрывов. Рассмотрим пункты меню **CONFIGURATION** по порядку.

1. **GAME FORMAT=X** (Тип игры). Здесь **X** - число от **0** до **7**. Именно этот параметр в основном и определяет сценарий игры. Вообще, принципиально различных типов игр в **Game Designer** не восемь, а четыре. Отличие одной четверки

от другой состоит лишь в способе управления. То есть если вы хотите использовать клавиатуру, вводите числа от 0 до 3, а если вы предпочитаете играть, орудуя джойстиком, - числа 4...7. Итак, что же означают эти числа (или иначе - коды)? Попробуем описать типы игр, получаемые при вводе каждого из них.

Тип А (коды 0 и 4). У вас имеются, как говорят в технике, две степени свободы. Можно двигаться по нижней кромке экрана только в двух направлениях - вправо и влево, а также стрелять вверх. Противники нападают сверху и при этом могут сбрасывать на вас бомбы.

Тип В (коды 1 и S). В этом случае вы занимаете круговую оборону и, поворачиваясь в разные стороны, обстреливаете противника. Дозволяется также двигаться по всему экрану, причем если вы уйдете за его край, то тут же появитесь с другой стороны. Движение возможно по вертикали, горизонтали и по диагоналям. Разворачивайтесь в нужную сторону, используя клавишу «вправо» для поворота по часовой стрелке или «влево» - в обратную сторону, а вместо акселератора жмите на клавишу «вверх», и при этом не забывайте время от времени стрелять.

Тип С (коды 2 и б). Пожалуй, этот тип лучше всего подходит для имитации космического боя. Ваш корабль летит на фоне звездного неба (о том, как его получить, будет сказано чуть позже). Впереди, то есть в правой части экрана, появляются вражеские корабли, стреляющие ракетами. Уничтожайте противника из своей пушки, лавируйте и не допускайте столкновений с врагом и попаданий его ракет. Вы можете перемещаться по левой половине экрана, двигаясь в любом направлении. При этом нажав клавишу «вправо», вы получаете ускорение и двигаетесь до середины экрана, отпустив - плавно откатываетесь назад. Клавиша «влево» не задействована.

Тип D (коды 3 и 7). Этот тип очень напоминает тип В. Отличие состоит, пожалуй, только в отсутствии диагонального движения, да управление несколько проще - какую клавишу нажмете, туда и поедете.

2. BACKGROUND = X (Задний фон). Этот пункт позволяет подобрать цвет фона всего экрана (аналогично операторам Бейсика PAPER X и BORDER X). Кодировка цвета (соответствие чисел и цветов, их обозначающих) стандартная, принятая в Бейсике:

- 0 - черный;
- 1 - синий;
- 2 - красный;
- 3 - фиолетовый;
- 4 - зеленый;
- 5 - голубой;
- 6 - желтый;
- 7 - белый.

3. FOREGROUND = X (Передний фон). Аналогично предыдущему пункту можно определить цвет звезд (если вы включили звездный фон, см. ниже), а также цвет всех надписей, в том числе и в таблице рекордов. Соответствует бейсик-оператору INK X.

4. SPECIAL FX = X (Специальная функция). Эта функция определяет порядок появления противника и способ его уничтожения, а также включает или выключает звездный фон. Противники в играх могут появляться либо поодиночке, либо сразу группой. Столкновение с любым из них ведет к гибели, поэтому их приходится уничтожать. Это

можно делать либо на расстоянии, стреляя из пушки, либо подойдя почти

вплотную, и ударив, например, дубинкой. Интересующее вас число можете выбрать из приведенной таблицы:

Противники появляются	Поодиночке		Группой	
	Нет	Есть	Нет	Есть
Звездный фон	Нет	Есть	Нет	Есть
Уничтожение на расстоянии	0	1	2	3
Уничтожение при сближении	4	5	6	7

Остальные пункты этого меню, как уже говорилось, позволяют менять характер звуковых эффектов.

5, 6, 7, 8. MISSILE SOUND (Звук вашего выстрела), BOMB SOUND (Звук выстрела противника), SHIP EXPLODE (Взрыв вашего корабля), ALIEN EXPLODE (Попадание снаряда во вражеский корабль). Все звуки изменяются одинаковым способом, поэтому нет необходимости описывать эти пункты в отдельности. При выборе одного из них на экране появляются пять диаграмм, отражающих пять характеристик звука (рис. 10.3):

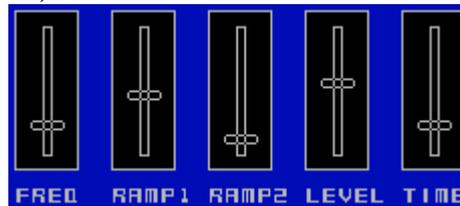


Рис. 10.3. Редактор звуков.

FREQ. - начальная частота. Чем выше «движок» на диаграмме, тем ниже получается тональность звука; RAMP1 и RAMP2 - составляющие огибающей звука (или проще - вибрации);

LEVEL - глубина вибрации; TIME - длительность звучания.

Изменяя эти параметры, можно получить звуки самого различного характера: от пулеметной очереди до жужжания бормашины.

На рис. 10.4 показана огибающая формируемого звукового сигнала: его частота периодически меняется. Рассмотрев этот рисунок, вы поймете, что означает тот или иной параметр звука.

Высота «движков» на диаграммах, а следовательно, и значения параметров, регулируется с помощью цифровых клавиш. Клавиши 1 и 2

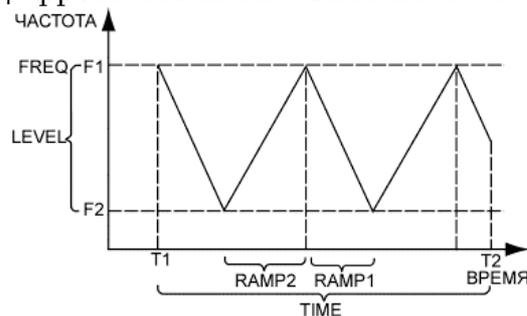


Рис. 10.4. Характеристики звука.

изменяют положение движка на диаграмме частоты звука (1 - поднять движок, 2 - опустить), 3 и 4 делают то же самое на второй диаграмме, следующую диаграмму изменяют клавиши 5 и 6, и так далее - то есть для регулировки каждого из параметров используется своя пара клавиш. Чтобы не действовать вслепую (точнее, вглухую), иногда нажимайте клавишу Symbol Shift, и вы услышите, какой звук получается в результате ваших манипуляций.

### **Перемещения противника**

После того, как вы задали тип игры, можно приступать к формированию траектории движения врага. Понятно, что если вы выбрали тип А, то есть можете передвигаться только по нижней кромке экрана, преимущественное направление движения противника должно быть сверху вниз, иначе не успеете вы сделать ни одного выстрела, как уже будете уничтожены. С типом С несколько проще, но все же неприятель должен нападать на вас спереди, а не сзади. Значит, здесь не стоит увлекаться перемещением слева направо. Для остальных двух типов выбор направлений в общем-то не имеет значения.

При выборе пункта MOVEMENT на экране появится картинка, показанная на рис. 10.5. Можно задать до восьми возможных траекторий движения противника, пронумерованных от 0 до 7 (N0 - номер траектории). Для каждой из них вводится:

PATTERN - ряд чисел, кодирующих направления (диаграмма в правом верхнем углу экрана подскажет вам, какому направлению какое число соответствует: 0 - вверх, 1 - вверх и вправо, 2 - вправо и т. д.)

NEXT - номер (N0) последовательности для продолжения движения. После прохождения по одной траектории, противники начинают



Рис. 10.5. Редактор траекторий.

двигаться по траектории, номер которой задается в этом пункте. Для продолжения движения можно указывать и ту траекторию, которую вы редактируете.

Для начала редактирования нажмите цифровую клавишу, соответствующую номеру изменяемой траектории (0..7). В правом нижнем углу экрана появится изображение траектории, напоминающее след мела на школьной доске, а в начале выбранной строки замигает курсор. Заменяя прежние числа новыми, вы обнаружите, что и траектория начнет меняться. Курсор перемещается по редактируемой строке клавишами 8 или 9. После того, как вы задали нужную траекторию, нажмите Enter. Теперь вы можете обработать аналогичным образом следующую последовательность, либо, еще раз нажав Enter, выйти в главное меню.

### **Уровни сложности**

Это, пожалуй, самый ответственный этап формирования игры, позволяющий вносить в нее наибольшее разнообразие.

Название режима ATTACK WAVES (волны атаки) говорит само за себя. Все, что вы в нем формируете, касается атакующей стороны, то есть вашего противника. Если вы любите риск и достаточно уверены в себе, можете дать сопернику хорошую фору. Но не переусердствуйте, чтобы не оказаться расстрелянным в первые же секунды боя.

После того как вы нажали клавишу 6 в главном меню, на экране появляется таблица, изображенная на рис. 10.6. Первая графа (N0 - номер уровня сложности)

не изменяется и служит лишь в качестве порядкового номера; числа в остальных шести графах могут быть изме-

NO.	ANIM	SCORE	PAT	MAX	SPD	NEXT
0	5	10	0	15	0	2
1	7	15	1	25	4	3
2	5	15	2	15	0	1
3	7	20	0	25	4	4
4	7	20	4	25	4	6
5	5	20	0	35	6	7
6	7	30	6	25	5	5
7	5	30	0	35	3	1

SHIFT FOR CURSOR CONTROL

Рис. 10.6. Режим ATTACK WAVES.

нены в предусмотренных пределах. Для коррекции того или иного параметра подведите мигающий курсор к нужной цифре с помощью стандартных курсорных клавиш:

Caps Shift и 5 - влево, Caps Shift и 7 - вверх, Caps Shift и б - вниз, Caps Shift и 8 - вправо, а затем введите новое число. Теперь мы попробуем разобраться с каждой графой таблицы в отдельности.

ANIM (Animation - оживление). Допустимый диапазон чисел - от 0 до 7. Первый шаг к «оживлению» мы уже сделали, задав траектории движения. Но движущийся объект может вращаться, внешне изменяться и т. п. В Games Designer для этого можно задействовать два или четыре спрайта (впрочем, ничто не мешает использовать и объекты, не меняющие своего внешнего вида). И здесь уже придется выбирать, что вам больше по душе: однотипные, но живые, шевелящие хвостами динозавры, либо разнообразные атакующие монстры, внешне мертвые, но не менее опасные. Кроме того, всю эту дичь можно заставить двигаться гуртом по одной траектории или пустить каждого по своей дорожке. Однако всего должно быть в меру, ибо увеличение сложности на одном уровне достигается за счет обеднения другого или сокращения общего количества уровней.

Объясняется это тем, что на все восемь уровней сложности даются 16 спрайтов, изображающих противника. Таким образом, для каждого уровня существует своя пара картинок, или для пары смежных уровней - четверка (см. раздел «Изменение графики»).

Так как же получить каждый из перечисленных вариантов? Чтобы не запутывать вас долгими рассуждениями, приводим таблицу, из которой без труда можно выбрать требуемое сочетание:

Код	Число спрайтов	Мультипликация	Число траекторий
0	2	нет	1
1	2	есть	1
2	2	нет	2
3	2	есть	2
4	4	нет	1
5	4	есть	1
6	4	нет	4
7	4	есть	4

Как видно из таблицы, коды 6 и 7 позволяют получить четыре группы врагов, каждая из которых движется по своей траектории. Но при этом можно использовать только траектории 0. .3. Таким образом, если использовать коды 6 и 7,

то в результате графа PAT (см. ниже), определяющая номер траектории, по которой начинают движение противники, оказывается совершенно бесполезной.

SCORE (Счет). В этой графе определяется количество очков, даваемое за попадание в цель на каждом из уровней. За удачный выстрел вы можете начислить себе от 0 до 99 очков. Логично предположить, что призовая сумма будет расти при переходе к более сложным уровням, хотя, конечно, вы вправе иметь на этот счет свое мнение.

PAT (Pattern, номер траектории). Эта графа определяет номер траектории, по которой начинают свое движение противники на каждом из уровней сложности. Естественно, что числа здесь могут быть от 0 до 7. Как уже говорилось, объекты могут перемещаться не только по одной траектории. При использовании четырех траекторий автоматически выбираются номера 0...3, а что касается двух, то в этом случае для определения второго пути берется следующий номер. Например, если в графе PAT задан номер 3, то одна группа противников будет двигаться по этой траектории, а вторая - по траектории 4 (или 0, если начальный номер 7).

MAX (Maximum). Здесь указывается общее количество врагов, которые будут бить вас на каждом уровне. Это число не может превышать 99, но столько, наверное, и не требуется, - можно ограничиться меньшим числом, иначе следующий уровень может оказаться недостижимым. Игра всегда начинается с уровня под номером 0 (причем ничто не мешает сделать его самым труднопроходимым). Впрочем, если вам захочется пропустить какой-либо уровень, задайте для него в графе MAX значение ноль. Это удобно делать также на этапе отладки: чтобы взглянуть, что творится на последних уровнях, не придется проходить всю игру от самого начала.

SPD (Speed - скорость). Помимо регулирования скорости перемещения врага, здесь вы решаете, давать ли ему оружие. Кроме того, в этой графе определяется общее количество одновременно атакующих вас противников. Ведь даже если в предыдущей графе задано максимально -возможное число противников, это вовсе не означает, что все они накинута на вас сразу. Они будут появляться постепенно: стоит уничтожить одного, как на подмогу товарищам спешит следующий. Одновременно же на экране может быть не больше четырех или не больше восьми вражеских объектов. Надо только помнить, что при меньшем их количестве атака будет более стремительной. К тому же стрельба по рассредоточенным мишеням потребует большей сноровки, так что это отнюдь не облегчит вашу участь. Из приведенной ниже таблицы выберите наиболее приемлемый для вас вариант и внесите соответствующий код в графу SPD.

Код	Число противников	Скорость	Стреляют ?
0	8	нормальная	нет
1	8	повышенная	нет
2	4	нормальная	нет
3	4	повышенная	нет
4	8	нормальная	да
5	8	повышенная	да
6	4	нормальная	да
7	4	повышенная	да

NEXT (Следующий). Как уже было сказано, игра всегда начинается с уровня под номером 0. А вот дальнейший порядок чередования уровней вам предлагается выбрать самостоятельно. И далеко не всегда наилучшим оказывается последовательный переход от уровня к уровню в порядке возрастания номеров.

Если в графе ANIM на четном уровне вы записали числа от 4 до 5, то лучше перескочить следующий уровень или вовсе исключить его из игры, так как графика на этом следующем уровне будет повторять предыдущую (вспомните: четверка спрайтов определяет графику двух соседних уровней).

### Изменение графики

Выбрав пункт 3 главного меню (ALTER SPRITES), вы увидите на экране изображения спрайтов (рис. 10.7), используемых в текущей игре (как вы помните, Games Designer «держит» в памяти одновременно до восьми игр). Для каждой игры имеется свой набор из 32 спрайтов, пронумерованных от 0 до 31 (спрайт-файл). Все спрайты имеют строго фиксированные размеры - 12 x 12 точек, и каждый «знает» свое место.

Первые 16 спрайтов (с номерами 00...15) - это ваши противники. Каждому уровню сложности, как вы помните, соответствует пара спрайтов, или двум соседним уровням - четверка (в зависимости от числа, заданного в графе ANIM режима ATTACK WAVES). То есть на уровне 0 могут использоваться спрайты 00 и 01 либо от 00 до 03; на уровне 1 - спрайты 02 и 03 либо та же четверка от 00 до 03; на следующем уровне - 04 и 05 или же 04...07 и т. д.

Следующие восемь спрайтов - это различные ракурсы того самого персонажа, которым управляете вы, нажимая на клавиши или поворачивая ручку джойстика. Правда, не во всякой игре понадобятся все восемь



Рис. 10.7. Таблица спрайтов.

картинок. Их количество зависит от типа игры, заданного в режиме CONFIGURATION, а точнее, от количества направлений стрельбы. Спрайт с номером 16 появляется при движении вверх, 17 - вверх и вправо, 18 - вправо, и так до спрайта с номером 23 (по часовой стрелке, с поворотом на 45 градусов). В результате получается, что в играх с типом А используется один только спрайт с номером 16, в типе В - все восемь спрайтов, в типе С - спрайт номер 18 и в типе D - спрайты с номерами 16, 18, 20 и 22.

Спрайты 24 и 25 изображают соответственно ваши и вражеские снаряды - по одному спрайту на снаряд.

Спрайты 28...31 соответствуют четырем фазам взрыва. Спрайт 27 используется только в том случае, если вы решили уничтожить противника при непосредственном контакте, то есть в опции CONFIGURATION задали значение специальной функции (SPECIAL FX) в диапазоне 4...7.

Что же касается спрайта 26, то автору этого описания не удалось выяснить его назначение: ни в одном варианте игры он не появляется. Не исключено, что это просто издержка производства или, что более вероятно, он предусмотрен создателями программы в качестве буфера. Во всяком случае, в этом качестве его очень удобно использовать при создании новой графики (см. ниже). А если

потребуется поменять местами два спрайта, то без такого буфера и в самом деле не обойтись.

Эти спрайты можно редактировать - изменять по своему вкусу. После того как вы вошли в режим ALTER SPRITES, на запрос Select sprite: [00-31] введите двузначное число, соответствующее номеру спрайта, который вы хотите изменить. Вы войдете в редактор спрайтов (рис. 10.8). На экране появится список управляющих клавиш редактора и два квадрата. Маленький квадрат внизу демонстрирует изменяемый



Рис. 10.8. Редактор спрайтов.

спрайт в натуральную величину, а большой квадрат справа - этот же спрайт в увеличенном масштабе.

Вверху большого квадрата помещены цифры от 1 до 6. Нажимая соответствующие цифровые клавиши, можно ставить или удалять точки в колонке под цифрой. Один раз нажали - поставили точку, еще раз нажали - убрали. Ряд, в котором происходят эти изменения, отмечается символом >, который можно передвигать вниз (клавиша 8) или вверх (клавиша 7). Но таким способом вам удастся нарисовать только левую половину спрайта. Чтобы перейти к правой стороне, нажмите клавишу 9, и цифры над большим квадратом зеркально отобразятся относительно его вертикальной оси. Теперь таким же образом можно ставить недостающие точки или убирать лишние и на правой стороне. Для возврата к левой половинке вновь нажмите клавишу 9.

Если вам нужно изменить цвет спрайта, нажмите клавишу Caps Shift и, не отпуская ее, цифровую клавишу, соответствующую коду цвета. Но это касается только цвета точек и не затрагивает фона (напомним, что фон устанавливается для всех спрайтов одинаковым с помощью функции BACKGROUND режима CONFIGURATION). Когда работа над спрайтом закончена, когда вы поставили последнюю точку и окрасили картинку в нужный цвет, - нажмите клавишу Enter для возврата к таблице спрайтов. Теперь можно точно так же изменить следующий спрайт или, еще раз нажав Enter, выйти в главное меню.

Прежде чем закончить этот раздел, упомянем еще об одной возможности. Для имитации движения, как вы понимаете, потребуется создать несколько почти одинаковых картинок, отличающихся одна от другой, быть может, совершенно незначительными деталями. Для этого не нужно перерисовывать весь спрайт от начала до конца, достаточно лишь слегка подправить уже нарисованный. Прежде всего понадобится получить копию спрайта в другом спрайте. В Games Designer есть такая возможность. Вводя номер спрайта, который вы хотите скопировать,

удерживайте клавишу Caps Shift, и вместо редактора спрайтов увидите внизу экрана подсказку

MOVE SPRITE XX TO

Теперь клавишу Caps Shift можно отпустить и ввести номер спрайта, в который

будет скопировано изображение. Если в последний момент вы передумали, то можете отменить операцию, нажав Enter. Когда же вы введете номер, то увидите, что получили в точности такой же спрайт на другом месте, а перемещаемый спрайт остался там же, где и был, без изменений.

В заключение расскажем о дополнительных возможностях, не предусмотренных в Games Designer, но которые, немного потрудившись, можно реализовать самостоятельно

### ***Дополнительные возможности***

Трудно сказать, в каком виде к вам попадет программа Games Designer. Но обычно она состоит из двух файлов:

- 1) gd - загрузчик на Бейсике, длина 270 байт, стартовая строка 1;
- 2) GD - программа в кодах, адрес загрузки 23600, длина 36400 байт.

Для внесения изменений и реализации дополнительных возможностей, о которых пойдет речь ниже, нужно слегка видоизменить кодовый блок. Не пугайтесь, это очень просто. Загрузите бейсик-блок gd оператором MERGE и приведите его к следующему виду:

```
1 BORDER 1: PAPER 1: INK 7: CLS : PLOT 58,148: DRAW 138,0: DRAW 0,-35: DRAW -138,0:
DRAW 0,35: PRINT AT 4,8;"Software Studios"; AT 6,9;"Games Designer"
2 PRINT AT 12,12; FLASH 1;"LOADING": INK 1:LOAD ""CODE 26000
5 RANDOMIZE USR 26000
10 LOAD "GD"CODE 24600: SAVE "gd" LINE 1: PAUSE 30:POKE PEEK 23631+256*PEEK
23632+2,181:SAVE "GD"CODE 27000,34000
```

Затем установите ленту на начало второго блока и введите с клавиатуры оператор GO TO 10. Включите магнитофон. Когда коды загрузятся, поменяйте кассету и сохраните программу целиком, нажав любую клавишу. Полученная программа будет загружаться, как обычно, командой LOAD "".

Заметим, что все сказанное никак не касается обладателей дисковой версии, так как все необходимое в этой версии уже сделано.

Теперь вы можете изменить количество «жизней» или получить бессмертие. Для этого в загрузчик нужно включить дополнительные операторы, выполняемые после загрузки кодового блока, но до старта программы, то есть до выполнения оператора RANDOMIZE USR 26000. Если вы хотите увеличить или уменьшить число попыток, введите дополнительную строку:

```
3 POKE 36919,X
```

X - это и есть требуемое число «жизней». Помните только, что оно не может превышать 10 и, уж конечно, не должно быть нулем. Стать «бессмертным» поможет другая строка:

```
4 POKE 38079,0: POKE 38080,195
```

И, наконец, самое полезное усовершенствование, которое мы можем предложить для облегчения работы с Games Designer. Если вы уже успели попробовать изменить спрайты, то, вероятно, заметили, что редактор спрайтов, мягко говоря, не слишком удобен в работе. Возможно, у вас и вовсе пропадет охота возиться с этим. Мы можем предложить вам значительно более простой и удобный способ получения новой графики с помощью любого графического редактора, например, The Artist II или Art Studio.

Этот способ состоит в следующем: картинки спрайтов рисуются в графическом редакторе, и экранный файл записывается на магнитный носитель. Далее экранный файл загружается в специальную программу - генератор спрайтов, которая «вытаскивает» спрайты из экранной картинки и сохраняет их на ленте. Естественно, можно пользоваться и готовыми экранными файлами.

При рисовании спрайтов нужно помнить несколько правил:

картинки имеют размеры 2x2 знакоместа, но по краям два ряда пикселей должны быть пустыми;

спрайты нужно расположить в два ряда вверху экрана в порядке возрастания номеров слева направо (рис 10.9);

имеет значение только цвет тона INK (цвет фона PAPER, как вы помните, устанавливается в программе одновременно для всех спрайтов).



Рис. 10.9. Изготовление спрайтов в графическом редакторе.

Когда вы получите экранный файл, запишите его на ленту, а затем запустите приведенную ниже программу. Если она набрана без ошибок, то через несколько секунд в верхнем левом углу экрана появится сообщение ОК (при получении сообщения ERROR IN DATA проверьте набор). Далее программа запросит экранную картинку (Screen\$:), загрузит и обработает ее. Затем на запрос Sprite: нужно ввести имя, под которым

будет записан кодовый блок со спрайтами. При повторном использовании программы ее можно запускать со строки 100.

Полученные таким способом спрайты могут быть включены в игру, если вы вставите в бейсик-загрузчик «gd» строку

```
3 LOAD ""CODE ADDR
```

где адрес загрузки спрайтов ADDR зависит от номера игры, в которой вы заменяете спрайты:

Номер игры	ADDR
1	29696
2	45056
3	47104
4	49152
5	51200
6	53248
7	55296
8	57344

Ниже приведен текст генератора спрайтов для Games Designer. Она рассчитана на работу с магнитофоном. Для работы с дисководом перед операторами LOAD и SAVE нужно написать RANDOMIZE USR 15619: REM: и в строке 310 максимальное количество символов в имени (число 10) заменить на 8.

```
10 CLS : RESTORE : LET s=0
20 FOR n=60000 TO 60107
30 READ a: POKE n, a: LET s=s+a
40 NEXT n
50 IF s<>12462 THEN PRINT "ERROR IN DATA": STOP
60 PRINT "OK"
100 LET a$="Screen$: ": GO SUB 300
110 LOAD n$CODE 16384
120 RANDOMIZE USR 60000
130 LET a$="Sprite: ": GO SUB 300: IF n$="" THEN GO TO 130
140 SAVE n$CODE 29696,1088 150 STOP
300 INPUT (a$); LINE n$
310 IF LEN n$>10 THEN LET n$=n$( TO 10)
320 RETURN
1000 DATA 33,0,64,17,0,116,221,33
1010 DATA 32,120,205,124,234,33,64,64
```

---

1020 DATA 205, 124, 234, 235, 6, 32, 54, 0  
1030 DATA 35, 16, 251, 201, 6, 16, 197, 229  
1040 DATA 229, 205, 175, 234, 205, 186, 234, 225  
1050 DATA 1, 32, 0, 9, 205, 186, 234, 205  
1060 DATA 175, 234, 225, 229, 124, 230, 24, 203  
1070 DATA 47, 203, 47, 203, 47, 198, 88, 103  
1080 DATA 126, 230, 7, 221, 119, 0, 225, 35  
1090 DATA 35, 221, 35, 193, 16, 208, 201, 175  
1100 DATA 6, 2, 18, 19, 18, 19, 36, 16  
1110 DATA 249, 201, 6, 6, 126, 230, 63, 18  
1120 DATA 35, 19, 126, 230, 252, 18, 43, 36  
1130 DATA 19, 16, 241, 201

## ПРИЛОЖЕНИЕ 1. НЕСКОЛЬКО ИГРОВЫХ ПРОГРАММ

После того, как вы немало потрудились, изучая совсем не простые вопросы создания игровых программ, неплохо было бы теперь поиграть, введя в компьютер одну из таких программ. Предлагаем несколько игр, к каждой из которых даются правила, и листинги программ, а для некоторых - еще и краткие пояснения на тот случай, если вы захотите более глубоко изучить особенности их построения и использовать в собственных разработках.

Если программа любой из предлагаемых игр будет набрана без ошибок, в нее сразу можно начинать играть, однако, если вы хотите нанести окончательный блеск, советуем проделать небольшую, но очень полезную работу - ввести в игру ее правила на русском языке, воспользовавшись программами 10 или 11.

Для этого предлагаем такую последовательность действий:

1. Введите в компьютер бейсик-программу нового набора символов и запустите ее оператором RUN. В результате этого в памяти компьютера образуется кодовый блок, содержащий русские буквы.

2. Запишите этот кодовый блок на ленту с помощью оператора

```
SAVE "NEW_SYMBOL"CODE 64256,768
```

3. Загрузите игровую бейсик-программу с ленты или введите ее из книги, после чего добавьте в текст строки:

```
1 CLEAR 64255
```

```
2 LOAD ""CODE: POKE 23607,250
```

и запустите оператором RUN. При этом с ленты потребуется ввести записанную ранее программу NEW\_SYMBOL.

4. После окончания загрузки на свободные строки игровой программы можно вводить русский текст. Но лучше всего образовать отдельную подпрограмму «Правила» где-нибудь в конце программы.

### **МИНЫ**

Цель игры состоит в том, чтобы пройти по полю, на котором случайным образом расставлены мины. Количество мин вы задаете сами в начале игры, после того как появится надпись RANG [10-150].



Рис. П.1. Игра МИНЫ.

Пройти необходимо из левого нижнего угла экрана в правый верхний угол по любой траектории, управляя человечком с помощью следующих клавиш:

- 2 - вверх, 3 - вверх и вправо,
- S - вниз, 1 - вверх и влево,
- E - вправо, D - вниз и вправо,
- Q - влево, A - вниз и влево.

В процессе игры в верхней части экрана выводится общее число пройденных шагов, а в нижней - количество мин в клетках вокруг человечка.

## Программа 34. МИНЫ.

```
30 POKE 23658,8
50 FOR N=0 TO 23
60 READ S
70 POKE USR "A"+N,S
80 NEXT N
90 DATA 56,56,16,254,56,56,40,40
100 DATA 129,90,36,90,90,36,90,129
110 DATA 170,85,170,170,170,85,170,170
115 BORDER 1: PAPER 1: INK 6: CLS
117 DIM P(30,18)
120 PRINT AT 3,6; "*****"
130 PRINT AT 7,6; "*****"
140 FOR N=4 TO 6
150 PRINT AT N,6; "*"
160 PRINT AT N,25; "*"
170 NEXT N
180 PRINT AT 5,11; INK 3; "M I N Y"
190 LET A$="PRESS ANY KEY TO CONTINUE"
195 LET C=2
200 FOR N=1 TO 25
205 IF INKEY$<>" " THEN GO TO 250
210 LET C=C+1: IF C>7 THEN LET C=2
220 PRINT AT 18,N+2; INK C;A$(N TO N)
230 NEXT N
240 IF INKEY$="" THEN GO TO 200
250 CLS
260 INPUT AT 13,0;AT 0,8;"RANG (10-150):";MIN
270 IF MIN<10 OR MIN>150 THEN GO TO 250
280 PRINT AT 11,10; INK 6;"PLEASE WAIT!"
300 FOR N=1 TO MIN
310 LET V=INT (RND*18+1)
320 LET H=INT (RND*30+1)
330 IF P(H,V)=1 THEN GO TO 310
340 LET P(H,V)=1
350 NEXT N
355 LET P(1,18)=0: LET P(30,1)=0
360 FOR N=1 TO 18
370 PRINT AT N,2; INK 4;PAPER 0;"CCCCCCCCCCCCCCCCCCCCCCCC"
380 NEXT N
390 LET SA=0: LET X=1: LET Y=18: LET X1=X: LET Y1=Y
400 PRINT AT 1,31; INK 6;"C"
410 IF INKEY$="2" THEN LET Y=Y-1: GO TO 500
415 IF INKEY$="S" THEN LET Y=Y+1: GO TO 500
420 IF INKEY$="Q" THEN LET X=X-1: GO TO 500
425 IF INKEY$="E" THEN LET X=X+1: GO TO 500
430 IF INKEY$="1" THEN LET X=X-1: LET Y=Y-1: GO TO 500
435 IF INKEY$="3" THEN LET X=X+1: LET Y=Y-1: GO TO 500
440 IF INKEY$="D" THEN LET X=X+1: LET Y=Y+1: GO TO 500
445 IF INKEY$="A" THEN LET X=X-1: LET Y=Y+1: GO TO 500
450 GO TO 510
500 LET SA=SA+1: PRINT AT 0,14; INK 7; PAPER 1;"SHAG:";SA:BEEP 0.01,-5
502 IF X>30 OR X<1 OR Y>18 OR Y<1 THEN LET X=X1: LET Y=Y1:GO TO 410
505 PRINT AT Y1,X1+1; INK 1; PAPER 4;"C"
510 IF X>30 OR X<1 OR Y>18 OR Y<1 THEN LET X=X1: LET Y=Y1:GO TO 410
512 LET KOL=0
515 IF X>1 THEN IF P(X-1,Y)=1 THEN LET KOL=KOL+1
520 IF X<30 THEN IF P(X+1,Y)=1 THEN LET KOL=KOL+1
```

```
525 IF Y>1 THEN IF P(X,Y-1)=1 THEN LET KOL=KOL+1
530 IF Y<18 THEN IF P(X,Y+1)=1 THEN LET KOL=KOL+1
535 IF X>1 THEN IF Y>1 THEN IF P(X-1,Y-1)=1 THEN LET KOL=KOL+1
540 IF X<30 THEN IF Y>1 THEN IF P(X+1,Y-1)=1 THEN LET KOL=KOL+1
545 IF X<30 THEN IF Y<18 THEN IF P(X+1,Y+1)=1 THEN LET KOL= KOL+1
550 IF X>1 THEN IF Y<18 THEN IF P(X-1,Y+1)=1 THEN LET KOL=KOL+1
570 IF KOL<>0 THEN PRINT AT 21,10; INK 2;"RADOM ";KOL;" MIN !": GO TO 590
580 PRINT AT 21,0;"oooooooooooooooooooooooooooo"
590 PRINT AT Y,X+1; INK 2; PAPER 5;"A"
600 IF P(X,Y)=1 THEN GO TO 650
605 LET X1=X: LET Y1=Y
610 IF X=30 AND Y=1 THEN GO TO 1000
615 GO TO 410
620 REM -----
650 FOR N=0 TO 7
655 BEEP 0.01,-10: BEEP 0.01,-10: BEEP 0.01,-1:BEEP 0.01,10
660 PAUSE 10: BORDER N
670 NEXT N
680 BORDER 1
690 PAUSE 50
700 FOR N=1 TO 30
710 FOR M=1 TO 18
720 IF P(N,M)=1 THEN PRINT AT M,N+1; PAPER 0; INK 3;"B"
730 NEXT M
740 NEXT N
750 PAUSE 500
755 PAPER 0: BRIGHT 1
760 FOR N=0 TO 10
770 PRINT AT N,0; INK 1;"ooooooooooooooooGAME "
780 PRINT AT 20-N,18; INK 2;"OVERoooooooooooo"
790 PAUSE 5: BEEP 0.01,N*3
800 NEXT N
810 BRIGHT 0
820 FOR N=0 TO 9
830 PRINT AT N,13;"oooooooooooooooooooo"
840 PRINT AT 20-N,0;"oooooooooooooooooooo"
850 PAUSE 5: BEEP 0.01,30-N*3
860 NEXT N
870 PRINT AT 21,0;"oooooooooooooooooooooooooooooooooooo"
880 FOR M=1 TO 5
890 FOR N=1 TO 7
900 PRINT AT 10,13; INK N;"GAME OVER"
910 BEEP 0.01,RND*40
920 NEXT N: NEXT M
930 GO TO 115
990 REM -----
1000 FOR N=1 TO 15
1010 FOR M=1 TO 18
1020 IF P(N,M)=1 THEN PRINT AT M,N+1; INK 3; PAPER 0;"B"
1030 IF P(31-N,M)=1 THEN PRINT AT M,32-N; INK 3; PAPER 0;"B"
1040 NEXT M
1050 NEXT N
1060 PAUSE 0
1070 GO TO 115
```

## **ЛИСЫ**

Охота на лис - это логическая игра, название которой дала игра радиолобительская, заключающаяся в том, что спортсмены с радиоприемниками передвигаются по пересеченной местности, пытаются «засечь» и найти спрятанные

передатчики («лисы») и получая за это определенные очки.



РИС. П.2. Игра ЛИСЫ.

Цель нашей игры - обнаружить и уничтожить пятерых «лис», спрятанных на поле размером 9x9 клеток. Играющий может просматривать клетки поля, вводя два числа, X и Y, где X номер клетки по горизонтали, а Y - номер клетки по вертикали. Номера клеток проставлены слева и сверху поля. После ввода координат клетки на поле в заданном месте загорается число. Это число показывает, сколько «лис» находится на одной линии с этой точкой, считая по горизонтали, вертикали и двум диагоналям вместе. Например, если на одной прямой по горизонтали (вертикали, диагонали) с вашей точкой есть три «лисы», то в этой клетке загорится число 3. Другой пример. Пусть на горизонтали Y=5 имеется 2 «лисы», на вертикали X=3 есть одна «лиса» и на диагоналях, проходящей через точку X=3, Y=5 также есть одна «лиса». Если теперь ввести эти координаты, то вы увидите в соответствующей клетке поля число 4.

Выстрел за выстрелом, на поле будут появляться числа, на основании которых можно вычислить положения всех «лис». Побеждает тот, кто за меньшее количество выстрелов обнаружит 5 «лис». Для выхода из игры необходимо вместо координат ввести два нуля, то есть X=0, Y=0.

Программа 35. ЛИСЫ.

```

10 BORDER 0: PAPER 0: INK 5
15 OVER 0: FLASH 0: BRIGHT 0
18 CLS
19 GO SUB 6100
20 GO SUB 7000
22 LET HOD=50: LET RE=50
40 GO SUB 5000
45 CLS
47 PAUSE 30
50 DIM M$(3,22)
55 LET F=0
60 LET M$(1)="nnnnnnSTART GAME"
62 LET M$(2)="nnnnnnSAVE program"
64 LET M$(3)="nnnnnnQUIT PROGRAM"
70 PRINT AT 2,5; INK 3;"***"; INK 6;" HUNTING THE FOX ";INK 3;"***"
72 INK 1: PLOT 39,59: DRAW 0,64: DRAW 177,0: DRAW 0,-64:DRAW -177,0
75 INK 5
80 FOR N=1 TO 3
90 PRINT AT N*2+6,5;M$(N)
100 NEXT N
102 PAPER 1: INK 5: PRINT AT 18,13;"nnnnnn": PRINT AT 18,15 ;RE: PAPER 0
103 PRINT AT 16,13; INK 2;"record"
105 PRINT AT 8,5; PAPER 1;M$(1)
107 PRINT AT 20,0; INK 4;"6-down, 7-up, ENTER-start"
110 LET No=1
120 LET Noo=No
130 LET a=CODE INKEY$
    
```

```
135 IF INKEY$="6" THEN LET No=No+1: GO TO 160
140 IF INKEY$="7" THEN LET No=No-1: GO TO 160
145 IF a=13 OR INKEY$="0" THEN GO TO 210
150 GO TO 130
160 IF No=0 THEN LET No=3
170 IF No=4 THEN LET No=1
180 PRINT AT No*2+6,5; PAPER 1;M$(No)
190 PRINT AT Noo*2+6,5; PAPER 0;M$(Noo)
195 BEEP 0.01,20: FOR N=1 TO 40: NEXT N
200 GO TO 120
210 BEEP 0.05,0
215 IF No=1 THEN GO TO 250
220 IF No=2 THEN GO TO 5800
230 IF No=3 THEN GO TO 6000
240 REM -----START GAME-----
250 DIM P$(9,9)
255 LET HOD=0: LET PROM=0:
257 LET POP=0
260 CLS
290 FOR N=1 TO 5
300 LET X=INT (RND*9+1)
310 LET Y=INT (RND*9+1)
320 IF P$(X,Y)="L" THEN GO TO 300
330 LET P$(X,Y)="L"
340 NEXT N
350 CLS
360 REM -----
400 FOR N=12 TO 156 STEP 16
410 PLOT N,163: DRAW 0,-144
420 NEXT N
430 FOR N=163 TO 15 STEP -16
440 PLOT 12,N: DRAW 144,0
450 NEXT N
455 INK 6
460 FOR N=1 TO 9
470 PRINT AT N*2,0;N
480 NEXT N
490 PRINT AT 0,2;"1 2 3 4 5 6 7 8 9"
491 INK 5
492 PLOT 163,163: DRAW 90,0: DRAW 0,-144: DRAW -90,0: DRAW 0,144: PLOT 165,161: DRAW
86,0: DRAW 0,-140: DRAW -86,0: DRAW 0,140
493 INK 3: PRINT AT 3,23;"SHOTS:": PRINT AT 8,24;"HITS:": PRINT AT 13,22;"X and Y":
PRINT AT 14,21;"LAST SHOT"
495 PAPER 1: PRINT AT 5,23;"#####": PRINT AT 10,23;"#####" 497 PAPER 0
500 REM -----
502 LET X=0: LET Y=0
510 LET X1=X: LET Y1=Y: GO SUB 2000
511 INPUT "X=";X,"Y=";Y
514 IF X=0 AND Y=0 THEN GO SUB 5000: GO TO 45
515 IF X<1 OR X>9 OR Y<1 OR Y>9 THEN BEEP 0.5,-23: GO TO 511
520 LET HOD=HOD+1
525 LET KOL=0
530 IF P$(X,Y)="L" THEN GO TO 1000
540 REM ----1----
550 FOR N=1 TO 9
560 IF P$(N,Y)="L" THEN LET KOL=KOL+1
570 NEXT N
580 REM ----2--
590 FOR N=1 TO 9
600 IF P$(X,N)="L" THEN LET KOL=KOL+1
```

```

610 NEXT N
620 REM ---3---
625 LET M=Y
630 LET N=X
650 LET M=M-1
670 LET N=N-1
675 IF N<1 OR M<1 THEN GO TO 700
680 IF P$(N,M)="L" THEN LET KOL=KOL+1
690 GO TO 650
700 LET N=X: LET M=Y
710 LET N=N+1: LET M=M+1
720 IF N>9 OR M>9 THEN GO TO 750
730 IF P$(N,M)="L" THEN LET KOL=KOL+1
740 GO TO 710
745 REM ----4----
750 LET N=X: LET M=Y
760 LET N=N-1: LET M=M+1
770 IF N<1 OR M>9 THEN GO TO 800
780 IF P$(N,M)="L" THEN LET KOL=KOL+1
790 GO TO 760
800 LET N=X: LET M=Y
810 LET N=N+1: LET M=M-1
820 IF N>9 OR M<1 THEN GO TO 850
830 IF P$(N,M)="L" THEN LET KOL=KOL+1
840 GO TO 810
850 GO SUB 1050
860 GO TO 510
1000 REM --P.P. PRINT FOX-
1010 PRINT AT Y*2,X*2; BRIGHT 1; INK 2;"A"
1015 FOR m=1 TO 2: FOR n=20 TO 0 STEP -5: BEEP 0.03,n: PAUSE 1: NEXT n: FOR n=0 TO 20
STEP 5: BEEP 0.03,n: PAUSE 1: NEXT n: NEXT m
1020 LET POP=POP+1
1025 GO SUB 2000
1030 IF POP>=5 THEN GO SUB 3000: GO SUB 5000: GO TO 45
1040 GO TO 510
1050 REM --P.P. PRINT --
1060 INK 4: BRIGHT 1: PRINT AT Y*2,X*2;KOL
1065 BRIGHT 0: BEEP 0.06,-5
1070 RETURN
2000 REM ----P.P. PRINT
2010 PAPER 1: INK 4: PRINT AT 5,25;HOD
2020 PRINT AT 10,25;POP
2030 PRINT AT 17,22;"h= u": PRINT AT 17,24;X1: PRINT AT 17,29;Y1
2035 PAPER 0
2040 RETURN
2140 REM -----
3000 REM ----KONEC----
3010 FOR N=1 TO 100: NEXT n
3020 PRINT AT 21,0; INK 2; BRIGHT 1;" BBBBBBBBBBBB ";BRIGHT 0; INK 4;HOD;BRIGHT 1;
INK 2;" BBBBBBBBBBBB":BRIGHT 0
3030 PAUSE 0
3035 IF RE>HOD THEN LET RE=HOD
3040 RETURN
5000 REM ---- Zastawka ----
5002 LET hod=50
5005 CLS
5010 DIM L$(9,32)
5020 LET L$(1)="nnnnnnbbb bnnnnbnnbbbnnnnnnnn"
5022 LET L$(2)="nnnnnnnnn bnnnnn bnnnnn bnnnnnnn"
5024 LET L$(3)="nnnnnbnnnn bnnnnn bnnnnnnnnnnnnn"

```

```

5026 LET L$(4)="nnnnnnnnn BnnnnBB BnnnnnnnnBBBBBnnn"
5028 LET L$(5)="nnnnnnnnn Bnnn B Bnnnnnnnnnnnnn B"
5030 LET L$(6)="nnnnnnnnn B Bnnn Bnnnnnnnnnnnnn B"
5032 LET L$(7)="nnnnnnnnn BBnnnn Bnnnnnnnnnnnnn B"
5034 LET L$(8)="nnnnnnnnn Bnnnnn Bnnnnn Bnnnnn B"
5036 LET L$(9)="BBnnnnnnn BnnnnnnnnBBBBBnnBBBBBnnn"
5040 BRIGHT 1
5045 FOR N=32 TO 0 STEP -1
5050 FOR M=9 TO 1 STEP -1
5060 PRINT AT M+6,0; INK 2;L$(M)(N+1 TO 32)
5070 NEXT M
5075 BEEP 0.003,20
5077 IF INKEY$<>" " THEN BRIGHT 0: RETURN
5080 NEXT N
5185 BRIGHT 0
5190 LET B$="SANKT-PETERBURG": LET YK=0: LET XK=8: LET CK=5 : GO SUB 5510
5200 LET B$="1993": LET YK=2: LET XK=13: LET CK=4:GO SUB 5500
5210 LET B$="AUTHOR:K. Igor": LET YK=4: LET XK=9: LET CK=6:GO SUB 5510
5220 LET B$="Press any key to continue": LET YK=20:LET XK=3 : LET CK=7: GO SUB 5510
5225 LET P=0
5230 REM -----
5240 LET C=INT (RND*6+1)
5245 LET B=INT <RND*1.1>
5250 LET Y=INT (RND*9+1)
5260 LET X=INT (RND*32+1)
5265 IF L$(Y,X)=" " THEN GO TO 5250
5270 PRINT AT Y+6,X-1; BRIGHT B; INK C;L$(Y,X)
5275 LET P=P+1
5280 IF INKEY$<>" " OR P>2000 THEN RETURN
5290 GO TO 5240
5499 STOP
5500 REM ---- B. KWADRAT ----
5510 LET S=LEN B$
5520 FOR N=1 TO S
5530 LET C$=B$(N TO N)
5540 PRINT AT YK,XK+N; INK CK;C$; PAPER 3;" "
5545 BEEP 0.03,10
5550 NEXT N
5555 PRINT AT YK,N+XK;" "
5557 PAUSE 15
5560 RETURN
5800 REM ----SAVE PROGRAM----
5810 CLS
5820 PRINT AT 9,1; INK 5;"SAVE program HUNTING THE FOX"
5830 SAVE "FOXES" LINE 10
5840 PAUSE 30
5850 GO TO 45
6000 REM --WYHOD IZ PROGRAMY--
6010 BORDER 1: PAPER 1: INK 5
6020 CLS
6030 STOP
6100 REM -- Graphics -----
6110 FOR N=0 TO 15
6120 READ S
6130 POKE USR "A"+N,S
6140 NEXT N
6150 RETURN
6160 DATA 2,30,46,254.30,127,15,15
6170 DATA 124,222,158,254,254,254,124,0
7000 REM ----BEEP----

```

```
7010 BEEP 0.1,0: BEEP 0.1,5:BEEP 0.1,10: BEEP 0.1,15:BEEP 0.1,20
7020 RETURN
```

### МАКСИТ

Начнем с того, что запишем на ленту маленькую программу-загрузчик:

```
10 LOAD ""CODE 16384 20 LOAD ""
```

В этой программе строка 10 загружает в экранную область памяти (с адреса 16384) картинку-заставку, а строка 20 - игровую программу МАКСИТ. Таким образом, после программы-загрузчика на ленте должен располагаться созданный вами ранее (см. главу 2) экранный файл, а вслед за ним - программа МАКСИТ, текст которой приводится ниже.

После загрузки игры на экране появляется поле в виде таблицы чисел 8x8 клеток и курсор в левом нижнем углу.



Рис. П.3. Игра МАКСИТ.

Каждое число задается программой случайным образом, но в пределах от 0 до 9. Справа от игрового поля выводятся имена играющих и набранные каждым из них очки. Играющие по очереди выбирают числа, причем выбранное число удаляется с игрового поля и прибавляется к счету играющего. Побеждает тот, кто набрал к концу игры большее количество очков.

Первый играющий может перемещать курсор только по горизонтали, выбирая любое число в строке. Его партнер может перемещать курсор только по вертикали и должен выбрать одно из чисел в столбце, отмеченном курсором. Можно играть, задав ограниченное время размышления над каждым ходом, для этого пригодится «секундомер», показывающий общее время игры (он расположен справа в нижней части экрана).

Программа 36. МАКСИТ.

```
10 POKE 23658,8
15 DIM a$(2,10): DIM t$(8,10): DIM o(8): DIM k$(6)
20 GO SUB 9050
30 LET ko1=1
40 PAPER 0: INK 3: OVER 0: FLASH 0: BRIGHT 0: BORDER 0
50 CLS
60 LET w$="": LET v$="": LET i=0
70 LET wx=12: LET wy=18
100 LET c=6: GO SUB 9000
110 PRINT AT 0,0;"*****ппМАКСИТпп*****"
115 PRINT #0;AT 1,0;INK 7-ko1; "*****"
120 RESTORE 9910: GO SUB 9600
130 LET x$="OFF": IF ko1=1 THEN LET x$="ON"
140 PRINT AT row+7,col+11; INK 8; PAPER 8;x$
160 LET c=3: GO SUB 9000
170 IF INKEY$="1" THEN GO SUB 9120: GO TO 1000
171 IF INKEY$="2" THEN GO SUB 9120: GO TO 200
```

```
172 IF INKEY$="3" THEN GO SUB 9120: GO SUB 400: PAUSE 0: GO TO 100
173 IF INKEY$="4" AND ko1=1 THEN GO SUB 9120: LET ko1=0: GO TO 50
174 IF INKEY$="4" AND ko1=0 THEN GO SUB 9120: LET ko1=1: GO TO 50
175 IF INKEY$="6" THEN GO SUB 9120: STOP
176 IF INKEY$="5" THEN GO SUB 9120: GO TO 700
180 GO SUB 9500
190 GO TO 170
200 REM ----DEFINE KEYS----
210 RESTORE 9920: GO SUB 9600
215 LET p=3
220 FOR m=1 TO 6
222 IF m=3 THEN LET p=5
225 PRINT AT (M-1)*2+row+p,col+12; INK 8; PAPER 8;"?"
230 PAUSE 0: LET b$=INKEY$
240 IF m=1 THEN GO TO 270
250 FOR n=1 TO m-1: IF b$=k$(n) THEN GO TO 230
260 NEXT n
270 IF b$<" " AND CODE b$<>13 THEN GO TO 230
280 LET k$(m)=b$
290 IF b$=" " THEN LET b$="SPACE"
300 IF CODE b$=13 THEN LET b$="ENTER"
310 GO SUB 9200
320 PRINT AT (M-1)*2+row+p,col+12; INK 8; PAPER 8;b$
330 NEXT m
340 GO TO 100
400 REM ---- HI SCORE ----
410 RESTORE 9930: GO SUB 9600
420 PRINT AT 6,8; PAPER 8; INK 8;"N1 ";t$(1);TAB 22;o(1)
430 LET c=6: GO SUB 9000
440 FOR n=2 TO 8
445 PRINT AT n*2+4,8; PAPER 8; INK 8;"N";n;" ";t$(n);TAB 22;o(n)
450 NEXT n
452 LET cc=0
465 IF ko1=0 THEN PAUSE 3: RETURN
470 IF cc>4 THEN LET cc=0
472 INK cc: LET cc=cc+1
475 PRINT AT row+1,col+7; PAPER 8;"HI SCORE"
480 IF INKEY$="" AND 1=0 THEN GO TO 470
485 GO SUB 9120
490 RETURN
700 REM -- SET TIME ---
710 RESTORE 9940: GO SUB 9600
720 LET c$="": PRINT AT row+1,col+14;
730 FOR n=1 TO 6
740 PAUSE 0: LET b$=INKEY$
750 IF b$<"0" OR b$>"9" THEN GO TO 740
760 LET c$=c$+b$
770 PRINT PAPER 8; INK 8;b$;
780 IF n=2 OR n=4 THEN PRINT PAPER 8; INK 8;":";
785 GO SUB 9120
790 NEXT n
800 LET ch=VAL c$( TO 2)
820 LET min=VAL c$(3 TO 4)
830 LET s=VAL c$(5 TO )
840 IF ch>24 OR min>60 OR s>60 THEN GO TO 700
850 LET tim=(ch*3600+min*60+s)*50
860 POKE 23674,INT (tim/65536): LET tim=tim-PEEK 23674*65536
870 POKE 23673,INT (tim/256): LET tim=tim-PEEK 23673*256
880 POKE 23672,INT tim
890 GO TO 100
```

```
1000 REM ---- START GAME ----
1001 FOR N=1 TO 24: RANDOMIZE USR 3582: BEEP .02,N: NEXT N
1005 LET wx=20: LET wy=16
1010 CLS
1020 GO SUB 9300
1030 CLS
1040 DIM Z(8,8)
1050 FOR n=1 TO 8
1060 FOR m=1 TO 8
1070 LET z(m,n)=INT (RND*10)
1075 PRINT AT n*2,m*2;z(m,n)
1080 NEXT m
1090 NEXT n
1095 LET c=6: GO SUB 9000
1100 FOR n=163 TO 35 STEP -16
1110 PLOT 12,n: DRAW 128,0
1120 NEXT n
1130 FOR n=12 TO 140 STEP 16
1140 PLOT n,35: DRAW 0,128
1150 NEXT n
1151 PAPER (7-6*ko1): LET c=6: GO SUB 9000
1152 PRINT AT 3,19;"oooooooooooo": REM 10 SPACE
1153 PRINT AT 8,19;"oooooooooooo": REM 10 SPACE
1155 IF ko1=0 THEN INK 0
1156 PRINT AT 3,19;v$
1157 PRINT AT 8,19;u$
1160 LET z(1,8)=10
1170 PRINT AT 16,2;" "
1171 LET c=6: GO SUB 9000
1172 PLOT 145,35: DRAW 0,128
1174 DRAW 93,0: DRAW 0,-128
1175 DRAW -93,0
1180 LET x=1: LET y=8
1190 LET x1=x: LET y1=y
1200 LET p1=1
1220 DIM u(2)
1300 REM ----
1310 LET c=6: GO SUB 9000
1320 PAPER ko1
1330 PRINT AT y*2,x*2; FLASH 1; OVER 1;" "
1335 IF p1=2 THEN GO TO 1360
1340 IF INKEY$=k$(1) THEN LET x=x-1: GO TO 1400
1350 IF INKEY$=k$(2) THEN LET x=x+1: GO TO 1400
1352 IF INKEY$=k$<5> THEN GO TO 1750
1354 IF INKEY$=k$(6) THEN GO TO 1500
1355 GO SUB 9500
1356 GO TO 1340
1360 IF INKEY$=k$(3) THEN LET y=y-1: GO TO 1400
1365 IF INKEY$=k$(4) THEN LET y=y+1: GO TO 1400
1370 IF INKEY$=k$(5) THEN GO TO 1750
1375 IF INKEY$=k$(6) THEN GO TO 1500
1377 GO SUB 9500
1380 GO TO 1360
1390 REM -----
1400 BEEP .0007,57
1405 GO SUB 9500
1406 IF x<1 OR x>8 OR y<1 OR y>8 THEN LET x=x1: LET y=y1:GO TO 1335
1430 PAPER 0: LET c=5: GO SUB 9000
1440 PRINT AT y1*2,x1*2; OVER 1;" "
1450 LET x1=x: LET y1=y
```

```
1460 GO TO 1310
1500 REM -----
1505 BEEP .0006,55: BEEP .0007,30
1510 IF z(x,y)=10 THEN GO TO 1400
1520 PRINT AT y*2,x*2; FLASH 1;" "
1530 IF p1=1 THEN LET u(1)=u(1)+z(x,y): LET p1=2: GO TO 1600
1540 LET u(2)=u(2)+z(x,y): LET p1=1
1600 LET z(x,y)=10
1605 LET end=1
1610 FOR n=1 TO 8
1620 IF p1=1 THEN IF z(n,y)<>10 THEN LET end=0: GO TO 1640
1630 IF p1=2 THEN IF z(x,n)<>10 THEN LET end=0
1640 NEXT n
1690 REM ---
1700 INK 7: PAPER 0
1702 IF p1=1 THEN INVERSE 1
1705 PRINT AT 5,23;u(1): INVERSE 0
1707 IF p1=2 THEN INVERSE 1
1710 PRINT AT 10,23;u(2)
1715 INVERSE 0
1720 IF end=0 THEN GO TO 1400
1730 PAPER 0
1740 PRINT AT y*2,x*2;" "
1750 REM ---- SCORE ----
1760 LET q$(1)=v$: LET q$(2)=w$
1765 LET i=1: GO SUB 400
1770 FOR m=1 TO 2
1780 IF u(m)<o(8) THEN GO TO 1860
1790 LET j=8: FOR n=1 TO 8
1800 IF u(m)>o(n) THEN LET j=n: GO TO 1820
1810 NEXT n
1820 FOR n=8 TO j+1 STEP -1
1830 LET o(n)=o(n-1): LET t$(n)=t$(n-1)
1840 NEXT n
1850 LET t$(j)=q$(m): LET o(j)=u(m)
1855 PAUSE 100: BEEP .1,0: GO SUB 400
1860 NEXT m
1865 PAUSE 100
1870 GO TO 40
9000 REM ---- color ----
9010 IF kol=1 THEN INK c: RETURN
9020 INK 7
9030 RETURN
9050 REM .---- DIM DATA ----
9060 RESTORE 9110
9070 READ t$(1),o(1)
9080 FOR n=1 TO 6
9090 READ k$(n)
9095 NEXT n
9100 FOR n=2 TO 8: LET o(n)=0: LET t$(n)=".....":NEXT n
9105 RETURN
9110 DATA "Igor K.",200,"0","P","Q","A","E","M"
9120 REM ---- BEEP 1 ----
9130 BEEP .0004,55
9140 BEEP .0006,40
9150 BEEP .0004,55
9160 RETURN
9190 REM ---- BEEP 2 ----
9200 BEEP .0009,30
9210 BEEP .003,10
```

```
9220 BEEP .0009,45
9230 RETURN
9300 REM ---- YOU NAME ----
9310 FOR n=1 TO 2
9320 LET c=6: GO SUB 9000
9330 PRINT AT 8,13;"PLAYER ";n
9340 PRINT AT 10,9;"ENTER YOU NAME:"
9350 LET c=2: GO SUB 9000
9360 PRINT AT 12,10;">nnnnnnnnnn<"
9365 INK 7
9370 PRINT AT 12,11;"....."
9375 LET c$=""
9380 LET c=5: GO SUB 9000
9390 FOR m=1 TO 10
9395 PAUSE 0: LET b$=INKEY$
9396 IF CODE b$=12 AND c$<>"" THEN LET c$=c$( TO LEN c$-1): LET m=m-2: GO TO 9405
9397 IF CODE b$=13 THEN GO TO 9430
9400 IF b$<" " OR b$>"Z" THEN GO TO 9395
9402 LET c$=c$+b$
9405 GO SUB 9490
9408 PRINT AT 12,11;"....."
9410 PRINT AT 12,11;c$
9420 NEXT m
9430 BEEP .01,10
9440 CLS
9450 IF n=1 THEN LET v$=c$: GO TO 9470
9460 LET w$=c$
9470 NEXT n
9480 RETURN
9490 REM ---- BEEP 3 ----
9495 BEEP .002,10
9496 BEEP .003,20
9497 BEEP .004,30
9498 BEEP .005,40
9499 RETURN
9500 REM ---- TIME ----
9510 LET tim=(PEEK 23672+256*PEEK 23673+65536*PEEK 23674)/50
9520 LET ch=INT (tim/3600): LET min=INT ((tim-ch*3600)/60): LET s=INT (tim-ch*3600-
min*60)
9530 IF ch>=24 THEN POKE 23672,0: POKE 23673,0:POKE 23674,0: GO TO 9500
9540 PRINT AT wy,wx; INK 8; PAPER 8; ("0" AND ch<10);ch;" ";("0" AND
min<10);min;" ";("0" AND s<10);s
9550 RETURN
9600 REM ---- WINDOW ----
9610 READ row,col,hgt,len,ink,pap
9620 PAPER pap: INK ink
9630 IF kol=0 THEN PAPER 7: INK 0
9640 LET x$="": FOR n=1 TO len: LET x$=x$+" ": NEXT n
9650 FOR n=row TO row+hgt-1
9660 PRINT AT n,col;x$: NEXT n
9670 LET len1=len*8-3: LET hgt1=hgt*8-3
9680 PLOT (col*8+1),(174-row*8): DRAW len1,0: DRAW 0,-hgt1:DRAW -len1,0: DRAW 0,hgt1
9690 READ nn: FOR n=1 TO nn
9700 READ y,x,x$: PRINT AT row+y,col+x;x$
9710 NEXT n
9720 PAPER 0: RETURN
9900 REM -- DATA WIND ---
9910 DATA 5,6,15,20,0,5
9911 DATA 6
9912 DATA 1,1,"1 - START GAME"
```

```

9913 DATA 3,1,"2 - REDEFINE KEYS"
9914 DATA 5,1,"3 - HI SCORE"
9915 DATA 7,1,"4 - COLOR"
9916 DATA 9,1,"5 - SET TIME"
9917 DATA 11,1,"6 - END OF GAME"
9920 DATA 4,7,17,18,0,3
9921 DATA 8
9922 DATA 1,1,"PLAYER 1"
9923 DATA 3,6,"LEFT:"
9924 DATA 5,5,"RIGHT:"
9925 DATA 7,1,"PLAYER 2"
9926 DATA 9,8,"UP:"
9927 DATA 11,6,"DOWN:"
9928 DATA 13,7,"END:"
9929 DATA 15,6,"FIRE:"
9930 DATA 3,5,19,22,7,1
9931 DATA 1
9932 DATA 1,7,"HI SCORE"
9940 DATA 10,3,3,26,7,2
9941 DATA 1
9942 DATA 1,1,"SET TIME:"
    
```

### ПИРАМИДА

В начале игры фишки находятся на клетках первого (нижнего) ряда игрового поля. Играющий имеет право выполнить один ход - по вертикали, горизонтали или дополнительное продвижение.

**Вертикальный ход.** Его имеет право выполнить только фишка нижнего ряда. При этом она перемещается на одну из двух свободных клеток следующего сверху ряда.

**Горизонтальный ход.** Фишка может перемещаться по своему ряду на любое количество свободных клеток. Прыгать через какие-либо фишки недопустимо.

**Дополнительные продвижения.** Чтобы получить преимущество над соперником, необходимо создавать и использовать возможности дополнительных продвижений. В процессе игры три любые фишки, в том числе свои и противника, могут занять положение треугольником две снизу,

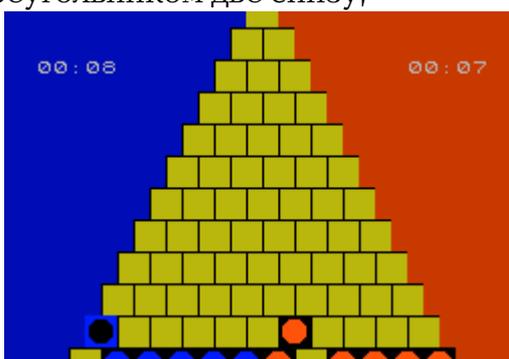


Рис. П.4. Игра ПИРАМИДА.

одна сверху. Хозяин верхней фишки после очередного хода или вместо него может продвинуть ее на клеточку следующего сверху ряда. Побеждает тот, кто умело используя дополнительные продвижения, первым достигнет вершины пирамиды. Клавиши управления.

В - выбор фишки. N - выбор поля для хода. M - выполнение хода. E - конец игры.

Программа 37. ПИРАМИДА.

```

5 BORDER 0: OVER 0: PAPER 0: FLASH 0: INK 0: CLS
7 LET pau=500
50 FOR n=0 TO 63: READ i: POKE USR "a"+n,i: NEXT n
    
```

```

55 DATA 128,128,128,128,128,128,128,128,255,128,128,128,
128,128,128,128,255,0,0,0,0,0,0,0,255,255,248,240,224,192,
192,192,255,255,31,15,7,3,3,3,192,192,192,224,240,248,255,255,
3,3,3,7,15,31,255,255,0,255,0,255,0,255,0,255
100 GO TO 3000
500 REM ----START GAME-----
505 BORDER 0: PAPER 0: INK 0: CLS
510 DIM a(13,13): DIM x(6): DIM y(6): DIM h(6): DIM v(6)
515 DIM o(14): DIM p(14): DIM r(6): DIM u(6)
520 LET c=1: LET wr1=0: LET wr2=0: LET wr11=0: LET wr12=0: LET mm1=0: LET mm2=0
600 LET t=1
605 FOR i=1 TO 11: LET t=t+1: LET a(t,i)=2: NEXT i
610 FOR i=1 TO 12: LET a(i,12)=1: NEXT i
615 FOR i=1 TO 6: LET u(i)=12: LET v(i)=12: LET R(i)=i: LET h(i)=i+6: NEXT i
620 REM *****
621 BRIGHT 0
622 FOR i=0 TO 21: PAPER 1:PRINT AT i,0;"oooooooooooooooo";PAPER
2;"oooooooooooooooo":NEXT i
625 BRIGHT 0
627 LET j=13: INK 0: PAPER 6
630 FOR i=1 TO 10: LET j=j-1: FOR t=1 TO i+1:PRINT AT i*2-1,j+t*2;"BC": PRINT AT
i*2,j+t*2;"A ":
NEXT t: NEXT i
635 PRINT AT 0,15;"nn"
640 PRINT AT 21,4: BRIGHT 1: PAPER 1;"DEDEDEDEDEDE";PAPER 2;"DEDEDEDEDEDE"; BRIGHT 0
642 PRINT AT 3,25: PAPER 2: INK 7;"00:00"
645 GO TO 1370
650 REM *****
651 GO TO 660
652 FLASH 0: LET x1=x(f): LET y1=y(f): GO SUB 2800: PRINT AT yy,xx;"DE": IF y(f)<>12
THEN PRINT AT yy+1,xx;"EG"
653 RETURN
660 LET f=1
662 BRIGHT 1
665 IF f>6 THEN LET f=1
667 PAPER c: INK 0
670 FLASH 1: LET x1=x(f): LET y1=y(f): GO SUB 2800: PRINT AT yy,xx;"DE": IF y(f)<>12
THEN PRINT AT yy+1,xx;"EG"
675 FLASH 0
680 IF INKEY$="b" OR INKEY$="B" THEN GO SUB 652: BEEP 0.02,50: LET f=f+1: GO TO 665
682 IF INKEY$="n" OR INKEY$="N" THEN BEEP 0.02,0: GO TO 750
683 IF INKEY$="e" OR INKEY$="E" THEN GO TO 3000
690 GO SUB 2100
700 GO TO 680
750 LET e=1
760 IF y(f)=12 THEN GO TO 800
765 IF a(x(f),y(f)*1)=1 AND a(x(f)+1,y(f)+1)=1 THEN GO TO 800
770 LET i=0: LET ii=y(f)
772 LET i=i+1: IF i>6 THEN GO TO 800
774 IF y(i)>ii THEN GO TO 900
775 GO TO 772
780 GO TO 900
800 IF x(f)-1<=0 THEN GO TO 805
801 IF a(x(f)-1,y(f)-1)<>1 AND a(x(f)-1,y(f)-1)<>2 THEN LET o(e)=x(f)-1: LET
p(e)=y(f)-1: LET e=e+1
805 IF a(x(f),y(f)-1)<>1 AND a(x(f),y(f)-1)<>2 THEN LET o(e)=x(f): LET p(e)=y(f)-1:
LET e=e+1
899 REM *****
900 LET k=x(f)
910 LET k=k+1

```

```
915 IF k>12 THEN GO TO 930
920 IF a(k,y(f))<>1 AND a(k,y(f))<>2 THEN LET o(e)=k: LET p(e)=y(f): LET e=e+1: GO TO
910
925 REM *****
930 LET k=x(f)
940 LET k=k-1
945 IF k<1 THEN GO TO 1000
950 IF a(k,y(f))<>1 THEN LET o(e)=k: LET p(e)=y(f):LET e=e+1: GO TO 940
960 REM *****
1000 LET i=1
1002 IF o(1)=0 OR p(1)=0 THEN GO TO 662
1005 IF o(i)=0 OR p(i)=0 THEN LET i=1
1007 IF p(i)=1 THEN PRINT AT 0,15; FLASH 1; PAPER 6; INK 0;BRIGHT 0;"HH": GO TO 1020
1010 LET x1=o(i): LET y1=p(i): GO SUB 2800: FLASH 1: BRIGHT 0: PAPER 6: INK 0: PRINT
AT yy,xx;"HH": IF y1<>12 THEN PRINT AT yy+1,xx;"HH"
1020 IF INKEY$="N" OR INKEY$="n" THEN GO SUB 2050:BEEP 0.02,0: LET i=i+1: GO TO 1005
1030 IF INKEY$="B" OR INKEY$="b" THEN GO SUB 2050: BEEP 0.02,50: FOR t=1 TO 14: LET
o(t)=0: LET p(t)=0: NEXT t: GO TO 665
1035 IF INKEY$="m" OR INKEY$="M" THEN BEEP 0.01,0: BEEP 0.01,10: BEEP 0.01,20: GO TO
1200
1036 IF INKEY$="e" OR INKEY$="E" THEN GO TO 3000
1037 GO SUB 2100
1040 GO TO 1020
1050 REM *****
1200 REM *
1201 FLASH 0
1205 IF p(i)=1 THEN BRIGHT 0: PAPER 6: INK 0: LET x1=x(f): LET y1=y(f): GO SUB 2800:
PRINT AT yy,xx;"BC": PRINT AT yy+1,xx;"A ": PRINT AT 0,15; PAPER c; INK 0; BRIGHT
1;"EG": GO TO 2300
1210 IF y(f)=12 THEN BRIGHT 0: PAPER 6: INK 0: LET x1=x(f): LET y1=12: GO SUB 2800:
PRINT AT yy,xx;"BC": BRIGHT 1: PAPER c: INK 0: LET x1=o(i): LET y1=p(i): GO SUB 2800:
PRINT AT yy,xx;"DE": LET a(x(f),y(f))=0: LET a(o(i),p(i))=1: LET x(f)=o(i): LET
y(f)=p(i): GO SUB 2000: GO TO 1300
1230 BRIGHT 0: PAPER 6: INK 0: LET x1=x(f): LET y1=y(f): GO SUB 2800: PRINT AT
yy,xx;"BC": PRINT AT yy+1,xx;"A ": BRIGHT 1: PAPER c: INK 0: LET x1=o(i): LET
y1=p(f): GO SUB 2800: PRINT AT yy,xx;"DE": PRINT AT yy+1,xx;"EG": LET a(o(i),p(i))=1:
LET a(x(f),y(f))=0: LET x(f)=o(i): LET y(f)=p(i)
1300 IF c=1 THEN GO TO 1310
1305 IF c=2 THEN GO TO 1340
1310 FOR t=1 TO 6: LET r(t)=x(t): LET u(t)=y(t): NEXT t
1320 GO TO 1350
1340 FOR t=1 TO 6: LET h(t)=x(t): LET v(t)=y(t): NEXT t
1350 IF c=1 THEN LET c=2: GO TO 1370
1360 IF c=2 THEN LET c=1
1370 IF c=1 THEN FOR t=1 TO 6: LET x(t)=r(t): LET y(t)=u(t): NEXT t
1380 IF c=2 THEN FOR t=1 TO 6: LET x(t)=h(t): LET y(t)=v(t): NEXT t
1385 FOR t=1 TO 14: LET o(t)=0: LET p(t)=0: NEXT t
1390 GO TO 660
1395 REM *****
2000 IF p(i)<>12 THEN PRINT AT yy+1,xx;"EG": RETURN
2010 RETURN
2050 LET x1=o(i): LET y1=p(i): GO SUB 2800: FLASH 0: BRIGHT 0: PAPER 6: INK 0: PRINT
AT yy,xx;"BC": IF y1<>12 THEN PRINT AT yy+1,xx;"A "
2055 LET x1=x(f): LET y1=y(f): GO SUB 2800: FLASH 1: BRIGHT 1: PAPER c: INK 0: PRINT
AT yy,xx;"DE": IF y1<>12 THEN PRINT AT yy+1,xx;"EG"
2060 RETURN
2100 REM *** VREM ***
2110 IF c=1 THEN GO TO 2130
2120 IF c=2 THEN GO TO 2200
2130 LET WR11=WR11+0.105
```

```
2140 LET WR1=INT (WR11)
2150 IF UR1=60 THEN LET WR11=0: LET wr1=0: LET mm1=mm1+1
2155 IF mm1>=60 THEN LET mm1=0
2160 INK 7: PAPER 1: BRIGHT 0: FLASH 0
2161 PRINT AT 3,1;" ";
2162 IF mm1<10 THEN PRINT AT 3,2;"0";
2164 PRINT mm1;"::"; IF wr1<10 THEN PRINT "0";
2166 PRINT wr1
2168 INK 0: BRIGHT 1
2170 RETURN
2200 LET WR12=WR12+0.105
2210 LET WR2=INT (WR12)
2220 IF WR2=60 THEN LET WR12=0: LET WR2=0: LET MM2=MM2+1
2230 IF MM2>=60 THEN LET MM2=0
2240 INK 7: PAPER 2: BRIGHT 0: FLASH 0
2250 PRINT AT 3,24;" ";
2260 IF mm2<10 THEN PRINT AT 3,25;"0";
2270 PRINT mm2;"::"; IF wr2<10 THEN PRINT "0";
2280 PRINT wr2
2285 INK 0: BRIGHT 1
2290 RETURN
2300 REM *** MUZYKA ***
2305 FOR j=4 TO 15
2310 FOR t=-20 TO 30 STEP j
2320 BEEP 0.02,t
2330 NEXT t
2340 NEXT j
2342 FOR i=1 TO 15: LET ch=INT (RND*20+10): BEEP 0.02,ch:NEXT i
2350 REM ** KONEC IGRY **
2360 GO TO 3000
2800 REM *****
2820 LET xx=2+x1*2+(12-y1): LET yy=y1*2-3
2830 RETURN
3000 REM *** MENQ ***
3010 PAPER 0: FLASH 0: BORDER 0: CLS
3015 PAPER 2
3020 FOR i=0 TO 21: PRINT AT i,0;" ": NEXT i: FOR i=0 TO 31: PRINT AT 21,i;" ": NEXT
i: FOR i=21 TO 0 STEP -1: PRINT AT i,31;" ": NEXT i: FOR i=31 TO 0 STEP -1: PRINT AT
0,i;" ": NEXT i
3025 PAPER 1
3030 FOR i=1 TO 20: PRINT AT i,1;" ": NEXT i: FOR i=1 TO 30: PRINT AT 20,i;" ": NEXT
i: FOR i=20 TO 1 STEP -1:PRINT AT i,30;" ": NEXT i: FOR i=30 TO 1 STEP -1:PRINT AT
1,i;" ": NEXT i
3035 PAPER 7
3040 FOR i=2 TO 19: PRINT AT i,2;" ": NEXT i: FOR i=2 TO 29: PRINT AT 19,i;" ": NEXT
i: FOR i=19 TO 2 STEP -1: PRINT AT i,29;" ": NEXT i: FOR i=29 TO 2 STEP -1: PRINT AT
2,i;" ": NEXT i
3045 PAPER 0
3050 PRINT AT 4,12; INK 6;"PIRAMIDA"
3060 PRINT AT 7,10; INK 1; BRIGHT 1;"Press any key";AT 9,11,"to continue"
3070 PRINT AT 12,8; INK 3;"SANKT-PETERBURG"
3080 PRINT AT 14,14; INK 7;"1992"
3082 PRINT AT 16,4; INK 3;"AUTHOR:KAPULTSEVICH IGOR"
3085 INK 7
3090 PLOT 24,87: DRAW 208,0: PLOT 24,85: DRAW 208,0
3100 IF INKEY$<>" " THEN BEEP 0.1,15: GO TO 500
3110 GO TO 3100
```

## ЦИФЕРТОН

Эта игра хорошо развивает зрительную и музыкальную память и может использоваться для проверки этих качеств у играющего и его друзей.

На игровом поле расположены несколько прямоугольников разного размера и цвета, причем четыре прямоугольника второго снизу ряда программа в случайном порядке зажигает и гасит подобно лампочкам новогодней гирлянды, сопровождая вспышку каждого из прямоугольников звуковыми сигналами (каждому прямоугольнику соответствует свой тон).

Задача играющего - пользуясь четырьмя управляющими клавишами, воспроизвести заданную компьютером последовательность зажигания



Рис. П. 5. Игра ЦИФЕРТОН.

прямоугольников в нижнем ряду экрана. Имейте в виду, что прямоугольники зажигаются и гаснут сериями, то есть вначале - один, например, красный, и вы должны ответить нажатием клавиши, включающей красный прямоугольник в нижнем ряду. Затем один за другим вспыхивают два прямоугольника - повторите и эту последовательность. Если ошибок не допущено, последует серия из трех вспышек и т. д. Когда вы сумеете правильно повторить последовательность из пяти цветов, игра перейдет на второй уровень и, сигнализируя об этом включит прямоугольник в верхней части игрового поля. Для перехода на третий уровень необходимо запомнить и быстро воспроизвести последовательность из десяти вспышек, а на четвертый - из пятнадцати.

Конечно же, дело не обойдется без ошибок. Когда это произойдет, программа остановится, включит длинный фиолетовый прямоугольник в средней части экрана, а потом игра начнется снова, но уже на том уровне, которого вы успели достигнуть к моменту ошибки.

Тем, у кого есть музыкальный слух, советуем воспользоваться звуковыми сигналами: так как каждому прямоугольнику соответствует свой тон, можно даже закрыть глаза и нажимать клавиши ориентируясь только по звукам.

Напомним, что время игры ограничено, а для его контроля служит постепенно уменьшающийся синий прямоугольник в левой части игрового поля.

После загрузки программы на экране появляется мерцающее меню:

- 1 - НАЧАЛО ИГРЫ
  - 2 - ВЫБОР УРОВНЯ "1"
  - 3 - ВЫБОР КЛАВИШ
  - 4 - ВЫХОД
- Выберите раздел ( 1 - 4 )

Нажав несколько раз клавишу 2, можете установить желаемый уровень игры - ему соответствует число в кавычках. Буква «П» означает ПРАКТИКА - это обычная игра, только несколько медленнее и без уровней. Третий раздел меню позволяет назначить удобные для вас клавиши управления. Стандартный набор высвечивается на экране после загрузки программы и будет установлен, если

нажать предложенные клавиши. Для начала игры достаточно нажать клавишу 1. Чтобы не забыть, какими клавишами пользоваться, под прямоугольниками нижнего ряда горят выбранные вами символы. Для перехода в меню в любой момент времени, нажмите клавишу E.

## Программа 38. ЦИФЕРТОН.

```

10 BORDER 0: PAPER 0: INK 7
20 CLS
30 PRINT AT 10,10;"PLEASE WAIT"
40 DIM k$(4): GO SUB 8300
50 GO SUB 9900
60 REM -- MENU ---
62 CLS
65 LET UR=1
70 PRINT AT 3,9; INK 2;"Ц И"; INK 1;" Ф Е"; INK 6;" Р Т"; INK 4;" О Н"
75 PRINT AT 20,5; INK 6;"ВЫБЕРИТЕ РАЗДЕЛ (1 - 4)"
76 IF UR=0 THEN PRINT AT 10,25; INK 3;"*****"; INK 4;"p"; INK 3;"*****": GO TO 80
77 PRINT AT 10,25; INK 3;"*****"; INK 4;UR; INK 3;"*****"
80 LET c=5: LET c1=7
90 PRINT AT 8,8; INK c;"1 - НАЧАЛО ИГРЫ"
95 PRINT AT 10,8; INK c1;"2 - ВЫБОР УРОВНЯ"
100 PRINT AT 12,8; INK c;"3 - ВЫБОР КЛАВИШ"
105 PRINT AT 14,8; INK c1;"4 - ВЫХОД"
120 IF INKEY$="1" THEN GO SUB 7000: GO TO 600
130 IF INKEY$="2" THEN GO SUB 7000: GO SUB 300
140 IF INKEY$="3" THEN GO SUB 7000: GO TO 400
150 IF INKEY$="4" THEN GO SUB 7000: STOP
160 IF c=5 THEN LET c=7: LET c1=5: GO TO 90
170 IF c=7 THEN LET c=5: LET c1=7: GO TO 90
300 REM ** ВЫБОР УРОВНЯ **
310 LET ur=ur+1
320 IF ur=4 THEN LET ur=0
330 IF ur=0 THEN PRINT AT 10,25; INK 3;"*****"; INK 4;"P"; INK 3;"*****": RETURN
340 PRINT AT 10,25; INK 3;"*****"; INK 4;ur; INK 3;"*****"
350 RETURN
400 REM ** ВЫБОР КЛАВИШ **
410 CLS
420 PRINT AT 3,10; INK 6;"ВЫБОРППКЛАВИШ"
430 PRINT AT 7,10; PAPER 2; INK 5;"КРАСНЫЙпп"; PAPER 0; " - "
435 PRINT AT 9,10; PAPER 1; INK 5;"СИНИЙпппп"; PAPER 0; " - "
440 PRINT AT 11,10; PAPER 6; INK 0;"ЖЕЛТЫЙппп"; PAPER 0; INK 5;" - "
445 PRINT AT 13,10; PAPER 4; INK 0;"ЗЕЛЕНыйпп"; PAPER 0; INK 5;" - "
450 FOR n=1 TO 4
455 PRINT AT 5+n*2,22;k$(n)
460 NEXT n
470 FOR n=1 TO 4
475 LET i$=INKEY$
480 IF i$="" OR I$="E" THEN GO TO 475
485 IF (CODE i$>64 AND CODE i$<91) OR (CODE i$>47 AND CODE i$<58) THEN GO TO 490
487 BEEP .05,-23: GO TO 475
490 IF n>1 THEN IF i$=k$(n-1) THEN GO TO 475
495 LET k$(n)=i$
500 PRINT AT 5+n*2,22; INK 3;i$
510 BEEP .002,50: BEEP .001,40
520 NEXT n
530 FOR n=1 TO 100: NEXT n
540 GO TO 60
600 REM ** НАЧАЛО ИГРЫ **
610 CLS
620 PRINT AT 1,9; INK 5;"Ц И Ф Е Р Т О Н"

```

```
625 INK 7
630 FOR N=31 TO 191 STEP 40
640 PLOT N,137: DRAW 34,0: DRAW 0,-10: DRAW -34,0: DRAW 0,10
650 NEXT N
660 FOR N=23 TO 239 STEP 56
665 PLOT N,96: DRAW 50,0: DRAW 0,-26: DRAW -50,0: DRAW 0,26
667 PLOT N,56: DRAW 50,0: DRAW 0,-18: DRAW -50,0: DRAW 0,18
670 NEXT N
675 PRINT AT 15,3; PAPER 2;"nnnnnn";AT 16,3;"nnnnnn";AT 15,10; PAPER 1;"nnnnnn";AT
16,10;"nnnnnn";AT 15,17; PAPER 6;"nn";AT 16,17;"nnnnnn"; AT 15,24; PAPER
4;"nnnnnn";AT 16,24;"nnn"
680 FOR N=1 TO 4
685 INK 3: LET X=N*7-2: PRINT AT 19,X;K$(N)
690 NEXT N
697 INK 7
700 FOR N=39 TO 207 STEP 56
705 PLOT N,25: DRAW 10,0: DRAW 0,-10: DRAW -10,0: DRAW 0,10
710 NEXT N
720 PLOT 23,113: DRAW 218,0: DRAW 0,-10: DRAW -218,0: DRAW 0,10
722 PLOT 7,7: DRAW 0,130: DRAW 10,0: DRAW 0,-130: DRAW -10,0
723 FOR N=5 TO 20: PRINT AT N,1; PAPER 1;" ": NEXT N
725 PLOT 0,0: DRAW 0,175: DRAW 255,0: DRAW 0,-175: DRAW -255,0
750 REM -----
755 DIM C(50)
760 LET KOL=0: LET PR=0
770 LET SP=0: LET P0=17
795 REM -----
800 FOR N=4 TO 24 STEP 5: PRINT AT 5,N;"nnnn": NEXT N
805 IF UR=0 THEN FOR N=4 TO 24 STEP 5: PRINT AT 5, N;PAPER 4;"nnnn": NEXT N: GO TO
850
810 PRINT AT 5,UR*5-1; PAPER 2;"nnnn"
845 REM -----
850 LET SP=30-UR*4: IF UR=0 THEN LET KOL=0: GO TO 860
853 IF P0=1 THEN GO SUB 2100: GO TO 60
855 LET KOL=UR*5-5: IF UR>1 THEN GO TO 862
860 LET KOL=KOL+1
862 DIM C(50)
865 FOR W=1 TO 100: NEXT W
870 IF KOL=UR*5 THEN LET UR=UR+1: GO SUB 2000: GO TO 800
880 FOR N=1 TO KOL
890 LET NO=INT (RND*4+1)
895 IF N<>1 THEN IF NO=C(N-1) THEN GO TO 890
900 LET C(N)=NO
905 LET M=NO*7-4
907 IF NO=1 THEN LET IN=2
910 IF NO=2 THEN LET IN=1
915 IF NO=3 THEN LET IN=6
920 IF NO=4 THEN LET IN=4
930 PRINT AT 10,M; PAPER IN;"nnnnnn";AT 11,M;"nnnnnn";AT 12,M;"nnnnnn"
932 IF NO=1 THEN LET NOTA=0
933 IF NO=2 THEN LET NOTA=3.86
934 IF NO=3 THEN LET NOTA=7.02
935 IF NO=4 THEN LET NOTA=10.88
940 BEEP .1,NOTA
950 FOR W=1 TO SP: NEXT W
955 PRINT AT 10,M; PAPER 0;"nnnnnn";AT 11,M;"nnnnnn";AT 12,M;"nnnnnn"
960 NEXT N
970 FOR N=1 TO KOL
980 IF INKEY$=K$(1) THEN LET NO=1: LET M=3: LET IN=2: GO TO 1000
985 IF INKEY$=K$(2) THEN LET NO=2: LET M=10: LET IN=1: GO TO 1000
```



```

9970 NEXT N
9975 DATA "B", 0, 124, 64, 124, 66, 66, 124, 0
9976 DATA "G", 0, 126, 64, 64, 64, 64, 64, 0
9977 DATA "D", 0, 28, 36, 36, 36, 36, 126, 66
9978 DATA "J", 0, 73, 73, 62, 73, 73, 73, 0
9979 DATA "I", 0, 66, 70, 74, 82, 98, 66, 0
9980 DATA "L", 0, 30, 34, 34, 34, 34, 98, 0
9981 DATA "P", 0, 126, 66, 66, 66, 66, 66, 0
9982 DATA "O", 0, 66, 66, 36, 24, 16, 96, 0
9983 DATA "C", 0, 68, 68, 68, 68, 68, 126, 2
9984 DATA "H", 0, 66, 66, 66, 62, 2, 2, 0
9985 DATA "N", 0, 65, 73, 73, 73, 73, 127, 0
9986 DATA "N", 0, 65, 73, 73, 73, 73, 127, 1
9987 DATA "E", 0, 60, 66, 30, 2, 66, 60, 0
9988 DATA "U", 0, 76, 82, 114, 82, 82, 76, 0
9989 DATA "A", 0, 62, 66, 66, 62, 34, 66, 0
9990 DATA "S", 24, 66, 70, 74, 82, 98, 66, 0
9991 DATA "R", 0, 64, 64, 124, 66, 66, 124, 0
9992 DATA "T", 0, 192, 64, 124, 66, 66, 124, 0
9993 DATA "F", 0, 62, 73, 73, 73, 62, 8, 0
9994 DATA "Q", 0, 66, 66, 114, 74, 74, 114, 0
9995 RETURN

```

В этой программе широко используются тексты на русском языке. Удобство их применения совершенно очевидно: тут и правила, которые можно вывести на экран перед началом игры, и пояснительные надписи по ее ходу, и тексты в кадрах заставки.

Посмотрим, как это сделано. Часть букв латинского алфавита (и цифра 3) просто использована как русские символы:

A, B, E, 3, K, M, O, P, C, T, X

Остальные сформированы как обычные графические символы в области UDG и соответствуют клавишам:

Q - Ы D - Д E - Э F - Ф R - Ъ G - Г T - Ъ H - Ч U - Ю J - Ж I - И L - Л O - У C - Ц P - П B - Б A - Я N - Ш M - Щ S - Й

Таким образом, чтобы на экране появилась, например, буква «Ы», нужно перейти в графический режим (курсор G) и нажать клавишу Q.

Подпрограмма русского алфавита занимает строки 9900...9995, а обращение к ней происходит в строке 50, то есть непосредственно перед тем, как русские буквы будут использоваться в строках 70, 90, ...

Рекомендуем набирать программу ЦИФЕРТОН, начиная со строк 9900...9995, после чего запустить ее оператором RUN и далее вводить все остальные строки.

## **ЛАБИРИНТ**

Через некоторое время после загрузки этой программы на экране появится лабиринт, стенки которого окрашены в белый цвет, а проходы заполнены зелеными звездочками. Цель игры состоит в том, чтобы за минимальное время собрать эти звездочки, передвигая по проходам маленького человечка.

Клавиши управления: P - вправо; O - влево; Q - вверх; A - вниз.





```
150 INK 2
160 PLOT 0,152: DRAW 255,0
170 INK 6
180 PRINT AT 4,8;"Sankt - Peterburg"
190 PRINT AT 6,14;"1993"
200 PAPER 7: INK 2
210 LET K=11
220 PRINT AT 8,K-1;"oooooooooooo"
230 FOR N=9 TO 16
240 PRINT AT N,K-1;"oooooooooooo"; PAPER 0;" "
250 NEXT N
260 PRINT AT 17,11; PAPER 0;"oooooooooooo"
270 PRINT AT 9,K;"Q.....UP"
280 PRINT AT 11,K;"A.....DOWN"
290 PRINT AT 13,K;"O.....LEFT"
300 PRINT AT 15,K;"P....RIGHT"
310 PAPER 1: INK 4
320 PRINT AT 20,3;"PRESS ANY KEY TO CONTINUE"
330 PAUSE 0
340 REM -- NACHALO IGRY ---
345 LET SC=0
350 CLS : INK 7
355 LET S=0
360 FOR N=1 TO 20
370 PRINT AT N,0;" ": PRINT AT N,31;" "
380 NEXT N
390 PRINT AT 0,0;"oooooooooooooooooooooooooooooooooooooooo"
400 PRINT AT 21,0;"oooooooooooooooooooooooooooooooooooooooo"
410 FOR N=1 TO 20
420 LET H=INT (RND*30+1)
430 LET V=INT (RND*20+1)
450 PRINT AT V,H; INK 2;"#"
460 NEXT N
470 FOR N=1 TO 20
480 LET H=INT (RND*30+1)
490 LET V=INT (RND*20+1)
510 PRINT AT V,H; INK 6;"@"
520 NEXT N
530 DIM X(25): DIM Y(25)
535 FOR N=1 TO 4: LET X(N)=10: LET Y(N)=10+N: NEXT N
540 PRINT AT Y(1),X(1);" ": LET V=-1: LET H=0: LET DL=4:LET DL1=4
545 REM -----
548 IF S=20 THEN GO TO 2000
550 IF INKEY$="O" THEN LET H=-1: LET V=0
560 IF INKEY$="P" THEN LET H=1: LET V=0
570 IF INKEY$="Q" THEN LET V=-1: LET H=0
580 IF INKEY$="A" THEN LET V=1: LET H=0
582 IF Y(DL1)=0 OR X(DL1)=0 THEN GO TO 586
585 PRINT AT Y(DL1),X(DL1);" "
586 IF DL<DL1 THEN LET DL=DL1: GO TO 590
590 FOR N=DL1 TO 2 STEP -1
600 LET X(N)=X(N-1)
610 LET Y(N)=Y(N-1)
620 NEXT N
625 LET X(1)=X(1)+H: LET Y(1)=Y(1)+V
640 FOR N=2 TO DL1
650 PRINT AT Y(N),X(N);"O"
660 NEXT N
665 REM -----
670 IF X(1)<1 OR X(1)>30 OR Y(1)<1 OR Y(1)>20 THEN GO TO 1000
```

```

680 IF SCREEN$ (Y(1),X(1))="#" THEN GO TO 1000
682 IF SCREEN$ (Y(1),X(1))="0" THEN GO TO 1000
690 IF SCREEN$ (Y(1),X(1))="@ " THEN BEEP .0008,56:BEEP .005,40: LET S=S+1: LET
DL1=DL1+1: LET SC=SC+10: PRINT #0;AT 0,10;"SCORE ";SC
695 REM -----
697 PRINT AT Y(1),X(1);"*"
700 GO TO 545
1000 CLS
1002 LET A$=" G A M E П П П О V E R "
1005 FOR N=1 TO 19: PRINT AT 10,6+N;A$(N): BEEP .03,30-(N*3+10): NEXT N
1010 IF INKEY$="" THEN GO TO 1010
1020 GO TO 10
2000 REM -----
2005 RESTORE
2010 FOR N=1 TO 14
2020 READ S,T
2030 BEEP S/1.5,T
2040 NEXT N
2045 GO TO 10
2050 DATA .4,20,.2,17,.2,20,.4,19
2055 DATA .2,16,.2,19,.4,18
2060 DATA .2,15,.2,18,.4,17
2070 DATA .45,14.3,.45,15.3,.45,16.3,.55,12
    
```

### ***СПРУТЫ***

Начало работы программы можно считать традиционным: появляется заставка с названием игры и информацией о том, как ее начать. Нажав клавишу S, вы запустите подпрограмму, создающую игровое пространство - дно моря со сложным рельефом и аквалангистов, укрывшихся в его расщелинах от страшных и чрезвычайно опасных осьминогов. Затем на поверхности моря появляется маленькая подводная лодка. Ее задача - спасти аквалангистов.

Клавиши управления подводной лодкой:

P - вправо; O - влево; Q - вверх; A - вниз.

Чтобы спасти аквалангиста, достаточно поставить подводную лодку на его место. Но действовать надо крайне осторожно: столкновение со дном моря или с осьминогом может погубить подводную лодку, и игра закончится. А чтобы вы были порешительнее и не слишком долго размышляли, куда двигаться и кого первым спасать, в левом верхнем углу экрана помещены часы-таймер, неумолимо отсчитывающие 150 секунд, отпущенные вам на проведение всей операции. Всего 150 секунд - и ни секундой больше!



Рис. П.8. Игра СПРУТЫ.

Из этой программы можно легко сделать несколько других, не менее занимательных игр. Можно заменить осьминогов на подводные лодки противника, а свою субмарину оснастить запасом торпед, можно создать на дне несколько



```

3320
3315 IF SCREEN$ (y+1,x)<>" " THEN GO TO 5000
3320 PRINT AT y,x;" ": LET y=y+1: PRINT AT y,x; INK 5;"A"
3330 RETURN
4000 IF x(z)<30 AND SCREEN$ (y(Z)+1,x(z))=" " THEN PRINT AT y(Z),x(z);" ": LET
x(z)=x(z)+1: IF x(z)>31 THEN LET x(z)=31
4005 PRINT AT y(z),x(z); INK 2;"B"
4010 RETURN
4100 IF x(z)>0 AND SCREEN$ (y(z),x(z)-1)=" " THEN PRINT AT y(z),x(z);" ": LET
x(z)=x(z)-1: PRINT AT y(z),x(z); INK 2;"B"
4110 RETURN
4200 IF y(z)>1 AND SCREEN$ (y(z)-1,x(z))=" " THEN PRINT AT y(z),x(z);" ": LET
y(z)=y(z)-1: PRINT AT y(z),x(z); INK 2;"B"
4210 RETURN
4300 IF y(z)<21 AND SCREEN$ (y(z)+1,x(z))=" " THEN PRINT AT y(z),x(z);" ": LET
y(z)=y(z)+1: PRINT AT y(z),x(z); INK 2;"B"
4310 RETURN
5000 BEEP 1,-40: PRINT AT y,x; INK 1;"*"
5020 LET l$="ПГGAME OVERПП"
5030 PAUSE 30: FOR f=1 TO LEN l$: PRINT AT 19,10; INK 2;BRIGHT 1;l$(1 TO f): BEEP
.1,-28: NEXT f
5040 PAUSE 200: RUN
5500 PRINT AT y+1,x;" ": LET l=l+1
5505 PRINT AT 0,26+l; INK 6;"C": BEEP .0007,56: BEEP .0005,40
5520 RETURN
6000 PAPER 0: INK 7: BORDER 0: CLS
6010 FOR f=0 TO 11: PRINT AT 10,f; INK 5;"A": PAUSE 5:PRINT AT 10,f;" ": NEXT f
6020 FOR f=12 TO 16: PRINT AT 10,f; INK 5;"A": PRINT AT 11,f; INK 6;"C": PAUSE 5:
PRINT AT 10,f;" ": NEXT f
6030 FOR f=17 TO 31: PRINT AT 10,f; INK 5;"A": PAUSE 5:PRINT AT 10,f;" ": NEXT f
6040 LET l$="BRAVO!"
6050 FOR f=1 TO LEN l$: PRINT AT 11,12;l$(1 TO f): BEEP.1,f+10: NEXT f
6060 PAUSE 30: FOR f=1 TO 30: BEEP .1,RND*20+20: BORDER RND*7: NEXT f
6070 PAUSE 100
6090 RUN

```

## ПРИЛУНЕНИЕ

Очень занимательная игра, хотя в ней нет ни стрельбы, ни лабиринтов, а только колонки цифр, за которыми, тем не менее, скрывается настоящая трагедия. Похожая ситуация возникла во время полета на Луну американского космического корабля Аполлон, когда на нем неожиданно



Рис. П.9. Картинка-заставка к игре ПРИЛУНЕНИЕ.

произошла авария, и astronautам пришлось брать управление на себя. Для благополучного возвращения на Землю пришлось вычислить единственно возможную траекторию полета и в соответствии с ней ориентировать корабль в пространстве, включая двигатели на строго заданное время (в нашей игре это

придется делать вам, определяя дозы топлива для двигателя).

Игра начинается с небольшого предисловия, после которого на экране появляется подробная информация о всех возможных параметрах прилунения. Эти параметры в процессе игры будут выводиться на экран в виде столбиков цифр, вы должны анализировать их и задавать программе одну-единственную величину - дозу топлива. Она может принимать значения от 8 до 200 условных единиц или 0. Последняя цифра означает, что ваш корабль свободно падает не расходуя топлива, при этом в соответствии с законами тяготения скорость падения непрерывно возрастает.

Задача игрока состоит в том, чтобы к моменту достижения поверхности Луны скорость корабля оказалась максимально близкой к нулю. Сколько на все маневры понадобится времени и сколько останется горючего, особого значения не имеет, но за последним параметром надо постоянно следить. Программа сделана так, чтобы горючего только-только хватило на идеальную посадку (высота = 0, скорость = 0).

Когда вы достигнете поверхности, появится информация о характере прилунения. Держим пари, что с первого раза никто из вас не посадит корабль не только ПРЕКРАСНО, но даже НЕУДАЧНО. Почти уверены, что и со второй попытки вам это не удастся, а вот когда удастся - зависит не столько от ваших математических способностей, сколько от сообразительности.

Программа 42. ПРИЛУНЕНИЕ.

```

10 BORDER 0: PAPER 0: INK 7: CLS
20 PRINT AT 10,10;"PLEASE WAIT"
30 GO SUB 9900
41 GO SUB 1700: CLS
42 INK 3
45 PRINT AT 10,8; FLASH 1;"ПОСАДИТЕ ЛУННИК"
47 PAUSE 250
50 GO SUB 1000
60 CLS
70 GO SUB 1500
90 REM ** НАЧАЛЬНЫЕ УСЛОВИЯ **
100 LET L=0
101 LET A=120
102 LET V=1
103 LET M=33000
104 LET N=16500
105 LET G=0.001
106 LET Z=1.8
107 LET K=0
108 LET YP=3
110 CLS
111 PRINT AT 0,11; INVERSE 1; INK 3;"ПОСАДКА" 120 INK 6
125 PRINT AT 2,0;"ДОЗА ВРЕМЯ ВЫСОТА СКОР. ГОРЮЧЕЕ" 130 PRINT AT 4,0;
210 IF A<=1 THEN GO TO 213
211 LET H=INT A
212 GO TO 214
213 LET H=1000*(A-INT (A)): LET H=INT H
214 LET U=INT ((M-N)/10)
215 LET E=INT (V*3600)
220 IF YP=20 THEN LET YP=2: FOR C=3 TO 21: PRINT AT C,0; TAB 31: NEXT C: PRINT AT
4,0;
221 LET YP=YP+1: INK 5: PRINT K;TAB 6;L;TAB 13;H;TAB 19;E;
TAB 27;U
230 INPUT "ВВЕДИТЕ ДОЗУ:";K: LET T=10
240 IF K=0 THEN GO TO 310 250 IF K>=8 AND K<=200 THEN GO TO 310
260 PRINT #0;AT 0,0; INK 2;"ТАКОЕ ""K"" НЕВОЗМОЖНО,ВВЕДИТЕ НОВОЕ": PAUSE 200

```

```
270 GO TO 230
310 IF ((M-N)-0.001)<0 THEN GO TO 410
315 IF (T-.001)<0 THEN GO TO 210
320 LET S=T
340 IF ((N+S*K)-M)<=0 THEN GO TO 350
345 LET S=(M-N)/K
350 GO SUB 900
355 IF I<=0 THEN GO TO 710
360 IF V<=0 THEN GO TO 380
365 IF J<0 THEN GO TO 810
380 GO SUB 600
385 GO TO 310
410 CLS
415 INK 5: PRINT "ГОРЮЧЕЕ ВЫШЛО НА ";L;" СЕКУНДЕ"
440 LET S=(-V+SQR (V*V+2*A*G))/G
445 LET V=V+G*S
450 LET L=INT (L+S)
510 PRINT "ЛУНЫ ДОСТИГЛИ НА ";L;" СЕК"
515 LET W=V*3600: LET X=INT W
520 PRINT "СО СКОРОСТЬЮ ";X;" КМ/Ч"
523 IF (M-N)<=0 THEN GO TO 530
525 PRINT "ГОРЮЧЕГО ОСТАЛОСЬ ";INT ((M-N)/10);" ЕД"
530 PRINT : INK 6
535 IF (-W+1)<=0 THEN GO TO 550
540 PRINT "ПРЕКРАСНОЕ ПРИЛУНЕНИЕ": PRINT
545 GO TO 590
550 IF (-W+10)<=0 THEN GO TO 560
553 PRINT "УДАЧНОЕ ПРИЛУНЕНИЕ"
555 GO TO 590
560 IF (-W+25)<=0 THEN GO TO 570
563 PRINT "СЕЛИ НЕУДАЧНО": PRINT
565 GO TO 590
570 IF (-W+60)<=0 THEN GO TO 580
573 PRINT "ЛУННИК ПОВРЕЖДЕН": PRINT
575 GO TO 590
580 PRINT "ИЗВИНИТЕ, НО ЛУННИК": PRINT
582 PRINT "ИСЧЕЗ, А ПОЯВИЛСЯ": PRINT
584 PRINT "НОВЫЙ ЛУННЫЙ КРАТЕР": PRINT
585 PRINT "ДИАМЕТРОМ ";INT (W*0.0847);" М"
590 PRINT : INK 1;"ПОПРОБУЕТЕ ЕЩЕ? (Д/Н)"
592 IF INKEY$="D" OR INKEY$="d" THEN LET AWT=0: GO TO 100
594 IF INKEY$="N" OR INKEY$="n" THEN GO TO 596
595 GO TO 592
596 CLS : LET C=0
597 PRINT AT 10,8; INK C;"ДО НОВЫХ ВСТРЕЧ! "
598 LET C=C-1: IF C<0 THEN LET C=7
599 PAUSE 5: GO TO 597
600 REM -----
602 LET L=INT (L+S)
605 LET T=T-S
610 LET M=M-S*K
615 LET A=I
620 LET V=J
630 RETURN
710 IF (S-.005)<0 THEN CLS : GO TO 510
715 LET S=2*A/(V+SQR (V*V+2*A*(G-Z*K/M)))
730 GO SUB 900
735 GO SUB 600
740 GO TO 710
810 LET W=(1-M*G/(Z*K))/2
```

```
815 LET S=M*V/(Z*K*(W+SQR(W*M+V/Z)))+.05
820 GO SUB 900
830 IF I<0 THEN GO TO 710
835 GO SUB 600
840 IF (-J)<=0 THEN GO TO 310
845 IF V<=0 THEN GO TO 310
850 GO TO 810
900 LET Q=S*K/M
905 LET J=V+G*S+Z*LN (1-Q)
940 LET I=A-G*S*S/2-V*S
945 LET C=Q/2+Q^2/6+Q^3/12+Q^4/20+Q^5/30
950 LET I=I+Z*S*C
960 RETURN
1000 REM ** ИНФОРМАЦИЯ **
1002 CLS
1005 PRINT AT 0,8; INK 3;"** "; INK 6; "ИНФОРМАЦИЯ";INK 3;" **"
1010 INK 5
1020 PRINT "ДОЗУ ГОРЮЧЕГО ""К"" МОЖНО"
1030 PRINT "УСТАНОВИТЬ РАВНОЙ НУЛЮ ЛИБО"
1040 PRINT "ЛЮБОЙ ВЕЛИЧИНЕ ОТ 8 ДО 200"
1050 PRINT "ВСЕГО ГОРЮЧЕГО 1650 ЕД."
1060 INK 4 1070 INVERSE 1
1090 PRINT AT 11,4;"ПППАРАМЕТРЫ ПРИЛУНЕНИЯПП"
1100 INVERSE 0
1110 INK 5
1120 PRINT "-ВРЕМЯ (СЕК)"
1130 PRINT "-ВЫСОТА (КМ,М)"
1140 PRINT "-СКОРОСТЬ (КМ/Н)"
1150 PRINT "-ГОРЮЧЕЕ (ЕД)"
1160 PRINT "-ДОЗА (ЕД)"
1162 PRINT #0; INK 2;ТАВ 6;"НАЖМИТЕ ЛЮБУЮ КЛАВИШУ"
1165 IF INKEY$="" THEN GO TO 1165
1170 RETURN
1500 INVERSE 1: INK 4
1505 PRINT AT 0,7;"ППВИДЫ ПРИЛУНЕНИЯПП"
1510 INVERSE 0 1520 INK 5
1530 PRINT "1. ПРЕКРАСНОЕ ПРИЛУНЕНИЕ"
1540 PRINT "2. УДАЧНОЕ ПРИЛУНЕНИЕ"
1550 PRINT "3. СЕЛИ НЕУДАЧНО"
1560 PRINT "4. ЛУННИК ПОВРЕЖДЕН"
1570 PRINT "5. ЛУННИК ИСЧЕЗ," " "ППППОЯВИЛСЯ НОВЫЙ КРАТЕР"
1580 INK 6
1590 PRINT AT 21,6;"НАЖМИТЕ ЛЮБУЮ КЛАВИШУ"
1600 IF INKEY$="" THEN GO TO 1600
1610 RETURN
1700 REM ** ПРИЧИНЫ ПОСАДКИ **
1710 CLS : INK 6
1720 PRINT "2093 GOD.": PRINT
1725 INK 5
1730 PRINT "ВАМ БЫЛО ПОРУЧЕНО ДОСТАВИТЬ ГРУЗ С ЗЕМЛИ НА МАРС. НО ЧЕРЕЗ НЕКОТОРОЕ
ВРЕМЯ ПОСЛЕ СТАРТА С ЗЕМЛИ У ВАС ПРОИЗОШЛА ПОЛОМКА И ВЗРЫВ АТОМНОГО РЕАКТОРА."
1740 PRINT "В МОМЕНТ ВЗРЫВА КОРАБЛЬ НАХОДИЛСЯ НЕДАЛЕКО ОТ ЛУНЫ."
1750 PRINT "ВЫ СРОЧНО ПОКИДАЕТЕ РАДИОАКТИВНЫЙ КОРАБЛЬ В СПАСАТЕЛЬНОЙ КАПСУЛЕ. ПРИ
ПОДЛЕТЕ К ЛУНЕ ВЫ ОБНАРУЖИВАЕТЕ, ЧТО ВЗРЫВОМ БЫЛ ПОВРЕЖДЕН КОМПЬЮТЕР, УПРАВЛЯЮЩИЙ
ПОСАДКОЙ. ЭТО ЗНАЧИТ, ЧТО ВАМ ПРИДЕТСЯ САЖАТЬ КАПСУЛУ ВРУЧНУЮ."
1760 PRINT AT 20,6; INK 3;"НАЖМИТЕ ЛЮБУЮ КЛАВИШУ"
1770 IF INKEY$="" THEN GO TO 1770
1780 RETURN
9899 STOP
9900 REM -- RUS.ALF.-
```

```

9910 FOR N=1 TO 20
9920 READ A$
9930 FOR M=0 TO 7
9940 READ S
9950 POKE USR A$+M,S
9960 NEXT M
9970 NEXT N
9975 DATA "B",0,124,64,124,66,66,124,0
9976 DATA "G",0,126,64,64,64,64,0
9977 DATA "D",0,28,36,36,36,36,126,66
9978 DATA "J",0,73,73,62,73,73,73,0
9979 DATA "I",0,66,70,74,82,98,66,0
9980 DATA "L",0,30,34,34,34,34,98,0
9981 DATA "P",0,126,66,66,66,66,66,0
9982 DATA "O",0,66,66,36,24,16,96,0
9983 DATA "C",0,68,68,68,68,68,126,2
9984 DATA "H",0,66,66,66,62,2,2,0
9985 DATA "N",0,65,73,73,73,73,127,0
9986 DATA "M",0,65,73,73,73,73,127,1
9987 DATA "E",0,60,66,30,2,66,60,0
9988 DATA "U",0,76,82,114,82,82,76,0
9989 DATA "A",0,62,66,66,62,34,66,0
9990 DATA "S",24,66,70,74,82,98,66,0
9991 DATA "R",0,64,64,124,66,66,124,0
9992 DATA "T",0,192,64,124,66,66,124,0
9993 DATA "F",0,62,73,73,73,62,8,0
9994 DATA "Q",0,66,66,114,74,74,114,0
9995 RETURN
    
```

### **КОРОЛЕВСТВО**

Эта текстовая игра сделана целиком на русском языке, поэтому подробных пояснений к ней не требуется; Достаточно читать выводимые на экран вопросы и, опираясь на логику и свой жизненный опыт, давать ответы.

Вы являетесь монархом маленькой аграрной страны где-то в центре Европы. Земли королевства плодородны, крестьяне счастливы, да вот беда: соседи, как всегда, завистливы. Но, несмотря на это, вы должны обеспечить процветание своей страны и даже попытаться расширить свои владения. Эта задача не так легка, как может показаться на первый взгляд - ведь против вас действует изощренная компьютерная программа. Она насылет на королевство всевозможные напасти: от холеры и чумы до нашествия саранчи и крыс. Да и враги не теряют зря времени, пытаясь отхватить то один, то другой кусочек ваших владений.



Рис. П. 10. Заставка к игре КОРОЛЕВСТВО.

В вашем распоряжении три показателя, которыми можно распоряжаться в зависимости от ситуации: население, земля и зерно. Например, чтобы защититься от врагов, можно воспользоваться услугами наемников, но им надо чем-то платить.

Вот и решайте: то ли отдать часть зерна и заставить голодать своих подданных, то ли уплатить земель, что в конце концов тоже грозит недобором зерна и голодом. Вот и приходится

королю постоянно лавировать между желаемым и возможным, не забывая о том, что при любых обстоятельствах надо сохранить корону, а иначе...

Программа 43. КОРОЛЕВСТВО.

```

10 LET B$=" БУШЕЛЕЙ": PAPER 0: INK 7
20 POKE 23658,8
30 RANDOMIZE
50 BORDER 0
60 OVER 0: FLASH 0: BRIGHT 0
70 CLS
80 PRINT AT 10,10;"Please WAIT"
90 GO SUB 9900
100 CLS : FOR N=1 TO 100: NEXT N
102 LET H$="К О Р О Л Е В С Т В О"
103 FOR N=6 TO 26 STEP 2
104 LET A=INT (RND*7+1): PAPER A: FOR M=14 TO 16: PRINT AT M,N;" ": NEXT M
105 PRINT AT 18,N; INK A; PAPER 0;H$(N-5)
106 NEXT N
108 PRINT #0;"пппппИНСТРУКЦИЯ НУЖНА? (Д/Н)"
109 LET C=1
110 BRIGHT 1: LET C=C+1: IF C=4 THEN LET C=6
112 IF C>7 THEN LET C=2
115 INK C
120 PLOT 110,110: DRAW 40,0: DRAW 30,60: DRAW -40,-30: DRAW -10,30: DRAW -10,-30:
DRAW -40,30: DRAW 30,-60
122 IF INKEY$="D" THEN BRIGHT 0: PAPER 0: GO SUB 8000
123 IF INKEY$="N" THEN BRIGHT 0: GO TO 150
125 LET A=INT (RND*7+1)
127 LET X=INT (RND*11+1)
129 BRIGHT 0
130 FOR N=14 TO 16: PAPER A: PRINT AT N,X*2+4;" ": NEXT N
132 PRINT AT 18,X*2+4; PAPER 0; INK A;H$(X*2-1)
140 GO TO 110
150 DIM Z(4): DIM E(10): DIM L(8)
151 PAPER 0
152 LET Z(1)=1500
153 LET Z(4)=1500
155 LET L(1)=100
156 LET L(8)=100
158 LET EC1) =5000
159 LET E(10)=5000
160 LET GOD=0
161 LET TE=0
162 LET UR=0 164 LET SAR=0
165 LET SAR1=0
167 LET MOB=0
168 LET ARM=0
169 LET NA=0
200 REM -----
210 LET GOD=GOD+1
215 INK 4: CLS
220 PRINT BRIGHT 1; INK 2;"***** "; INK 7;"GOD ";GOD;INK 2;" *****"
230 PRINT INK 5;"НАСЕЛЕНИЕ:"; INK 4;L(8);" ЧЕЛОВЕК"
240 IF GOD=1 THEN GO TO 275
245 PRINT "РОДИЛОСЬ:";L(2)
250 PRINT "УМЕРЛО ЕСТЕСТВЕННОЙ СМЕРТЬЮ:";L(3)
255 PRINT "УМЕРЛО ОТ ГОЛОДА:";L(4)

```



```
585 GO SUB 7100
587 BEEP .1.-25: PAUSE 200: GO TO 555
590 LET E(5)=-R*K0
591 LET E(1)=E(1)+E(5)
592 LET Z(2)=K0
593 LET Z(1)=Z(1)+Z(2)
595 GO TO 850
600 LET R=R-1
610 PRINT "СКОЛЬКО АКРОВ ЖЕЛАЕТЕ ПРОДАТЬ""ПО ";R;" БУШЕЛЬ/АКР"
620 INPUT "СКОЛЬКО АКРОВ? ";K0
625 IF K0<0 THEN BEEP .1.-25: PAUSE 200: CLS : GO TO 610
630 IF K0=0 THEN GO TO 850
635 IF K0<=Z(1) THEN GO TO 670
640 GO SUB 7150
650 BEEP .1.-25: PAUSE 200: CLS : GO TO 610
670 IF K0<=Z(1)/10 THEN GO TO 800
675 LET R=R-1
680 INK 1: PRINT "ПРОДАТЬ ТАК МНОГО ВЫ МОЖЕТЕ ""ТОЛЬКО ПО ";R;" БУШЕЛЬ/АКР" 685 INK
5
690 PRINT "БУДЕТЕ ПРОДАВАТЬ:"""А-ПО ЭТОЙ ЦЕНЕ, ""В-ДРУГОЕ КОЛИЧЕСТВО, ""С-НЕ БУДЕТЕ
ПРОДАВАТЬ. ""НАЖМИТЕ А, В ИЛИ С"
700 IF INKEY$="А" THEN GO TO 800
710 IF INKEY$="В" THEN CLS : LET R=R+1: GO TO 610
720 IF INKEY$="С" THEN GO TO 850
730 GO TO 700
800 LET E(5)=K0*R
810 LET Z(2)=-K0
820 LET E(1)=E(1)+E(5)
830 LET Z(1)=Z(1)+Z(2)
850 INK 6 860 CLS
870 PRINT "СКОЛЬКО АКРОВ ВЫ ЖЕЛАЕТЕ ЗАСЕЯТЬ?"
880 INPUT "СКОЛЬКО АКРОВ? ";K0: LET K0=INT K0
890 IF K0<0 THEN GO TO 860
900 IF K0<=E(1) THEN GO TO 950
910 GO SUB 7100 915 GO SUB 7000
920 GO TO 860
950 IF K0<=Z(1) THEN GO TO 1000
960 GO SUB 7150
965 PAUSE 300
970 GO TO 860
1000 IF K0<L(1)*10 THEN GO TO 1050
1010 GO SUB 7200 1015 PAUSE 300
1020 GO TO 860
1050 LET E(4)=-K0
1060 LET E(1)=E(1)+E(4)
1070 CLS 1080 PRINT "СКОЛЬКО ЗЕРНА ВЫ ДУМАЕТЕ""УПОТРЕБИТЬ В ПИЩУ?"
1090 INPUT "СКОЛЬКО БУШЕЛЕЙ? ";K0: LET K0=INT K0
1100 IF K0<0 THEN CLS : PRINT "НЕ ВАЛЯЙ ДУРАКА!!!": PAUSE 200: GO TO 1070
1110 IF K0<=E(1) THEN GO TO 1150
1120 GO SUB 7100
1130 PAUSE 300
1140 GO TO 1070
1150 LET L(4)=-L(1)-INT (K0/40))
1151 IF L(4)>=0 THEN LET TE=TE-1: LET L(4)=0: GO TO 1153
1152 LET TE=TE+INT (L(4)/2)
1153 LET E(3)=-K0
1154 LET L(1)=L(1)+L(4)
1155 LET E(1)=E(1)+E(3)
1160 REM -----
1170 LET UR=INT (RND*3+5)
```

```
1200 REM --НАСЕЛЕНИЕ--
1210 LET L(2)=INT ((9.0E-02+5.0E-02*RND)*L(1))+1
1220 LET L(3)=-INT ((4.0E-02+3.0E-02*RND)*L(1 ))
1230 LET L(1)=L(1)+L(2)+L(3)
1300 REM -----
1305 CLS
1310 IF (RND>.05 AND SAR=0) OR GOD<5 THEN GO TO 1400
1320 PRINT "СЕМИЛЕТНЕЕ НАШЕСТВИЕ САРАНЧИ!"
1330 IF SAR=0 THEN LET SAR1=7: LET SAR=1
1340 IF SAR1>0 THEN PRINT "GOD ";8-SAR1: LET UR=INT (RND*3+ 2): GO SUB 7000: LET
SAR1=SAR1-1: GO TO 1400
1350 LET SAR=0
1400 REM -----
1410 LET E(2)=INT (UR*ABS E(4))
1420 LET E(1)=E(1)+E(2)
1500 REM -- КРЫСЫ -----
1501 REM
1502 IF RND>.25 THEN GO TO 1532
1504 LET E(7)=-INT (E(1)/10)
1506 LET E(1)=E(1)+E(7)
1507 PRINT "КРЫСЫ НАВОДНИЛИ ВАШИ АМБАРЫ": GO SUB 7000
1530 REM -- ЧУМА -----
1531 REM
1532 IF RND>.01 THEN GO TO 1550
1533 PRINT "ЧЕРНАЯ ЧУМА ПОРАЗИЛА КОРОЛЕВСТВО": GO SUB 7000
1534 LET L(6)=-INT (L(1)/2)
1540 REM -- ХОЛЕРА -----
1541 REM
1550 IF RND>.2 THEN GO TO 1580
1560 PRINT "ЭПИДЕМИЯ ХОЛЕРЫ ПОРАЗИЛА""КОРОЛЕВСТВО.":GO SUB 7000
1570 LET L(6)=L(6)-INT (L(1)/20)
1580 LET L(1)=L(1)+L(6)
1590 REM -- ВОЙНА -----
1591 REM
1600 IF RND>.15 OR GOD<3 THEN GO TO 3000
1620 CLS
1630 PRINT "СОСЕДНЕЕ КОРОЛЕВСТВО УГРОЖАЕТППPP""ВОЙНОЙ."; INK 5;"ВЫ ОБЪЯВИТЕ
МОБИЛИЗАЦИЮ? (Д/Н)"
1640 IF INKEY$="D" THEN GO TO 1670
1650 IF INKEY$="N" THEN LET MOB=0: GO TO 1750
1660 GO TO 1640
1670 REM -- МОБИЛИЗАЦИЯ ----
1680 CLS
1685 PRINT "СКОЛЬКО НАРОДУ ВЫ МОБИЛИЗУЕТЕ?"
1690 INPUT AT 20,0;AT 0,0;"СКОЛЬКО ЧЕЛОВЕК? ";KO: LET KO=INT KO
1695 IF KO>(L(1)-(L(1)/3)) THEN GO SUB 7250: PAUSE 200: GO TO 1680
1700 IF KO<=0 THEN LET MOB=0: GO TO 1750
1710 LET NOV=KO
1740 REM -- НАЕМНИКИ -----
1750 CLS
1760 PRINT "НАЕМНИКИ ТРЕБУЮТ ПО 80 БУШЕЛЕЙППPP""ЗЕРНА В ГОД. СКОЛЬКО
НАЕМНИКОВППП""ВЫ ПРИВЛЕЧЕТЕ ?"
1770 INPUT AT 17,0;AT 0,0;"СКОЛЬКО НАЕМНИКОВ? ";KO: LET KO= INT KO
1780 IF KO<=0 THEN LET KO=0: GO TO 1800
1790 IF KO*80>E(1) THEN GO SUB 7100: PAUSE 200: GO TO 1750
1792 REM -- ИТОГИ -----
1793 REM
1800 LET ARM=MOB+KO
1802 LET NA=KO*80
1810 PRINT "МОБИЛИЗОВАНО:";MOB
```

```
1820 PRINT "НАНЯТО:";KO
1830 PRINT "ИТОГО:"; INK 2;ARM; INK 4;" СОЛДАТ"
1910 IF ARM=0 THEN LET W=.2: GO TO 1920
1915 LET W=.1
1920 IF RND>W THEN GO TO 2000
1922 REM -- ПЕРЕГОВОРЫ -----
1923 REM
1930 GO SUB 7000: CLS
1940 PRINT "НАЧАЛИСЬ МИРНЫЕ ПЕРЕГОВОРЫ:"
1950 PRINT "ВОЙНЫ НЕ БУДЕТ, ЕСЛИ ВЫ ОТДАДИТЕППП" INT (Z(1)/3) ;" АКРОВ ЗЕМЛИ И "; INT
(E(1)/3);B$;" ЗЕРНА."
1955 PRINT "ВЫ СОГЛАСНЫ С ЭТИМИ УСЛОВИЯМИ?ППП" ""(Д/Н)"
1960 IF INKEY$="D" THEN GO TO 1975
1962 IF INKEY$="N" THEN CLS : GO TO 2020
1970 GO TO 1960
1975 LET Z(3)=-INT (Z(1)/3)
1980 LET E<8)=-INT (E(1)/3)
1985 LET E(1)=E(1)+E(8)
1990 LET Z(1)=Z(1)+Z(3)
1995 GO TO 2300
2000 REM -- СРАВНЕНИЕ СИЛ --
2001 REM
2010 GO SUB 7000: CLS
2020 PRINT "ВАШИ ВОЙСКА:";ARM;" ЧЕЛОВЕК"
2030 LET ARM1=INT (RND*L(1)+E(1)/120)
2035 PRINT "ВОЙСКА ПРОТИВНИКА:";ARM1;" ЧЕЛОВЕК"
2040 IF ARM<ARM1 THEN GO TO 2200
2045 REM -- ВОЙНА ВЫИГРАНА --
2046 REM
2050 PRINT "ВОЙНА ВЫИГРАНА"
2060 LET L(5)=-INT (MOV/4.5): LET TE=TE-2
2070 LET Z(3)=INT (Z(1)/1.5)
2080 LET E(8)=INT (E(10)/2+NA/2)
2090 LET L(1)=L(1)+L(5)
2100 LET Z(1)=Z(1)+Z(3)
2110 LET E(1)=E(1)+E(8)
2120 GO TO 2300
2200 REM -- ВОЙНА ПРОИГРАНА
2210 PRINT "ВОЙНА ПРОИГРАНА!"
2220 LET L(5)=-INT (L(1)/2): LET TE=TE+!NT (L(8)/2)
2230 LET Z(3)=-INT (Z(1)/2)
2240 LET E(8)=-INT (E(1)/2)
2250 LET L(1)=L(1)+L(5)
2260 LET Z(1)=Z(1)+Z(3)
2270 LET E(1)=E(1)+E(8)
2300 REM -- НАЕМНИКИ -----
2310 IF NA<=E(1) THEN GO SUB 7000: GO TO 2400
2320 GO SUB 7000: CLS
2330 PRINT "У ВАС НЕ ХВАТИТ ЗЕРНА ЗАПЛАТИТЬПП" ""НАЕМНИКАМ!!!"
2340 LET L(7)=-INT (.75*L(1))
2350 LET E(9)=-INT (.5*E(1))
2360 LET L(1)=L(1)+L(7)
2370 LET E(1)=E(1)+E(9)
2380 GO SUB 7000: GO TO 2420
2390 REM -----
2400 LET E(6)=-NA
2410 LET E(1)=E(1)+E(6)
2420 GO TO 200
3000 REM -----
5000 GO TO 200
```



## Приложение 2. Основные операторы, функции и команды Spectrum-Бейсика

<b>ABS m</b>	Стр. 247. Абсолютное значение числа $m^1$ . PRINT ABS -12 <i>12</i>
<b>ACS p</b>	Функция арккосинус; p задается в пределах от -1 до 1, результат печатается в радианах. PRINT ACS 0.5 <i>1.04719761</i>
<b>AND</b>	Стр. 54, 57. Логический оператор. LET x=7:IF x>5 AND x<10 THEN PRINT "O'kay" <i>O'kay</i>
<b>ASN p</b>	Функция арксинус, p задается в пределах от -1 до 1, результат печатается в радианах. PRINT ASN 0.7 <i>0.7753975</i>
<b>AT y,x</b>	Стр. 15. Оператор задает знакоместо для вывода данных в PRINT и INPUT. Параметр y определяет номер строки (0...21), а параметр x - номер столбца (0...31).
<b>ATN p</b>	Функция арктангенс, результат печатается в радианах. PRINT ATN 20.742 <i>1.5226223</i>
<b>ATTR (y,x)</b>	Функция возвращает число от 0 до 255, соответствующее атрибутам знакоместа. Позиция знакоместа задается целочисленными величинами: (0...23) и (0...31). 10 FLASH 1:PAPER 3:1NK 7:BRIGHT 0 20 PRINT AT 10,12;" " 30 PRINT ATTR (10,12) <i>159</i>
<b>BEEP t,h</b>	Стр. 30. Оператор генерирует звук длительностью t (в секундах) и высотой h (в полутонах). Число h = 0 соответствует ноте ДО первой октавы. BEEP 0.05,0.3
<b>BIN S</b>	Стр. 46. Команда переводит двоичное число s в десятичное (нули в левой части числа можно опускать). PRINT BIN 10011010 <i>154</i>
<b>BORDER c</b>	Стр. 14. Оператор задает цвет поля вокруг рабочей области экрана и цвет фона служебного окна. Параметр c принимает

<sup>1</sup> Во всех примерах результаты, выводимые на экран, будем выделять курсивом.

целочисленные значения в пределах 0...7.

```
BORDER 1:REM синий цвет бордюра
```

**BRIGHT a** Стр. 55. Оператор устанавливает яркость выводимого символа. Число *s* принимает одно из целочисленных значений: 0 - для нормальной яркости, 1 - для повышенной яркости, 8 - сохраняет существующую яркость.

```
PRINT "RRR "; BRIGHT 1;"RRR"; BRIGHT 0;"RRR"
```

**CAT** Ключевое слово в Spectrum- Basic не задействовано.

**CHR\$ k** Стр. 54. Функция возвращает символ по указанному, в аргументе коду *k*. Код принимает значения 0...255.

```
PRINT CHR$ 35
```

```
#
```

**CIRCLE x,y,r** Стр. 163. Оператор строит на экране окружность радиуса *r*, центр которой имеет горизонтальную координату *x* (0...255), а вертикальную - *y* (0...175).

```
CIRCLE INK 1;100, 50,42
```

**CLEAR n** Стр. 60. Оператор уничтожает все переменные и очищает занимаемую ими память. Выполняет RESTORE и CLS, устанавливает PLOT-позицию в нижнюю левую точку экрана. Изменяет системную переменную RAMTOP на *n* и задает новый GO SUB стек.

```
CLEAR 40000
```

**CLOSE #a** Оператор используется при работе с потоками и каналами

**CLS** Стр. 15. Оператор очищает основной экран и окрашивает его в текущий постоянный цвет фона.

**CODE A\$** Стр. 43. Функция возвращает код первого символа символьного значения *A\$* в соответствии с таблицей символов ZX Spectrum [ ].

```
PRINT CODE "Symbol"
```

```
83
```

**CONTINUE** Команда продолжает выполнение программы с оператора, на котором она была прервана.

**COPY** Оператор распечатывает копию 22 строк экрана на принтере.

**COS p** Функция вычисляет косинус угла *p*, заданного в радианах.

```
PRINT COS 1.4
```

```
0.169967140
```

**DATA a,b,c,g\$,k\$** Стр. 21. Оператор задания списка данных - произвольного набора числовых (*a*, *b*, *c*) и символьных (*g\$*, *k\$*) значений. Оператор DATA можно помещать в любом месте программы.

```
DATA 56, 0,128,37, 11, 0,"Sokoban", 68
```

- DEF FN**  
**R(x,y)=E** Оператор определяет пользовательскую функцию (дополнительно к «встроенным» в Бейсик). Здесь R - имя функции, x, y - параметры функции, E - числовое выражение, задающее саму функцию. Вызов пользовательской функции осуществляется ключевым словом FN.  
 100 DEF FN L(x,y)=xt2+4.5\*x-0.8\*y+9.6  
 300 PRINT FN 1(3.69,0.23)  
 41.03214
- DIM R (m)** Стр. 21. Оператор задаст одномерный числовой массив на m элементов,  
 10 DIM V(128)
- DIM R(m,n)** Стр. 22. Оператор задает двухмерный числовой массив, рассчитанный на m строк по n чисел в каждой.  
 DIM Z(20,5)
- DIM R\$(m,n)** Стр. 22. Оператор задает двухмерный символьный массив, рассчитанный на m слов по n символов в каждом.  
 DIM K\$(15,8)
- DRAW x,y**  
**[,r]** Стр. 23. Оператор строит на экране отрезок прямой линии или дугу окружности. Относительное расстояние (в пикселях) от начала отрезка до его конца измеряется величиной x по горизонтали и y по вертикали.  
 DRAW INK 5; 100, 20, 8
- ERASE** Ключевое слово в Spectrum- Basic не задействовано.
- EXP x** Экспоненциальная функция, возводит число  $e=2.7182818$  в степень x.  
 PRINT EXP 1  
 2.7182818
- FLASH p** Стр. S9. Оператор устанавливает режим мерцания при отображении символьной и графической информации. Число p принимает одно из целочисленных значений: 0 - постоянное свечение, 1 - включает режим мерцания, 8 - сохраняет предыдущее состояние.  
 PRINT FLASH 1;"ZX Spectrum"
- FN** Возвращает значение пользовательской функции, заданной оператором DEF FN.
- FOR n=a TO b** Стр. 19. Оператор цикла, позволяющий выполнять [STEP s] часть программы, заключенную между операторами FOR и NEXT, заданное число раз. Управляющая переменная n принимает значения от n=a до n=b с шагом s. Если шаг не задан, то по умолчанию он принимается равным 1.  
 200 FOR n=1 TO 10  
 210 PRINT "\*";  
 220 NEXT n

<b>FORMAT</b>	Ключевое слово в Spectrum- Basic не задействовано.
<b>GO SUB n</b>	Стр. 33. Оператор передачи управления подпрограмме. Номер строки, с которой начинается подпрограмма, задается целочисленным значением n. Текст подпрограммы должен заканчиваться оператором RETURN, который возвращает управление оператору, следующему за GO SUB. 120 GO SUB 3000 130 GO SUB 500
<b>GO TO n</b>	Стр. 25. Оператор безусловного перехода на строку с номером n, который должен иметь целочисленное значение. GO TO 330
<b>IF...THEN..</b>	Стр. 26. Оператор условного перехода, причем условие ставится за IF. Если это условие выполняется, то управление передается на оператор (операторы), стоящий за ключевым словом THEN. Если же оно не выполняется (ложно), то управление переходит к следующей строке программы. 40 LET x=7 50 IF x<5 THEN PRINT "good": GO TO 300 60 PRINT "bad" <i>bad</i>
<b>IN m</b>	Стр. 109. Функция считывает число (байт) из порта. Адрес порта задается целочисленным значением m (0... 65535), записываемым за ключевым словом. 480 LET jo=IN 31
<b>INK c</b>	Стр. 15. Оператор устанавливает цвет тона, то есть цвет, в который окрашиваются выводимые на экран символы, точки, линии. Код цвета c должен иметь одно из целочисленных значений (0...9). 50 PRINT AT 10,12;INK 4;"Sinclair"
<b>INKEY\$</b>	Стр. 26. Функция возвращает символ, соответствующий нажатой клавише. Используется для обнаружения факта нажатия клавиши на клавиатуре. 100 IF INKEY\$="P" THEN GO TO 1000 120 IF INKEY\$="O" THEN GO TO 2000
<b>INPUT a,b,t\$</b>	Стр. S3. Оператор ввода данных в программу: присваивает числовым или символьным переменным a, b, t\$, расположенным за ключевым словом, соответственно, числовые или символьные значения, набираемые на клавиатуре. 20 INPUT INK 1;"name:";k\$
<b>INT x</b>	Стр. 24. Функция преобразует числовое значение x до ближайшего (меньшего) целого числа. PRINT INT 23.59;" ";INT-3.84 <i>23 -4</i>
<b>INVERSE p</b>	Стр. 213. Оператор устанавливает режим инвертированного отображения графической и символьной информации. Число p

может принимать одно из следующих целочисленных значений:  
 0 - устанавливает нормальный режим цветопередачи; 1 - при печати символов цвет тона и цвет фона меняются местами; графическая информация отображается цветом фона и становится невидимой.

```
10 PAPER 7: INK 0:CLS
20 INVERSE 1: PRINT "Moon"
```

**LEN a\$** Стр. 27, 28. Функция возвращает длину (число символов) символьного значения a\$.

```
PRINT LEN "Compiler"
8
```

**LET x=a** Стр. 20. Оператор присваивает переменной x значение a.

```
30 LET y=3.1415926
40 LET b$="Tape"
50 LET s(12.7)=128
```

**LINE m** Стр. 19. Ключевое слово ставится после оператора SAVE и указывает, что программа после ее загрузки оператором LOAD будет сразу запущена со строки с номером m.

```
SAVE "Star" LINE 10
```

**LIST m** Оператор выводит на экран листинг бейсик-программы, находящейся в памяти компьютера, начиная со строки m. Если номер не указан, выводится вся программа.

**LLIST m** Оператор распечатывает на принтере листинг бейсик-программы, находящейся в памяти компьютера, начиная со строки m.

**LN z** Функция натурального логарифма положительного числа z.

```
PRINT LN 5.5
1.704748
```

**LOAD a\$** Стр. 19. Оператор загрузки бейсик-программы в память компьютера. Имя программы задается значением, a\$. LOAD с «пустым» именем загружает первую встретившуюся на ленте программу.

```
LOAD "Express"
LOAD ""
```

**LOAD a\$ CODE [n], [l]** Стр. 43. Оператор загрузки в память компьютера кодового блока (последовательности байтов). Имя файла, в котором был сохранен блок на магнитной ленте, задается символьным значением a\$. За ключевым словом CODE указывается адрес n размещения блока кодов в памяти компьютера и количество загружаемых байт - l.

```
LOAD "Raiders" CODE
LOAD "Maxit" CODE 16384,6912
LOAD "" CODE
```

**LOAD a\$ DATA r[\$] ( )** Оператор загружает с магнитной ленты в память компьютера массив. Имя файла, содержащего массив, задается символьным значением a\$. Следом за DATA указывается имя массива г (буква

или буква со знаком \$) с пустыми скобками.

```
LOAD "Fighter" DATA r()
LOAD "Monitor" DATA s$()
```

**LOAD a\$  
SCREEN\$**

Оператор загружает с магнитной ленты блок кодов непосредственно в экранную область памяти, и на экране воспроизводится ранее записанная картинка. Имя файла, в котором она была сохранена на ленте, - a\$.

```
LOAD "Zastawka" SCREEN$
```

**MERGE a\$**

Стр. 146. Оператор подгружает бейсик-программу с магнитной ленты в память компьютера, не стирая уже находящейся там программы. Имя загружаемой программы задается значением a\$. Оператор MERGE соединяет две программы, располагая их строки по порядку возрастания номеров.

```
MERGE "Command"
```

**MOVE**

Ключевое слово в Spectrum-Basic не задействовано.

**NEW**

Стр. 144. Оператор очищает область памяти, отведенную для размещения и работы бейсик-программы.

**NEXT n**

Стр. 19. Оператор цикла, позволяющий выполнять часть программы, заключенной между FOR и NEXT, заданное число раз. Управляющая переменная n принимает значения от n = a до n = b с шагом h. Если он не задан, то по умолчанию принимается 1.

```
200 FOR n=1 TO 10
210 PRINT "*";
220 NEXT n
```

**NOT**

Стр. 55. Логический оператор отрицания условия, стоящего за ключевым словом: истинное условие делает ложным, ложное - истинным.

```
60 LET x=5:IF NOT (x>10) THEN PRINT "No"
```

Функция возвращает 0, если числовое значение, следующее за ключевым словом, не равно нулю; 1 - если значение равно нулю

```
PRINT NOT(7*8-56)
```

```
1
```

**OR**

Стр. 34. Оператор логического сложения условий. Комбинация условий, объединенных операторами OR, истинна, если истинно хотя бы одно из них.

```
50 LET y=64:
```

```
60 IF y=32 OR y=64 OR y=128 THEN PRINT "Left"
```

Функция возвращает значение, стоящее перед ключевым словом, если ее аргумент равен 0 или возвращает 1, если аргумент не равен 0.

```
PRINT 33 OR 6
```

```
1
```

**OUT s,b**

Стр. 113. Оператор записывает в порт целочисленное значение b (0...255), адрес которого задается целочисленным значением s (0...65535).

```
OUT 254,3
```

- OVER s** Стр. 33. Оператор устанавливает режим наложения изображений на экране. Число *s* может принимать значения:  
 0 - устанавливает нормальный режим отображения информации; 1 - новое изображение накладывается на уже существующее на экране, не стирая его. Точки пересечения принимают окраску фона; 8 - задает режим прозрачности, при котором режим наложения соответствует ранее установленному для данного знакоместа.  
 10 OVER 1  
 20 PRINT AT 10,10;"1 2 3 "  
 30 PRINT AT 10,11;"a b c"  
*1a2b3c*
- PAPER c** Стр. 15. Оператор устанавливает цвет фона - цвет, в который окрашивается экран. Число *c* принимает целочисленное значение в пределах от 0 до 9 и является кодом цвета.  
 10 BORDER 0: PAPER 0: INK 7
- PAUSE t** Стр. 17. Оператор приостанавливает работу программы на время, задаваемое целочисленным значением *t* (0... 65535). Пауза равна *t*/50 сек. Оператор PAUSE 0 продолжит работу программы только после нажатия какой-либо клавиши.  
 PAUSE 200 (пауза на 4 сек.)
- PEEK n** Стр. 138. Функция считывает число в интервале от 0 до 255 (байт) из ячейки памяти. Адрес ячейки задается целочисленным значением *n* (0... 65535).  
 60 POKE 40000,209  
 70 PRINT PEEK 40000  
*209*
- PI** Стр. 84. Функция возвращает значение числа «пи» (3.1415926).  
 PRINT PI  
*3.1415926*
- PLOT x , y** Стр. 23. Оператор ставит на экране точку (включает пиксель). Позиция точки задается двумя целочисленными значениями координат: горизонтальной - *x* (0...255) и вертикальной - *y* (0...175).  
 300 PLOT INK 1; INVERSE 1; *x,y*
- POINT x , y** Функция проверяет состояние пикселя. Если он включен (окрашен в цвет тона), функция возвращает 1, а если выключен (окрашен в цвет фона) - 0. Числа *x* и *y* соответствуют горизонтальной (0...255) и вертикальной (0...175) координатам.  
 570 IF POINT (*x,y*)=1 THEN GO TO 240
- POKE n, b** Стр. 46. Оператор записывает в память целочисленное значение *b* (0...255). Адрес ячейки памяти задается целочисленным значением *n* (0... 65535).  
 100 POKE 23609, 155  
 110 PRINT PEEK 23609  
*155*

**PRINT a [b\$]** Стр. 15. Оператор выводит на экран данные - числовые a и символьные b\$ значения, следующие за ключевым словом. Перед блоками данных могут ставиться слова, задающие атрибуты (AT, TAB, INK, PAPER, FLASH, INVERSE и OVER). 25 PRINT AT 15,10;INK 1;PAPER 4;"Piraimda"

Числовые выражения, стоящие за PRINT, вычисляются, и на экране отображается результат в десятичной форме. Числа больше  $10^8$  и меньше  $10^{-5}$  отображаются в экспоненциальной форме.

```
PRINT 500000000
5E8
```

Для управления печатью используются следующие символы: ; - блоки отображаются один за другим без пробелов; , - текущая позиция печати переводится на середину текущей строки или в начало следующей, в зависимости от позиции последнего напечатанного символа; ' - данные, отделенные апострофом, печатаются с новой строки.

**RANDOMIZE k** Стр. 64. Оператор задает начало последовательности случайных чисел, генерируемой оператором RND. Целочисленное значение k (1... 65535) устанавливает фиксированное начало последовательности.

```
RANDOMIZE 51000
```

**READ w** Стр. 21. Оператор присваивает переменной w значения, считываемые из списка данных, следующих за оператором DATA.

```
50 READ x
```

**REM** Стр. 18. Оператор отделяет текст комментариев от текста программы. Следом за REM могут стоять любые символы.

```
100 GO SUB 5000: REM Подпрограмма заставки.
```

**RESTORE n** Стр. 121. Оператор указывает строку n DATA, с которой нужно считывать данные следующим по программе оператором READ. Если n не указано, либо n=0, то оператор READ обращается к первому в программе оператору DATA.

**RETURN** Стр. 29. Оператор заканчивает подпрограмму (GO SUB) и передает управление на оператор, следующий сразу за GO SUB.

**RND** Стр. 24. Функция возвращает псевдослучайное число в интервале от 0 до 1, исключая саму 1. Изменить последовательность можно выполнением оператора RANDOMIZE.

```
FOR j=1 TO 5:PRINT RND:NEXT j
0.26338196
0.75448608
0.58673096
0.00527954
0.39709473
```

**RUN [n]** Стр. 18. Оператор запуска бейсик-программы. Для запуска с определенной строки, должно быть задано целочисленное

значение n.

**SAVE a\$**  
**[LINE n]** Стр. 19. Оператор записывает бейсик-программу на магнитную ленту. Программе присваивается имя, задаваемое символьным значением a\$, которое должно содержать не более десяти символов и не может быть «пустым». Ключевое слово LINE указывает, что программа после ее загрузки оператором LOAD будет сразу запущена со строки с номером n.  
SAVE "Fox" LINE 5

**SAVE a\$ CODE**  
**n, l** Стр. 18. Оператор сохранения на магнитной ленте блока кодов (последовательности) байтов. Имя файла, в который записывается блок, задается символьным значением a\$. Имя должно состоять не более чем из десяти символов и не может быть «пустым». За ключевым словом CODE указывается адрес n размещения блока кодов в памяти компьютера и количество загружаемых байт l.  
SAVE "Spruts" CODE 16384,6912

**SAVE a\$ DATA**  
**r[\$] ( )** Оператор сохраняет на магнитной ленте массив. Имя файла, в котором сохраняется массив, задается символьным значением a\$. Имя должно состоять не более чем из десяти символов и не может быть «пустым». Следом за DATA указывается имя массива r (буква или буква со знаком \$) с пустыми скобками.  
SAVE "Monitor" DATA r()  
SAVE "Fighter" DATA s\$()

**SAVE a\$**  
**SCREEN\$** Оператор записывает на магнитную ленту экранное изображение. Имя файла, в котором сохраняется экранное изображение, задается символьным значением a\$. Имя должно состоять не более чем из десяти символов и не может быть «пустым».  
SAVE "Koro1" SCREEN\$

**SCREEN\$**  
**(y, x)** Стр. 229. Функция возвращает символ, помещенный в указанном знакоместе экрана. Позиция знакоместа задается двумя целочисленными значениями:  
y (0...21) - номер строки; x (0...31) - номер столбца.  
200 PRINT AT 10, 12; "@"  
210 LET a\$=SCREEN\$ (10, 12)  
220 PRINT a\$  
@

**SGN g** Стр. 172. Функция определяет знак числового значения. Она возвращает следующие значения:  
1 - если аргумент положителен; -1 - если аргумент отрицателен; 0 - если аргумент равен 0.  
PRINT SGN15  
1  
PRINT SGN -130  
-1

**SIN p** Стр. 84. Функция вычисляет синус угла p, заданного в радианах.  
PRINT SIN 1.2  
0.93203909

- SQR k**            Стр. 239. Функция вычисляет квадратный корень числа k.  
                       PRINT SQR 49  
                       7
- STEP**            Стр. 19. Оператор используется для построения цикла FOR...NEXT.  
                       Устанавливает шаг приращения управляющей переменной.  
                       10 FOR n=12 TO 2 STEP-2  
                       20 PRINT n;" ";  
                       30 NEXT n  
                       12 10 8 6 4 2
- STOP**            Стр. 34. Оператор останавливает выполнение бейсик-программы.  
                       После остановки программы ее выполнение может быть продолжено оператором CONTINUE.
- STR\$ w**          Стр. 131. Функция преобразует числовое значение w в символьное. STR\$ возвращает значение аргумента в виде символьной константы.  
                       70 LET A=747920  
                       80 A\$=STR\$ A  
                       90 PRINT A\$  
                       747920
- TAB x**            Стр. 55. Оператор задает позицию вывода на экран в текущей строке. Целочисленное значение x (0..31) указывает номер колонки. Если x меньше текущей позиции вывода, символ помещается в заданную позицию, но строкой ниже. TAB используется только с PRINT и INPUT.  
                       PRINT TAB 1;"\*";TAB 5;"\*";TAB 3;"|";TAB 3;"\_"  
                       \* \*  
                       /  
                       -
- TAN p**            Функция вычисляет тангенс угла p; заданного в радианах.  
                       PRINT TAN 1.3  
                       3.6021024
- THEN**            Стр. 26. Ключевое слово, используется в операторе IF...THEN...
- TO**                Стр. 19. Как оператор, используется для построения циклов FOR...NEXT.  
                       Как функция, выделяет подстроку (выполняет сечение) символьного значения. Начальная и конечная позиции сечения задаются двумя целыми числами, которые записываются через ключевое слово TO в скобках.  
                       PRINT "PIRAMIDA" (3 TO 6)  
                       RAMI
- USR n**            Стр. 144. Функция с целочисленным значением передает управление подпрограмме в машинных кодах, расположенной по адресу n (0... 65535).

**USR "a"** Стр. 47. Функция с символьным значением возвращает адрес первого из восьми байтов, задающих изображение символа, определяемого пользователем. Следом за USR указывается одна из 20 букв A,B,C,...U.

```
PRINT USR "A"
63368
```

**VAL c\$** Стр. 164. Функция возвращает символьное значение, представляющее собой символьную запись числового значения, в числовую константу.

```
PRINT VAL "4+9"
-49
```

**VAL\$ "c\$"** Функция возвращает строку символов, представляющую собой результат вычисления символьного выражения, записанного в виде символьной константы.

```
10 LET a$="ROBIN":LET b$="WOOD"
20 LET c$="a$+";" in the ";"+b$"
30 PRINT VAL$ c$
ROBIN in the WOOD
```

**VERIFY a\$** Стр. 146. Оператор проверяет правильность записи программы на магнитную ленту. Имя проверяемой программы задается символьным значением a\$. Если программа, записанная на ленте, не совпадает с программой, находящейся в памяти компьютера, то выдается сообщение об ошибке:

Tape loading error.

Таким же образом можно проверить правильность записи массивов **VERIFY...DATA**, экранного изображения - **VERIFY...SCREEN\$** и блока кодов - **VERIFY...CODE**.

```
VERIFY" " VERIFY"Miny"
VERIFY"Sokoban"DATA r()
VERIFY"Space war"SCREEN$
VERIFY"Tanx"CODE
VERIFY"Tanx"CODE 16384,6912
```

### Приложение 3. Команды графического редактора Art Studio

<b>FILE</b>	<b>Запись, чтение, проверка файла</b>
<b>Save file...</b>	Запись экранного файла.
<b>Load file ...</b>	Загрузка экранного файла.
<b>Load file</b>	Загрузка первого встреченного на ленте экранного файла. (Только для кассетной версии.)
<b>Load current</b>	Загрузка экранного файла с последним введенным именем. (Только для дисковой версии.)
<b>Verify file...</b> <b>Verify file</b>	Проверка экранного файла, записанного на ленту. (Только для кассетной версии.)
<b>Merge file...</b> <b>Merge file</b> <b>Merge current</b>	Наложение экранного файла с кассеты/диска на картинку.
<b>Catalogue</b>	Просмотр каталога диска. (Только для дисковой версии.)
<b>Erase file ...</b>	Удаление файла с диска. (Только для дисковой версии.)
<b>ATTRIBUTES</b>	<b>Выбор цветовой палитры</b>
<b>Set Ink</b>	Выбор цвета тона. Для выбора одного из восьми цветов появившейся на экране палитры наведите на него стрелку и нажмите стрельбу. Можно установить пассивный цвет (TRANSPARENT), который не будет влиять на раскраску экрана во время изображения объектов.
<b>Set paper</b>	Выбор цвета фона.
<b>Set border</b>	Выбор цвета бордюра.
<b>Bright</b>	Выбор градации яркости. Можно установить низкую (BRIGHT OFF), высокую (BRIGHT ON) либо пассивную (TRANSPARENT) градацию яркости.
<b>Flash</b>	Выбор эффекта мерцания.
<b>Over</b>	Выбор/отмена режима инвертирования. Выполняется аналогично оператору Бейсика OVER.
<b>Inverse</b>	Выбор/отмена режима удаления. При выбранном режиме изображение объекта осуществляется не нанесением точек, а их удалением.
<b>Transparent</b>	Установка всех цветовых атрибутов пассивными. Удобно

использовать при редактировании готовых картинок.

**Standard** Установка стандартных цветов (белый бордюр, белый фон, черный тон).

## **PAINT**

## **Рисование кистью, пером, распылителем**

**Pen** Рисование пером.

**Spray can** Рисование распылителем.

**Brush** Работа кисточкой.

**Edit** Редактирование формы кисти. Выберите кисть, форма которой вас не устраивает. На экране появится ее изображение в восьмикратном увеличении. Наводя стрелку на нужные точки и нажимая стрельбу, можно создать кисть любой формы - они заменит ту, которую вы вызывали на редактирование.

**BrushInverse** Выбор/отмена режима удаления. Повторяет INVERSE в меню ATTRIBUTES

## **MISCELLANEOUS**

## **Разное**

**View screen** Просмотр экрана До нажатия клавиши Огонь картинка выводится на вес 24 строки экрана.

**Clear screen** Полная очистка экрана.

**Bright grid 1** Нанесение яркостной сетки 1. Знакоместа отделяются друг от друга переменной градации яркости. Удобно использовать при планировании и раскраске картинки.

**Bright grid 2** Нанесение яркостной сетки 2. В отличие от предыдущего режима выделяется не каждое знакоместо, а квадраты из четырех знакомест.

**Remove grid** Удаление яркостной сетки.

**Change colour** Изменение цвета в заданном окне. Перед тем, как выбрать эту функцию, в режиме SET INK (меню ATTRIBUTES) установите цвет, который требуется изменить, и в режиме SET PAPER (меню ATTRIBUTES) - цвет, на который необходимо изменить. Цвет будет изменен на другой независимо от того, был ли это цвет фона или цвет тона.

## **WINDOWS**

## **Работа с окнами**

**Define window** Задание окна произвольного размера.

<b>Last window</b>	Вызов последнего заданного окна. Информация о последнем заданном окне хранится в памяти компьютера, и оно может быть снова вызвано на экран.
<b>Whole screen</b>	Определение окна размером во весь экран.
<b>Clear window</b>	Полная очистка окна.
<b>Cut &amp; paste window</b>	Копирование окна. Вместо стрелки вам «на руки» будет выдано окно. Установив его на нужное место, нажмите клавишу Огонь, и содержимое исходного окна будет повторено на указанном месте.
<b>Cut, clear &amp; paste</b>	Перенос окна. Применяется аналогично предыдущей функции, но окно, которое служило источником, очищается.
<b>Invert window</b>	Инвертирование содержимого окна.
<b>Re-scale window</b>	Копирование окна с изменением масштаба. Определите второе окно, после чего содержимое исходного будет повторено в нем независимо от размеров обоих.
<b>Clear &amp; re-scale</b>	Перенос окна с изменением масштаба. Применяется аналогично предыдущей функции, но окно, которое служило источником, очищается.
<b>Flip horizontal</b>	Поворот окна вокруг горизонтальной оси симметрии.
<b>Flip vertical</b>	Поворот окна вокруг вертикальной оси симметрии.
<b>Rotate 1/4, 1/2, 3/4</b>	Поворот окна по часовой стрелке на 90, 180 и 270 градусов.
<b>Merge</b>	Выбор/отмена режима наложения. Используется при операциях, связанных с перемещением окон (CUT & PASTE WINDOW, CLEAR & RE-SCALE и т. п.). При выбранном режиме перемещаемое изображение накладывается на изображение, находящееся на экране. В противном случае старое изображение затирается.
<b>Multiple</b>	Выбор/отмена режима многократного повторения операций копирования и переноса.

**FILL****Закрашивание объектов**

<b>Solid fill</b>	Полное закрашивание объекта.
<b>Textured fill</b>	Заполнение объекта текстурой.
<b>Wash texture</b>	Замена текстуры. Укажите, какую текстуру вы хотите видеть на месте предыдущей. Кроме замены, можно также производить и наложение текстур одну на другую.
<b>Edit texture</b>	Редактирование текстуры.

---

**MAGNIFY** **Редактирование в увеличенном масштабе**


---

**Magnify x2, x4, x8** Редактирование деталей изображения с увеличением в 2, 4 и 8 раз.

**Grid** Выбор/отмена сетки. Сетка используется для отделения точек друг от друга при восьмикратном увеличении фрагмента картинка.

---

**TEXT** **Работа с текстами**


---

**Left to right** Печать текста слева направо.

**Downwards** Печать текста сверху вниз.

**Normal height** Установка высоты букв. Можно выбрать нормальную (NORMAL), удвоенную (DOUBLE) и утроенную (TREBLE) высоту.

**Double height**

**Treble height**

**Normal width** Установка ширины букв. Можно выбрать нормальную, удвоенную или утроенную ширину.

**Double width**

**Treble width**

**Sideways** Выбор/отмена печати букв, повернутых на 90 градусов.

**Bold** Печать жирным шрифтом.

**Caps lock** Фиксация верхнего регистра.

**Snap horizontal** Выбор/отмена установки курсора на ближайшее знакоместо по горизонтали. При необходимости поместить символ точно в знакоместо, используйте эту и следующую функции.

**Snap vertical** Выбор/отмена установки курсора на ближайшее знакоместо по вертикали.

**Font editor** Редактирование набора символов. Это не просто функция, а самостоятельная программа, которая сама по себе требует отдельного описания, поэтому ей будет посвящен специальный раздел.

---

**SHAPES** **Построение геометрических фигур**


---

**Points** Нанесение точек.

**Lines** Проведение прямых линий.

**Continious lines** Проведение ломаных линий.

**Rectangles** Построение прямоугольников.

**Triangles** Построение треугольников.

**Circles** Построение окружностей.

<b>Rays</b>	Построение отрезков с общим началом (лучей).
<b>Elastic</b>	Режим подсказки. При выбранном режиме фигуры, которые вы рисуете, отображаются на экране пунктирной линией от базовой точки к текущему положению курсора еще в процессе выбора размеров и местоположения фигуры. Изображение фиксируется нажатием клавиши Огонь.
<b>Snap horizontal</b>	Привязка точки к знакоместу по горизонтали
<b>Snap vertical</b>	Привязка точки к знакоместу по вертикали.

**FONT EDITOR****Редактор наборов символов****CHARACTER      Операции над отдельным символом**

<b>Clear1</b>	Очистка символьного поля.
<b>Invert</b>	Инверсия символа.
<b>Flip horizontal</b>	Поворот символа вокруг горизонтальной оси.
<b>Flip vertical</b>	Поворот символа вокруг вертикальной оси.
<b>Rotate 1/4</b>	Поворот символа на 90 градусов по часовой стрелке.
<b>Scroll right</b>	Сдвиг символа вправо на одну точку.
<b>Scroll down</b>	Сдвиг символа вниз на одну точку.

**FONT      Операции над набором символов**

Это меню аналогично предыдущему, разница лишь в том, что действие его функций распространяется сразу на весь набор символов.

**MISCELLANEOUS      Разное**

<b>Capture font</b>	Копирование символов с экрана. Функция позволяет «вытаскивать» изображения символов из загруженной в Art Studio картинки. Для этого в главном меню необходимо определить окно (функция DEFINE WINDOW меню WINDOWS), из которого графические изображения будут копироваться в область набора символов. Копирование происходит следующим образом. Из левого верхнего угла окна выбирается квадрат размером 8x8 точек и помещается на место символа, отмеченного указателем. Следующий блок копируется в символ, находящийся после отмеченного, и т. д. до исчерпания окна либо набора символов.
<b>Copy ROM</b>	Загрузка из ПЗУ стандартного набора символов.
<b>Menu</b>	Возврат в главное меню.

## Приложение 4. Команды музыкального редактора Wham

### **LOAD TUNE**

### **Загрузка мелодии**

Этот режим позволяет загрузить файл, содержащий мелодию (будем называть его редактируемым файлом), в буфер для дальнейшей работы с ним. После перехода в режим (клавиша 1) программа спросит, с какого устройства вы хотите загружать файл. Выбор устройства осуществляется нажатием одной из клавиш: T - кассета, M - оперативная память, D - дискета.

### **SAVE TUNE**

### **Сохранение мелодии**

Эта функция позволяет записать редактируемый файл. После выбора устройства для записи клавишами T (кассета), M (память) или D (дискета) введите имя файла, в котором будет сохранена мелодия.

### **EDIT MODE**

### **Режим редактирования**

Для ввода мелодии используются клавиши двух нижних рядов, а два верхних ряда клавиш служат для выбора функций. Для получения дополнительных шумовых эффектов нужно нажать клавишу 8. На экране появится меню функций выбора частоты (WAVEFORM) и длительности (DURATION) шумовых эффектов. Выбрать один из предоставляемых программой 16 вариантов длительности и 8 вариантов частоты (формы колебаний) шумовых эффектов можно с помощью следующих клавиш:

- 5,8 - выбор редактируемого шумового эффекта;
- 6 - переход к функции выбора частоты;
- 7 - переход к функции выбора длительности;
- 0 - редактирование эффекта.

После настройки эффектов можно выйти обратно в режим редактирования мелодии (клавиша Enter) и использовать их в работе с помощью клавиш Y, U и I.

### **HEAR TUNE**

### **Прослушивание мелодии**

При прослушивании мелодии в режиме EDIT MODE, она звучит в несколько искаженном виде. Чтобы получить представление об истинном звучании мелодии, то есть услышать, как она будет исполняться после компиляции (см. ниже), воспользуйтесь функцией HEAR TUNE. После прослушивания нажмите любую клавишу.

### **SET TEMPO**

### **Изменение темпа**

В этом режиме устанавливается необходимый темп исполнения мелодии. Клавишей 8 можно ускорить темп, клавишей 5 - замедлить его.

### **WHAMPILER**

### **Компиляция**

Если нужно музыкально оформить программу, написанную на Бейсике или ассемблере, необходимо записать мелодию в виде подпрограммы, работающей независимо от редактора. В качестве примера предлагаем откомпилировать одну из 5 мелодий, загружаемых из памяти (см. LOAD TUNE - MEMORY), допустим, тему под номером 1 - FREEDOM. Загрузите ее и, нажав клавишу 4, войдите в режим компиляции. На запрос

TUNENAME?

введите имя, под которым вы хотите сохранить исполняемый файл, например, FREE.MUS. Далее программа попросит ввести адрес, с которого будет располагаться и запускаться исполняемый файл:

```
ASSEMBLY ADDRESS?
```

Укажите десятичный адрес, например, 60000, после чего компилятор выведет на экран следующую информацию:

```
TUNE NAME FREE.MUS
ASSEMBLY ADDRESS 60000
RETURN OPTION KEYPRESS
WHITE NOISE - NONE -
CHANNEL 1 LENGTH 313 количество шагов в 1-м канале
CHANNEL 1 LOOP START
CHANNEL 2 LENGTH 313 количество шагов во 2-м канале
CHANNEL 2 LOOP START
1. KEYPRESS
2. ALWAYS
3. TUNEEND
RETURN OPTION 1,2 OR 3
```

Цифрами 1, 2 и 3 обозначены условия компиляции. В зависимости от выбранного условия получается определенная модификация исполняемого файла:

**KEYPRESS** - проигрывание мелодии завершается при нажатии любой клавиши;

**ALWAYS** - проигрывание мелодии осуществляется одновременно с работой программы (подробно этот режим описан ниже);

**TUNEEND** - проигрывание завершается по окончании мелодии или при нажатии любой клавиши.

Запуск компиляции произойдет сразу после выбора клавишами 1, 2 или 3 одного из перечисленных условий. Нажмите, к примеру, клавишу 1.

По окончании компиляции на экран будет выведена длина исполняемого файла в байтах и сообщение о нормальном завершении операции. Затем появится информация, необходимая для настройки исполняемого файла уже в процессе использования его в вашей программе (напомним, что в нашем примере ASSEMBLY ADDRESS = 60000);

```
REPLAY SPEED 60035, (230 TO 255)
```

Этой надписью программа сообщает, что, записывая в ячейку памяти с адресом 60035 (ASSEMBLY ADDRESS + 35) число от 230 до 255, можно изменять темп проигрывания мелодии;

```
BORDER COLOR: 60026, (0 TO 7)
```

Это сообщение говорит о том, что, записывая в ячейку с адресом 60026 (ASSEMBLY ADDRESS + 26) числа от 0 до 7, вы можете на время исполнения мелодии устанавливать требуемый цвет бордюра (по умолчанию- фиолетовый);

```
TO RUN - RANDOMIZE USR 60000
```

Эта надпись напоминает, что для запуска исполняемого файла из Бейсика необходимо использовать команду

```
RANDOMIZE USR 60000 (ASSEMBLY ADDRESS)
```

После вывода этих сообщений производится запись исполняемого файла. На экране появится стандартное сообщение

```
START TAPE, THEN PRESS ANY KEY
```

после чего нужно включить магнитофон на запись и нажать любую клавишу.

**ЛИТЕРАТУРА**

1. Ларченко А. А., Родионов Н. Ю. ZX Spectrum для пользователей и программистов- 2-е изд. СПб.: Питер, 1993.
2. Диалекты Бейсика для ZX Spectrum (под редакцией Родионова Н. Ю., Ларченко А. А.)- СПб.: Питер, 1992.
3. Родионов Н. Ю. Адаптация программ к системе TR-DOS- СПб.: Питер, 1992.
4. 600 игр для ZX Spectrum.- СПб.: Питер, 1993.
5. Системные программы для ZX Spectrum. Выпуск 1. (под редакцией Родионова Н. Ю.)- СПб.: Питер, 1993.
6. Александров В. В. и др. ЭВМ: игра и творчество.- СПб.: Машиностроение, 1989.
7. Кершан Б. и др. Основы компьютерной грамотности.- Москва: Мир, 1989.

# Содержание

---

<b>ВВЕДЕНИЕ.....</b>	<b>3</b>
<b>1. СТРУКТУРА ИГРОВОЙ ПРОГРАММЫ .....</b>	<b>6</b>
Заставка .....	6
Игровое пространство .....	7
Блок взаимодействия с играющим.....	8
Блок оценки игровой ситуации.....	9
<b>2. ЗАСТАВКА .....</b>	<b>10</b>
Простейшие заставки .....	11
Бегущая строка .....	21
Печатающий квадрат .....	22
Прыгающие буквы .....	24
Заставка из нескольких кадров.....	26
Делаем картинку-заставку с помощью ART STUDIO .....	30
<b>3. ПОСТРОЕНИЕ ИГРОВОГО ПРОСТРАНСТВА.....</b>	<b>34</b>
Из чего состоит игровое пространство .....	34
Создание графических объектов .....	34
Спрайт-генератор .....	39
Русификация программ.....	45
Изготовление спрайтов с помощью ART STUDIO .....	49
<i>Редактирование увеличенного изображения.....</i>	<i>50</i>
<i>Кодирование спрайта методом «окон» и запись его на ленту.....</i>	<i>51</i>
<i>Использование полученного спрайт-файла.....</i>	<i>53</i>
<i>Создание спрайтов с учетом их конфигурации .....</i>	<i>53</i>
Построение пейзажей .....	54
<b>4. ПРИЕМЫ ПРОГРАММИРОВАНИЯ ИГР .....</b>	<b>61</b>
Составление блок-схемы программы.....	61
Модульное построение программы .....	65
Составление программы меню.....	71
Звуковое сопровождение игр .....	75
Музыкальный редактор Wnam .....	76
<i>Сочинение новой мелодии.....</i>	<i>76</i>
<i>Компиляция и запись мелодии.....</i>	<i>80</i>
<b>5. БЛОК ВЗАИМОДЕЙСТВИЯ С ИГРАЮЩИМ.....</b>	<b>82</b>
Управление спрайтами с помощью клавиатуры .....	82
Управление спрайтами с помощью джойстика .....	84
Особенности управления клавиатурой .....	90
Взаимодействие играющего с программой .....	92
<b>6. ОЦЕНКА ИГРОВОЙ СИТУАЦИИ.....</b>	<b>98</b>
Элементы оценок.....	98
Комплексная оценка игры.....	100

<b>7. СИСТЕМНЫЕ ПЕРЕМЕННЫЕ.....</b>	<b>104</b>
КЛАВИАТУРА.....	104
ЭКРАН.....	105
НАБОРЫ СИМВОЛОВ.....	107
РАЗНОЕ .....	108
ПОДПРОГРАММЫ ПЗУ.....	110
<b>8. LASER BASIC В ИГРОВЫХ ПРОГРАММАХ.....</b>	<b>111</b>
ВВЕДЕНИЕ В LASER BASIC .....	111
ЗАСТАВКА НА LASER BASIC .....	113
ИГРОВАЯ ПРОГРАММА .....	117
<b>9. ПЕРВЫЕ ШАГИ К КОДОВОЙ ПРОГРАММЕ .....</b>	<b>120</b>
ПРОГРАММА SUPERCODE .....	120
ПРИМЕНЕНИЕ NEW SUPERCODE .....	127
ИГРОВАЯ ПРОГРАММА «ЗВЕЗДНАЯ ВОЙНА» .....	128
КОМПИЛЯЦИЯ ИГРОВЫХ ПРОГРАММ .....	132
<b>10. ГЕНЕРАТОР ИГР GAMES DESIGNER.....</b>	<b>135</b>
ГЛАВНОЕ МЕНЮ.....	135
ИГРА.....	137
ПОРЯДОК ПРОЕКТИРОВАНИЯ ИГРЫ .....	138
ВНЕШНИЙ ВИД ИГРЫ .....	138
ПЕРЕМЕЩЕНИЯ ПРОТИВНИКА.....	141
УРОВНИ СЛОЖНОСТИ .....	141
ИЗМЕНЕНИЕ ГРАФИКИ .....	144
ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ.....	146
<b>ПРИЛОЖЕНИЕ 1. НЕСКОЛЬКО ИГРОВЫХ ПРОГРАММ .....</b>	<b>149</b>
МИНЫ.....	149
ЛИСЫ .....	151
МАКСИТ.....	156
ПИРАМИДА .....	161
ЦИФЕРТОН .....	165
ЛАБИРИНТ .....	169
АНАКОНДА .....	171
СПРУТЫ.....	173
ПРИЛУНЕНИЕ .....	175
КОРОЛЕВСТВО .....	179
<b>ПРИЛОЖЕНИЕ 2. ОСНОВНЫЕ ОПЕРАТОРЫ, ФУНКЦИИ И КОМАНДЫ СПЕКТРУМ-БЕЙСИКА.....</b>	<b>186</b>
<b>ПРИЛОЖЕНИЕ 3. КОМАНДЫ ГРАФИЧЕСКОГО РЕДАКТОРА ART STUDIO... 197</b>	
<b>ПРИЛОЖЕНИЕ 4. КОМАНДЫ МУЗЫКАЛЬНОГО РЕДАКТОРА WHAM.....</b>	<b>202</b>
<b>ЛИТЕРАТУРА .....</b>	<b>204</b>