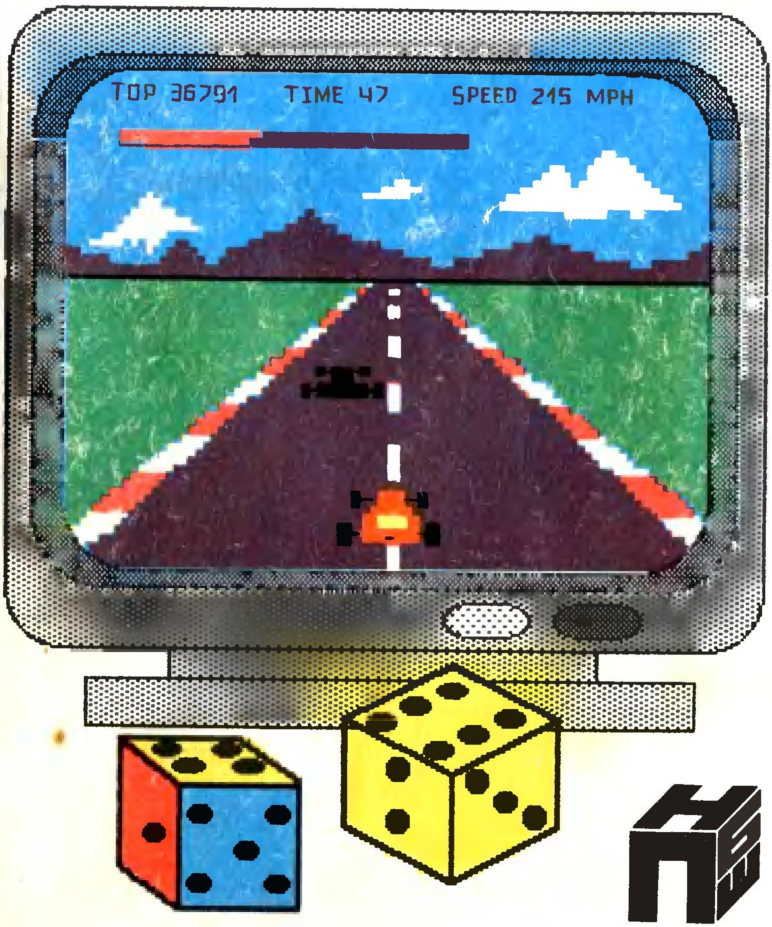
В.В.АЛЕКСАНДРОВ
А.И.АЛЕКСЕЕВ
А.И.СЕМЕНКОВ

ЭВМ: игра и творчество



НАУЧНО-ПОПУЛЯРНАЯ БИБЛИОТЕКА ШКОЛЬНИКА

НАУЧНО-ПОПУЛЯРНАЯ БИБЛИОТЕКА ШКОЛЬНИКА
Основана в 1986 году

В.В.АЛЕКСАНДРОВ
А.И.АЛЕКСЕЕВ
А.И.СЕМЕНКОВ

ЭВМ: игра и творчество

Под общей редакцией
д-ра техн наук **В. В. Александрова**

Ленинград
„Машиностроение"
Ленинградское отделение
1989

ББК 32.973
А46
УДК 681.3

Рецензент д-р техн. наук Н. А. РУМЯНЦЕВ

Александров В. В. и др.

А46 ЭВМ игра и творчество В. В. Александров, А. И. Алексеев, А. И. Семенков; Под общ. ред. д-ра техн. наук В. В. Александрова.— Л.: Машиностроение. Ленингр. отд-ние, 1989.— 128 с., ил.— (Науч.-попул. б-ка школьника).

ISBN 5-217-00698-6

В книге в популярной форме рассказано о создании и применении игровых обучающих программ. Приведена обобщенная структурная схема, которая позволяет создавать программы для обучения с максимальным использованием игрового компонента. Рассмотрены различные классы игр и их влияние на обучение и творческую деятельность. На конкретных примерах показана организация популярных в настоящее время игр на ЭВМ и даны принципы построения алгоритмов и программ таких игр.

Книга рассчитана на учащихся старших классов и ПТУ.

А $\frac{2404000000-965}{038(01)-89}$ КБ—53—18—87 ББК 32.973

ISBN 5-217-00698-6

ПРЕДИСЛОВИЕ

«Дело осложняется тем, что, спасая чьи-то минуты, ты тратишь минуты единственной жизни своей Это делает все серьезным, а не забавой »
А Вознесенский Из поэмы «О»

Феномен массового увлечения компьютерами вызвал лавинообразный поток научно-популярных публикаций, и, самое интересное, наиболее частыми словами, употребляемыми совместно со словом «ЭВМ», стали те, которые всегда однозначно ассоциировались только с интеллектом человека: знание, понятие, творчество, игра, обучение. Поэтому так однозначен и выбор названий для книг, освещающих, как правило, разные стороны сложного и многогранного объекта, имя которому компьютер.

Уже в момент сдачи этой книги в издательство в магазинах появился замечательный перевод книги Д. Мичи и Р. Джонстона «Компьютер — творец». В оригинале эта книга имеет важный подзаголовок, который более четко ориентирует читателя: машинный интеллект и знание человека.

Этот подзаголовок одновременно отражает и симбиоз двух авторских личностей — представителя науки Д. Мичи (одного из родоначальников искусственного интеллекта) и журналиста Р. Джонстона. Их методологическая концепция относительно ЭВМ, обучения и знаний отчетливо видна из следующего отрывка: *«В когнитивной психологии известна задача о мартышке, которая, находясь в комнате, пытается завладеть гроздью бананов, подвешенных на ниточке к потолку так, что ей до них не дотянуться В то же время в углу комнаты стоят садовые ножницы с ручками такой длины, что ими обезьяна вполне может перерезать нитку. Как мартышке догадаться, каким образом можно достать бананы? Представьте себе теперь не живую мартышку, а робота по имени Мартышка, и вместо бананов набор инструментов, которые ему приказано принести. Как же заложить в робота все необходимое для этого знания? Передать ему содержание учебников по физике? Хотя с помощью этой информации робот сможет получить массу самых разнообразных сведений, без которых он вполне мог бы обойтись (например, определить*

силу натяжения шнура или конечную скорость, с которой инструменты грохнутся на пол, если шнур оборвется), Мартышка так и не узнает, что нужно сделать для того, чтобы шнур оборвался, т. е. не получит никаких сведений об особенностях данной ситуации, о возможных действиях и о важных для этого случая причинно-следственных связях. Как видим, рассматривать учебник по физике в качестве исчерпывающего и действенного описания окружающего физического мира нет никаких оснований». Читая эту книгу, физически чувствуешь, как авторы «спасают чьи-то минуты, тратя свои...»

Авторы настоящей книги спасают свое время тем, что сосредоточивают внимание на материале, не нашедшем еще достаточного отражения в популярном виде.

С помощью ЭВМ не умеющие рисовать строят красивые, великолепно воспринимаемые графики и картины. Имитация космических полетов, съемка фантастических фильмов, создание новых звуковых форм также не обходятся без помощи компьютерной техники. Специфика и сложность математического мышления и воображения оказываются более доступными при образном и наглядном представлении на экране ЭВМ. Компьютерные программы точно так же, как и телевизионные, превращают безжизненную аппаратуру в нечто такое, с чем было бы приятно провести некоторое время. Программы, как и сценарии телевизионных передач, пишут, но не на обычном, а на специальном языке, составленном на базе английского и понятном компьютерам. Большинство настольных ЭВМ понимают язык под названием Бейсик (Basic). Эти и другие возможности стимуляции воображения и расширения творческих возможностей человека популярно описаны в настоящей книге.

Мы рассматриваем различные классы игр с точки зрения их влияния на обучение и творческую деятельность. В книге приведены примеры организации популярных игр на персональных ЭВМ и сформулированы принципы алгоритмического построения с помощью ЭВМ игровых и творческих компонентов. Особое внимание уделяется тому, что игровой компонент качественно меняет и интенсифицирует процесс обучения, накопления и представления знаний. Симбиоз творческого и игрового начал в ЭВМ дает каждому из нас благодарного, дружелюбного, терпеливого и знающего партнера, который полностью управляется нами и подчиняется нашим желаниям.

ВВЕДЕНИЕ

«Саламандр, предназначенных для синґапурского рынка, обучали говорить на "Basic-English". »

К Чапек Война с Саламандрами 1935 г

Использование в названии книги таких объемных понятий, как ЭВМ, ИГРА и ТВОРЧЕСТВО, отражает точку зрения авторов на то, что игровые и творческие компоненты в организации взаимодействия человека с ЭВМ приводят к качественно новой форме интеллектуального общения и, прежде всего, к неразрывности таких противоположных, казалось бы, сфер деятельности, как развитие и познание, игра и работа. Игра — это деятельность, в которой доставляет удовольствие сам процесс, безотносительно к пользе. Работа же — деятельность, при которой усилия направлены на производство чего-либо полезного самым простым и легким способом. И вот сегодня мы становимся свидетелями того, как персональная ЭВМ учит нас, помогает оформлять тексты, исправлять ошибки, решать различные задачи по физике и механике. Но самым удивительным феноменом является *ненавязчивое* возникновение психологической комфортности, увлеченности, желания познать свои и компьютерные возможности без комплекса неуверенности в себе, без страха за свое место в классе, коллективе и т. д.

Но это только начальный эффект, первая стадия знакомства, а затем ЭВМ предоставляет разнообразные возможности каждому проверить свои творческие способности в науке (например, не доказать свойство модели, а построить ее визуальный образ на экране дисплея) и почувствовать себя писателем-фантастом. А их предсказания и проекты, значительно опережая возможности своего времени, тем не менее с меньшим или бóльшим запаздыванием оказались (с трудно объяснимой высокой степенью) реализованными в нашей повседневной жизни. Например, из 108 предвидений Ж. Верна сбылось 98,

из 86 Г. Уэллса — 75, из 50 А. Беляева — 47... Хотя, справедливости ради, стоит обратить внимание читателя на то, что в большинстве своем они совпадают с предсказаниями и предвидениями В. Одоевского, опубликованными в 1840 г. в повести «4338-й год», за четверть века до первых романов Жюля Верна и за полвека до Герберта Уэллса. Приведем только две небольшие цитаты из этой повести, которые, как нам кажется, с нетрадиционной стороны подчеркивают симбиоз ЭВМ и творчества.

Первая: «*...Замечательно и то, что аэростат, локомотивы, все роды машин, независимо от прямой пользы, ими приносимой в их осуществлении, действуют на просвещение людей самим своим происхождением, ибо, во-первых, требуют от производителей и ремесленников приготовительных познаний и, во-вторых, требуют такой гимнастики для разумения, каковой вовсе не нужно для лопаты или лома...*»

Эта цитата удивительнейшим образом предвосхитила то, что сегодня становится реальностью — обучение посредством ЭВМ как ее собственным возможностям, так и различным предметным дисциплинам. Из всех «родов машин» ЭВМ, или компьютер [1], обладает действительно фантастическими способностями. ЭВМ — сложнейшая из машин — все больше и больше берет на себя такие функции, как доступное и простое обучение, психологически совместимое и комфортное взаимодействие с человеком. В дальнейшем именно с позиции реализации этих функций мы и будем рассматривать использование игровых компонентов в ЭВМ.

Вторая цитата свидетельствует о том, что в области накопления, представления и распространения знаний как результата исследований искусственного интеллекта действительность значительно опередила представления фантастов (высокоскоростные вычислительные сети, базы знаний, экспертные системы, анализ и синтез видеообразов и т. д.):

«*...изобретение книги, в которой посредством машины изменяются буквы в несколько книг.*

Машины для романов и для отечественной драмы.

---

[1] Сегодня на русском языке все чаще употребляется англоязычный аналог слова ЭВМ По-видимому, кроме всех прочих причин, это связано с более широкими возможностями употребления таких понятий, как компьютерная технология, компьютеризация общества и т д

Настанет время, когда книги будут писаться слогом телеграфических депешей; из этого обычая будут исключены разве только таблицы, карты и некоторые тезисы на листочках. Типографии будут употребляться лишь для газет и для визитных карточек; переписка заменится электрическим разговором; проживут еще романы, и то недолго — их заменит театр, учебные книги заменятся публичными лекциями. Новому труженику науки будет предстоять труд немалый: поутру облетать (тогда вместо извозчиков будут аэростаты) с десяток лекций, прочесть до двадцати газет и столько же книжек, написать на лету десяток страниц и по-настоящему поспеть в театр; но главное дело будет: отучить ум от усталости, приучить его переходить мгновенно от одного предмета к другому; изощрить его так, чтобы самая сложная операция была ему с первой минуты легкою; будет приискана математическая формула для того, чтобы в огромной книге нападать именно на ту страницу, которая нужна, и быстро расчислить, сколько затем страниц можно пропустить без изъяна».

Уже сегодня реализована возможность распределенной передачи информации: звуковой, текстовой, образной между любыми точками земного шара. Компьютер же полностью видоизменяет характер взаимодействия между абонентами. Нет никакой необходимости физически присутствовать, например, при чтении лекции или прочитывать «двадцать газет». Компьютер формирует и выдает сведения на заданную тему, тем самым «отучая ум от усталости, переходя мгновенно от одного предмета к другому».

Из анализа творчества, и не только писателей-фантастов, следует вывод, что цель творчества — предсказание и прогнозирование, при этом свободный полет воображения, как это ни странно, не отрывается от принципиально реализуемых человеком идей. Трудно только предвидеть форму, технологию и временной интервал широкого распространения в обществе предсказанных идей. Этот тезис подтверждается и метаморфозой, которая произошла с оценкой творчества таких, например, художников, как А. Родченко, В. Вазорелли, М. Эсхер, Ф. Леже и др. Особенно поучителен и интересен тот факт, что разработанные ими подходы активно используются в науке и технике.

«...В отличие от множества других художников и писателей того времени, которые, проявляя интерес к машине, удовлетворились воспеванием движения, восхищением перед машинами или их осмеянием, Леже хочет понять машину. Он отождествляет себя с нею, можно сказать, проникает внутрь ее, встраивается посреди ее шестеренок. Даже мир изображаемых им людей художник переделывает по образцу машинного мира, каким бы ужасным ни был этот образец. Он празднует установление над городом, над человеком господства индустриальной цивилизации, причем празднует, несомненно, слишком рано (ведь неизбежность этого господства и его отрицательные стороны мы обнаруживаем лишь сейчас)» — писал Гастон Диль в книге «Фернан Леже».

Как рекорд в спорте, который через некоторое время становится общедоступным, так и предвидения фантастов и творческие концепции художников приближаются к нуждам современников. Сегодня нам достаточно знаком образ индустриализации, и цель приведенной цитаты — лишний раз подтвердить потребность в более целеустремленном поиске сути гармонического симбиоза ЭВМ, ИГРЫ и ТВОРЧЕСТВА при переходе от индустриального общества к информативному.

Одним из явных признаков такого перехода является и интенсивное развитие различного рода компьютерных игр. Сейчас их популяризация находится в центре внимания газет, журналов, радио и телевидения, и, как часто бывает в таких случаях, на поверхности оказалась «пена», а истинная роль и возможности компьютерных игр остались в тени.

Да, полезно детям тренировать ловкость пальцев и глазомер на игровых автоматах, управляемых компьютером. Но мы хотим здесь рассмотреть другую сторону компьютерных игр — имитацию и тренинг интеллектуальной деятельности человека. И с этой точки зрения важно то, что все предыдущие машины являлись «усилителями» отдельных органов человека: рук (экскаватор и т. д.), ног (автомобиль и т. д.), глаз (микроскоп, радар и т. д.).

И вот впервые ЭВМ выступает в роли усилителя и внутренней интеллектуальной деятельности, и такая ее разновидность, как игра, приобретает новый оттенок. Это легко пояснить на примере статьи в журнале «Техника молодежи» № 11 за 1987 г. «Охота с компьютером в

топологической пуще». Приведем из нее маленький отрывок:

«Новая компьютерная графика извлекает из тайников топологических пущ шедевры, которые не снились художникам-абстракционистам. В свою очередь, математики получили эффективное подспорье в своих исследованиях. Так обогащается выразительность искусства и ускоряется развитие науки. Так неожиданно возникает глубинная связь между, казалось бы, несопоставимыми высшей математикой и живописью, в частности абстрактной, осуждаемой (или непонимаемой?) многими искусствоведами. Как видим, новое, интересное, неожиданное рождается не только вдохновением и пылким воображением художника, но и пером, и кистью тонких и сложных компьютерных программ».

Топология — часть абстрактной математики, доступная пониманию узкого круга специалистов, становится образно доступной на экране дисплея компьютера.

Компьютерные программы усиливают наше воображение через зрительное представление математических свойств изучаемых явлений. Но принципиально компьютер позволяет построить на дисплее образ, объективно не существующий ни под объективом камеры, ни как результат математического моделирования. Таковы муаровые изображения, отражающие случайный характер интерферирующих наложений, например разработанная авторами программа КАЛЕЙДОСКОП (SCOPE) с дробным числом симметрии. Это принципиально новый подход к использованию компьютера. Вы задаете ряд произвольных чисел, а на экране дисплея рождаются «фантастические» картины. Кавычки стоят здесь потому, что человек не в состоянии заранее вообразить и представить их. И здесь мы опять должны обратить внимание на такой значительный факт, как совпадение муаровых изображений, получаемых на ЭВМ, с образами, которые возникают у человека под воздействием наркотика. Авторы испытали на себе действие гипноза от возникающих и развивающихся на экране дисплея картин, пробуждается азарт погони за красивыми образами, которые легко порождаются изменением значений одного, двух или более параметров.

А теперь — ближе к программам и действиям.

1.
ИГРА В НАШЕЙ ЖИЗНИ

«И что мне помешает
воздвигнуть все миры,
которых пожелает
закон моей игры?»
Ф. Сологуб

Все дети любят играть. Это относится не только к людям, но и к птенцам и детенышам млекопитающих, исключая разве что пресмыкающихся. Но только для человека игра — упражнение в моделировании и имитации. Детские игрушки представляют собой материальные модели взрослого мира. Правда, вопреки тому, что думают проектировщики и изготовители игрушек или любящие родители, детям важна не окраска куклы или игрушечного автомобиля, не точность, с какой игрушка копирует реальные объекты, а степень их функциональности. Каждому ребенку известно, что игрушечный автомобиль — не настоящая машина, что кукла — не живой аналог человека, но тем не менее дети, а иногда и взрослые, по-прежнему любят играть с ними. Ведь, играя, они осваивают и познают модели поведения, и поэтому игра — это прежде всего познание мира через моделирование и имитацию его связей.

Моделирование очень активно применяется в науке. Многие законы не были бы открыты, если бы не были использованы такие модели, как небесные сферы, твердые тела, неделимые атомы, силы упругости, колебания эфира, планетарная структура атомов и многие-многие другие. Цель исследователя, ученого, инженера заключается в формулировании количественных прогнозов относительно поведения простых моделей.

Но для того чтобы модели отражали рассматриваемую сторону реальной действительности, они должны быть достаточно приближены к ней. А с другой стороны, модель должна быть достаточно простой, чтобы давать возможность понять ее количественные соотношения. И если модель хорошо согласуется с действительностью, то сфор-

мулированные с ее помощью прогнозы в значительной степени реализуются.

Поскольку удачную модель легко принять за действительность, то успешное решение некоторой задачи, быстрое достижение цели в игре рождают ошибочное представление, что модель и есть реальная действительность.

В существующих сегодня компьютерных видеоиграх человеку, сидящему за экраном дисплея, предлагается своеобразный цветной, очень занимательный мир, который может увлечь своей фантастичностью и заставить поверить в его реальность. В самом же деле этот мир — мнимый, это мир моделей и имитаций, расположенный на плоскости экрана дисплея.

В науке в прошлом также существовали модели, которые своей реальностью обманывали ученых. Геоцентрическое строение Вселенной, эфир как носитель света, электронные орбиты — все эти модели выглядели настолько убедительно, что ученые, и не только ученые, очень долго оставались их рабами.

Вселенная и даже отдельный атом обладают бесконечным числом степеней свободы. С другой стороны, число элементов ЭВМ, а также другие ресурсы и время анализа моделей ограничены и могут охватить только конечное число переменных. Следовательно, для составления прогнозов человек не может оперировать элементами самой Вселенной, которая бесконечна, он вынужден создавать ее модели, которые могут быть осмыслены мозгом или обработаны на ЭВМ.

Модели видоизменяются в процессе познания, они развиваются, их число растет, и, отбирая и модифицируя некоторые из них, мы совершенствуем нашу способность познавать окружающий мир, предвидеть события и соответствующим образом изменять окружающую нас среду. Хотя процесс поиска бесконечен, его последовательные стадии все больше и больше приближают нас к цели. И в этом продвижении в сфере познания ЭВМ начинает играть решающую роль. Почему?

Ответом могут служить слова Шерри Теркла [1]:

«...В отличие от машины обычного типа, с которой у человека устанавливается только один вид взаимоотно-

---

[1] **Шерри Теркл.** Компьютер и человек // Диалог США М Мир, 1985 № 31, с 18—26

шений, компьютер становится партнером в самых различных аспектах.

Когда за один и тот же компьютер для одной и той же работы садятся разные люди, стили их работы оказываются различными. Особенно это сказывается в программировании. Для многих программирование превращается в создание особых миров. Одни создают миры предсказуемые и, опираясь на свой опыт, вызывают в этих мирах желаемые события, чувствуя себя их повелителями. Другие удовлетворяют иные потребности, иные желания: они создают миры, все более и более сложные, готовые в любой момент вырваться из-под контроля; при этом создатели ощущают себя волшебниками и мастерами эквилибристики.

Хамелеонные свойства компьютера, т. е. его способность при программировании становиться тем или иным произведением программиста-автора, делают его идеальным средством создания множества различных произведений и через них средством самопознания. Но компьютеры — это не просто экраны, на которые проецируется личность. Они уже сделались фактором, формирующим новое поколение. Для взрослых и детей, увлеченных электронными играми, для всех, кто с помощью компьютеров манипулирует словами, информацией, зрительными образами, в особенности же для тех, кто учится программировать, компьютеры становятся фактором, формирующим их личность, их „я".

Приобретаемый в работе с компьютерами опыт становится точкой отсчета при обдумывании и обсуждении других сторон жизни. Компьютеры вызывают дискуссии о народном образовании, общественной жизни, политике и человеческой природе. В этом смысле компьютер становится „философской машиной". Он побуждает и детей философствовать на детском уровне. Компьютер создает обстановку, в которой детская мысль сталкивается со многими фундаментальными вопросами, в том числе и с вопросом „Что такое жизнь?".

В мире взрослых специалисты-компьютерщики спорят о том, станет ли компьютер когда-нибудь подлинно «искусственным интеллектом», способным мыслить самостоятельно, по-человечески. Но какого бы уровня ни достиг интеллект компьютеров в будущем, они уже сейчас побуждают детей думать и высказываться о таких понятиях,

как живое и неживое, способное сознавать и не способное».

Психологи и педагоги рассматривают усвоение и применение знаний как две стороны активного процесса обучения. Ими доказано, что школьники обладают предметно-практическим мышлением. И если учитель отвергает одну альтернативу, то ученики не примут другую до тех пор, пока не представят ее в виде зримого образа. Школьники хотят уловить, построить и вновь разобрать то, что они себе представили. Способность к абстрактному логическому мышлению формируется только в старших классах, и то не у всех, поэтому в изучении всех дисциплин модели играют такую большую роль. Все модели, используемые в науке, и модели, используемые при преподавании научных дисциплин, с той или иной степенью трудности представимы в ЭВМ. Причем в современных персональных компьютерах с их богатой графикой и цветовыми решениями такие модели более наглядны. Но главное в другом — эти модели можно исследовать, их можно трогать, поворачивать, воздействовать на них любыми средствами, которые доступны внутри компьютерного мира. А запрограммировать для компьютерного мира можно любые воздействия, которые, конечно, имеют свою формальную модель.

Одна из основных педагогических проблем заключается в том, что описание таких, например, моделей, как материальная точка, абсолютно твердое тело, несжимаемая жидкость, идеальный газ, углеродный цикл в экосфере, неизменчивость видов и т. д., проводится без установления и обоснования взаимосвязи между ними. Школьная программа по химии объясняет периодическую систему (таблицу) элементов и химические связи на основе теории о структуре атомов. Наряду с этим в школьном курсе физики корпускулярная модель газа представляется как беспорядочное движение твердых шариков. В наглядных пособиях орбита электрона изображается в виде эллипса, в центре которого расположено ядро. Таким образом, в одном учебнике электрон представлен в виде шарика, в другом — в виде волны, а в третьем — как облако вероятности.

Преподавателю понятно, что в зависимости от обстоятельств следует применять ту или иную модель. Вообще же эти модели альтернативны, а для ученика каждая из них фиксируется в статусе окончательной истины.

Отсюда следует вывод, что учить надо не слепому запоминанию уже построенных моделей, а самостоятельному моделированию.

Для обучения самостоятельному моделированию лучше всего подходит ЭВМ — инструмент, способный помочь в построении модели, ее исследовании и трансформации.

Всю информацию, которая необходима для жизни в условиях стремительных перемен (об источниках энергии, о новых видах технологии, о генной инженерии и о социальных последствиях всех явлений), запомнить заранее, на школьной скамье, невозможно, потому что наши знания являются неполными. Но с помощью ЭВМ учителя могут вооружить детей стратегией познания, которой эффективно пользуются ученые и которую можно свести к следующим приемам:

1) внимательное наблюдение за реальной действительностью и уважительное отношение к фактам;

2) умение выделить существенное и отобрать необходимые данные;

3) создание модели, которая позволяет интерпретировать изменение этих данных, имеет ограниченное число степеней свободы и достаточно проста для понимания;

4) использование этой модели для прогнозирования будущих событий;

5) экспериментальная проверка модели и изучение пределов ее применимости;

6) практическое использование модели в пределах ее применимости;

7) адаптация модели к явлениям, лежащим за пределами ее возможностей, и, в случае необходимости, разработка новой модели, что приводит к повторению цикла на другом уровне.

Для того чтобы эта стратегия была воспринята с интересом и сама «впиталась», не вызывая скуки, необходимо использовать поистине безграничные возможности персональных компьютеров.

Подчеркнем, что пункты 1—7 — не просто операции, которые необходимо заучить, а *основная цель обучения*, и этой целью является формирование специальных навыков.

Рассмотрим игровые модели, которые позволяют изучить некоторые пункты стратегии познания.

ПЕРВАЯ ИГРА

Для того чтобы показать, как большое число случайных событий может привести к некоторому четко определенному конечному состоянию, Пауль Эренфест (иностранный член-корреспондент АН СССР, физик) придумал следующую игру.

Встречаются две собаки — белая и черная. Первую из них только что вымыли в ванне, она чистая и опрятная. Вторая всю жизнь провела в грязной конуре, ее никогда не мыли, и она полна блох. Собаки радуются встрече, обнюхивают друг друга, начинают играть. Блохи тоже рады, потому что у них увеличивается «жизненное пространство», и начинают беспорядочно скакать с одной собаки на другую. Какая из собак будет сильнее чесаться через некоторое время после знакомства?

Эту ситуацию легко [illegible] графически на экране персонального компьютера [illegible] соприкасающихся кругов (две собаки). Заштрихованный [illegible] из кругов — это черная собака. Возьмем для [illegible] объяснения шесть фишек, пронумерованных [illegible] могут быть произвольной формы [illegible] гораздо меньше кругов. Эти фишки изображают шесть блох. Теперь многократно программным способом бросаем шестигранную игральную кость. Блоха [illegible] выпадает соответствующий номер, перепрыгивает [illegible] собаку.

Сначала на [illegible] больше блох, чем на черную, [illegible] собаке уменьшается. После некоторого [illegible] число блох на обеих собаках [illegible] одинаковым, а перемещение блох [illegible] — одинаково интенсивным. Чис[illegible] колеблется вокруг некоторого [illegible] равновесия. Конечное число [illegible] первоначального состояния.

Эта игра поз[illegible] распределяется с одинаков[illegible] комнатах [illegible] почему ма[illegible] смешивается с водой.

Игра различ[illegible] компьютере. Э[illegible] ционные изобра[illegible] с сиропом и вод[illegible] пронумерованных [illegible] двух

смежных комнатах. Как это конкретно реализовать, мы рассмотрим дальше. Сейчас важно подчеркнуть, что игра, которая будет тем или иным способом представлена в компьютерном мире, может научить, дать знания в доступной, комфортной, легкой — игровой — форме..

Для того чтобы показать, как игра может дать конкретные знания, скажем, по физике, рассмотрим другую игру, похожую на предыдущую по способу реализации и основной идее, но дающую другие знания.

ВТОРАЯ ИГРА

На широких ступенях лестницы сидят, например, шестеро детей. На экране персонального компьютера (ПК) с помощью графического блока нарисуем десять горизонтальных линий и между ними с помощью другого блока (датчика случайной последовательности 1—6) расположим шесть пронумерованных фигурок. Программно бросаем две шестигранные игральные кости — с помощью блока случайных чисел 1—6. Фигурка, номер которой выпал на первой игральной кости, передвигается на одну ступеньку вниз; фигурка, номер которой выпал на второй игральной кости, поднимается на одну ступеньку вверх.

Таким образом, общая подъемная сила остается постоянной, а так как ниже последней ступеньки спуститься нельзя, то бросок игральной кости, обозначающий такую команду, во внимание не принимается.

Как распределятся фигурки на лестнице после определенного числа бросков игральных костей?

Это распределение достигает некоторого равновесия, причем число фигурок уменьшается снизу вверх по принципу геометрической прогрессии. Средняя высота распределения фигурок на линиях определяется величиной первоначального подъема, т. е. суммой высот, на которые были подняты фигурки при их первоначальном распределении по линиям.

В физической модели, реализованной на ПК, эти фигурки обозначают молекулы, а программные броски игральных костей моделируют переходы энергии при случайных столкновениях.

Естественно, что и в первой, и во второй играх желательно использовать как можно большее число передвигающихся элементов.

Единственное различие между первой и второй играми заключается в условии сохранения энергии

Вторая игра иллюстрирует барометрическое распределение молекул воздуха, или, пользуясь более абстрактнымми категориями, больцмановское распределение квантов энергии в кристалле Эйнштейна

ТРЕТЬЯ ИГРА (УПРОЩЕННЫЙ ВАРИАНТ)

Нарисуем на экране с помощью блока графики шесть квадратов, обозначающих участки для охоты в джунглях. Пронумеруем участки от 1 до 6. На каждом участке живут волк или пантера, обладающие одинаковой силой. Если кто-то из них сможет неожиданно напасть на соперника и убить его, то территория жертвы достанется потомству победителя.

Поместим трех волков (темные фигурки) и трех пантер (светлые фигурки) на эти участки и бросим программно две шестигранные игральные кости. Цифра, выпавшая на первой кости, обозначает номер жертвы, которая убирается с квадрата, который она занимает. Цифра, выпавшая на второй кости (втором датчике случайных чисел от 1 до 6), обозначает победителя, а на территории жертвы должна быть нарисована фигурка победителя.

На первом этапе игры численность представителей обоих видов подвергается случайным колебаниям. Но после того как все квадраты заняты одним видом, другой вид оказывается полностью уничтоженным и уже никак не может возродиться.

Эта игра представляет собой модель самопроизвольного нарушения симметрии. Первоначальное состояние и правила игры обеспечивают полную симметрию между волками и пантерами. Однако эта экосистема характеризуется отсутствием стабильности, и предсказать, какой вид выживет, невозможно. Даже если первоначально будет пять волков и одна пантера, может случиться так, что выживут пантеры.

В отличие от первых двух игр, результат этой игры менее предсказуем И действительно, история играет определенную роль в формировании флоры и фауны данного региона. Возможно, что именно таким образом правосторонние аминокислоты были устранены в процессе биологической эволюции Теперь у всех живых существ имеются только левосторонние молекулы аминокислот

ЧЕТВЕРТАЯ ИГРА

Эта игра рассматривается под девизом поиск и может явиться основой для формирования пунктов 1—6 стратегии построения эффективных моделей явления.

Группа игроков занимает места перед экраном ПК. Один из игроков является ведущим; он играет роль «природы». Задумав определенный закон «природы», он «раздает» каждому из играющих электронные карты, которые могут представлять собой нумерованные квадраты или прямоугольники различного цвета. Раздача электронных карт из первоначально сформированной колоды осуществляется программным раздатчиком, использующим соответствующий датчик случайных чисел.

Остальные игроки выступают в роли «ученых». Каждый из них получает некоторое число электронных карт. Начинается игра. Ведущий объявляет одну карту, помещает ее в центр экрана и предлагает партнерам помещать рядом с этой исходной картой по одной своей так, чтобы были видны все предыдущие.

О загаданном законе ведущий не рассказывает «ученым». Если электронная карта, помещенная «ученым», соответствует этому закону, то ведущий говорит «правильно». Если электронная карта не соответствует задуманному закону, то он говорит «неправильно», и «ученый» должен забрать свою карту.

Тот, кто первым избавляется от своих карт, т. е. действует в соответствии с законом «природы», объявляется победителем. В следующей игре ему поручается роль «природы».

Игра представляет интерес благодаря тому, что закон известен только «природе», т. е. ведущему, поскольку он, естественно, придумывает его до начала игры. В процессе игры «ученые» внимательно следят за правильными или неправильными ходами и у них появляются предположения о характере этого закона.

По мере накопления эмпирических данных, т. е. последовательности правильных ходов, каждый «ученый» пытается использовать различные аналоги при выборе хода. Неудачный ход заставляет его вносить изменения в свою рабочую гипотезу.

Когда эксперимент, т. е. определенная карта, подтверждает его прогноз, «ученый» пытается обобщить правило. Как только игрок открывает закон, он начинает играть правильно, и число имеющихся у него карт начинает быстро уменьшаться.

Разумеется, закон, выбираемый «природой», должен быть четким и недвусмысленным. Более того, если «природа» ошибается, это не остается незамеченным, так как

в конце игры игроки обсуждают выбранный ею закон и проверяют свои ходы. Кстати, настоящая Природа никогда не мошенничает!

Эта игра увлекательна, если избранный закон относительно прост, но не тривиален. Например, закон чередования цвета карт — красная, черная, красная, черная и т. д.— прост, но слишком очевиден. Все отгадывают его очень быстро и игра теряет интерес. То же самое происходит, если закон слишком сложен и никто не может его открыть.

Эта игра имитирует научный поиск, подобно тому, как в шахматах имитируется сражение. Она знакомит молодых людей с научным подходом к исследованиям; кроме того, она позволяет выявить тех, кто способен к творческому мышлению.

Игровая тренировка при помощи персонального компьютера не является необходимым условием открытия новых законов Природы, но она пробуждает полезный для играющего интерес, подвигающий к открытиям в компьютерном мире И если научить открытию еще непознанных законов Природы практически невозможно, то научить работать с необходимой степенью совершенства на персональном компьютере можно и нужно, по мнению авторов, любого и каждого

Игровая программа обязательно несет на себе яркий отпечаток личности того, кто ее создал, и именно в этом ее основная ценность

Наиболее важной характеристикой игровой программы является ее популярность. На вопросах о том, какая программа может завоевать и удержать популярность, почему она вообще может стать или не стать популярной, мы остановимся в следующей главе

2.
ИГРА, КОМПЬЮТЕР, ЧЕЛОВЕК

« Нужен разум, чтобы знать,
что давать машине»
Норберт Винер

Когда-то компьютерная грамотность связывалась только с обучением программированию, затем появились электронные экзаменаторы, но лишь теперь, с распространением персональных компьютеров и специальных учебных программ, учащимся предлагаются увлекательные упражнения, вызывающие желание вновь и вновь обращаться к компьютеру.

Если учащийся делает ошибку, компьютер не раздражается, не нервничает, а терпеливо объясняет и, если нужно, повторяет объяснение много раз подряд. Не каждому учителю это доступно.

Компьютеризованное обучение значительно трансформирует всю систему образования. Функции высокопрофессиональных консультантов смогут выполнять экспертные системы, связанные с банками данных. Лекционные занятия со временем могут быть перенесены в дисплейные классы, так же как и практические занятия, и лишь лабораторные работы останутся пока нетронутыми.

Однако добиться того, чтобы персональный компьютер стал хорошим преподавателем,— дело чрезвычайно трудное. Пока еще существует очень много сложных вопросов во многих изучаемых дисциплинах, вопросов, которые порождают встречные и уточняющие вопросы и так далее — лавинообразный развивающийся процесс. Компьютер не может поговорить по душам с тем, кому он задает вопросы, а это так много значит. Очень важными являются вопросы нравственности, а они отсутствуют в компьютерном мире. И, естественно, там, где уроки служат формированию у учащихся мировоззрения и нравственных качеств, без учителя не обойтись.

Шестнадцатилетний разработчик программ для научных исследований пока еще не типичен для нашей страны,

и информационная техника в виде персональных компьютеров еще не вошла широко и уверенно в учебные классы системы среднего образования, но, в то время как ПК для многих учителей все еще книга за семью печатями, их ученики ведут дискуссии о преимуществах различных программных систем и обмениваются впечатлениями о различных компьютерных играх

Надо помнить, что информатика — самый юный из всех школьных предметов, а методика ее преподавания пока не располагает окончательными концепциями. Педагоги спорят о том, когда (начиная с 1-го, 2-го или 5-го класса) следует начать обучение вычислительной технике и программированию, а точнее, алгоритмизации и алгоритмическим языкам программирования. Какие игры и в каком виде следует «врезать» в программу преподавания различных предметов? Что вредно, а что полезно в богатом опыте, накопленном с помощью компьютерных видеоигр, в изобилии поступающих через гибкие диски?

Вопросов очень много, и не на все из них получены к настоящему времени ответы. Мы надеемся, что некоторые ответы сможет подсказать материал настоящей книги.

Какие идеи являются определяющими на сегодняшний день в организации взаимодействия человека с ЭВМ? Для правильной ориентации при создании учебных компьютерных видеоигр это существенный вопрос, так как учащийся должен проводить за экраном терминала много времени.

В последние годы наметилась четкая тенденция использования графических возможностей видеотерминалов для организации объектно-ориентированного взаимодействия человека с ЭВМ. Это взаимодействие осуществляется по принципу «Вы видите то, что имеете». Это означает, что Вы управляете объектами, которые расположены на плоскости экрана, не с помощью написания специальных команд, а непосредственным изменением этого объекта.

Такой подход использован в современных экранных редакторах, текстовых процессорах, интерактивных графических системах

Команды человека ассоциированы с элементарными действиями: нажатиями на клавиши, перемещениями «мыши», сдвигом рукоятки «джойстика». На экране эти действия связаны с состояниями объектов, с которыми работает человек.

Например, Вам нужно переписать информацию из одного места хранения в другое. В этом случае нет необходимости вносить команду записи информации в новое место хранения и стирать ее в старом месте для освобождения памяти, достаточно захватить курсором имя (рамку) блока, хранящего нужную информацию, и курсором же перенести его (ее) в новое место хранения, т. е. совершить как бы действительное перемещение упаковки заданной информации. Если эта информация не была еще упакована, то такими же действиями курсора ее можно упаковать.

Подобные элементы ассоциаций использованы для редактирования файлов на физическом уровне, для работы с заголовками файлов, с оглавлением (списком) файлов и т п.

Таким образом, вместо работы с командами операционной системы происходит взаимодействие человека с *миром компьютерных объектов.* Это взаимодействие реализуется с помощью принципа *непосредственного редактирования информации.*

Что же такое *компьютерный объект?* Это предмет, с которым осуществляются манипуляции в широком житейском смысле, т. е. это любой объект, который узнаваем по своему изображению на экране дисплея.

Непосредственное редактирование означает, что манипулирование предметами (компьютерными объектами) осуществляется без посредничества каких-либо дополнительных языков или программных систем. В этом случае пользователь не ведет диалога, а, нажимая на клавиши, изменяет компьютерные объекты, наблюдая эти изменения на экране дисплея (рис. 1).

При реализации принципа непосредственного редактирования необходимо помнить следующие положения.

1. Работа человека на ЭВМ состоит в целенаправленном изменении объекта, который имеет внутреннюю структуру и содержание, а также внешнее алфавитно-цифровое и (или) графическое представление. В качестве объекта могут рассматриваться: программы; модель физической системы; модель явления; база данных; реальный физический объект; математические агрегаты...

2. Процесс воздействия на объект можно назвать *редактированием,* а соответствующую программную систему, осуществляющую процесс воздействия,— *редактором.*

3. Принцип непосредственного редактирования состоит в следующем:

а) в процессе работы с редактором человек должен «находиться» в пространстве объекта (в «мире» объекта), с которым осуществляются манипуляции;

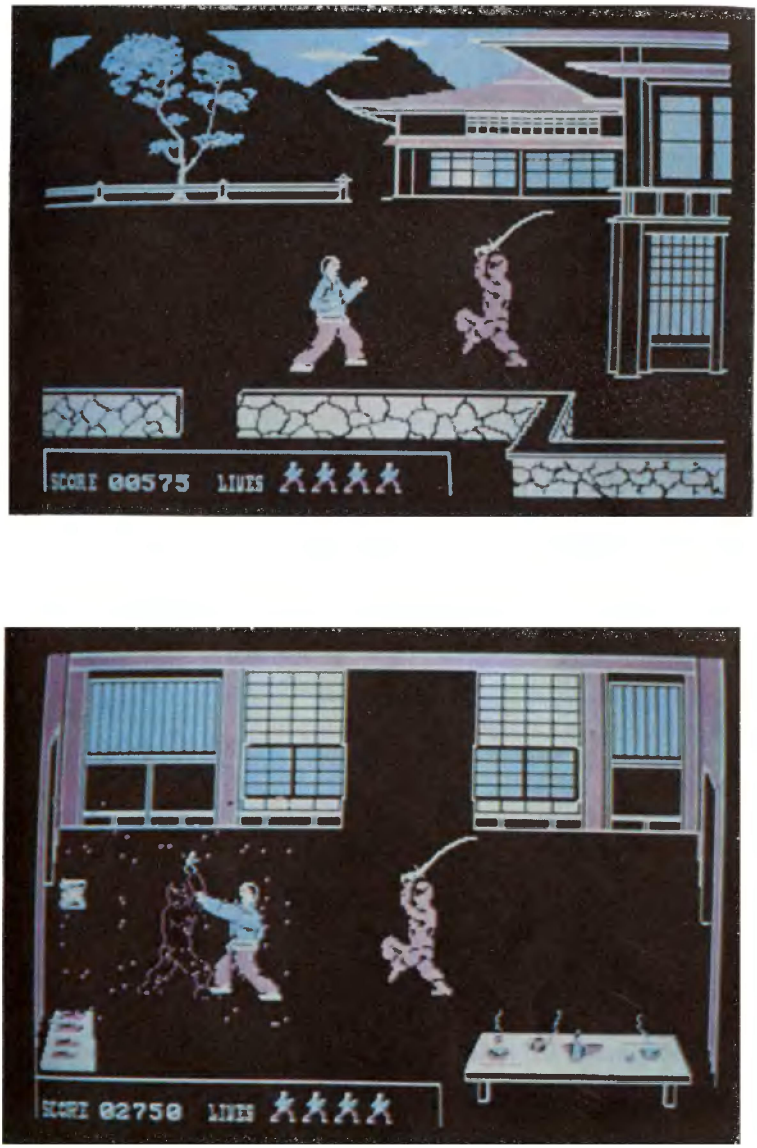


Рис. 1. Экран персональной ЭВМ: *а* — объекты компьютерного мира; *б* — работа с объектами компьютерного мира

Рис. 2. Одна из сцен компьютерного мира — положение в игре ТЕТРИС (TETRIS)

б) на экране отображаются редактируемая часть «мира» и положение в нем человека;

в) внешний вид манипуляций с объектом, которые позволяет производить редактор, должен соответствовать интуитивным представлениям человека об этих манипуляциях;

г) должны отображаться все состояния объекта.

Положение человека в «мире» должно определять большинство аргументов действий с объектом. Отображение редактора на экране должно представлять собой существенные для работающего в «мире» понятия в основном в графической форме (цветовые контрасты).

Программа РЕДАКТОР н е д о л ж н а отображать на экране элементы своего построения и понятия, связанные с конкретной реализацией «мира» и самого редактора (рис. 2).

В режиме непосредственного редактирования информации на экране отображаются состояния объектов, подвергающихся редактированию. Н е о т о б р а ж а ю т с я команды редактирования и ответы программ редактора

(реакция программы редактора) на осуществляемые операции над объектами.

Использование расположения человека в «мире» для определения аргументов команд позволяет заменить традиционную схему операций над объектом (т. е. последовательность: команда — аргумент, ..., команда — аргумент) схемой вида: аргумент — команда —...— команда...

Основным преимуществом разработки систем на основе аналогии «человек в мире объектов» и принципа непосредственного редактирования является то, что сама система в качестве посредника между человеком и объектом как бы исчезает, а вместе с ней исчезают и диалоги между человеком и системой в традиционном смысле этого слова, т. е. наблюдается эффект исчезновения системы-посредника.

Большинство тренирующих и обучающих видеоигр базируется на принципе непосредственного редактирования. Это означает, что объектами игры играющий манипулирует не с помощью набора текста управляющих команд, а непосредственно изменяя положение этих объектов в реальном масштабе времени с помощью клавиатуры, «мыши» или джойстика. Рассмотрим, например, полет на компьютерном самолете. На экране мы видим кабину самолета изнутри, как будто сидим в кресле пилота. Прямо впереди видна взлетно-посадочная полоса, ниже — панель с приборами, рукоятка управления элеронами и рулями высоты, еще ниже видны педали руля поворота. Для того чтобы начать полет, нужно включить двигатель, нажав кнопку на приборной панели, и мы делаем это, нажимая на клавиатуре соответствующую клавишу. Затем нужно увеличить или уменьшить газ и, управляя элеронами и рулем поворота, начать движение. Все эти действия являются имитацией настоящих движений летчика, ведущего самолет по взлетно-посадочной полосе. Реакция самолета на эти действия отображается на экране персонального компьютера с помощью системы непосредственного редактирования.

О создании программы тренирующей и обучающей видеоигры мы поговорим в следующей главе.

3. ПРОГРАММИРОВАНИЕ ИГРЫ

«...Но неуемный разум разложил
и этот мир, построенный на ощупь
вникающим и мерящим перстом».
М. Волошин

Довольно часто бывает так, что желание играть во все доступные компьютерные игры довольно быстро приедается и хочется самому придумать какую-нибудь особенную игру, чтобы испытать своих друзей, знакомых и даже незнакомых людей ну и, конечно, самому попробовать подняться на следующую ступень мастерства. При всем этом игра занимает все более важное место в обучении. Изучение различных предметов в игре — вот огромная сфера приложения игровых программ, дающая наибольший эффект с перспективой, направленной в будущее.

Но для того чтобы понять, какая программа может быть поистине игровой и учебной, следует обратиться немного к истории вопроса. Давайте рассмотрим структуру игровой программы, с которой работает один играющий, причем в реальном масштабе времени и без применения специальных приспособлений типа «мышек», «джойстиков», специальных клавиатур.

Прежде всего такая компьютерная игра с точки зрения этого одного играющего представляет собой некоторую, вполне определенную задачу. Как и в любой задаче, в ней есть условия, цель и средства достижения цели. Что же такое в игре решение задачи? Это некоторый процесс (например, перемещение объектов задачи, изменение их ориентации, скорости и ускорения, изменение их формы и цвета), которым управляет играющий по некоторым правилам, которые определяет сама игра. То, как проходит этот процесс (является ли он выигрывающим, ведущим к цели, или, наоборот, проигрывающим, уводящим от цели), оценивается игровой программой по установленному в ней критерию: например, подсчитывается сумма баллов, которые даются играющему за тот или иной ход, или

меняется ситуация на игровой площадке (ускоряется или замедляется темп движения объектов игровой задачи, появляются новые объекты).

БЛОКИ ИГРОВОЙ ПРОГРАММЫ

Давайте зададимся вопросом: какие основные блоки должна иметь игровая программа, чтобы быть по-настоящему игровой, а кроме того, еще и обучающей?

Анализ действующих в настоящее время и пользующихся популярностью компьютерных игр показал, что таких блоков должно быть три (рис. 3):

1) блок игровой среды;
2) блок взаимодействия с играющим;
3) блок оценки игровой ситуации.

Пожалуй, это минимальное число блоков, которое может содержать работоспособная игровая программа. Понятно, что они тесно взаимосвязаны и взаимозависимы, но в целях анализа могут быть рассмотрены как относительно самостоятельные.

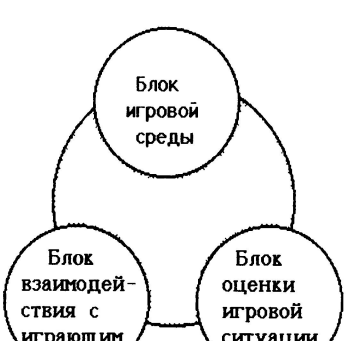


Рис. 3. Структура игровой программы

Блок игровой среды — это та сцена, тот трехмерный компьютерный мир, в котором есть все, что стоит, висит, лежит, движется, появляется и исчезает в соответствии со смыслом, законами игры. Действия на этой сцене представлены сочетаниями светящихся точек на плоском экране дисплея, однако знание законов восприятия человеком света и тени позволяет почти нулевую третью координату раздвинуть от играющего до исчезающей точки на горизонте, как это показано на рис. 4.

Так, в довольно простой игре GO-MOKU (ПЛОСКИЕ КРЕСТИКИ — НОЛИКИ) блок игровой среды определяет то поле, в котором ведется игра (квадратная сетка, обычно размером 15×15 клеток), два типа фигур (крестик и нолик) и правила расстановки фигур на игровом поле. Для очень распространенной игры PAC-MAN (УПАКОВЩИК) — это лабиринт, в котором передвигаются персонажи игры (один, которым управляете Вы, и несколько «врагов», которые стараются причинить Вам гору неприятностей), управляемые программой Эта сцена изображена на рис. 5.

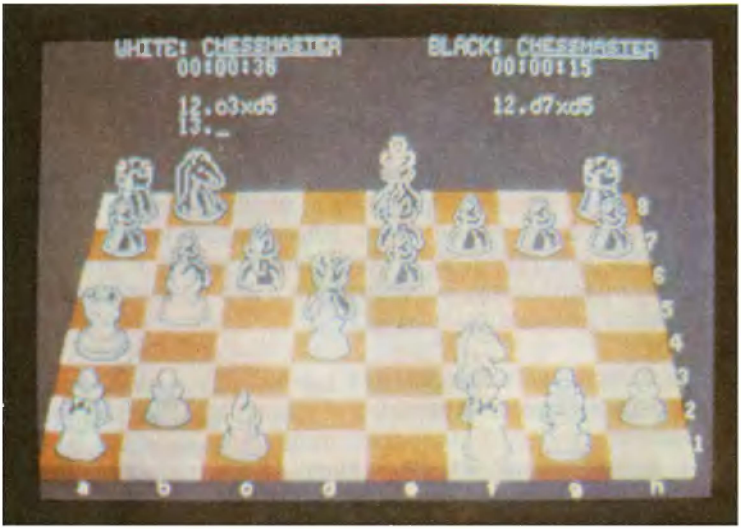


Рис. 4. Изображение сцены компьютерного мира с перспективой

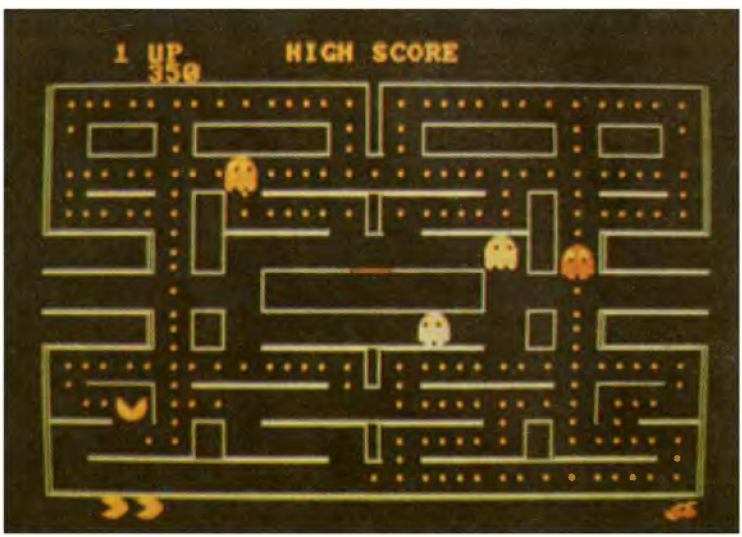


Рис. 5. Одно из положений в игре УПАКОВЩИК (PAC-MAN)

Как Вы увидите в дальнейшем, этот блок чрезвычайно важен: он задает тон всей игре и от него в основном зависит, даст игра тот решающий эффект, на который рассчитывал ее создатель, или нет.

Блок взаимодействия с играющим — это все то в программе, что позволяет играющему, т. е. Вам, изменять «все, что Вам доступно изменять», точнее, изменять то, что предусмотрено блоком игровой среды. Так как в нашей игре задействована только клавиатура персональной машины, то, естественно, мы можем менять что-либо на сцене только нажатием на определенные клавиши.

Сразу же заметим, что для динамических компьютерных игр, например PAC-MAN, где обстановка на сцене меняется в реальном масштабе времени, очень важно, в каком темпе Вы нажимаете на клавиши. Для других игр, таких как GO-MOKU, это не существенно, зато чрезвычайно важна последовательность нажатия клавиш, так как от этого сильно зависит ситуация, возникающая в игре.

Блок оценки игровой ситуации — это условия для играющего и для объектов игры на игровой сцене. В одних играх — это подсчет числа очков по той или иной системе (как в уже упоминаемой игре PAC-MAN), описание или показ начальной ситуации в игре и объяснение, описание или показ конечной ситуации на игровой сцене.

Подчеркнем еще раз, что среди рассматриваемых блоков, которые будут реализованы Вами в игровой программе, важнейшим является блок игровой среды. От него в основном зависит, насколько интересна и познавательна будет игра, что она даст играющему и Вам, ее создателю.

ЧАСТИ ПРОГРАММЫ

В программе, которая будет представлять Вашу игру, следует предусмотреть две части. Одна из них должна реализовать всю логику игры и представить эту логику в виде совокупности машинных структур данных и алгоритмов, а вторая — отобразить ситуацию на сцене и всю динамику объектов сцены на экране терминала. Таким образом, первая часть определит поле игры и правила, а вторая — форму поля, цвета команд

и число светящихся точек на экране. Труднее всего, с точки зрения программирования, представить достаточно хорошо игру на экране. Трудно не потому, что трудно программировать, а потому, что нужно ответить на множество вопросов о необходимости и достаточности изобразительных средств на экране, так как экран дисплея может быть и недогружен, когда изображений мало для успешного обучения, и перегружен, когда различных деталей и элементов слишком много. В обоих случаях игра становится неинтересной и не дает желаемого результата.

ВНУТРЕННЯЯ СТРУКТУРА ИГРОВОЙ ПРОГРАММЫ

Во всем множестве компьютерных игр реализовано небольшое число исходных идей, но они представлены очень большим количеством отображений на экране дисплея. Изображение сцены игры на экране — очень важный элемент, но сейчас мы остановимся на внутренней структуре игры, той структуре, которая реализует основную идею.

В программах компьютерных игр удается выделить три иерархических уровня, которые позволяют правильно построить схему игры и затем правильно провести программирование. Итак, в игре можно выделить *оперативный, тактический* и *стратегический* уровни, соотношение между которыми, как мы увидим в дальнейшем, во многом определяет, насколько популярной окажется игра.

Эти три уровня (рис. 6) очень часто могут быть непосредственно соотнесены с текстом игровой программы, и такая связь помогает нам определить не только логическую и программную структуру игры, но и рассматривать соотношение между чисто игровым и обучающим компонентами в различных играх.

Обычно первое, с чем мы сталкиваемся в игре,— необходимость в соответствии с правилами игры изменять объекты на игровой сцене. Нажатие на определенную клавишу вызывает вполне определенные изменения: передвижение движущихся персонажей, изменение на один шаг параметров задачи, которые отображены на экране в числовой форме, изменение на определенный угол положения фигур на сцене и т. д. Это и есть о п е р а т и в н ы й у р о в е н ь. Сюда относятся также выполнение всех действий программы между двумя последовательными нажатиями клавиш и программный опрос — было ли сде-

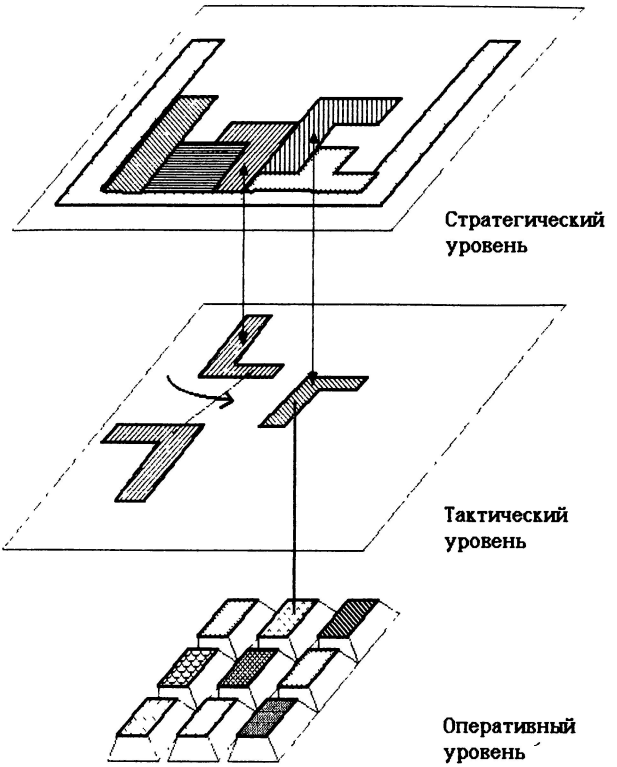


Рис. 6. Три иерархических уровня игры

лано действие играющим и, если оно было совершено, произведены ли соответствующие действия программы.

Результатом действия оперативного уровня в программе должно быть отображение всех перемещений и изменений на экране дисплея. Играющему должно быть четко видно, что реакция программы на его действия существует и что эти его действия ощутимо влияют на ситуацию в игре.

Этот уровень почти напрямую связан с блоком взаимодействия с играющим. Как уже было сказано, для игр реального времени важен темп нажатия клавиш, поэтому достаточного времени для существенного изменения картины на игровой сцене нет, а значит, блок игровой среды меняется незначительно. В этом случае темп игры высок

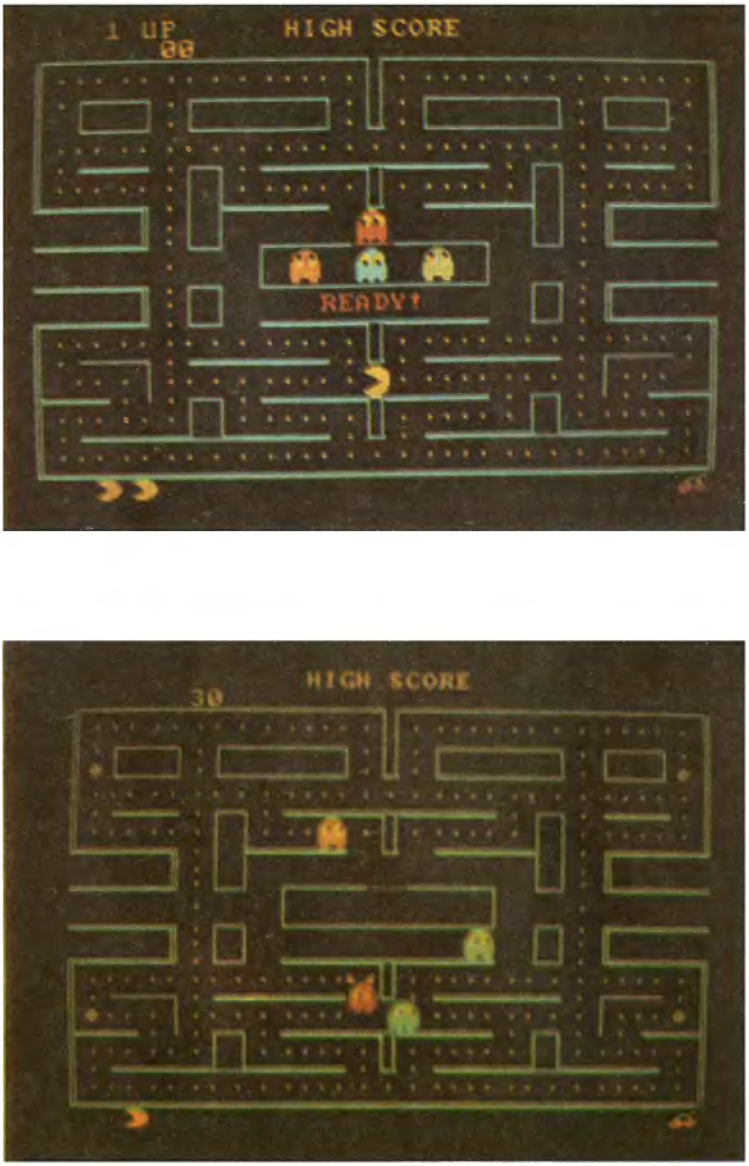


Рис. 7. Два положения в игре УПАКОВЩИК (PAC-MAN) после двух последовательных нажатий на управляющие клавиши

и изображение на сцене должно быть впечатляющим, чтобы позволить оценить ситуацию в очень короткие интервалы времени, как на рис. 7.

Теперь поднимемся на один этаж выше и рассмотрим тактический уровень игры. Действия, которые выполняет программа на этом уровне, ведут к достижению некоторой, вполне определенной локальной цели. Про этот уровень можно сказать, что он представляет собой область с расплывчатыми (размытыми) границами и в некоторых играх может даже отсутствовать. И все-таки для большинства игр этот уровень достаточно важен, так как именно на нем играющий достигает заметного улучшения (или ухудшения) положения в игре, после чего блок игровой среды существенно изменяет обстановку на игровой сцене и начинается следующий этап в игре, отличающийся по своим параметрам от предыдущего. Например, все объекты игры начинают двигаться по сцене быстрее, по другим траекториям и в других условиях.

Естественно, в программе должна быть такая часть, которая разворачивает этот игровой этап. Именно эта часть программы имеет отношение к оперативному уровню, точнее, тактический уровень программы включает в себя оперативный уровень, но часто не сводится к нему и имеет свои отдельные блоки. С помощью этих блоков более детально оценивается игровая ситуация (не только тот момент, в который нажимается клавиша), и изменения, производимые блоком игровой среды, более значительны, чем на оперативном уровне.

Интересна та игра, сложность которой изменяется в самом процессе, возрастая от этапа к этапу. Такое наращивание сложности игры происходит на тактическом уровне. Это один из критериев определения необходимости существования в разрабатываемой Вами игре тактического уровня.

Чаще всего тактический уровень можно соотнести с блоком оценки игровой ситуации.

Стратегический уровень включает в себя тактический уровень и задействует некоторое число самостоятельных блоков: для ввода на игровую сцену всех объектов, для определения, задания и визуализации их начальных параметров и вообще организации всей игровой сцены. Сюда относятся также проверка критерия окончания игры, если это игра с окончанием, а также

фиксация и визуализация результатов всей игры в целом и результатов прошлых игр.

Пожалуй, именно эта часть игры наиболее существенна: от того, как реализован блок игровой среды, который часто напрямую соотносится со стратегическим уровнем, зависит внешний интерес к игре.

ОБОБЩЕННАЯ СХЕМА ИГРОВОЙ ПРОГРАММЫ

Исходя из анализа различных компьютерных игр и выделенных структурных уровней игровых программ, соотносимых с блоками, включаемыми в тексты программ, можно предложить обобщенную схему, реализующую все, что было сказано выше. Очевидно, для структурного определения обобщенной программы компьютерной игры следует воспользоваться нотацией языка Паскаль (см. текст обобщенной игровой программы на языке Паскаль).

Вообще языковой вопрос, т. е вопрос о том, на каком языке рациональнее всего писать игровые программы, является довольно сложным, так как рабочих языков используется в настоящее время много, а выбор конкретного языка сильно зависит от класса разрабатываемых программ Язык Паскаль хорошо отражает структуры сложных систем, а Бэйсик достаточно прост и гибок. И здесь уместно обратить внимание на то, что К. Чапек не только дал в пьесе R U.R. (1920 г.) миру слово и понятие «робот» — сегодня это широко известно, но и понятие Бэйсик в романе «Война с Саламандрами» (1935 г.), совпадающее по смысловому значению с тем, которое сегодня используют специалисты по ЭВМ:

«Вместе со школьным обучением саламандр появился на свет языковой вопрос. Какой из существующих на свете языков должны прежде всего изучать саламандры? Саламандры, родом с тихоокеанских островов, естественно, говорили на „Pidgin-English", который они переняли от туземцев и матросов; многие изъяснялись по-малайски или на других местных наречиях. Саламандр, презназначенных для сингапурского рынка, приучили говорить на „Basic-English", то есть на научно-упрощенном английском языке, который обходится несколькими сотнями выражений и опускает устарелые грамматические формы, этот реформированный стандартный английский язык стали поэтому называть „саламандер-инглиш". В образцовых Ecooles Zimmermann саламандры объяснялись на языке Корнелия, однако вовсе не по националистическим соображениям, а лишь потому, что этого требует высшее образование; наоборот, в реформированных школах их обучали эсперанто, как языку удобопонятному Кроме того, в то время появились еще пять или шесть универсальных языков, которые должны были прийти на смену вавилонской путанице и дать всему миру — как людям, так и саламандрам — единый общий язык, конечно, было много споров о том, какой из этих универсальных языков наиболее целесообразен, благозвучен и универсален В конечном счете получилось, что каждая нация пропагандировала свой собственный Универсальный Язык».

```
BEGIN  Main                                  Начало основной программы
Initialize_All;
Draw_Poster;
IF_Help_Needed
    THEN Write_Help_Text;
        ++++ Strategic level +++++ '         Стратегический уровень
REPEAT                                       |
    Define_Players_Level;                    |
    Initialize_Game;                         |
    WHILE  Not_End_Of_Game  DO               |
        ++++ Tactic level +++++              |  Тактический уровень
      BEGIN                                  |  |
      Initialize_MicroGame                   |  |
       (Level_Of_Player, Current_Result)     |  |
      WHILE  Not_End_Of_MicroGame  DO        |  |
       ' +++++ Operating level +++++ '       |  |  Оперативный уровень
            BEGIN                            |  |  |
              Step;                          |  |  |
              Ask_Management;                |  |  |
              Delay ( Current_delay )        |  |  |
            END;                             |  |  |
       ' +++++ Operating level +++++ '       |  |  Оперативный уровень
      Change_Score;                          |  |
      END '                                  |  |
        ' ++++ Tactic level ++++             |  Тактический уровень
    Save_Mark;                               |
    Save_Result '                            |
UNTIL  Once_More;                            |
        +++++ Strategic level +++++ '        Стратегический уровень
END  'Main.                                  Конец основной программы
```

Если в приведенном отрывке заменить слово Саламандра на слово Компьютер, то содержание этого отрывка полностью соответствует проблемным статьям ученых в области разработки языков программирования В такой интерпретации можно считать, что École Zimmermann — школа Дейкстры или А А Ершова, а язык Корнелия — языки Си, Ада и др

В соответствии с эпиграфом к введению не только саламандры, по мнению К Чапека, должны знать Basic-English (в понятии автора — ограниченный тезаурус), а и все те, кто взаимодействует с ними

И еще одно серьезное замечание: мы приводим тексты программ на английском языке, в том числе для того, чтобы читатель воспользовался важнейшим игровым компонентом из связки ЯЗЫК — ТЕКСТ — СМЫСЛ и СМЫСЛ — ТЕКСТ — ЯЗЫК, что можно выразить словами

«обучаясь — понимать, понимая — обучаться». Замечено, что, работая на персональном компьютере с программами, имеющими развитый интерфейс на английском языке, даже не изучающие этот язык в школе значительно быстрее его осваивают.

Содержание процедур, к которым происходит обращение внутри данной программы, определяется конкретными правилами игры. Некоторые из игр, которые будут рассмотрены далее, разделены на блоки в соответствии с приведенной обобщенной схемой.

Рассмотрим процедуры, используемые на *стратегическом уровне.*

В процедуре Initialize-All инициализируются все элементы и параметры блока игровой среды, которые не будут изменяться в ходе всей игры. Такие действия, как Вы увидите, необходимы, так как многие языки программирования не позволяют иметь константы или инициализировать переменные определенной структурной сложности.

В процедуре Draw-Poster на экране дисплея визуализируются название игры и сопровождающая название справочная информация.

Логическая функция Help-Needed задает играющему вопрос, знаком ли он с правилами игры. Если нет, то процедура Write-Help-Text выдает на экран терминала сведения о том, как пользоваться игрой.

Цикл REPEAT-UNTIL реализует стратегический уровень игры. Далее в процедуре Define-Players-Level у играющего уточняется уровень сложности, на котором он хотел бы играть, и устанавливаются соответствующие параметры. В зависимости от этих параметров в процедуре Initialize-Game устанавливаются начальные значения для блока игровой среды, а именно для тех переменных, которые будут изменяться в ходе игры.

Стратегический уровень заканчивается вызовами двух процедур: Give-Mark, которая дает дифференцированную оценку проведенной игре, и Save-Result, которая служит для запоминания результатов игры в таблице рекордов. После окончания цикла с помощью логической функции Once-More играющему задается вопрос, хочет ли он сыграть еще партию.

Тактический уровень реализуется внутри процедуры WHILE Not-End-of-Game DO BEGIN...END и построен аналогично стратегическому. Сначала в процедуре Initialize-Micro-Game генерируется очередной игровой этап.

Последняя процедура тактического уровня Change-Score служит для подведения результатов для очередного локального этапа в игре. Цикл тактического уровня управляется логической функцией Not-End-of-Game, в которой проверяются все условия окончания игры.

Далее следует цикл, который реализует *оперативный уровень*. Оперативный уровень — самый внутренний цикл в обобщенной схеме. Им управляет, как и тактическим уровнем, логическая функция Not-End-of-Micro-Game, которая проверяет логические условия перехода к следующей игровой ситуации.

Процедура Step производит все локальные изменения в блоке игровой ситуации за один шаг игры.

Процедура Ask-Management проверяет, было ли в течение данного цикла управляющее воздействие, и (если оно было) вносит соответствующие изменения в ход игры.

Процедура Delay реализует задержку для создания приемлемого темпа игры. У этой процедуры есть параметр, который необходим для того, чтобы поддерживать равномерный темп игры.

Введение обобщенной схемы в анализ уже созданных игр и использование ее в качестве основы для синтеза новых игр показывают, что она позволяет выделить ряд стандартных программных блоков, из которых, как из кирпичиков или из элементов конструктора. можно строить разнообразные по сложности, красоте, выразительности и содержанию компьютерные игры, сообразуясь с потребностью, ради которой разрабатываетя игра.

4.
ТРЕНИРОВКА И ОБУЧЕНИЕ ИГРОЙ

«Игра это искра, зажигающая огонек пытливости и любознательности»
В А Сухомлинский

Иногда бывает очень полезно задаться вопросом: а стоит ли играть в эти компьютерные игры, ведь весь компьютерный мир, в который мы можем заглянуть, смотря на экран дисплея, имеет лишь аналогию с реальным миром. Он лишь похож на него, а вот законы, которые действуют внутри компьютерного мира, там, по ту сторону экрана, совсем другие, они не похожи на реальные законы природы. Например, двигаясь в скоростном автомобиле по шоссе с виражами, я попадаю в катастрофу — столкновение с другими гоночными машинами; в компьютерном мире мне сразу же, одним нажатием на клавишу, дается другая жизнь, и я опять могу продолжать гонку либо начать ее сначала. Чему может научить такая игра? Безрассудству, смертельному риску в реальной жизни, пренебрежению к жизням других людей?

Так вот, чтобы определить, насколько полезна или вредна та или иная игра (для психологического тренинга или обучения некоторому предмету, например физике), нужно уметь разделять игры на группы, состав которых зависит от некоторого критерия.

КЛАССИФИКАЦИЯ КОМПЬЮТЕРНЫХ ИГР, ОСНОВАННАЯ НА АНАЛИЗЕ ОБОБЩЕННОЙ СТРУКТУРНОЙ СХЕМЫ ИГРЫ

Возьмем в качестве критерия полезности игры преобладание в ней одного из структурных уровней, которые были рассмотрены в предыдущей главе.

Оказывается, каждый из рассмотренных структурных уровней может быть соотнесен с определенным характером

деятельности в игре. А это позволит нам, определив, какой из уровней (оперативный, тактический или стратегический) в игре преобладает, понять, что может дать такая компьютерная игра.

Анализ оперативного уровня сразу показывает, что он ближе всего к таким видам психофизиологической деятельности человека, как ощущение и психомоторика. Значит, если в игре преобладает этот уровень, то от играющего требуется очень быстро и четко нажимать на клавиши, причем в определенной последовательности, да еще необходимо внимательно следить за быстро меняющейся ситуацией в компьютерном мире — на сцене по ту сторону экрана дисплея. Обычно обстановка меняется в так называемом реальном масштабе времени, игра получается очень динамичной, она протекает за короткое время и сопровождается большим эмоциональным напряжением. Игры такого типа очень нравятся детям младшего и среднего школьного возраста. Надо сказать, что думать в такой игре просто некогда.

Если посмотреть на архитектуру такой игры, то видно, что вся нагрузка ложится на оперативный уровень, а тактический и стратегический уровни обеднены. Таким образом, эти игры почти ничем не отличаются от игр, реализованных в виде игровых автоматов.

В качестве примеров таких игр следует, наверное, назвать SPASE WAR (КОСМИЧЕСКАЯ ВОЙНА), PANGO (ЛОТОК), XONIX (КСОНИКС), SC (SPASE COMMANDERS — КОСМИЧЕСКИЕ ВОЙСКА, рис. 8). Все они рассчитаны только на тренировку скорости реакции.

Игры с преобладанием тактического уровня следует, пожалуй, отнести к играм, требующим более высокого уровня психической деятельности. Стратегическая задача в таких играх обычно довольно примитивна, но зато в них много разнообразных ситуаций, персонажей, препятствий, целей. Они интересны, но захватывающе интересны только в процессе осваивания игры. А затем, когда успешные реакции на типичные ситуации будут закреплены достаточно твердо, Вы будете играть в такую игру лишь для достижения высоких результатов.

Что же требуется от нас, чтобы успешно играть в игры, в которых основная тяжесть ложится на тактический уровень, и что тренируется такими играми? Чему мы обучаемся, играя в них?

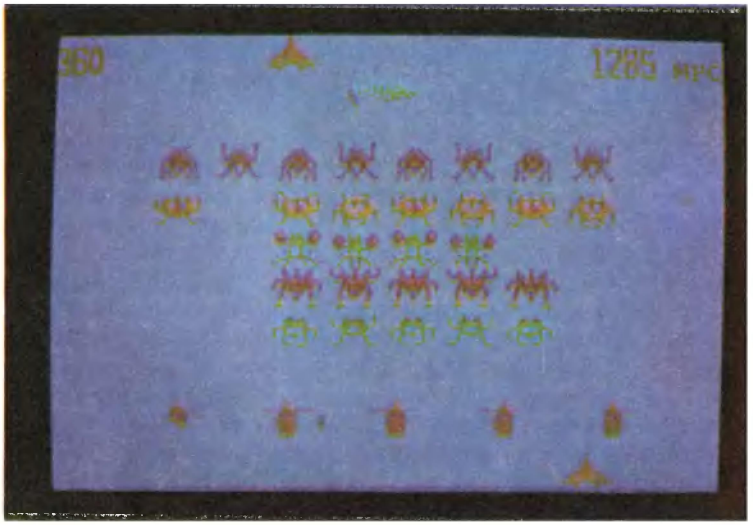


Рис. 8. Положение в игре КОСМИЧЕСКИЕ ВОЙСКА (SC)

Во-первых, нужна и тренируется эмоциональная устойчивость, во-вторых, зрительная память, в-третьих, адекватное восприятие и осознание быстроменяющейся ситуации. И, наконец, в-четвертых, необходимо и тренируется умение быстро принимать решения.

Игры такого плана представляют собой наиболее популярный и захватывающий класс игр. В качестве примеров можно привести JET (УПРАВЛЕНИЕ ИСТРЕБИТЕЛЕМ), DECATHLON (ДЕСЯТИБОРЬЕ), TETRIS (ОРИЕНТАЦИЯ И КОМБИНАЦИЯ ПЛОСКИХ ТЕЛ В ПРОСТРАНСТВЕ, рис. 9).

Давайте рассмотрим теперь игры со сложной задачей на стратегическом уровне. Такие игры, как правило, протекают в медленном темпе: на оперативном уровне на каждый ход отводится очень много времени, и тактический уровень чаще бывает сравнительно беден. Игры этого класса требуют преимущественно интеллектуальной деятельности. Для успешной игры необходимы и тренируются воображение, логическое мышление, комбинаторное мышление и т. п. Однако обычно это скучноватые игры, популярные у ограниченной аудитории. К играм

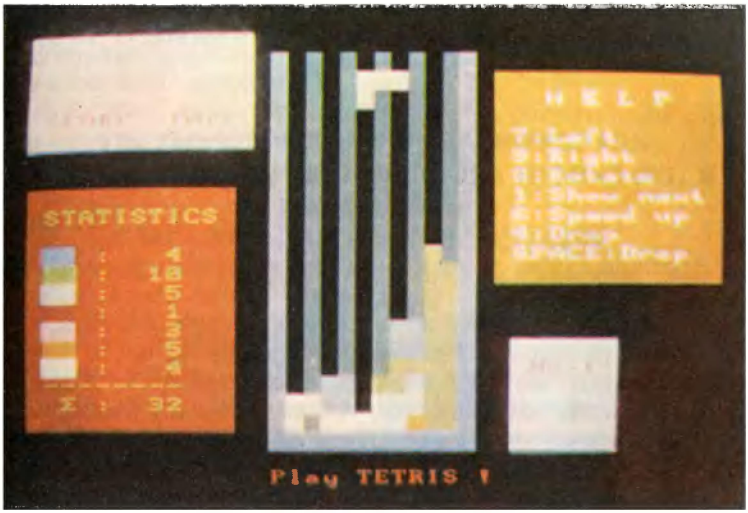


Рис. 9. Две позиции в игре ТЕТРИС (TETRIS)

этого класса можно отнести GO-MOKU (ГО-МОКУ), CHESS MASTER (ШАХМАТНЫЙ МАСТЕР), CHESS (ШАХМАТЫ).

КЛАССИФИКАЦИЯ КОМПЬЮТЕРНЫХ ИГР В ПЛОСКОСТИ ПРОБЛЕМНО-ФУНКЦИОНАЛЬНОГО НАПОЛНЕНИЯ

Прежде всего необходимо отметить социально-имитационную функцию компьютерных игр. Очень многие играют, и часто это воспринимается как успешное овладение языком программирования и навыками работы на персональных компьютерах. Однако в большинстве случаев эти успехи мнимые; мало того, те, кто не хочет учиться работать на персональных машинах, еще более укрепляются в своем нежелании.

Затем следует отметить функцию отдыха — переключение деятельности. Практика программистов показывает, что умеренное использование развлекательных видеоигр может быть полезным, оказывая разгружающий, восстанавливающий эффект.

Существенной является функция самоиспытания. Пожалуй, каждому интересно определить, на что способен в той или иной ситуации, найти границы этих способностей, понять, что ему лично нужно развивать. Некоторые игры не имеют четкого предметного содержания и непонятны специалистам и ученым, но они понятны педагогам и психологам, так как в ходе такой игры человек получает в основном информацию о самом себе, о том, как он может взаимодействовать с некоторой отвлеченной моделью мира. Например, каждый знает, что для объезда препятствия справа руль автомобиля надо поворачивать вправо, но далеко не каждый знает, успеет ли он сделать это известное и понятное ему действие в определенный ограниченный отрезок времени. При этом компьютерная игровая самодиагностика имеет массу важных преимуществ:

а) конфиденциальность — играющий может получить информацию о себе, не раскрывая ее другим людям;

б) возможность широкого социального сравнения, так как обычно игру сопровождает таблица рекордов и каждый играющий может сравнить себя с достаточно большим числом людей, выполняющих аналогичную деятельность;

в) преимущества имитации — видеоигра имитирует реальную деятельность, но ошибки в игре нельзя сравнить с ошибками в реальной действительности, которые обходятся гораздо дороже;

г) игровой характер самопознания и оперативная компенсация слабых результатов.

КЛАССИФИКАЦИЯ ИГР ПО СТЕПЕНИ И УСТОЙЧИВОСТИ ИХ ИСПОЛЬЗОВАНИЯ

Сначала нужно выделить такие специфические игровые программы, которые сами являются генераторами некоторого множества игр. Казалось бы, таких программ должно быть очень много, так Вы уже успели заметить, что вообще игровых программ существует такое количество, что только для их перечисления не хватит и толстой книги. Однако оказывается, что таких программ мало. Авторам известна только одна — WOLD PUZZLER (СЛОВЕСНЫЙ КРОССВОРД). Эта программа позволяет создать множество кроссвордов по заданной пустой схеме.

Несомненно, особую группу составляют игровые программы, с помощью которых проводится обучение некоторым разделам физики, математики, лингвистики, и не только обучение, а и тренировка, т. е. проверка знаний, умений и натренированности ума. Таких программ довольно много, и здесь мы приведем лишь их названия: UNIVERSAL PATTERNER (УНИВЕРСАЛЬНЫЙ ПОСТРОИТЕЛЬ УЗОРОВ), TRUE OR FALSE (ИСТИНА ИЛИ ЛОЖЬ), CLOCKMAN (ЧАСОВЩИК), BINOMIAL DISTRIBUTION (БИНОМИАЛЬНОЕ РАСПРЕДЕЛЕНИЕ), CALEIDOSCOPE (КАЛЕЙДОСКОП), MULTIPLICATION TRAIN (УМНОЖАЮЩИЙ ПОЕЗД), GRAPHIC GENERATOR (ГРАФИЧЕСКИЙ ГЕНЕРАТОР), ALPHABET (АЛФАВИТ), MEMORY (ПАМЯТЬ), PHYSICS (ФИЗИКА), BUILDER (СТРОИТЕЛЬ), KEYBOARD (КЛАВИАТУРА), QUADRATIC EQUATIOUS (КВАДРАТНЫЕ УРАВНЕНИЯ), MUSIC MACHINE (МУЗЫКАЛЬНАЯ МАШИНА), LINEAL REGRESSION (ЛИНЕЙНАЯ РЕГРЕССИЯ), 5 FACTORS (5 ФАКТОРОВ), CONSTELLATIONS (СОЗВЕЗДИЯ), U.K. MAP (КАРТА ВЕЛИКОБРИТАНИИ), SCROLL (ПЕРЕМОТКА), MAKING WAVES (ГЕНЕРАЦИЯ ВОЛН), CHORDS

(АККОРДЫ), SPEARMAN'S RANK (РАНГОВАЯ КОРРЕЛЯЦИЯ СПИРМАНА), MONSTER MATH (МАТЕМАТИЧЕСКОЕ ЧУДОВИЩЕ), SEMAPHORE (СЕМАФОР), WORD PROCESSOR (ОБРАБОТЧИК СЛОВ), ADDING MACHINE (СУММИРУЮЩАЯ МАШИНА), UTILITY DRAW (ПОЛЕЗНОЕ РИСОВАНИЕ).

Каждая из этих программ стоит того, чтобы ее подробно описать, но это отдельная книга.

Обратимся теперь к классификации игровых программ, рассмотренной в гл. 3.

Наиболее удобной игрой, т. е. такой, от которой не устают, к которой постоянно возвращаются, результаты которой удовлетворяют, а неудачи не приносят заметного ощущения дискомфорта, является игра, в которой нагрузка на играющего равномерно распределена по трем выделенным уровням: оперативному, тактическому, стратегическому. В качестве примера приводилась игра TETRIS (ТЕТРИС), которая заключается в плотной упаковке плоских фигур, составленных из определенного количества квадратов; упаковка осуществляется управляемым вращением и автоматическим перемещением сверху вниз на плоскости ограниченных размеров. Опыт авторов показал, что за год эксплуатации этой игры интерес к ней не уменьшился. Хотелось бы, чтобы игр с подобным распределением нагрузки по уровням было как можно больше, но их очень мало: это — PANGO (ЗАДАТЬ ЖАРУ), SNAKE (ЗМЕЯ), DIGGER (ЗОЛОТОИСКАТЕЛЬ).

Довольно много игр, для которых определяющим является оперативный уровень, на него ложится основная нагрузка. Играя в эти игры, некогда думать, нужно быстро нажимать на клавиши, отслеживая изменившуюся ситуацию. Перечислим игры такого типа: ASTEROIDS AHEAD (ВПЕРЕДИ АСТЕРОИДЫ), LASER CANNON (ЛАЗЕРНАЯ ПУШКА), ROCKET ATTACK (РАКЕТНАЯ АТАКА), SHARP SHOOTER (МЕТКИЙ СТРЕЛОК), SHOOTING RANGE (СТРЕЛЯЮЩИЙ РЯД), SKY SHOOT (СТРЕЛЬБА В НЕБО), BOMB (БОМБЕЖКА), BOMB RUN (БОМБАРДИРОВКА), AIR-SEA RESCUE (ВОЗДУШНО-МОРСКИЕ СПАСАТЕЛЬНЫЕ РАБОТЫ). BLACK HOLES (ЧЕРНЫЕ ДЫРЫ), SHIPS (КОРАБЛИ), COMPUTER COMBAT (КОМПЬЮТЕРНОЕ СРАЖЕНИЕ), ROCKET ATTACK (РАКЕТНАЯ АТАКА). Даже не анализируя содержания этих игр, а только просмотрев их названия, каждый может понять, что игры

быстры, эмоциональны, внешне зрелищно красивы, могут сильно увлечь играющего, но это начальное увлечение быстро проходит и интерес к играм теряется.

Интерес к игре поддерживается более долгое время, если нагрузка захватывает и тактический уровень (см. рис. 6. Это означает, что в общем плане игры происходят закономерные или случайные изменения ее интенсивности. Для игры TETRIS (ТЕТРИС) изменения на тактическом уровне означают увеличение скорости появления и падения плоских фигур, которые нужно успеть плотно упаковать. К таким играм относятся: PARATROOPER (ПАРАШЮТИСТЫ), CAT (КОШКА), PAC-MAN (УПАКОВЩИК), BOWLING (КАТАНИЕ ШАРОВ), KNIGHT FIGHT (РЫЦАРСКИЙ ТУРНИР), TREASURE HUNTER (КЛАДОИСКАТЕЛЬ), CLIMBER (АЛЬПИНИСТ), STAR GATES (ЗВЕЗДНЫЕ ВОРОТА), FRUIT GUARD (СТРАЖА ФРУКТОВ), STICKS AND STONES (ПАЛКИ И КАМНИ), JELLYMEN (ЧЕЛОВЕЧКИ ИЗ ЖЕЛЕ).

К особому классу относятся игры, в которых почти вся нагрузка сосредоточена на стратегическом уровне. В таких играх следует обязательно выработать определенную стратегию поведения, которая должна привести к выигрышу, и все же быть достаточно гибкой, чтобы при изменении ситуации поменять и стратегию поведения. Перечислим игры, относящиеся к этому классу: CHESS MASTER (ШАХМАТНЫЙ МАСТЕР), CHESS (ШАХМАТЫ), LARRY (ЛАРРИ), CTOCK CONTROL (УПРАВЛЕНИЕ СКЛАДСКИМ ЗАПАСОМ), CHANNEL CROSSING (ПЕРЕСЕЧЕНИЕ КАНАЛА), PUZZLE 15 (ГОЛОВОЛОМКА 15), 3-D HUNT (ТРЕХМЕРНАЯ ОХОТА), BEAN CUP (ЧАШКА С БОБАМИ), KINGDOM (КОРОЛЕВСТВО), 3-D NOUGHTSCROSSES (ТРЕХМЕРНЫЕ НОЛИКИ И КРЕСТИКИ), PONTOON (ПОНТОН).

5. КОМПЬЮТЕРНОЕ ОБУЧЕНИЕ

« Взгляни, учитель, и рассей тревоги,—
сказал я — Вот кто нам подаст совет,
когда ты сам не ведаешь дороги»
Данте Алигьери «Божественная комедия»

В настоящее время игра с персональным компьютером представляет собой основной элемент активного внедрения компьютерной грамотности. Причем игра в этом случае для нас является тем наиболее безболезненным путем преодоления психологического барьера, который отделяет школьника и студента от богатейших возможностей компьютеров.

Компьютерные видеоигры и процесс обучения с помощью компьютеров, постепенно сближаясь, неизбежно сольются и скоро окажутся единым и нерасторжимым понятием. Именно поэтому то поколение, которое начнет свою трудовую деятельность в производственной сфере, насыщенной вычислительной техникой, должно еще в детской игре с ЭВМ узнать и усвоить границы возможного и невозможного для мира ЭВМ.

ЧЕМ ХОРОШ КОМПЬЮТЕР ПРИ ОБУЧЕНИИ!

Прежде всего в компьютерном мире нет физических ограничений. Границы реальных возможностей этого мира соприкасаются с границами фантазии, но при этом в ней действуют специфические и отличающиеся от реального мира законы.

На протяжении всей своей истории человек стремился избавиться от всевозможных конфликтов. И чем больше в этом преуспевал, тем активнее искал этим конфликтам замену, в первую очередь в виде самых разных игр — от рулетки до шахмат и от пятнашек до футбола. Природа этой игровой активности связана, по-видимому, с глубинными механизмами психики.

В начале XX в был проведен математический анализ салонных стратегических игр (шахмат, бриджа, покера), что привело к появлению еще одной математической теории, получившей название теории игр. А в 70-е годы игровые постановки задач были успешно использованы лауреатом Нобелевской премии по химии М. Эйгеном для объяснения самого загадочного этапа эволюции — возникновения жизни на Земле.

Что же все-таки такое для нас игра? Ведь для того, чтобы научиться ставить и решать задачи в игре, прежде всего нужно определиться.

Игрой условимся называть набор правил (например, правил игры в шахматы), *партией* — одну из возможных реализаций этих правил, *ходом* — выбор одним из игроков какой-либо возможности (альтернативы); кроме того, ходом называют и саму выбранную альтернативу.

Теория игр предполагает, что имеется некоторое число игроков, заданы правила игры и каждый игрок принимает решение, т. е. выбирает одну из альтернатив, стремясь увеличить свой «доход».

Система игр, выполняющих функцию обучения, должна быть построена с учетом усложнения последовательности программных задач по рассматриваемому разделу знаний. Каждая игра должна быть основана на практическом применении знаний предыдущих ступеней и знания новых ступеней должны формироваться на практических примерах.

Таким образом, игра представляет собой неформальное средство воспитания, причем в настоящее время, когда компьютерных игр появляется все больше, мы обнаруживаем, что они превращаются в инструмент социального воспитания молодого поколения.

Персональный компьютер, если рассматривать его как игрушку, представляет собой лучшую игрушку на сегодняшний день, так как именно он и только он может служить универсальным посредником между реальным миром и миром учащегося, позволяя рассмотреть любые ситуации реального и даже фантастического мира. Именно потому, что посредством компьютера можно в форме видеофильма реализовать практически любые фантастические идеи о нереальных мирах, причем с отличающимися от реального мира законами, мы имеем право утверждать, что он является идеальной материальной опорой, относительно которой можно конструировать обучающие кур-

сы, и тем самым строить мосты между учащимся и законами окружающего мира.

Обучающие программы позволяют легко, в игровой форме проводить эксперименты и сравнения, строить различные аналогии, т. е. обогащать опыт знаниями об известном и неизвестном.

Между учащимся и персональным компьютером существуют связи, подобные тем, которые соединяют его с людьми и предметами его окружения. Оказывается, что часто компьютер помогает учащемуся вступать в контакт с другими учащимися и преподавателями.

Все вы хорошо знаете, что игра и обучение редко успешно сосуществуют. «Всему свое время: есть время для игры и время для работы», «Мы здесь не развлекаться собрались!» — к подобным высказываниям учащиеся привыкли давно. Некоторые преподаватели благосклонно допускают возможность дополнения учебы игрой, но в подавляющем большинстве они пока еще не осознали, что эти два процесса могут происходить одновременно. Чаще они считают, что время, потраченное на игру, сокращает время на обучение.

С другой стороны, величайшие представители педагогической мысли — от Платона до Шиллера и от Коменского до Руссо — постоянно утверждали, что игра для детей — лучший способ познания.

Так почему же существует такое сильное различие между словами, которые часто звучат, и делами, которые осуществляются при обучении? Скорее всего это происходит потому, что теоретики педагогики считают игру состоянием ума, которое должно быть присуще всему процессу обучения, а преподаватели-практики считают, что если игра и может применяться в обучении, то она должна быть обучающей.

Необходимо сразу же сказать, что определение «обучающая игра» совершенно некорректно, так как под этим подразумевается, что можно познавать, испытывая скуку, или играть, ничего не познавая. Фундаментальный анализ теории игр доказывает, что не существует хорошей увлекательной игры, которая бы ничему не учила. Точно так же нет эффективных учебных упражнений, в которых не содержался бы элемент игры. Поэтому хорошая игрушка, с которой охотно играют, всегда чему-то учит, а хорошие учебники и учебные пособия — это, в сущности, своего рода игрушки.

Обучение с использованием ЭВМ, весьма популярное в настоящее время, показывает, насколько трудно в целом провести грань между обучением и игрой, учебным материалом и игрушкой, игрушкой и так называемой обучающей игрой.

В 1985 г. Центр по изучению систем и прогрессивных технологий (CESTA) выпустил каталог обучающих программных средств, существующих на французском языке. Программное обеспечение в нем систематизировано по разделам, соответствующим различным учебным предметам (иностранные языки, французский язык, математика и т. д.); кроме того, имеется раздел „обучающих игр". При помещении той или иной программы в раздел игр составители встретились с определенными трудностями, так как не определено, какую из программ с уверенностью можно считать игрой. Многие педагоги привыкли отождествлять „игры" с упражнениями на компьютере.

Интересно, что обратная тенденция также широко распространена: многие программы, описываемые как чисто развлекательные, являются, по сути, обучающими играми.

Для большинства обучающих игр ставится задача, не пересказывая учебник, смоделировать процессы, которые не могут быть должным образом показаны книгой, и обеспечить проверку знаний.

Специалисты по информатике, являющиеся передовым эшелоном науки, считают, что игра и обучение представляют собой вполне совместимые понятия. А что думают об этом учащиеся? Мы имеем такую информацию от школьников старших классов. Сначала обучение с помощью ЭВМ привлекает их, потому что им кажется — и совершенно правильно, что это своего рода видеоигра. Но они достаточно быстро понимают свою ошибку, так как очень часто программы для так называемых обучающих компьютерных игр почти не содержат игрового компонента. Упражнения на Бэйсике, к которым в большинстве случаев сводятся программы, обычно служат цели повторения и проверки знаний, а в этом нет ничего особенно увлекательного и способного пробудить интерес. Внимание учащегося привлекает не столько конкретная программа, сколько сама «игрушка», т. е. компьютер, и лишь впоследствии может появиться интерес к собственно информатике.

Когда Вы в первый раз подходите к компьютеру и нажимаете на клавиши, то на экране появляются слова, изображения, слышатся звуки, компьютер для вас сначала — прежде всего игрушка, таинственный предмет, и, управляя им, Вы испытываете огромное удовольствие. Подобно герою научно-фантастических фильмов Вы чувствуете, что обладаете поистине неограниченными возможностями. Когда же Вы садитесь за компьютер в двадцатый раз, то в двадцать раз больше осознаете, насколько мощным средством обучения является этот прибор. Разрабатывая первые, небольшие и простые программы на самом простом языке — Бэйсик,— Вы открываете для себя возможности объективизации структуры своего мышления. Таким образом, персональный компьютер представляет собой нечто большее, чем обучающая машина или репетитор, он — устройство, вместе с которым можно думать и которое, помогая нам думать и освобождая ум, непроизвольно заставляет нас играть.

Зададимся вопросом, чем обладает обучающая игра и обладает ли она чем-либо, что заставляет нас играть и, играя, познавать. Прежде всего заметим, что отличительной чертой игр вообще является удивительное сочетание повторения и неожиданности.

Например, в игре TETRIS, в которой необходимо укладывать падающие фигурки на «дно колодца» без зазоров, непрерывное падение фигурок одновременно притупляет и напрягает внимание до тех пор, пока внезапное исчезновение всех правильно набранных «слоев» не нарушает тишину (рис 10) Это как в музыке, когда простой и продолжительный ритм создает фон, в который врывается тема То же самое можно заметить в теннисе или футболе, где монотонные обмены мячом неожиданно прерываются резким ударом или проходом.

Следовательно, обучающая игра ведет мысль играющих через легкие и равномерные процедуры вопросов и ответов, обменов и чередований, поддерживает и обостряет интерес введением новых элементов и использует созданное таким образом оживление для того, чтобы доверительно и с энтузиазмом вовлечь обучающихся в процесс, несущий с собой и неожиданность, и удовольствие, и трудность открытия.

Таким образом, получается, что искусство составителя обучающих компьютерных игр заключается в правильной дозировке повторения и неожиданности. Это очень трудно, да и к тому же не существует компьютерной игры, которая подходила бы всегда и для всех. Проявляющаяся в игре

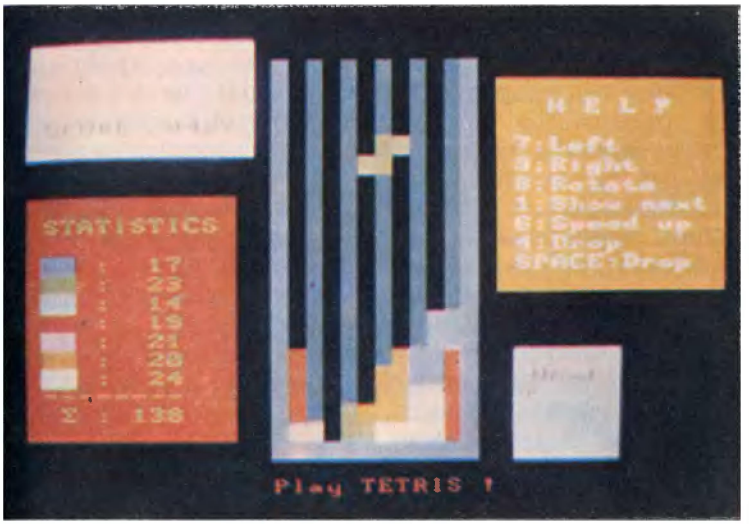


Рис. 10. Положение в игре ТЕТРИС (TETRIS), характерное для резкого изменения темпа игры

новизна резко стимулирует одних, но отталкивает других, настойчивое повторение одних ободряет, а других утомляет. И это происходит потому, что игра — дело играющего. Конечно, обучающие компьютерные игры представляют собой программы, с помощью которых мы играем и познаем, но прежде всего игра — это желание играть, причем в высшей степени индивидуальное. Это желание возникает у играющего, и никакие внешние воздействия не могут гарантировать его появления. Внешние воздействия могут способствовать возникновению образа игры. Образ игры, согласно теории Павлова об условных рефлексах,— представление и мысли об игре, в которую Вы играли несколько раз. Это одна из причин, по которым дети так любят играть в традиционные игры (например, «дочки-матери», «классы», «скакалка»), освященные радостью предшествующих поколений.

Создатели обучающих игр хорошо знают особенности традиционных игр и лучшие из обучающих компьютерных игр похожи на них, но не более того.

Многие педагоги, специализирующиеся на игре и игрушках, заблуждаются, считая, что сам по себе предмет —

игра или игрушка — может спровоцировать игру. А ведь любая деятельность и любой предмет могут стать источником игры, а значит, и познания, но ничто не гарантирует участия в ней учащегося. Каждый играет во что и когда ему нравится То же самое относится и к учению: учащегося можно заставить выполнить определенную учебную работу, но это не означает, что он в результате чему-то научился. Учению, как и игре, присущи чисто добровольные начала. Так как учение — игра, то многие преподаватели считают, что можно обучать с помощью компьютерных игр. В таком подходе содержатся две существенные ошибки. Во-первых, атрибутика игр не ведет автоматически к игре, а следовательно, и к обучению. Во-вторых, учение и преподавание — это два связанных, но не идентичных процесса. Игра, хоть она и применима в обучении, не гарантирует учения.

Игра, бесспорно, сможет расширить кругозор учащегося и тем самым способствовать его общему развитию, однако очень мало надежды заставить учащегося усвоить все предметы учебной программы на основе использования набора компьютерных игр или программ. Попытки в этом направлении предпринимаются, но прямых положительных результатов еще нет и, вероятно, не будет.

Для того чтобы обеспечить постоянную связь между игрой и учением, необходимо внести дух игры в разносторонние и многообразные программы обучения.

Именно это авторы и старались сделать с помощью того материала о программах для персональных компьютеров, который представлен в книге.

ОБЩЕЕ НАЗНАЧЕНИЕ ПРОГРАММ

Когда же появились игровые программы? Они появились в составе программного обеспечения почти всех вычислительных центров вместе с созданием и развитием больших систем, содержащих в своем составе пакеты прикладных программ. И первоначально почти всегда игровые программы предназначались для переключения программиста, проводящего у вычислительной машины очень много времени, на другой вид деятельности, т. е., другими словами, игровые программы предназначались, создавались и распространялись для отдыха. Но уже тогда,

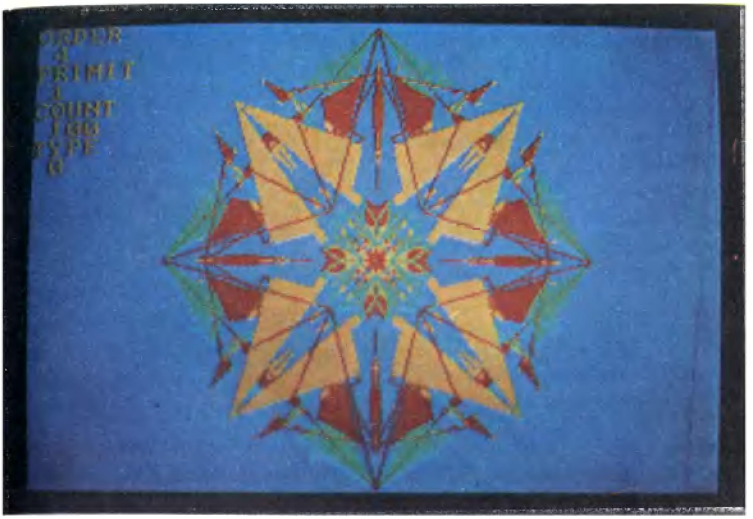
в 1950-е годы, было замечено, что успехи программистов в их профессиональной деятельности достаточно сильно зависят от выигрышей в «детских» игровых программах. Это отразилось и в фольклоре программистов. Можно, например, привести высказывание: «Как только достаточно сложная и серьезная программа создана, она становится никому не нужной».

Такая особенность компьютерных игр несколько позже была с успехом использована создателями больших программных систем для обучения пользователей программированию и введению в «мир» разработанных и внедряемых в практику программных продуктов. Для этого применяли специальные вводные программы, которые в игровой форме обучали оператора взаимодействию с программами и с ЭВМ. Они в процессе игры объясняли все используемые программным продуктом понятия, могли дать короткую и развернутую справку об установленных в программах структурах данных. Современные вводные программы не только обучают, например, языку программирования, но и заставляют в процессе игры (причем совершенно незаметно) сдать экзамен с оценкой и даже иногда назначают переэкзаменовку тем, кто недостаточно хорошо усвоил материал. Переэкзаменовка, естественно, тоже игровая. При этом игра легко снимает (ломает) барьер недоверия к своим возможностям и возможностям компьютера тоже.

Хорошо известно, что впервые персональные ЭВМ были созданы именно как игровые ЭВМ, как электронные игрушки. Правда, более половины первого тиража этих машин было раскуплено для использования непосредственно по специальностям их владельцев. Такая покупная пропорция сохраняется и по сегодняшний день. Таким образом, игровой компонент становится в настоящее время определяющим в обучении программированию и использованию ЭВМ в различных сферах деятельности.

Персональный компьютер является на сегодня уникальным инструментом, позволяющим работать с любыми объектами, которые могут быть информационно описаны. Из таких информационных объектов программистами всех рангов (от начинающих до профессионалов) создается компьютерный мир. Этот мир иногда похож на реальный, но чаще все же не сравним с ним. Вы и сами можете создавать свои собственные информационные объекты, заставлять их действовать в пределах Вашего Компьютер-

Рис. 11. Результаты работы програм



мы КАЛЕЙДОСКОП (SCOPE)

ного Государства, наблюдать за ними, обучать их действовать в этом государстве и одновременно узнавать нечто новое о реальном мире, окружающем Вас.

Как только заходит разговор о творчестве, сразу же возникает в памяти вся атрибутика искусства. И если говорить о творчестве в живописи, то персональный компьютер достаточно высокого класса, скажем, такой, как ЕС-1841, предоставляет любому человеку поистине исключительные художественные возможности. Немного экспериментов на Бэйсике — и Вы убедитесь, что программы могут творить чудеса.

ПРОГРАММА SCOPE (КАЛЕЙДОСКОП)

Обратившись к тексту программы КАЛЕЙДОСКОП, можно видеть, что оператор 20 передает управление подпрограмме 9000, в которой ожидается ввод некоторых исходых данных. Оператор 9010 подготавливает запрос ввода угла, под которым расположены зеркала модели калейдоскопа — того мысленно представляемого Вами калейдоскопа, одного из многих возможных, с которым Вы хотите немного поиграть в творчество. Далее необходимо ввести число, соответствующее форме того примитива, который будет отражаться в зеркалах калейдоскопа. Это может быть: 2 — окружность; 3 — треугольник; 4 — случайные колебания между окружностью и треугольником. Затем программа запрашивает (оператор 9050) количество различных изображений, которые Вы хотите увидеть на экране дисплея. И, наконец, последним программа запрашивает (оператор 9070) тип получающихся в калейдоскопе изображений, причем ответом на запрос программы должно быть введение одного из чисел: 1 — изображение, вписывающееся в окружность, т. е. все полученные фигуры в калейдоскопе заполняют окружность; 2 — изображение калейдоскопа вписывается в ромб; 3 — изображение вписывается в квадрат; 4 — изображение меняет форму поля заполнения.

После того как Вы ввели все необходимые исходные данные, программа начинает генерацию изображений. Электронный калейдоскоп очень эффектен. С неустанной монотонностью он генерирует исключительно разнообразные узоры, среди которых иногда встречаются довольно неожиданные «творческие удачи» программы (рис. 11).

```
10 DIM X(4), FF(4,180)                    Задание размеров рабочих
                                          массивов
15 SCREEN 1: COLOR 8,1: KEY OFF           Переход в графический режим
                                          работы и выбор палитры цветов.
20 GOSUB 9000: LOCATE 1,7: INPUT OD       Вызов п/п печати данных, ввод
                                          порядка симметрии
21 IF OD=0 THEN STOP                      (если =0 то конец работы)
22 PI=3.14159265358#: PI2=PI*2: P2I=PI/2
25 LOCATE 3,7: INPUT PRI                  Ввод  исходных
27 LOCATE 5,7: INPUT NMAX                       данных
30 TYPE=0: IF PRI=1 THEN LOCATE 7,7:
31 INPUT TY
32 GOSUB 9000                             Печать введенной информации
35 IF INT(OD)=OD THEN L=2*OD              Определение четная/нечетная
      ELSE L=180                          степень симметрии.
40 ALO=PI/OD: SA=SIN(ALO/2): CA=COS(ALO/2)
55 N=1: N1=1
56 PR=PRI: TYPE=TY                        Определение типа примитива,
                                          который необходимо рисовать,и
57 IF PR=2 THEN GOSUB 4000: GOTO 95       вызов соответствующей
58 IF PR=3 THEN GOSUB 5000: GOTO 95       подпрограммы
59 IF PR=4 THEN PR=INT(3*RND1+1: Если тип примитива любой (4), то
               GOTO 57                    выбирается произвольный
60 IF TY=0 THEN TYPE=INT(4*RND+1)         аналогично при любом типе
                                          треугольников (TYPE=0)
61 IF TYPE=1 THEN GOSUB 1000              Вызов процедур формирования
70 IF TYPE=2 THEN GOSUB 1500              треугольников
80 IF TYPE=3 THEN GOSUB 2000
90 IF TYPE=4 THEN GOSUB 2500
91 KOL=3
95 K=0
100 FOR I=1 TO KOL                        Формирование KOL симметричных
                                          фигур
130  AL2=ALO-FF(I,1)                      Определение их местоположения
150  FOR J=2 TO L                         на плоскости(определяется угол и
155     FF(I,J)=(J-1)*ALO+AL2             расстояние от начала координат
175     AL2=ALO-AL2                       каждой вершины фигуры)
180     NEXT J
190 NEXT I
192 NO=N: IF N=4 THEN N=0: NO=0
195 GOSUB 9500                            __ Вызов подпрограммы 9500 __
200 FOR I=1 TO L                          Построение L симметричных фигур
```

```
205 IF PR=2 THEN GOTO 6000          Рисование прямоугольника
206 IF PR=3 THEN GOTO 7000          Рисование круга
207 X1=X(1)*COS(FF(1,1))+160        _____ Рисование треугольника
208 Y1=X(1)*SIN(FF(1,1))+100        Определение координат вершин
209 X2=X(2)*COS(FF(2,1))+160        треугольника
210 Y2=X(2)*SIN(FF(2,1))+100
211 X3=X(3)*COS(FF(3,1))+160
213 Y3=X(3)*SIN(FF(3,1))+100
214 LINE (X1,Y1)-(X2,Y2),N0         Рисование треугольника
220 LINE (X2,Y2)-(X3,Y3),N0
230 LINE (X3,Y3)-(X1,Y1),N0
250 XS=(X1+X2+X3)/3                 определение его центра.
260 YS=(Y1+Y2+Y3)/3
265 IF K=1 THEN GOTO 280            Если возможно, то
270 PAINT(XS,YS),N0,N0              закрашивание фигуры
280 NEXT I                          Завершение цикла построения
                                    симметричных фигур
290 N=N+1: N1=N1+1                  Проверка на окончание работы по
300 IF N1>NMAX THEN                 количеству построеных фигур
      A$=INPUT$(1): GOTO 20         NMAX. Если "да", то то ожидание
                                    нажатия любой клавиши и повтор
                                    всей работы (на 20),
310 GOTO 56                         иначе 56.
1000 ' rem                          _ Подпрограмма формирования
1020 FOR I=1 TO 3                   треугольника с вершинами,
1030   F=AL0*RND                    задаваемыми случайно
1035   C=COS(F)                     (генератором случайных чисел
1037   GOSUB 9700: X(I)=X0*RND      RND)
1040   FF(I,1)=F
1052   NEXT I
1055 ' rem
1081 X1=X(1)*COS(FF(1,1)):          с 1081 по 1092 определение
                Y1=X(1)*SIN(FF(1,1)) возможности
1082 X2=X(2)*COS(FF(2,1)):          закрашивания треугольника (с
                Y2=X(2)*SIN(FF(2,1)) помощью п/п 3000)
1083 X3=X(3)*COS(FF(3,1)):
                Y3=X(3)*SIN(FF(3,1))
1084 GOSUB 3000: IF K=1 THEN RETURN
1085 X1=X(2)*COS(FF(2,1)):
                Y1=X(2)*SIN(FF(2,1))
1086 X2=X(3)*COS(FF(3,1)):
                Y2=X(3)*SIN(FF(3,1))
1087 X3=X(1)*COS(FF(1,1)):
                Y3=X(1)*SIN(FF(1,1))
```

```
1088 GOSUB 3000: IF K=1 THEN RETURN
1089 X1=X(3)*COS(FF(3,1)):
                Y1=X(3)*SIN(FF(3,1))
1090 X2=X(1)*COS(FF(1,1)):
                Y2=X(1)*SIN(FF(1,1))
1091 X3=X(2)*COS(FF(2,1)):
                Y3=X(2)*SIN(FF(2,1))
1092 GOSUB 3000: IF K=1 THEN RETURN
1094 RETURN
```

```
1500 X(1)=100*RND: FF(1,1)=0
1510 FOR I=2 TO 3
1520   F=ALO*RND: C=COS(F): FF(I,1)=F
1525   GOSUB 9700: X(I)=XO*RND
1540   NEXT I
1550 GOTO 1055
```

Подпрограмма формирования треугольника,одна из вершин которого лежит на зеркале.

```
2000 X(1)=100*RND: FF(1,1)=0
2010 FF(2,1)=ALO: C=COS(ALO)
2020 X(2)=100*RND
2030 F=ALO*RND: C=COS(F): FF(3,1)=F
2035 GOSUB 9700: X(3)=XO*RND
2050 GOTO 1055
```

Подпрограмма формирования треугольника,две вершины которого лежат на разных зеркалах.

```
2500 F=ALO/2: C=COS(F)
2501 GOSUB 9700: R=XO*RND
2503 R1=XO-R
2504 XS=R*COS(ALO/2):
                YS=R*SIN(A:O/2)
2510 IF YS<R THEN R=YS
2520 IF R1<R THEN R=R1
2530 FOR I=1 TO 3
2540   F=PI2*RND
2550   X1=XS+R*SIN(F): Y1=YS+R*COS(F)
2555   X(I)=SQR(X1*X1+Y1*Y1)
2556 IF X1=0 THEN FF(I,1)=PI
                ELSE FF(I,1)=ATN(Y1/X1)
2560   NEXT I
2570 GOTO 1055
```

Подпрограмма формирования треугольника, ни одна из вершин которого не лежит на зеркалах (одна вершина на биссектрисе угла между зеркалами, две другие произвольно)

```
3000 IF Y1=Y2 AND X1=X2 THEN D=0:
                GOTO 301
3005 D=((Y1-Y2)*X3+(X2-X1)Y3+X1
       *Y2-X2*Y1)/SQR((Y1-Y2)*
       (Y1-Y2)*(X2-X1)*(X2-X1))
3010 IF D<2 THEN K=1
3020 RETURN
```

Подпрограмма определения возможности закрашивания треугольника

(если его толщина <2 точек то закрашивание невозможно K=1)

```
4000 KOL=4: K=0
4005 F2=ALO: F=F2: GOSUB 9700:
                   H1=RND*H0
4006 H(1)=H1: FF(1,1)=F2
4007 F1=0: F=F1: GOSUB 9700:
                   H2=RND*H0
4008 H(3)=H2: FF(3,1)=F1
4010 F3=RND*(F2-F1)+F1:
                   IF H2<H1 THEN H1=H2
4015 H2=RND*H1: H(2)=H2: FF(2,1)=F3
4020 F3=RND*(F2-F1)+F1: F=F3:
                   GOSUB 9700
4025 H1=RND*(H0-H2)+H2:
                   FF(4,1)=F3: H(4)=H1
4030 H1=H(4)*COS(FF(4,1)):
       H2=H(2)*COS(FF(2,1))
4040 Y1=H(4)*SIN(FF(4,1)):
       Y2=H(2)*SIN(FF(2,1))
4050 D=SQR((H1-H2)*(H1-H2)
       +(Y1-Y2)*(Y1-Y2))
4060 IF D<2 THEN K=1: GOTO 4170
4065 H1=H(1)*COS(FF(1,1)):
       H2=H(3)*COS(FF(3,1))
4070 Y1=H(1)*SIN(FF(1,1)):
       Y2=H(3)*SIN(FF(3,1))
4080 D=SQR((H1-H2)*(H1-H2)
       +(Y1-Y2)*(Y1-Y2))
4085 IF D<2 THEN K=1
4170 RETURN
```

Подпрограмма формирования четырехугольника, две вершины которого лежат на зеркалах остальные произвольно.

```
5000 KOL=1
5010 F=ALO*RND: C=COS(F):
                   GOSUB 9700: R=H0
5013 R2=R*RND
5020 FF(1,1)=F: H(1)=R2
5030 Y1=R2*SIN(F): Y2=R2*SIN(ALO-F)
5040 RD=R-R2
5050 IF RD>R THEN RD=R2
5060 IF RD>Y1 THEN RD=Y1
5070 IF RD>Y2 THERN RD=Y2
5080 RD=RD*RND: IF RD<2 THEN K=1
5090 RETURN
```

Подпрограмма формирования круга произвольного радиуса и местоположения.

```
6000 H1=H(1)*COS(FF(1,1))+160:
```

Подпрограмма построения четырехугольника
Определение декартовых

```
      Y1=X(1)*SIN(FF(1,1))+10
6001 X2=X(2)*COS(FF(2,1))+160:
      Y2=X(2)*SIN(FF(2,1))+100
6002 X3=X(3)*COS(FF(3,1))+160:
      Y3=X(3)*SIN(FF(3,1))+100
6003 X4=X(4)*COS(FF(4,1))+160:
      Y4=X(4)*SIN(FF(4,1))+100
6005 LINE (X1,Y1)-(X2,Y2),NO
6006 LINE (X2,Y2)-(X3,Y3),NO
6007 LINE (X3,Y3)-(X4,Y4),NO
6008 LINE (X4,Y4)-(X1,Y1),NO
6010 XS= (X1+X2+X3+X4)/4
6020 YS=(Y1+Y2+Y3+Y4)/4
6030 GOTO 265
7000 XS=X(1)*COS(FF(1,1))+160
7001 YS=X(1)*SIN(FF(1,1))+100
7010 CIRCLE(XS,YS),RD,NO
7020 GOTO 265
9000 CLS
9010 LOCATE 1,1: PRINT "ORDER"
9020 LOCATE 2,1: PRINT OD
9030 LOCATE 3,1: PRINT "PRIMIT"
9040 LOCATE 4,1: PRINT PRI
9050 LOCATE 5,1: PRINT "COUNT"
9060 LOCATE 6,1: PRINT NMAX
9070 LOCATE 7,1: PRINT "TYPE"
9080 LOCATE 8,1: PRINT TY
9090 RETURN
9700 ' rem

9710 XO=100*CA/ISA*SIN(F)
              +CA*COS(F)
9720 RETURN
```

координат вершин

Построение

Определение центра четырехугольника

Подпрограмма построения круга радиуса RD, в точке (XS,YS) цвета NO

Подпрограмма печати исходных данных.

Подпрограмма определения максимально возможного расстояния от начала координат до вершины

Игра с программой заключается в том, что, запуская различные варианты узоров калейдоскопа, Вы можете на себе ощутить холодную беспристрастность электронного генератора «программных творений». Некоторые из его произведений могут Вам и не понравиться, но через некоторое время Вам придется согласиться с тем, что ряд изображений может претендовать на оценку «очень интересное изображение» и «творческая удача» программы. Однако Ваше мнение может не совпадать с мнением

других исследователей, которые собрались вместе понаблюдать за машинным творчеством. Поэтому интересно провести игровой конкурс с программой: кто, задавая те или иные исходные данные, сможет заставить программу генерировать достаточно большое число «гениальных» узоров. Во всяком случае, когда программа КАЛЕЙДОСКОП генерирует один узор за другим, никогда не повторяясь в течение довольно продолжительного времени, это производит гипнотическое действие.

Программа оперирует с четырьмя цветами, учитывая и цвет фона, на котором строится изображение. В реальном калейдоскопе можно получить изображение и большего количества цветов, достаточно только в трубочку калейдоскопа положить нужное число стекол разного цвета. Количество разных цветов у программы ограничено возможностями графики персонального компьютера IBM PC-AT, но даже те, у кого дисплей черно-белый, смогут оценить возможности электронного калейдоскопа, так как часто черно-белые изображения своей графичностью превосходят цветные по воздействию и «художественному» решению.

Следует отметить, что программа SCOPE является экспериментальной и может быть существенно доработана, т. е. сокращена по числу операторов, но она вполне работоспособна и предоставляет широкие возможности по проведению различных экспериментов.

Рассмотрим программы, которые позволяют углубиться в анализ некоторых математических агрегатов, или, если хотите, конструкций, которые при расчетах на компьютере могут выдавать алогические решения и позволяют в игровой форме проверить Вашу интуицию и способности к предсказанию.

Текст программы АТТРАКТОР (ATTRACT)

```
20 PRINT "Input initial point X":
      INPUT X0                          Ввод начальной точки
21 KEY OFF                              Очистка экрана
22 N%=0                                 Установка переменной цвета в 0
25 SCREEN 1: CLS: COLOR 8,1             Переход в графический режим,
                                        выбор палитры.
27 FOR J%=1 TO 200                      Цикл по параметру R. (на экране
28   R=2.5+J%*0.0075: X=X0              200 строк).
```

```
30  FOR I%=1 TO 200                    Холостой ход алгоритма (разгон)
40    X=R*X*(1-X)
50    NEXT I%
70  FOR I%=1 TO 300                    Собственно аттрактор
80    X=R*X*(1-X)
90    L%=X*200                         Масштабирование точки решения
100   PSET(L%,J%),1: N%=N%+1           Отображение точки решения в
                                       строке J%, цветом N%.
105   IF N%=4 THEN N%=0                Установка цвета для следующего
                                       решения (всего 4 цвета).
110   NEXT I%
120 NEXT J%
130 A$=INPUT$(1)                       После окончания работы
                                  алгоритма- ожидание нажатия любой клавиши
140 STOP
```

ПРОГРАММА ATTRACT (АТТРАКТОР)

Программа (см. текст программы АТТРАКТОР) очень проста. Она имеет ядро, которое можно описать с помощью шести условных инструкций:

$x \Leftarrow 0{,}3$
for $i \Leftarrow 1$ to 200
$x \Leftarrow r \cdot x \cdot (1-x)$
for $i \Leftarrow 1$ to 300
$x \Leftarrow r \cdot x \cdot (1-x)$
plot $(200x, 100)$

Что же осуществляет эта программа? Она решает итерационным методом уравнение $x \Leftarrow x \cdot (1-x)$. При этом исходное значение переменной сначала лучше задать равным 0,3. Затем можно будет поэкспериментировать со значением этой величины и наблюдать явную зависимость от него. Первоначально программа входит в цикл, итерирующий основное уравнение 200 раз. Это необходимо, как мы увидим в дальнейшем, для того, чтобы выйти из промежуточного неустановившегося режима. Неустановившийся режим присущ свойствам самого уравнения и не является результатом приближенных вычислений. Затем программа входит в новый цикл, итерирующий уравнение еще 300 раз и высвечивающий каждое новое значение x на экране дисплея.

Параметр 100 в условной графический инструкции plot (она реализуется на Бэйсике в программе оператором 100) принят произвольно: просто принято, что экран имеет размеры 200×200; получается, что горизонтальная координата $200x$ распределяет вычисленные и всегда находящиеся в интервале от 0 до 1 (кстати, почему?) значения x вдоль одной строки на экране, установленной на высоте 100, т. е. посередине экрана.

В зависимости от управляющего параметра R (в тексте програмы величина R определяется оператором 28) программа высветит либо одну и ту же точку на экране 300 раз, либо несколько точек меньше чем 300 раз каждую. Может также получиться, что будут высвечены 300 различных точек.

Что представляет собой эта программа? Она реализует идею «странного аттрактора». Аттрактор — это обобщение понятия равновесия, это то состояние некоторой системы, к которому она в конце концов приходит, т. е. к которому она как бы «притягивается».

Простым примером физической системы, иллюстрирующей понятие аттрактора, служит маятник Представим себе обычный движущийся маятник, который под воздействием сил трения замедляется и в конце концов останавливается В данном случае точка останова выступает в роли аттрактора, поскольку «притягивает» к себе маятник Если описать движение маятника в прямоугольной системе координат, где по одной оси откладывается угол, составляемый маятником с вертикалью, а по другой — скорость изменения этого угла, то движение маятника в такой системе будет представлять точка, движущаяся вокруг точки начала отсчета

По мере того как маятник замедляется, эта точка приближается к точке начала отсчета, где через некоторое время и останавливается Эта точка и является аттрактором (рис 12).

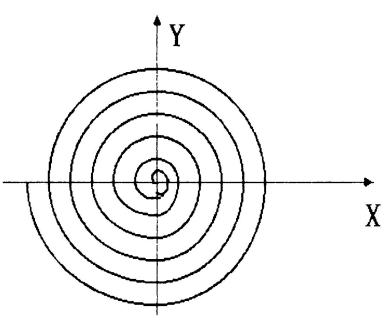


Рис. 12. Аттрактор в центре координат

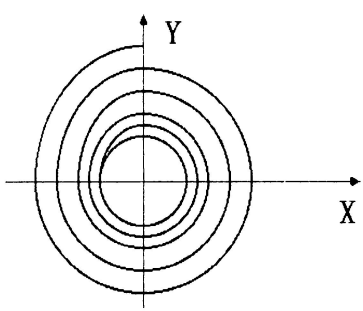


Рис. 13. Система, демонстрирующая аттрактор — окружность

Естественно, аттрактор, состоящий всего лишь из одной точки, не является странным.

Несколько странным уже можно назвать аттрактор старинных часов с маятником и грузом на длинной цепочке. Груз предназначен для подкачивания энергии к маятнику.

Если такие остановленные часы запустить слабым толчком, то маятник ведет себя подобно уже рассмотренному и аттрактор один. Но если запустить часы сильным толчком, то маятник раскачивается в сильном темпе, который затем замедляется до некоторого стабильного На фазовой диаграмме движение точки, представляющей такой маятник, выглядит в виде спирали, обвивающейся все плотнее вокруг некоторой окружности Здесь аттрактор является окружностью (рис. 13). Но и в данном случае окружность не представляется нам более странной, чем точка в случае простого затухающего маятника.

Обыкновенному маятнику можно задать хаотическое поведение, подвергнув его вертикальной вибрации. В этом случае маятник будет качаться совершенно непредсказуемым образом, без всяких признаков периодичности.

Для того чтобы приблизиться к действительно странным аттракторам, рассмотрим простую систему из трех усилителей (рис. 14), первый из которых (У1) вырабатывает на выходе сигнал x, поступающий на входы двух других усилителей. Второй усилитель (У2) выдает сигнал $1 - x$, имея на входе сигнал x На вход третьего усилителя (У3) поступают два сигнала: x и $1 - x$. На выходе он дает произведение входных сигналов $x\,(1-x)$, которое подается на вход усилителя У1. Сюда же подается управляющий сигнал R. Эту систему завершает еще один компонент (устройство) — прерыватель П1, который на короткие промежутки времени выдает на выходе тот сигнал, который в данный момент поступает ему на вход.

Описанная электронная схема на выходе порождает удивительный танец напряжения x по мере того как управляющее напряжение R

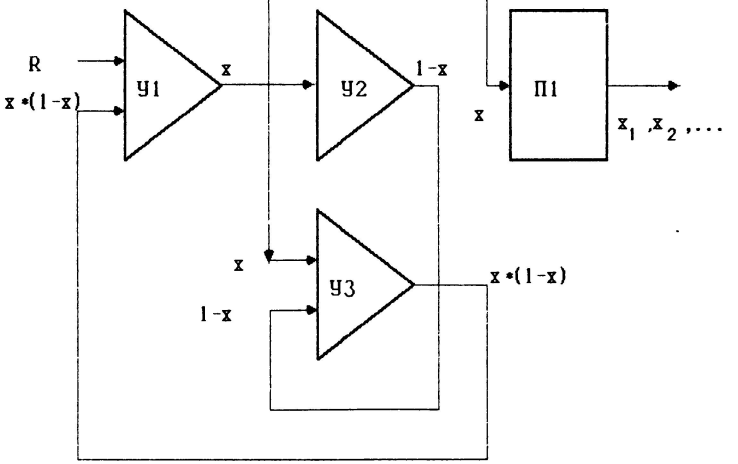


Рис. 14. Система, демонстрирующая наличие странных аттракторов

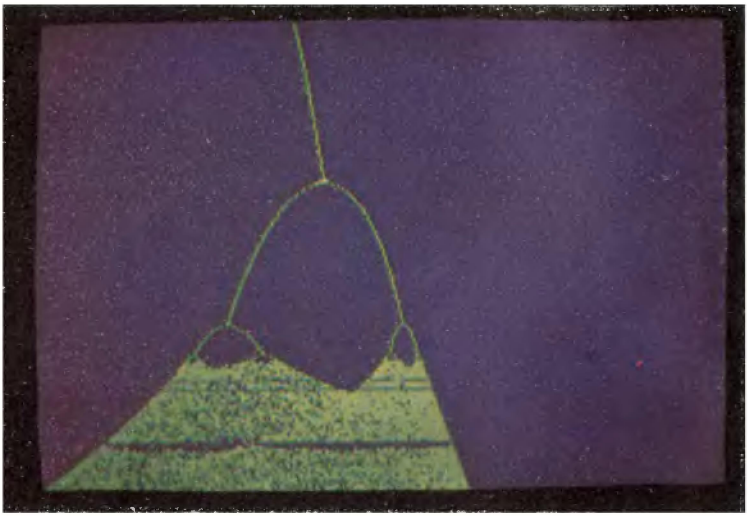


Рис. 15. Переход к хаосу в программе АТТРАКТОР (ATTRACT)

постепенно увеличивается. При $R<3$ и начальном значении $x=0$ система в течение короткого времени выдает колебательный сигнал, но затем приходит к некоторому фиксированному значению x. Это значение представляет собой аттрактор в виде одной точки. Если довести напряжение R до значения, незначительно превышающего 3, то сигнал на выходе системы начинает колебаться между двумя значениями. Аттрактор будет состоять уже из двух точек. Число точек аттрактора возрастает по мере увеличения напряжения R, а когда оно достигает значения $R\approx3{,}57$, то сигнал, вырабатываемый системой, начинает «гоняться» за простыми повторяющимися закономерностями, которые были присущи ему ранее, и теперь его поведение определяется странным аттрактором, обладающим бесконечным числом значений, в результате получается хаос.

С помощью текста программы АТТРАКТОР можно увидеть появление все большего числа точек аттрактора, а затем и вхождение в хаос, так как эта программа моделирует на персональном компьютере систему, изображенную на рис. 14.

Полная картина поведения рассмотренной системы с усилителями возникает по мере того, как программа строит ряд изображений, расположенных одно под другим (рис. 15). Изображения соответствуют ряду значений управляющего параметра R, изменяющегося от 2,9 до 4,0, что задано оператором 28 в тексте программы. Эти измене-

ния реализуются за 200 шагов от верхнего до нижнего края экрана.

Те, кто заинтересуется игрой, генерирующей подобные рисунки на персональном компьютере, наверняка смогут детально разобраться в природе лежащего в основе простого итерационного уравнения. Интересен вопрос, являются ли случайными значения параметра R, которые приводят к хаотическому поведению аттрактора.

ПРОГРАММА ATTRACT 2 (АТТРАКТОР 2)

Более сложные динамические системы описываются уравнениями Энона, названными так в честь французского математика Мишеля Энона. Так называемое отображение Энона не только описывают физические системы, подобные движущемуся астероиду или водопроводному крану, из которого капает вода, но генерируют также чудесные изображения.

Текст программы АТТРАКТОР 2 (ATTRACT 2)

```
220 PRINT "Initial point X,Y";:          Ввод параметров аттрактора
                INPUT X,Y                2-го типа
221 PRINT "increment of x,y";:
                INPUT DX,DY
223 PRINT "amount of tragectory";:
                INPUT NT
240 GOTO 70
250 XX=X*COS(A)-(Y-X*X)*SIN(A)           ------------------------------
260 Y=X*SIN(A)+(Y-X*X)*COS(A)            Подпрограмма вычисления
270 X=XX                                 аттрактора 1-го типа
272 IF ABS(X)>5 THEN X=SGN(X)*5          Формирование координат точки
274 IF ABS(Y)>5 THEN Y=SGN(Y)*5          для вывода
275 J=(X+2)*25: K=(Y+2)*25
280 RETURN
290 XX=SUN(A*Y)-COS(B*X)                 ------------------------------
300 Y=SIN(C*X)-COS(D*Y)                  Подпрограмма вычисления
310 X=XX                                 аттрактора 2-го типа
313 IF ABS(X)>5 THEN X=SGN(X)*5
315 IF ABS(Y)>5 THEN Y=SGN(Y)*5
317 J=(X+2)*25+20: K=(Y+2)*25+20
320 RETURN
```

Рассмотрим текст программы АТТРАКТОР 2. Одно из отображений Энона, которое реализовано в первой части программы АТТРАКТОР 2 (операторы 250 и 260), состоит

из пары уравнений, которые в условной алгоритмической записи выглядят следующим образом:

$$\left.\begin{aligned} x &\Leftarrow x\ \cos a - (y - x^2)\ \sin a; \\ y &\Leftarrow x\ \sin a + (y - x^2)\ \cos a. \end{aligned}\right\} \qquad (1)$$

По текущим значениям двух переменных x и y в правых частях уравнений вычисляются новые значения (также обозначаемые x и y) в левых частях.

В программе ATTRACT 2 эти два уравнения, а также два уравнения:

$$\left.\begin{aligned} x &\Leftarrow \sin\ (ay) - \cos\ (bx); \\ y &\Leftarrow \sin\ (cx) - \cos\ (dy) \end{aligned}\right\} \qquad (2)$$

используются для создания изображений порядка и хаоса, присущих широкому классу динамических систем. Основные фрагменты программы ATTRACT 2 очень похожи на аналогичный фрагмент программы ATTRACT. Различия между этими программами объясняются двумя причинами: в программе ATTRACT 2 мы имеем две итерируемые переменные вместо одной и, поскольку у нас два уравнения, необходимо ввести временную переменную xx для нового значения x, в то время пока текущее значение x используется еще во втором уравнении:

```
input x, y
for i ⇐ 1 to 1000
  xx ⇐ xcosa − (y−x^2) sina
   y ⇐ xsina+(y−x^2) cosa
   x ⇐ xx
  plot (100x, 100y)
```

К тому же система уравнений, которая описывается отображением Энона, является консервативной. Это означает, что предварительный итерационный цикл, предназначенный для исключения промежуточных знаний, как это было в программе ATRACT, в неустановившемся режиме может быть опущен.

В программе ATTRACT 2 оператор 30 (см. текст программы) запрашивает тип аттрактора, который соответствует двум системам уравнений: $T=1$ — соответствует системе (1); $T=2$ — соответствует системе (2).

При выборе в программе $T=1$ моделируется поведение системы, описываемой уравнениями (1). В этом случае

Рис. 16. Результаты работы программы АТТРАКТОР 2 (ATTRACT 2) при $A=0{,}264$

Рис. 17. Результаты работы программы АТТРАКТОР 2 (ATTRACT 2) при $A=1{,}5732$

Рис. 18. Результаты работы программы АТТРАКТОР 2 (ATTRACT 2) при коэффициентах A_1, B_1, C_1, D_1

Рис. 19. Результаты работы программы АТТРАКТОР 2 (ATTRACT 2) при коэффициентах A_2, B_2, C_2, D_2

нужно задать [1] значение a. При $a=0{,}264$ получаются аттракторы вида, представленного на рис. 16, а при $a=1{,}5732$ — на рис. 17.

Изображения на рис. 18—20 получены программой ATTRACT 2 при $T=2$, т. е. итерированием уравнений (2). В этом случае в качестве исходных нужно задавать четыре параметра: a, b, c, d.

[1] В текстах программ задаваемые параметры записываются прописными буквами: $a\rightarrow$A; $b\rightarrow$B и т. д.

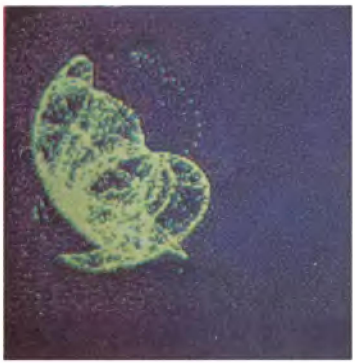


Рис. 20. Результаты работы программы АТТРАКТОР 2 (ATTRACT 2) при коэффициентах A_3, B_3, C_3, D_3

Вам интересно будет самим поиграть с программой, пытаясь получить изображения такие же, как на рисунках.

Для тех, у кого это не получается, подскажем, что изображение на рис. 18 получено по программе ATTRACT 2 при $a=2{,}01$; $b=-2{,}53$; $c=1{,}61$; $d=-0{,}33$. При $a=-2{,}7$; $b=-0{,}9$; $c=-0{,}86$ и $d=-2{,}2$ получено изображение, представленное на рис. 19, а при $a=-2{,}24$; $b=0{,}43$; $c=-0{,}65$ и $d=-2{,}43$ — на рис. 20.

Вы можете также изобрести свои собственные итерационные формулы и провести с ними вычислительные эксперименты. Персональный компьютер предоставляет для этого замечательные возможности.

Компьютер может генерировать затейливые узоры по Вашей программе. Образцы, приведенные на рис. 21 и 22, получены с использованием двух нескольких отличающихся друг от друга методов.

Программа основана на использовании формулы окружности $R=\sqrt{x^2+y^2}$, но результат работы этой программы показывает, что окружности сливаются и дают в результате довольно устойчивые изображения квадратов.

ПРОГРАММА COVER 3 (УЗОРЫ)

Б. Мартин из Астонского университета в Бирмингеме предложил для построения узоров две формулы, которые и реализованы в программе COVER 3 (УЗОРЫ). Здесь реализованы следующие формулы (см. текст программы УЗОРЫ):

$$x \Leftarrow y - \text{sign}\ (x)\ [\text{abs}\ (bx-c)];$$

$$y \Leftarrow a-x.$$

```
20 SCREEN 1: CLS: KEY OFF: COLOR 8,1     Установка графического режима
22 GOSUB 125                             Вызов подпрограммы печати
                                         исходных данных
25 N=0: N1%=0: LOCATE 1,3:
                INPUT NMAX
30 LOCATE 3,3: INPUT SIZE                Ввод исходных данных
40 LOCATE 5,3: INPUT A: LOCATE 7,3:
                INPUT B
45 LOCATE 9,3: INPUT C
50 XMIN%=70: XMAX%=319: YMIN%=0
51 YMAX=199:
      XMID%=(XMAX%-XMIN%)/2              Установка поля рисунка
52 XDIF%=XMAX%-XMIN%
53 YDIF%=YMAX%-YMIN%
54 YMID%=(YMAX%-YMIN%)/2
55 XDS%=XDIF%/250:
                YDS%=YDIF%/200
60 X=0: Y=0: CLS: GOSUB 125              Вызов подпрограммы печати
                                         введенных данных
62 LINE (XMIN%,YMIN)-
                (XMAX%,YMAX%),2,B        Рисование рамки
67 X1%=XMID%+X*XDS%*SIZE+75              Определение координат точки,
                                         выводимой на поле рисунка
68 Y1%=YMID%+Y*YDS%*SIZE+75
69 IF X1%<XMIN% OR X1%>XMAX% OR          Определение возможности вывода
      Y1%<YMIN% OR Y1%>YMAX%             точки (попадания в поле
                THEN GOTO 72             рисунка).Если нет то 72.
71 PSET(X1%,Y1%),N1%                     Рисование точки (X1%,Y1%)
                                         цветом N1% .
72 IF N=NMAX THEN GOSUB 125:             Определение окончания работы.
                GOTO 105                 "ДА" - печать данных (п/п 125) и
                                         переход на 105
75 XX=Y-SGN(X)*SQR(ABS(B*X-C))           Вычисление следующей точки.
80 YY=A-X
83 N=N+1
85 X=XX: Y=YY: N1%=N1%+1
90 N1%=N1%-INT(N1%/4)*4                  Изменение цвета точки (всего 4)
100 GOTO 67
105 LOCATE 15,1: PRINT "OK ?":
                A$=INPUT$(1)             Ожидание нажатия клавиши
106 LOCATE 15,1: PRINT "      ":         Стирание надписи, подготовка к
                N=0                      новой работе Вызов п/п печати
107 GOSUB 125: GOTO 25                    данных и переход на 25
```

```
125 LOCATE 1,1: PRINT "N=":
                LOCATE 2,1: PRINT NMAX
126 LOCATE 4,1: PRINT "S=":
                LOCATE 4,1: PRINT SIZE
127 LOCATE 6,1: PRINT "A=":
                LOCATE 6,1: PRINT A
128 LOCATE 8,1: PRINT "B=":
                LOCATE 8,1: PRINT B
129 LOCATE 10,1: PRINT "C=":  .
                LOCATE 10,1: PRINT C
130 RETURN
```

Подпрограмма печати исходных данных

Функция sign x принимает значение $+1$ или -1 в зависимости от того, положителен или отрицателен аргумент. Функция abs (u) дает абсолютное значение выражения u.

Изображения могут изменяться довольно значительно в зависимости от выбираемых значений a, b и c — числовых параметров в формуле (рис. 21).

Используемые формулы, как и ранее, записаны в сокращенной математико-алгоритмической форме. Напомним, что при такой записи предполагается, что каждый раз новые значения x и y вычисляются в правой части формул и полученный результат присваивается x и y в левой части, после чего эти значения вновь подставляются в правую часть для вычисления следующих x и y.

Скачки в значениях x и y начинаются с точки $x=0$, $y=0$, как это задано в программе, т. е. в условном начале координат. И следующая точка может появиться, скажем, справа сверху, а точка за ней — слева снизу.

Так как точки на экран дисплея выводятся очень быстро, то возникает ощущение миниатюрного электронного дождя, падающего на экран, на нем появляются сотни, а затем и тысячи точек. Вскоре начинает вырисовываться узор. Константы, используемые в формулах, показаны на рис. 21.

Не менее интересные узоры (рис. 22) дает и использование формул

$$x \Leftarrow y - \sin x;$$

$$y \Leftarrow a - x.$$

В этих формулах требуется указать значение лишь одного параметра a.

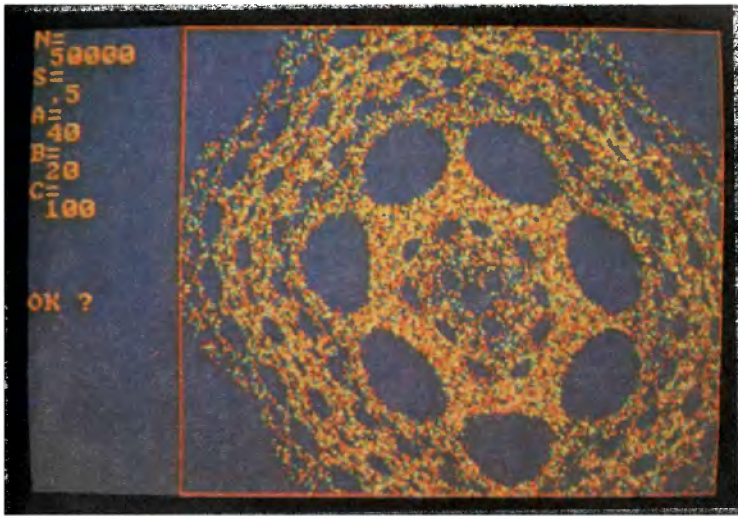


Рис. 21. Результаты работы программы УЗОРЫ (COVER 3)

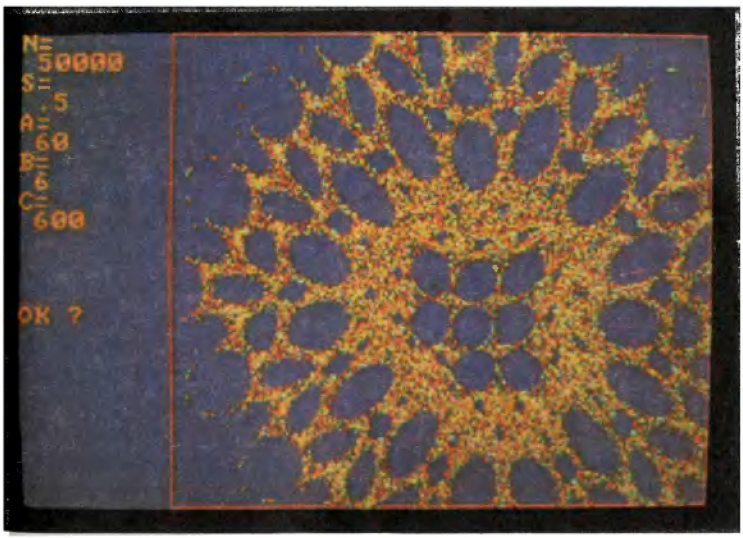


Рис. 22. Результаты работы второго варианта программы УЗОРЫ (COVER 3)

Игра состоит в том, чтобы, изменяя значения параметра, получить узор с заданным заранее характером рисунка и определенным числом осей симметрии.

Программа запрашивает число точек N, которые будут выводиться на экран, размер изображения S (представляющий собой коэффициент, определяющий масштаб) и значения коэффициентов A, B и C (операторы программы 125—130).

Значения параметров, при которых получены изображения, выведены на экран дисплея (см. рис. 21, 22).

Давайте рассмотрим программу, которая интересна тем, что позволяет наблюдать за поведением «компьютерных человечков». По мнению психологов, эти человечки, которых мы увидим как квадратики, потому что смотрим на них сверху, ведут себя подозрительно похоже на нас. Но вообще-то ничего удивительного в такой похожести нет, так как в основу этой программы положены результаты работ ученых-психологов.

ПРОГРАММА PARTY (ГОСТИ)

Во-первых, в компьютерном мире создается комната, которая для нас на экране выглядит как большой прямоугольник. В этой комнате есть стол, он тоже выглядит как прямоугольник, заполненный буквами T. В созданную комнату можно пригласить до двадцати «компьютерных человечков» — гостей, их имена обозначены первыми двадцатью символами (буквами) латинского алфавита. Каждому человечку нужно задать желаемую «идеальную» дистанцию как относительно стола (предполагается, что на столе что-то есть), так и относительно каждой другой персоны, которая приглашена в эту комнату. Те, кто был оставлен Вами на диете, предпочитают держаться на расстоянии полутора-двух метров от стола, в то время как другие чувствуют себя очень неуютно, если расстояние до стола больше полуметра. Вы можете проверить сами, что, закодировав вероятные взаимоотношения между гостями в компьютерной комнате, можно при помощи компьютерной модели предсказать, как будет развиваться общение между приглашенными в комнату человечками.

Программа работает, рассматривая каждого приглашенного по очереди: она «мысленно» перемещает человечка на каждый из прилегающих восьми квадратов. В

```
10 DIM FAM$(20), ROOM(20,21),
   EH(20),H(20), Y(20), H1(20), Y1(20)
20 FAM$(1)="A": FAM$(2)="B":
                  FAM$(3)="C"
30 FAM$(4)="D": FAM$(5)="E":
                  FAM$(6)="F"
40 FAM$(7)="G": FAM$(8)="H":
                  FAM$(9)="I"
50 FAM$(10)="J": FAM$(11)="K":
                  FAM$(12)="L"
60 FAM$(13)="M": FAM$(14)="N":
                  FAM$(15)="O"
70 FAM$(16)="P": FAM$(17)="R":
                  FAM$(18)="S"
80 FAM$(19)="U": FAM$(20)="W"
90 PRINT "amount of guests";:
                  INPUT NN
91 FOR I=1 TO NN
92   EH(I)=0
93   NEHT I
95 FAM$(NN+1)="T"
100 FOR I=1 TO NN
110  FOR J=1 TO NN
120     ROOM(I,J)=2
130     NEHT J
135  ROOM(I,I)=0: ROOM(I,NN+1)=1
140  NEHT I
141 GOSUB 800
142 KEY OFF: SCREEN 1:
       COLOR 1,2: CLS
143 LINE (0,0)-(319,199),1,B
145 GOSUB 340
146 IF N>NN THEN GOTO 150

147 IF EH(N)=0 THEN H(N)=20:
       Y(N)=1: EH(N)=1: N=N+1
150 FOR I=1 TO NN
160  PRSUM=1000000!

165 HH(I)=H(I): YY=Y(I)

170  FOR DH=-1 TO 1
```

Задание рабочих массивов

Указание идентификаторов гостей

от A до Ш

Ввод количества гостей

Исходное формирование матрицы присутствия гостей в комнате

Указание идентификатора "стол"

Задание стандартной матрицы комфортных расстояний

Вызов подпрограммы 800

Переход в графический режим, выбор палитры

Рисование поля комнаты

Вызов подпрограммы 340

Определение все ли гости собрались

Ввод очередного гостя

Цикл формирования расположения гостей на очередном шаге алгоритма

Определение местоположения гостя

Циклы просмотра соседей

```
180    FOR DY=-1 TO 1                          выбранного гостя
190            SUM=0
200            X=X(I)+DX                       Определение местоположения
210            Y=Y(I)+DY                       соседних мест (куда можно
                                               переместиться)
220            IF X<W1+1 OR                    Анализ возможности перемещения
       X>W2-1 OR Y<W3+1                        в эту точку (ограничения размера
       OR Y>W4+1 THEN GOTO 290                 комнаты,
230            IF (Y>T1-1 AND                  расположение стола)
       Y<T2+1) AND (X>T3-1
       AND X<T4+1) THEN GOTO 290
240        FOR J=1 TO NN                       Цикл подсчета оценки
                                               комфортности в данной точке
241    IF EX(J)=0 THEN GOTO 270                (без учета еще отсутствующих
                                               гостей
245    IF ROOM(I,J)=-1 THEN GOTO 270           и гостей "с улицы")
250    D2=(X-X(J))*(X-X(J))+(Y-Y(J))*          Подсчет расстояния от выбранной
       (Y-Y(J)): D=SQR(D2)                     точки дорассматриваемого (J-го)
                                               гостя.
260    SUM=SUM+ABS(D-ROOM(I,J))                Подсчет дискомфорта от данного
                                               гостя в данной точке комнаты.
270            NEXT J                          Переход к следующему гостю.
275    GOSUB 500                               Вызов подпрограммы 500
276 SUM=SUM+ABS(D-ROOM(I,NN+1))                Учет дискомфорта от удаления
                                               от стола
280    IF SUM<PRSUM THEN XX=X:                 Сравнение данной оценки
       YY=Y: PRSUM=SUM                         дискомфорта с ранее подсчитаной,
                                               выбор наименьшей и запоминание
                                               этой точки (если оценка меньше)
290            NEXT DY
300    NEXT DX
305 X1(I)=XX> Y1(I)=YY                         Перемещение гостя в точку с
                                               наименьшей оценкой дискомфота.
310 NEXT I                                     Выбор следующего гостя.
320 GOSUB 340                                  Вызов подпрограммы 340.
325 A$=INKEY$> IF A$<>"" THEN STOP             При нажатии любой клавиши
                                               работа заканчивается
330 GOTO 146                                   Переход к следующему шагу
                                               алгоритма
340 FOR I=1 TO NN
345 IF EX(I)=0 THEN GOTO 360                   Подпрограмма рисования
                                               расположения гостей в комнате
350 LOCATE Y(I)<X(I): PRINT " ."               (вначале стирается старое
360 NEXT I                                     расположение, а
```

```
370 FOR I=1 TO NN                          затем рисуется новое). Гости не
375  IF EX(I)=0 THEN GOTO 390              вошедшие не рассматриваются.
380  LOCATE Y1(I),X1(I): PRINT FAM$(I)     Рисование идентификатора гостя
381  Y(I)=Y1(I): X(I)=X1(I)
390 NEXT I
400 FOR I=T1 TO T2
410  FOR J=T3 TO T4
420     LOCATE I,J> PRINT "T"              Рисование "стола"
430     NEXT J
440 NEXT I
450 RETURN
500 IF X<T3 THEN GOTO 60                   Подпрограмма вычисления
                                           расстояния от точки до стола по
510 IF X>T4 THEN GOTO 700                  оси Y (T1,T2) и по оси X (T3,T4)
520 IF Y<T1 THEN D=T1-Y: RETURN
530 D=Y-T2: RETURN
600 D2=(T3-X)*(T3-X)
610 IF Y<T1 THEN D2=D2+(Y-T1)*(Y-T1):
611         D=SQR(D2): RETURN
620 D=T3-X: RETURN
700 D2=(X-T4)*(X-T4)
710 IF Y<T1 THEN D2=D2+(Y-T1)*(Y-T1):
711         D=SQR(D2): RETURN
720 IF Y>T2 THEN D2=D2+(T2-Y)*(T2-Y):
721         D=SQR(D2): RETURN
730 D=X-T4: RETURN
800 PRINT "Other distance";:               Подпрограмма корректировки
        INPUT A$                           матрицы комфортных расстояний.
801 IF A$="n" OR A$="N"                    Если корректироватьне надо (N
        THEN GOTO 890                      или n), то 890.
805 N1=0: N2=0
810 PRINT "Comfortable distance for";      Ввод идентификаторов гостей
820 INPUT F1$,F2$                          или гость и стол, между которыми
                                           необходимо задать новое
                                           расстояние.
840 FOR I=1 TO NN+1
850  IF FAM$(I)=F1$ THEN N1=I
860  IF FAM$(I)=F2$ THEN N2=I
880 NEXT I
885 IF N1=0 OR N2=0 THEN GOTO 890          Если введены идентификаторы не
                                           участвующие, конец работы (890).
886 INPUT ROOM(N1,N2): GOTO 805            Ввод расстояния и на 805.
890 RETURN
```

каждом положении программа вычисляет суммарное ощущение дискомфорта, испытываемое данным человечком. Тот из соседних квадратов, в котором сумма неприятных ощущений минимальна, выбирается в качестве нового положения гостя.

Структурно программа довольно проста (см. текст программы ГОСТИ). Динамика перемещений человечков задается в ней следующими условными операторами:

```
ПОВТОРИТЬ
    ДЛЯ i ⇐ До 20
        ПЕРЕМЕСТИТЬ i-ГО ГОСТЯ
    ВЫВЕСТИ НА ЭКРАН
ДО НАЖАТИЯ ПРОБЕЛА
```

Перемещение *i*-го гостя представляет собой более сложную операцию, поскольку его новая позиция будет зависеть от расположения всех других человечков с учетом тех расстояний, на которых данный человечек предпочитает находиться от остальных. Именно поэтому программа должна содержать таблицу, в каждом элементе которой дано расстояние (в условных единицах), идеальное для человечка, указанного номером строки, по отношению к гостю, указанному номером столбца:

	A	B	C	D	E	F	G	H		T
A	0	15	7	2	6	9	4	12		1
B	8	0	6	6	3	4	2	8		2
C	11	4	0	5	12	2	8	1		1
D	6	9	3	0	10	7	13	5		0 5
E	3	10	5	7	0	1	7	14		1
F	7	8	14	10	4	0	2	9		2
	.			.	.	.	.	.	. .	
T	.	.			.	.			.	0

В последнем столбце таблицы указывается идеальное для каждого человечка расстояние до стола.

Такой набор данных должен быть оформлен в программе либо как файл данных, либо как набор операторов DATA. В двух массивах X и Y хранятся текущие координаты определенных Вами человечков. Во время любого вычислительного цикла *j*-й человек занимает положение с координатами X и Y. Сумма неприятных ощущений вычисляется следующим образом:

```
PRSUM ⇐ SUM
    SUM ⇐ 0
    for j ⇐ 1 to 20
    D2 ⇐ [x (i) − x(j)] ² + [y(i) − y(j)] ²
    D2 ⇐ SQRT (D2)
    SUM ⇐ SUM + ABS [D2 − ROOM (i, j)]
```

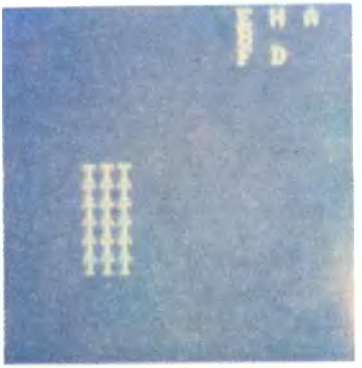
Очень интересно поэкспериментировать с различным числом приглашенных в комнату человечков и различными массивами идеальных социальных расстояний. Некоторые строки массива можно заполнять с помощью блока генератора случайных чисел, принадлежащих определенному диапазону, например от 1 до 25. В этом случае Вы получите модели приглашенных «с улицы» человечков, для которых массив идеальных социальных расстояний заранее не известен.

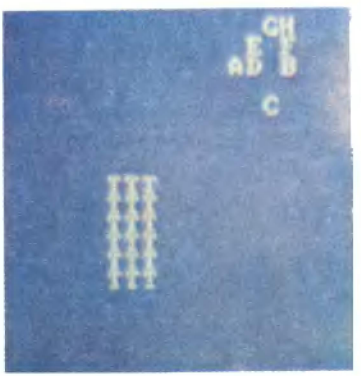
Некоторые из проигранных ситуаций показаны на рис. 23—25.

А теперь перед нами программа, генерирующая необыкновенные узоры. В основе этой программы лежит идея самовоспроизводящегося клеточного автомата Э. Фредкина. Оказывается, этот очень простой алгоритмический клеточный автомат может порождать удивительные узоры, возникающие при тонких вариациях самовоспроизведения.

Что же представляет собой клеточный автомат Фредкина? Представьте себе бесконечную на плоскости (двумерную) решетку из квадратных клеток. В каждый данный момент компьютерного времени каждая клетка может находиться в одном из двух возможных для нее состояний: она может быть либо живой, либо мертвой. Если клетка жива, то она высвечивается на экране программным блоком «отображение», если мертва, то она не высвечивается (сливается с фоном).

Вся система клеток управляется воображаемыми компьютерными часами, при тактовых сигналах которых могут меняться состояния клеток. Первая разновидность клеточного автомата: судьба каждой клетки определяется четырьмя окружающими ее соседними клетками, как показано на рис. 26, *а*. Второй вариант — это когда судьба клетки определяется восемью окружающими ее соседними клетками (рис. 26, *б*).





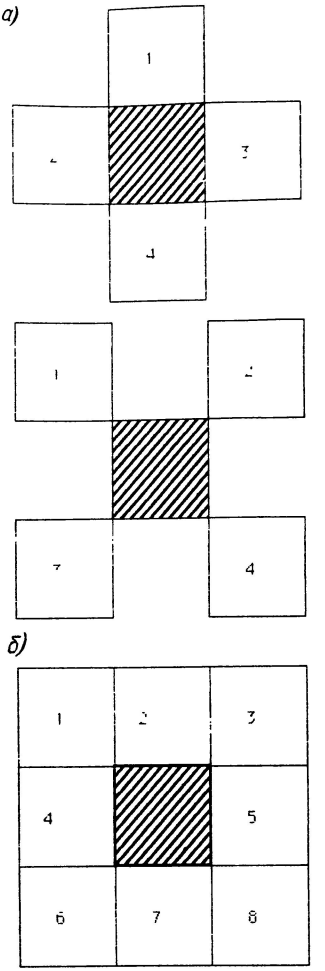


Рис. 26. Разновидности клеточных автоматов

Рис. 23. Исходное положение в программе ГОСТИ (PARTY)
Рис. 24. Промежуточные результаты работы программы ГОСТИ (PARTY)
Рис. 25. Изображение устойчивого конечного положения в игре ГОСТИ (PARTY)

Общий закон для каждой клетки таков: если число живых соседей на одном такте часов является четным, то на следующем такте данная клетка будет мертвой, независимо от ее предыдущего состояния; если же число живых соседей на одном такте часов оказывается нечетным, то на следующем такте клетка будет живой. Это правило применяется одновременно ко всем клеткам решетки.

Исходную конфигурацию можно либо задавать с помощью блока генератора случайных чисел, либо создавать самостоятельно симметричную структуру при помощи блока задания начальных значений.

После ввода исходной конфигурации для каждой клетки квадрата вычисляется ее состояние, причем клетки, находящиеся за границами квадрата, считаются мертвыми, так как они сливаются с фоном и не видны.

ПРОГРАММА LIFE (ЖИЗНЬ)

Рассмотрим текст программы, которая названа ЖИЗНЬ (LIFE). С помощью оператора 5 программа запрашивает размер поля, в котором будет наблюдаться развитие компьютерной жизни, точнее, размер поля в программе выбран стандартным, а в зависимости от числа клеток в квадрате меняются размеры клеток (рис. 27).

Рис. 27. Результаты работы программы ЖИЗНЬ (LIFE)

```
1 DIM A(60,60),B(60,60),C(24,21,$(3)      Указание рабочих массивов
2 LLK=1: LLK0=51: PRINT "type":           Выяснение типа анализируемой
                  INPUT T                 окрестности
3 IF T=1 THEN L=4              - 1-й тип  - крестообразная окрестность
4 IF T=2 THEN L=8              - 2-й тип  - полная окрестность 8 элементов
5 PRINT "SIZE": INPUT N:                  Определение кол-ва клеток в поле
      SIZE=INT(160/N): XS=80
6 YS=20: PRINT "Role of LIFE":            Определение типа правила
                  INPUT RL                "жизни"
8 PRINT "Type (random, regular)":         Определение начальной
                  INPUT I                 конфигурации клеток поля
9 IF I=0 THEN GOTO 140                    При случайной - на 140
10 PRINT "First point": GOTO 50           при не случайной - 50
11 KEY OFF: CLS: SCREEN 1: COLOR 8,1:     Перенод в графический режим
                  X1=XS+N*3:              и выбор палитры
12 Y1=YS+N*3
13 LINE (XS-1,YS-1)-(XS+N*SIZE+1,         Рисование поля
                  YS+N*SIZE+11,3,BF
14 C(1,1)=-1: C(2,1)=0: C(3,1)=1:         ____________________________
                  C(4,1)=0
15 C(5,1)=-1: C(6,1)=1: C(7,1)=1:
                  C(8,1)=-1               Матричное задание окрестностей 1
16 C(1,2)=0: C(2,2)=-1: C(3,2)=0:         и 2 типа
                  C(4,2)=1
17 C(5,2)=-1: C(6,2)=-1: C(7,2)=1:        ____________________________
                  C(8,2)=1
30 GOTO 375
50 INPUT L0                               Ввод цвета первой клетки (левый
                                          верхний угол)
60 FOR I=1 TO N
70   K=I+L0-1: L2=K-INT(K/3)*3            Формирование остальных клеток
80   FOR J=1 TO N                         поля (чередование цветов - 3
90     L1=(L2+J-1)-INT((L2+J-1)/3)*3            цвета начиная с L0)
100    B(I,J)=L1
110    NEXT J
120 NEXT I
130 GOTO 11
140 FOR I=1 TO N                          Задание цветов клеткам поля
150 FOR J=1 TON                           случайным образом
160     B(I,J)=INT(2.9999999#*RND)
170    NEXT J
180 NEXT I
```

```
181 GOTO 11
190 KOL=0
200 FOR J=1 TO N                          Определение для каждой клетки
210  FOR I=1 TO N                         поля числа соседей различного
215     S(1)=0: S(2)=0: S(3)=0            цвета
220     FOR K=1 TO L                      Определение координат соседней
230               I1=I+C(K,2)             клетки с учетом типа окрестности
240               J1=J+C(K,1)
245               M=1
250  IF I1<1 OR J1<1 OR I1>N OR J1>N      Определение выхода за границы
                  THEN GOTO 270           поля
260               M=A(I1,J1)+1            Определение цвета соседней
270               S(M)=S(M)+1             клетки. Подсчет кол-ва клеток
                                          цвета M
280               NEXT K
290     IF INT(S(1)/2)*2=S(1) THEN        ____________________________
                  R=0: GOTO 330
300     R=2
310     IF S(2)>S(3) THEN R=1             Реализация трех привил "жизни"
320     IF RL=0 THEN GOTO 330
325     IF RL=1 AND A(I,J)>0 THEN
                  R=A(I,J): GOTO 330
327     IF RL=2 AND A(I,J)=2 THEN R=2     ____________________________
330     B(I,J)=R                          Присвоение точке цвета в
                                          соответствии с правилом
340     IF B(I,J)=A(I,J) THEN KOL=KOL+1   Проверка не изменения состояния
                                           поля клеток
350     NEXT I
360  NEXT J
370  IF KOL=N*N THEN GOTO 470
375  LOCATE 2,11: PRINT "LIFE STEP ";
                  LLK
377  LLK=LLK+1
380  FOR I=1 TO N                         _____________Печать поля
390   FOR J=1 TO N                              клеток
400     X1=SIZE*(I-1)+XS
410     Y1=SIZE*(J-1)+YS
420     LINE (X1,Y1)-(X1+SIZE,Y1+SIZE)    Рисование клетки (I,J) цветом
                  ,B(I,J),BF               B(I,J)
430     A(I,J)=B(I,J)
440     NEXT J
450  NEXT I
452  IF LLK<LLK0 THEN GOTO 460            Сравнение номера текущего шага с
                                          граничным
```

```
453 LOCATE 12,1: PRINT "CONTINUE "   Если совпадают, то выяснение
454 LOCATE 13,1: INPUT A$            продолжать ли работать дальше,
455 IF A$="y" OR A$="Y" THEN         если да ("y" или "Y"), то
        GOTO 456 ELSE GOTO 480           - 456, иначе 480
456 LOCATE 12,1: PRINT "                "  Стирание надпи-
457 LOCATE 13,1: PRINT "      "                  сей
458 LLK0=LLK+50                     Определение следующего граничного
                                       значения шага алгоритма.
460 FOR I=1 TO 1000                    Задержка для визуального
465 NEXT I                             анализа состояния поля клеток
467 GOTO 190                           Переход на 190
470 SCREEN 0                           Переход в цифровой режим
480 PRINT "NEXT LIFE ?": INPUT A$      Продолжать ли работу
490 IF A$="y" THEN GOTO 2              Если да ("y"), то 2, иначе 500
500 STOP                               (конец)
```

Вы никогда не думали, что можете очень легко стать архитектором и уверенно и быстро проектировать высотные здания и создавать красивые и надежные телевизионные вышки почти произвольной высоты? За экраном дисплея персонального компьютера вы можете все это проделать непринужденно и в игровой форме. Для этого необходимо лишь ознакомиться с программой СОЗДАНИЕ (CREATE).

ПРОГРАММА CREATE (СОЗДАНИЕ)

Программа предназначена для проектирования многоэтажного здания, каждый этаж которого конструктивно состоит из треугольных крепящих элементов.

При обращении к программе (см. текст программы СОЗДАНИЕ) она запрашивает о высоте здания, которое Вы хотите спроектировать. В программу необходимо ввести величину H_{max}, которая выражает высоту конструкции в некоторых условных единицах. Для самого верхнего этажа необходимо также задать высоту самого этажа, это величина H_0. Каждый этаж состоит из того или иного числа треугольников, самый верхний этаж должен представлять собой только один треугольник. Поэтому необходимо задать угол наклона боковой стороны верхнего треугольника к его основанию. Это величина Al_0. Программа запрашивает также величину MU, которая представляет собой коэффициент эстетики — минимальное увеличение или уменьшение основания базового треугольника

следующего уровня по сравнению с предыдущим. Как будет видно в дальнейшем, этот коэффициент определяет архитектурный стиль конструируемого здания или башни. Программе нужно также знать, из какого материала будет строиться сооружение, и для этого необходимо ввести величину PO. Программа по Вашему заказу в диалоговом режиме будет строить высотное многоэтажное здание, и поэтому необходимо задать коэффициент эластичности K1 и два коэффициента предельной точности материала на сжатие и на растяжение P10, P20. Понятно, что все задаваемые величины относительны, и Ваша задача как конструктора и архитектора состоит в том, чтобы вариацией задаваемых величин добиться желаемых результатов. Программа же осуществит все расчеты и произведет оптимизацию, стараясь сэкономить материалы и затраты на строительство.

Текст программы СОЗДАНИЕ (CREATE)

```
1 KEY OFF: SCREEN 1: COLOR 8,1          Переход в графический режим,
                                        выбор палитры
3 DIM FL(100)                           Задание рабочего массива
10 CLS: PRINT "Hmax": INPUT HM          Ввод исходной информации (высота
                                        сооружения,
20 PRINT "H0": INPUT H0                 начальная высота треугольника,
30 PRINT "AI0": INPUT AL0:              начальный угол)
       AL0=AL0*3.141596/180
40 HD=H0/5                              Определение шага возможного
                                        уменьшения высоты треугольника
50 HMIN=HD                              и минимального размера
60 PRINT "MU": INPUT MU                 (коэффициент минимально
                                        возможного увеличения ширины
                                        следующего этажа,
70 PRINT "P0": INPUT P0                 вес единицы материала
80 PRINT "K1": INPUT K1                 и коэффициент его
                                        сжатия/растяжения)
85 FOR I=1 TO 100                       Начальное задание количества
90   FL(I)=2                            треугольников на этажах
95   NEXT I
100 PRINT "P10,P20": INPUT KR1,KR2      (предельно допустимые нагрузки
                                        сжатие/растяжение)
110 XT=160: MS=200/HM: YT=0             Начальная установка параметров
                                        (координаты, вес конструкции,
120 P=0: FL0=1                          этаж, угол треугольников на
```

```
130 CLS: KOL=1: AL=ALO: H=HO: HT=0
140 A=H/SIN(AL)
150 HH=A*COS(AL)

160 HB=2*HH*KOL
170 P=2*KOL*(A+HH)*PO+P
180 K=INT((1+2*MU)/(2*MU))


 90 HO=HT-HO: 2*MS: II=1

200 FOR I=1 TO 2*KOL
205 HOO=HH*MS
210 X=XO+HOO*(I-1)
220 II=II*(-1)
230 Y1=YT+(1-II)*H/2*MS
240 Y2=YT+(1-(-1)*II)*H/2*MS
250 LINE (X,Y1)-(X+HOO,Y2),2
260 NEXT I
270 YT=YT+H*MS: HT=HT+H
280 LINE -(XO,YT),2



293 H=HO

294 IF HT+HO>HM THE GOTO 460

295 KK=FL(FLO)


300 FOR I=KK TO K
310 KOL=I: L=HB/(2*(I-1))

315 AL=ATN(H/L)
316 IF L>H*2.5 THEN GOTO 375
320 R=K1*P*SIN(AL)/(H*(KOL+1))
330 P1=R*COS(AL)
340 P2=R*SIN(AL)/KOL
350 IF P1>KR1 THEN GOTO 375
360 IF P2>KR2 THEN GOTO 375



370 GOTO 420
375 H1=H-HO
```

данном этаже
высоте треугольников.
Определение размеров сторон треугольников,
размера основания данного этажа,
веса всей конструкции.
Определение максимального количества треугольников, исходя из MU.
Определение начальной точки следующего этажа.
Рисование данного этажа
(ломаная линия начиная от начальной точки, количество звеньев равно количеству сторон в треугольникак на этаже)
Рисование стороны
Рисование основания от конечной точки до начальной следующего этажа).
Восстановление значения высоты треугольника.
Проверка на окончание: да - на 460.
Определение начального количества треугольников на этаже.
Цикл от начального до максимального кол-ва треугольников на этаже
Определение угла треугольников.
Определение нагрузок на этаже
Определение выхода за допустимые значения нагрузок (сжатие/растяжение). Если да - 375
Все нормально 420
Уменьшение высоты треугольника

```
377 IF H1<HMIN THEN H=H0: GOTO 382   Проверка на минимальность
                                     высоты (да - 382)
379 L=L*H1/H: H=H1:                  Определение кол-ва
            KOL=INT(HB/(2*L))+1      треугольниковна данном этаже
                                     при уменьшении высоты.
380 IF KOL>K THEN GOTO 382           Если их меньше чем K,
            ELSE GOTO 315            то 315, иначе 382.
382 NEXT I                           382 - увеличить кол-во
                                     треугольников
420 ' REM
435 FLO=FLO+1                        Переход на следующий этаж
440 GOTO 140                         На - 140 (расчет конструкции)
450 PRINT "It's imposible to create" Невозможно построить
                                     конструкцию с такими
                                     параметрами.
460 PRINT "Try again": INPUT A$      Пробовать снова ?
470 IF A$="n" THEN STOP              Если нет (n), то конец, иначе 480
480 PRINT "Other number of treang.": Изменение начального
            INPUT A$                 кол-ва треугольников ?
500 IF A$="n" THEN GOTO 10           Если нет (n) то 10, иначе 510
510 PRINT "Number of floor";:        Указать номер этажа
            INPUT I                  и
520 PRINT "Number of treang.'";:     начальное количество
            INPUT FLI(I-1)           треугольников на нем.
530 GOTO 110                         На 110
```

При некоторых сочетаниях параметров построить конструкцию программа не сможет, поэтому на экране Вы, возможно, увидите надпись: «It is impossible to create — «Это построить невозможно». Но Вам сразу же предоставляется следующая попытка, и на вопрос «Try again?» («Нужно ли повторить?») надо ответить «Y», если вы хотите продолжить, и «N» в противном случае. Если вы ответили «Y» при построенном уже здании, то будет предпринята попытка изменить число конструктивных элементов — треугольников — на задаваемом Вами этаже. Программа спросит «Other number of treang?» («Другое количество треугольников?»), и если вы ответите «N», то вновь будет осуществляться запрос исходных данных. Если вы ответите «Y», то необходимо указать номер этажа (→Number of floor), считая первым самый верхний этаж, и число треугольников на нем.

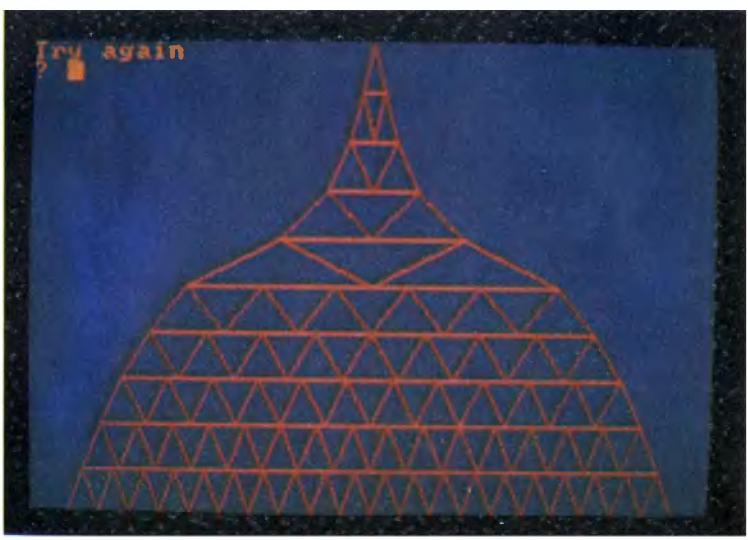


Рис. 28. Три варианта работы

Задавая различные коэффициенты и параметры, можно получить на экране и Эйфелеву башню, и останкинскую телевизионную вышку, и средневековый кафедральный собор. Работая и играя с этой программой, Вы ощутите разницу между архитектурными стилями и поймете взаимосвязь стилей с материалом конструкций, оцените значимость экономии материалов и бесполезную нагрузку архитектурных излишеств. На рис. 28 представлены результаты работы программы СОЗДАНИЕ, а вот при каких параметрах и коэффициентах получены эти конструкции — это Вы определите сами.

Всегда интересно исследовать некоторый неизвестный объект, если для этого под рукой есть необходимые способы и инструменты. В этом отношении персональный компьютер действительно незаменим, так как позволяет моделировать и представлять как натуральный любой из известных приборов. На компьютере можно моделировать и линейку, и треугольник, и осциллограф, т. е. известные измерительные приборы, но с помощью персонального компьютера можно также создать фантастический прибор, скажем, для измерения объемов тел — в пяти-

программы СОЗДАНИЕ (CREATE)

мерном пространстве. Мы не будем рассматривать программу такого прибора, хотя она вполне реальна, а обратим Ваше внимание на небольшую программу, позволяющую исследовать математические агрегаты. Предположим, что нам известно только, что этот агрегат плоский, т. е. все точки его лежат в некоторой плоскости, в которой мы можем построить систему координат.

ПРОГРАММА POINTS (ТОЧКИ)

Текст программы ТОЧКИ помогает исследовать свойства плоского математического агрегата. Это динамический агрегат, т. е. такой, который изменяется со временем, не является статическим, застывшим. Математически его можно выразить в виде

$$\left.\begin{aligned} x &= (x+y)\ \sin y; \\ y &= a - bx. \end{aligned}\right\}$$

Программа производит поточечное построение области значений этой функции в прямоугольной декартовой системе координат, область определения задается исследователем. Необходимо сначала ввести временной диапазон T(t), который показан также и на экране дисплея персонального компьютера, постоянные коэффициенты A(a), B(b) и число точек, которые будут программой выведены на экран N(n). Необходимо подчеркнуть, что ввод каждого исходного числа осуществляется нажатием соответствующей клавиши. Это означает, что если, например, нужно ввести или поменять значение N, то следует нажать на клавишу клавиатуры с литерой N и затем набрать соответствующее численное значение, нажимая на цифровые клавиши клавиатуры. Для того чтобы получить на экране изображение результатов работы программы, необходимо «запустить» вывод результатов, нажав клавишу C.

Четыре отдельных результата работы программы ТОЧКИ представлены на рис. 29. В них легко разобраться, так как на плоскость изображения результата работы программы выводятся также значения исходных данных.

Можно поменять и сам исследуемый математический агрегат для этого достаточно изменить операторы 110 и 120. Таким образом, программа представляет собой

программный блок, предназначенный для исследования математических плоских объектов, который может использоваться в математической серии обучающих программ.

Текст программы ТОЧКИ (POINTS)

```
5 REM program POINTS
10 N%=100: T=2: A=1: B=1: N11%=0      Задание исходных данных "по
                                      умолчанию"
11 GOSUB 500                          Вызов подпрограммы 500.
15 SIZE=6*T+1                         Определение размера рисунка
20 XMAX%=30: XMIN%=319:
       YMAX%=199: YMIN%=0
30 XDIF%=XMAX%-XMIN%:
       XDS=XDIF%/SIZE
40 YDIF%=YMAX%-YMIN%:                 Определение поля рисунка
       YDS=YDIF%/SIZE
42 XMID%=(XMAX%+XMIN%)/2
44 YMID%=(YMAX%+YMIN%)/2
50 SCREEN 1: CLS: COLOR 8,1           Переход в графический режим,
                                      выбор палитры.
60 LINE (XMIN%,YMIN%)-                Рисование рамки рисунка
       (XMAX%,YMAX%),2,B              (прямоугольник),
65 LINE (XMID%,YMIN%)-                   осей Y и
       (XMID%,YMAX%),3
67 LINE (XMIN%,YMID%)-                          X
       (XMAX%,YMID%),3
70 X=0: Y=0: GOSUB 500                Задание начального условия,
                                      вызов п/п 500
80 FOR I%=1 TO N%                     _____ Вычисление N точек .....
90   LOCATE 1,1:                      (печать № точки)
       PRINT USING "####";I%
100    X1=X: Y1=Y
110    X=(X1+T)*SIN(Y1)               Вычисление координат точки
                                      фигуры (X и
120    Y=A-B*X1                                Y)
130    X%=XMID%+XDS*X                 Пересчет координат точки в
140    Y%=YMID%+YDS*Y                 координаты поля рисунка
150    PSET (X%,Y%),N11%              Печать точки цветом N11%
155    N11%=N11%+1: IF N11%=4         Определение цвета следующей
       THEN N11%=0                    точки фигуры (всего 4 цвета)
160    NEXT                           _____
200 GOSUB 800                         Вызов подпрограммы 800 и 830
210 GOSUB 830                         (рез-тат символ нажатой клавиши)
```

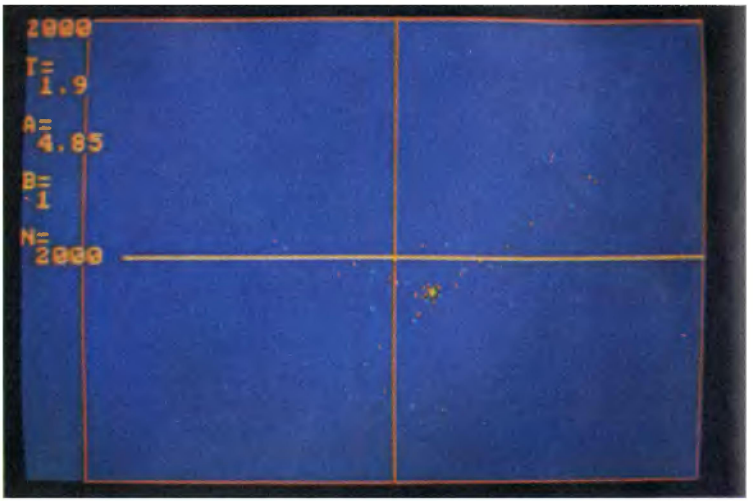
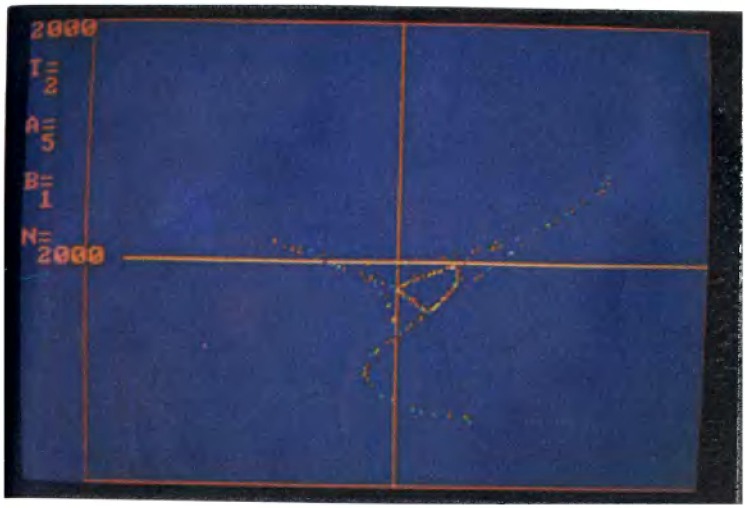
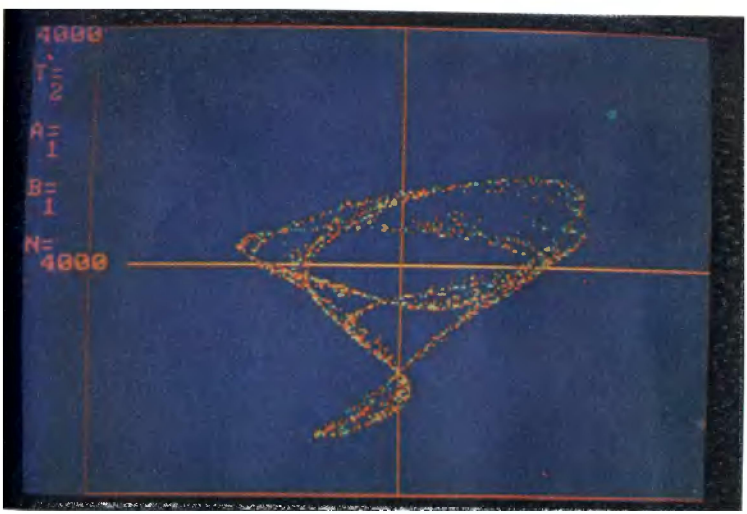


Рис. 29. Результаты работы





программы **ТОЧКИ (POINTS)**

```
220 IF I$="t" PR I$="T" THEN
       GOSUB 400
230 IF I$="a" PR I$="A" THEN
       GOSUB 420
240 IF I$="b" PR I$="B" THEN
       GOSUB 440
250 IF I$="n" PR I$='N" THEN
       GOSUB 460
260 IF I$="c" PR I$="C" THEN GOTO 15

300 GOTO 210

460 LOCATE 17,1: INPUT T
405 LOCATE 17,1:  PRINT "    ":
       GOSUB 500: RETURN
420 LOCATE 17,1:  INPUT A
425 LOCATE 17,1:  PRINT "    ":
       GOSUB 500: RETURN
440 LOCATE 17,1: INPUT B
445 LOCATE 17,1:  PRINT "    ":
       GOSUB 500: RETURN
460 LOCATE 17,1: INPUT N%
465 LOCATE 17,1:  PRINT "    ":
       GOSUB 500: RETURN
500 LOCATE 3,1: PRINT "T="
510 PRINT T
520 LOCATE 6,1: PRINT "A="
530 PRINT A
540 LOCATE 9,1: PRINT "B="
550 PRINT B
560 LOCATE 12,1: PRINT "N="
570 PRINT N%
580 RETURN
800 I$=INKEY$

810 IF I$<>"" THEN GOTO 800
820 RETURN
830 I$=INKEY$
840 IF I$="" THEN GOTO 830

850 LE%=LEN(I$): I$=RIGHT(I$,1)

860 RETURN
```

Если нажато t (T), то вызов подпрограммы 400
Если нажато a (A), то вызов подпрограммы 420
Если нажато b (B), то вызов подпрограммы 440
Если нажато n (N), то вызов подпрограммы 460
Если нажато c (C), то переход на 15 (вычисления)

Если нажаты другие клавиши, то 210

— Подпрограмма ввода параметра T (ввод T, стирание введенного значения, вызов 500, возврат)

— Подпрограмма ввода параметра A (ввод A, стирание введенного значения, вызов 500, возврат)

— Подпрограмма ввода параметра B (ввод B,стирание введенного значения, вызов 500, возврат)

— Подпрограмма ввода параметра N (ввод N, стирание введенного значения, вызов 500, возврат)

— Подпрограмма печати параметров алгоритма

каждый параметр печатается в своем месте экрана

———————————

— Подпрограмма ожидания нажатия клавиши
Если не нажата, то на 800,
иначе возврат ———

— Подпрограмма определения нажатого символа. Если клавиша нажата, то на 830,
иначе определение длины символа и его сдвиг в праводля последующего сравнения ———

ПРОГРАММА LINE (ЛИНИЯ)

Программный модуль ЛИНИЯ (LINE) продолжает и развивает идеи, заложенные в основу программного модуля ТОЧКИ. Вместе с тем это вполне самостоятельная программа, предназначенная для исследования плоских математических агрегатов, но путем построения не точек, а области значений агрегата при помощи отрезков прямых линий.

Текст программы ЛИНИЯ (LINE)

```
20 DIM X(2500),Y(2500), X1%(2500),          Задание рабочих массивов
        Y1%(2500)
30 SCREEN 1: COLOR 8,1: N%=10               Переходж в графический режим
40 C1=.5: C2=.5: A1=.5                      Начальная установка параметров
                                            формулы
50 MIN1%=81: MAX1%=319:
     MIN2%=190: MAX2%=0
60 DIF1%=MAX1%-MIN1%:                       Задание размеров поля рисунка
     DIF2%=MAX2%-MIN2%
70 X(0)=0: Y(0)=0
80 XMIN=0: XMAX=0: YMIN=0:
        YMAX=0: MD%=N%/3
90 FOR I%=1 TO N%                           Реализация формулы для N точек
100  Z=C1*I%+C2*SIN(A1*I%)                  Вычисление координат для точки I
110  X(I%)=X(I%-1)+SIN(Z)
120  Y(I%)=Y(I%-1)+COS(Z)
130  IF XMIN>X(I%) THEN XMIN=X(I%)          Определение минимального и
140  IF YMIN>Y(I%) THEN YMIN=Y(I%)          максимального значений
                                            координат среди всех N
150  IF XMAX<X(I%) THEN XMAX=X(I%)          точек
160  IF YMAX<Y(I%) THEN YMAX=Y(I%)
170  NEXT
180  XDIF=XMAX-XMIN                         Определение размера рисунка
190  YDIF=YMAX-YMIN
200  IF XDIF>YDIF THEN DIF=XDIF
          ELSE DIF=YDIF
210  FOR I%=0 TO N%                         ____Формирование координат в
                                            поле рисунка
220  XO=(X(I%)-XMIN)/DIF
230  YO=(Y(I%)-YMIN)/DIF
240  X1%(I%)=MIN1%+DIF1%*XO
250  Y1%(I%)=MIN2%+DIF2%*YO
260  NEXT                                   ____
```

```
270 CLS: LINE(MIN1%,MIN2%)-
         (MAX1%,MAX2%),1,B
280 GOSUB 1000

300 PSET (X1%(0),Y1%(0))

302 FOR I%=1 TO N%
303  COL%=INT(I%/MD%)+1
304  LINE -(X1%(I%),Y1%(I%)),COL%

310 NEXT
320 I$=INKEY$:330 IF I$<>"" GOTO 320

335 GOSUB 400

340 I$=INKEY$
350 IF I$="c" OR I$="C" THEN
GOSUB 410
360 IF I$="1" THEN GOSUB 420

370 IF I$="2" THEN GOSUB 430

380 IF I$="3" THEN GOSUB 440

385 IF I$="n" OR I$="N" THEN
              GOSUB 450
390 GOTO 340

400 LOCATE 18,1: PRINT "CHANGE"
401 LOCATE 16,1: PRINT "C-calc."
402 LOCATE 19,1: PRINT "1-c1"
403 LOCATE 20,1: PRINT "2-c2"
404 LOCATE 21,1: PRINT "3-c3"
405 LOCATE 22,1: PRINT "N-num."
409 RETURN
410 LOCATE 25,1:
    PRINT "Calculation";: RETURN 80
420 LOCATE 6,1: INPUT C1:
    GOSUB 1000: RETURN
430 LOCATE 9,1: INPUT C2:
    GOSUB 1000: RETURN
440 LOCATE 12,1: INPUT A1:
    GOSUB 1000: RETURN
450 LOCATE 14,1: INPUT N%:
```

Рисование рамки поля рисунка

Вызов подпрограммы 1000 (Печать параметров)

Перенесение пера в начальную точку рисунка

—— Рисование

Определение цвета линии

Рисование линии от текущей точки до точки I

———

Ожидание нажатия клавиши. Если не нажата то 320

Если нажата - вызов подпрограммы 400

Ожидание нажатия клавиши:

если нажато c(C),то подпрограмма 410

если нажато 1,то подпрограмма 420

если нажато 2,то подпрограмма 430

если нажато 3,то подпрограмма 440 .

если нажато n(N),то подпрограмма 450

если эти клавиши не нажаты,то 340 (ожидание)

———— Подпрограмма печати

клавиш управления работой алгоритма

———

П/п передачи управления на 80 (вычисление)

П/п ввода параметра C1, вызов 1000 и возврат

П/п ввода параметра C2, вызов 1000 и возврат

П/п ввода параметра C3, вызов 1000 и возврат

П/п ввода параметра N, вызов

```
    GOSUB 1000: RETURN
900 LOCATE 20,35: END
1000 FOR L%=1 TO 14 LOCATE L%,1:
    PRINT "          ";
1001 NEXT
1005 LOCATE 1,1: PRINT "Size="
1010 LOCATE 2,1:
       PRINT USING "###.##";DIF;
1020 LOCATE 4,1: PRINT "c1="
1030 LOCATE 5,1:
       PRINT USING "###.##";C1
1040 LOCATE 7,1: PRINT "c2="
1050 LOCATE 8,1:
       PRINT USING "###.##";C2
1060 LOCATE 10,1: PRINT "c3="
1070 LOCATE 11,1:
       PRINT USING "###.##";A1
1080 LOCATE 25,14
1090 PRINT "Z(i)=c1*i+c2*sin(c3*i)";
1100 LOCATE 13,1: PRINT "N=";
1101 PRINT USING "###.##";N%
1110 RETURN
```

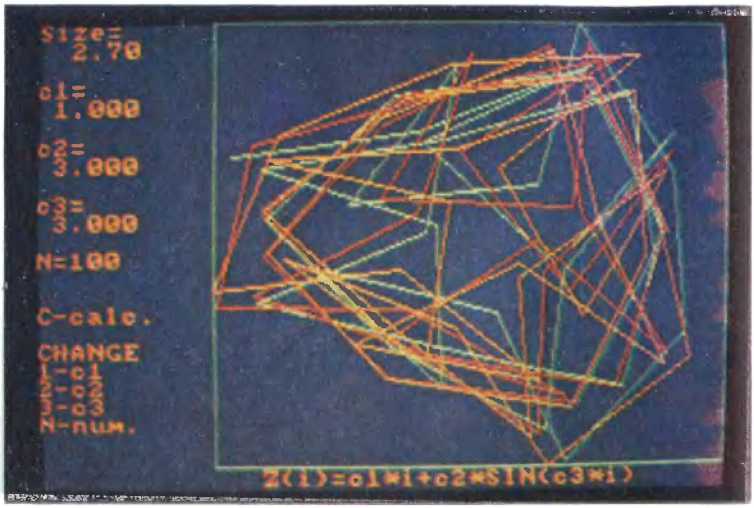
1000 и возврат

Подпрограмма печати параметров алгоритма (вначале очистка экрана 1000-1001, затем печать параметров на соответствующих местах экрана

В приведенном конкретном тексте программы ЛИНИЯ исследуется следующий агрегат:

$$\left.\begin{aligned} x_i &= x(i-1) + \sin(c1 \cdot i + c2 \cdot \sin(c3 \cdot i)); \\ y_i &= y(i-1) + \cos(c1 \cdot i + c2 \cdot \sin(c3 \cdot i)). \end{aligned}\right\}$$

Как видно из формул, этот агрегат плоский и статический, так как он не зависит от времени. В основе конструкции лежат тригонометрические функции. Структура агрегата задана в операторах 100, 110 и 120. Так же как и в предыдущей программе, в данном программном модуле необходимо ввести начальные данные. Чтобы ввести значения коэффициентов C1, C2 и C3, необходимо нажать соответственно клавишу 1, 2 или 3, а затем набрать на цифровой части клавиатуры соответствующее численное значение. Необходимо ввести также N — число линий построения. Для этого нужно нажать клавишу с символом «N», а затем на цифровой части клавиатуры набрать желаемое число. Число N определяет «детальность» построения области значений заданного математического агрегата. Что такое «детальность» построения, легко увидеть из рис. 30 и 31, на которых представлены результаты работы программного модуля ЛИНИЯ.



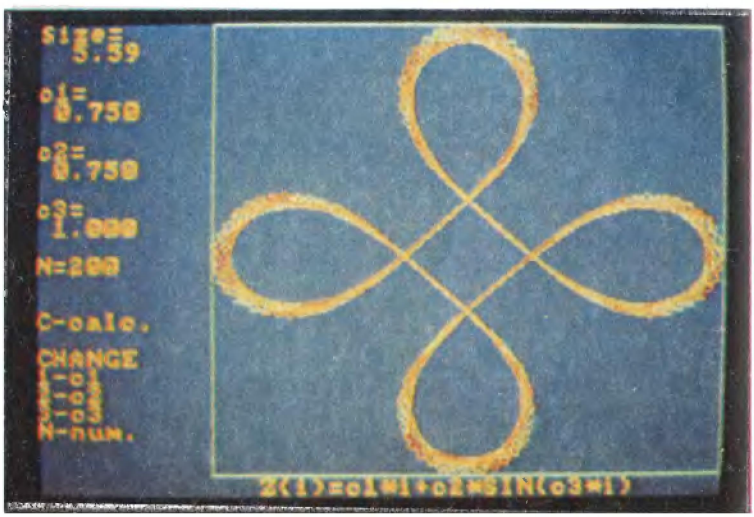
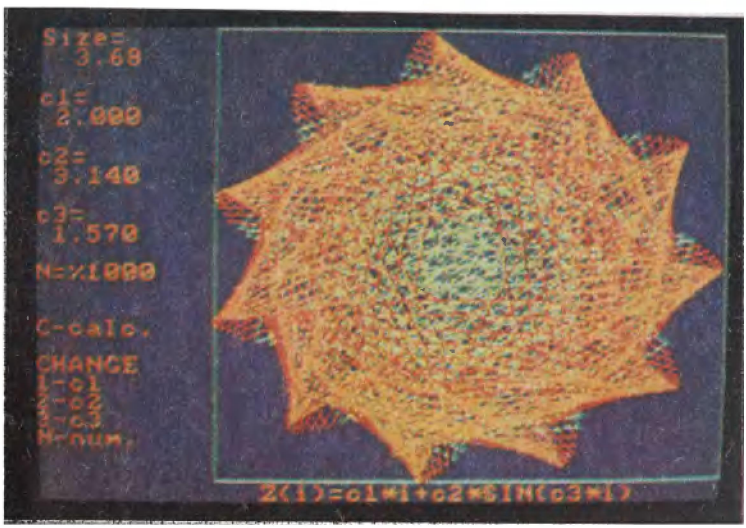
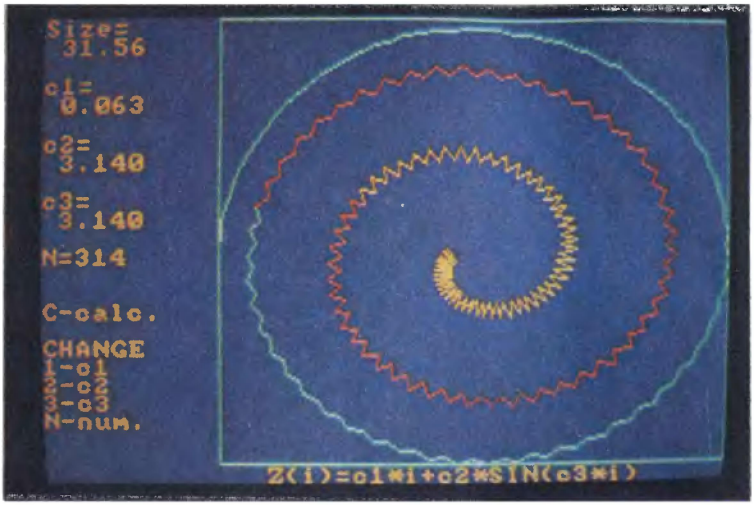


Рис. 30. Результаты работы



программы ЛИНИЯ (LINE)



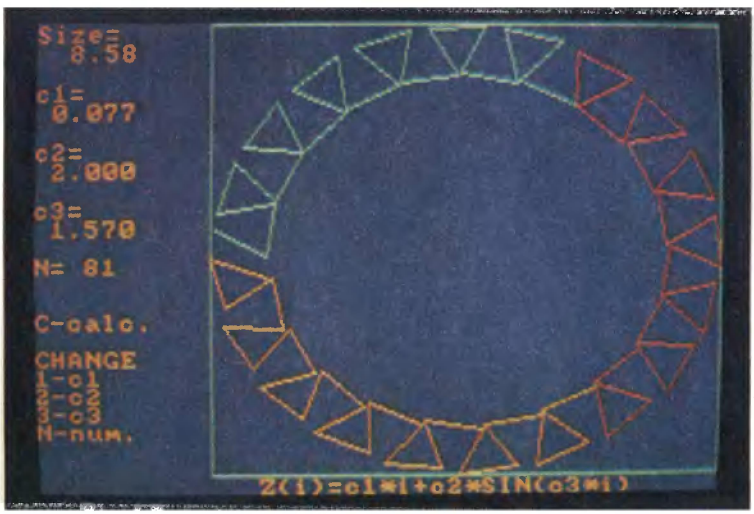
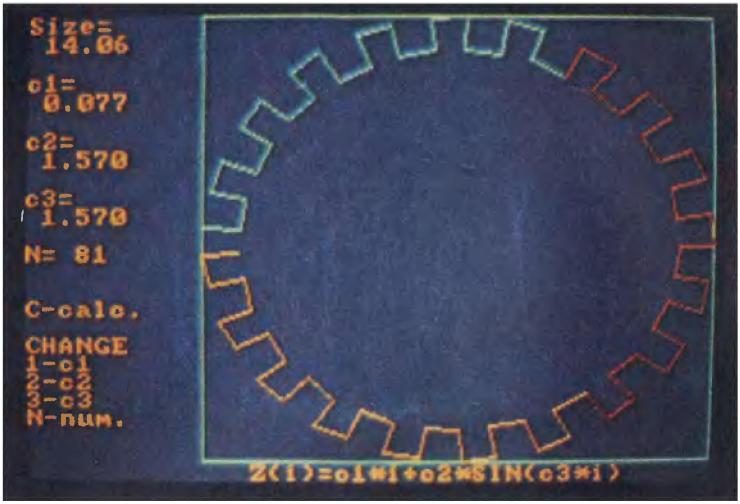
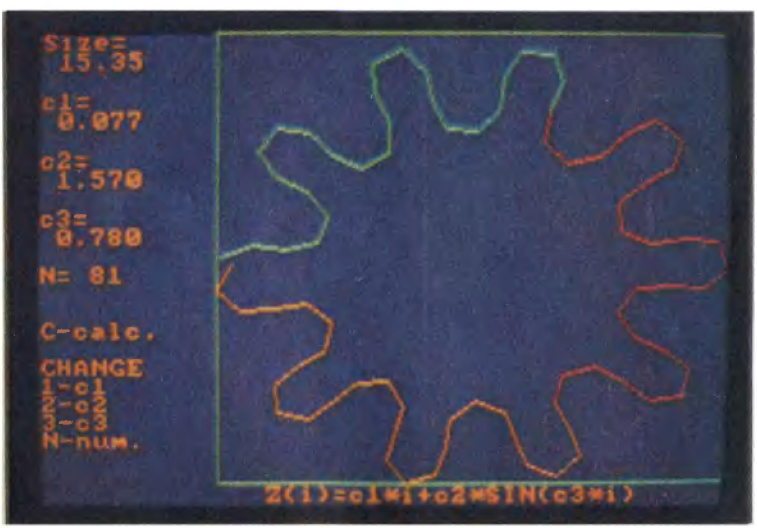


Рис. 31. Интересный результат ра-





боты программы ЛИНИЯ (LINE)

В связи с исследованием этого математического объекта возникает мысль о следующей игре, которую мы Вам и предлагаем. Попробуйте получить графики, изображенные на рис. 30 и 31, но закрыв, скажем, листом бумаги, значения всех исходных данных к задаче. Вы сумеете проанализировать свою интуицию и логику. Желаем успеха!

Чтобы создать впечатляющую картину внутреннего компьютерного мира, нужно уметь создавать на экране персонального компьютера различные изображения.

ПРОГРАММА TREEN (ДЕРЕВЬЯ)

Программа предназначена для создания изображений различного вида деревьев — это видно уже из ее названия. Примеры деревьев, которые можно «вырастить» в Вашем компьютерном саду с помощью этого программного модуля, показаны на рис. 32 и 33. Программа строит изображения сосен, елей, берез дубов и многих других видов деревьев. Для построения конкретного вида изображений создается индивидуальная программа, но можно создать программы, реализующие построение различных изображений.

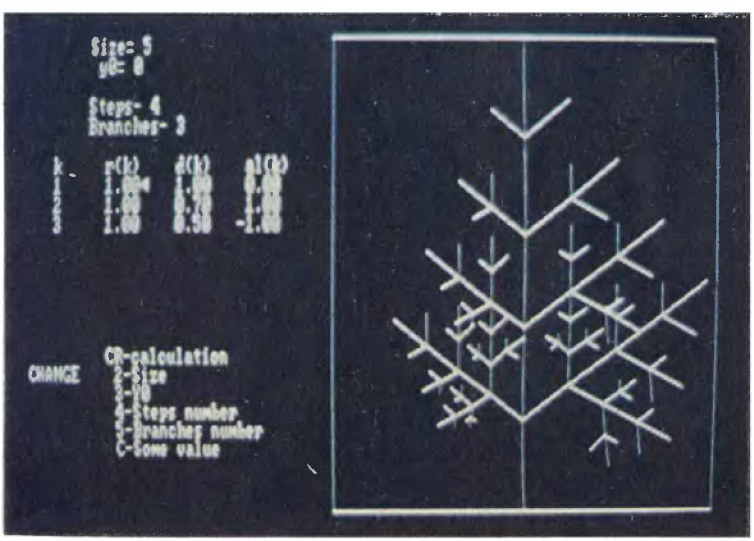


Рис. 32. Результаты работы программы ДЕРЕВЬЯ (TREEN) при малой глубине рекурсии

Операторы 250—447 (см. текст программы ДЕРЕВЬЯ) заняты построением собственно изображения дерева, а остальные операторы выполняют вспомогательные роли: построение рамки поля изображения, вывод на экран компьютера подсказывающей и определяющей режимы работы информации, а также вывод результирующих параметров построенного изображения.

Прежде всего отметим особенности программы с точки зрения ввода исходных данных и определения начальных величин: нажатие клавиш, указанных в левом нижнем углу экрана (CR, 2,3,...), вызывает следующие действия:

CR — построение дерева в соответствии с исходными данными (левый верхний угол экрана);

2 — SIZE — изменение размера плоскости рисунка;

3 — Y0 — определение координаты расположения ствола в плоскости рисунка (по оси y);

4 — Steps number — число шагов, которые необходимо выбрать для построения дерева (в связи с ограничениями по времени и пространству памяти этот параметр не может быть больше 10);

5 — Branche number — этот параметр определяет число подветвей, которые должны вырасти на ветви в процессе построения изображения дерева.

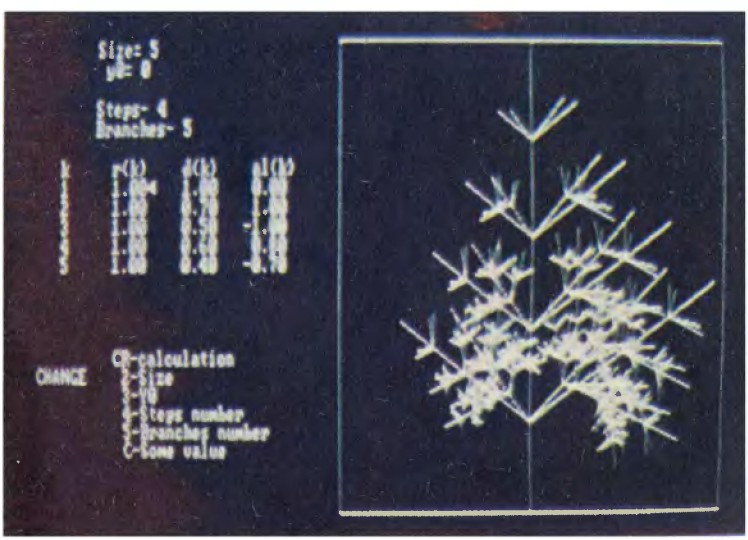


Рис. 33. Результаты работы программы ДЕРЕВЬЯ (TREEN) при большой глубине рекурсии

Величина C — Some value — определяет изменение параметра подветви, вырастающей на ветви, на которую указывает символ △ на экране компьютера (левый верхний угол экрана).

На это действие на экране появляется запрос «New value?». В ответ на этот запрос надо ввести новое значение. Указатель △ перемещается клавишами ↑ , ↓ , ← и →.

После запуска программа строит изображение, но можно сразу же поменять все исходные параметры и построить новое изображение. Эта программа может послужить исходным модулем для создания библиотеки различных изображений, которые будут заполнять сцену Вашей компьютерной игры.

Текст программы ДЕРЕВЬЯ (TREEN)

```
10 DIM X1(10),X2(10),Y1(10),Y2(10)      Задание рабочих массивов
20 DIM N(10),D(10), SN(10),CS(10)
30 DIM R(6),DO(6), AL(6),C1(6),S1(6),
                REG(6,3)
40 KK=2, NMAX1=3: SIZE=10: Y1(0)=3:     Задание основных исходных
                D(0)=1                  данных "по умолчанию"
60 NMAX=NMAX1-1
70 SN(0)=0
80 CS(0)=1
90 MAX1=635: MIN1=280: MAX2=0:          Задание параметров рисования
                MIN2=199
95 DIF1=MAX1-MIN1: DIF2=MAX2-MIN2
100 REG(0,1)=1: REG(1,1)=1:             Задание исходных данных о ветвях
                REG(2,1)=1              "по умолчанию"
110 REG(0,2)=1: REG(1,2)=1:
                REG(2,2)=.5
120 REG(0,3)=0: REG(1,3)=1:
                REG(2,3)=-1
125 GOSUB 1350                          Вызов подпрограммы 1350
130 FOR I=0 TO NMAX                     ———————
150  S1=SIN(AL(I))                     Вычисление значений sin и cos
160  C1=COS(AL(I))                     углов для ветвей дерева
170 NEXT                                ———————
180 DS1= DIF1/SIZE: DS2=DIF2/SIZE       Вычисление параметров рисования
200 X1(0)=SIZE/2: X2(0)=SIZE/2          Определение координат ствола
210 Y(0)=Y1(0)+D(0)                     дерева
220 SCREEN 2                            Переход в графический режим
                                        высокого разрешения
230 CLS                                 Очистка экрана
```

```
232 LINE MIN1,MIN2)-
      (MAX1,MAX2),,B
235 X1%=MIN1+X1(0)*DS1:
      X2%=MIN1+X2(0)*DS1
236 Y1%=MIN2+Y1(0)*DS2:
      Y2%=MIN2+Y2(0)*DS2
240 LINE (X1%,Y1%)-(X2%,Y2%)
250 FOR KMAX=1 TO KK

260 K=1
270 N(K)=-1
280 N(K)=N(K)+1

290 IF N(K)>NMAX THEN GOTO 420

300 R0=R(N(K))
310 X1(K)=X1(K-1)*(1-R0)
      +X2(K-1)*R0
320 Y1(K)=Y1(K-1)*(1-R0)
      +Y2(K-1)*R0
330 SN(K)=SN(K-1)*C1(N(K))
      +CS(K-1)*S1(N(K))
340 CS(K)=CS(K-1)*C1(N(K))
      -SN(K-1)*S1(N(K))
350 D(K)=D(K-1)*D0(N(K))
360 X2(K)=X1(K)+SN(K)*D(K)
370 Y2(K)=Y1(K)+CS(K)*D(K)
375 X1%=MIN1+X1(K)*DS1:
      X2%=MIN1+X2(K)*DS1
376 Y1%=MIN2+Y1(K)*DS2:
      Y2%=MIN2+Y2(K)*DS2
380 LINE (X1%,Y1%)-(X2%,Y2%)
390 IF K=KMAX THEN GOTO 280

400 K=K+1
410 GOTO 270

420 K=K-1

430 IF K=0 THEN GOTO 447
440 GOTO 280
447 NEXT
450 GOSUB 1000: GOSUB 1040:
      GOSUB 1200: GOSUB 800
```

Рисование рамки рисунка (прямоугольник)
Определение координат ствола дерева в поле рисунка

Рисование ствола
_____ Определение координат всех ветвей для всех шагов алгоритма и их рисование

Номер (тип) рассматриваемой ветви

Рассмотрены ли все ветви, "Да" на 420

Вычисление начальных координат рассматриваемой ветви

Пересчет sin и cos для рассматриваемой ветви

Вычисление длины ветки
Определение конечных координат ветви
Вычисление координат рассматриваемой ветви в поле рисунка

Рисование ветви
Последний шаг алгоритма (если да - 280,
нет - 400).
Переход к следующему шагу алгоритма.
Переход к предыдущему шагу алгоритма

_____

Вызов подпрограмм 1000, 1040, 1200 и 800

```
456 KH1%=0: KH2%=1                    Установка параметров указателя
                                      (курсора)
457 GOSUB 1300                        Вызов подпрограмм 1300 и 830
460 GOSUB 830                         (рез-тат символ нажатой клавиши)
461 IF I$="1" THEN END                Если нажата 1, то конец
462 IF I$="2" THEN GOSUB 520          Если нажата 2, то вызов п/п 520
463 IF I$="3" THEN GOSUB 530          Если нажата 3, то вызов п/п 530
464 IF I$="4" THEN GOSUB 540          Если нажата 4, то вызов п/п 540
465 IF I$="5" THEN GOSUB 550          Если нажата 5, то вызов п/п 550
466 IF I$="H" THEN GOSUB 560          Если нажата стрелка вверх, то
                                      вызов п/п 560
467 IF I$="P" THEN GOSUB 570          Если нажата стрелка вниз, то
                                      вызов п/п 570
468 IF I$="K" THEN GOSUB 580          Если нажата стрелка влево, то
                                      вызов п/п 580
469 IF I$="M" THEN GOSUB 590          Если нажата стрелка вправо, то
                                      вызов п/п 590
470 IF I$=CHR$(13) GOTO 125           Если нажато CR (ВК), то на 125
                                      (вычисления)
471 IF I$="C" OR "c" THEN GOSUB 490   Если нажата C (c), то вызов п/п
                                      490
472 IF I$="R" OR "r" THEN GOSUB 500   Если нажата R (r), то вызов п/п
                                      500
480 GOTO 460                          Если нажаты не эти клавиши, то
                                      460
490 LOCATE 32,1: INPUT                ____ Подпрограмма ввода
    "New value";REG(KH1%,KH2%)        значений параметров ветвей.
492 LOCATE 23,1: PRINT                Ветвь и параметр определяется
    "                              "  расположением указателя
                                      (KH1,KH2). Введенное значение
494 LOCATE KH1%+8,KH2%*8-2            стирается и отображается в
496 PRINT USING "#####.##";           таблице параметров
        REG(KH1%,KH2%)
498 RETURN                            _____
500 CLS: GOSUB 100:                   _____ Подпрограмма подготовки к
        GOSUB 1040: GOSUB 1200:       работе (очистка экрана, печать
501 RETURN                            текущих параметров и меню ____
520 LOCATE 23,1: INPUT                _____
        "New size";SIZE
525 GOSUB 1000: GOSUB 1090: RETURN
530 LOCATE 23,1: INPUT                Подпрограммы 520-535, 540-545,
        "New Y0";Y1(0)                550-555 -- ввод
535 GOSUB 1000: GOSUB 1090:           основных параметров
        RETURN                        алгоритма
```

```
540 LOCATE 23,1: INPUT
      "New steps number";KK
545 GOSUB 1000: GOSUB 1090:
      RETURN
550 LOCATE 23,1: INPUT
    "New branches number";NMAX1
552 NMAX=NMAX1-1: GOSUB 1000
555 GOSUB 1040: GOSUB 1310: KH1=0:
    GOSUB 1300: RETURN
560 GOSUB 1310
561 KH1%=KH1%-1: IF KH1%<0 THEN
      KH1%=NMAX
565 GOSUB 1300: RETURN
570 GOSUB 1310
571 KH1%=KH1%+1: IF KH1%>NMAX
      THEN KH1%=0
575 GOSUB 1300: RETURN
580 GOSUB 1310
581 KH2%=KH2%-1: IF KH2%<1
      THEN KH2%=3
585 GOSUB 1300: RETURN
590 GOSUB 1310
591 KH2%=KH2%+1: IF KH2%>3
      THEN KH2%=1
595 GOSUB 1300: RETURN
```

Подпрограммы 560-565, 570-575, 580-585, 590-595 -- подпрограммы управления указателем (курсором) в таблице параметров ветвей

```
800 I$=INKEY$
810 IF I$<>"" THEN GOTO 800
820 RETURN
```

Подпрограмма ожидания освобождения клавиатуры, если нет, то на 800, иначе возврат

```
830 I$=INKEY$
840 IF I$="" THEN GOTO 830

850 LE%=LEN(I$): I$=RIGHT$(I$,1)
860 RETURN
```

Подпрограмма ввода символа нажатой клавиши. Если клавиша не нажата, то на 830, иначе запись символа клавиши в I$

```
1000 LOCATE 1,8: PRINT "Size=";SIZE
1010 LOCATE 2,9: PRINT "y0=";Y1(0)
1020 LOCATE 4,8: PRINT "Steps-";KK
1030 LOCATE 5,8: PRINT "Branches-"
                 ;NMAX1
1035 RETURN
1040 LOCATE 7,1

1041 PRINT "  k    r(k)   d(k)   al(k)"
```

Подпрограмма печати основных параметров алгоритма

Подпрограмма печати таблицы параметров ветвей. Печать заголовка

```
1045 FOR I%=0 TO NMAX:
       LOCATE I%+8,1
1046   PRINT
     "                                    "
1047   NEXT
1050 FOR I=0 TO NMAX
1060   LOCATE I+8,1
1061   PRINT USING "####"; I+1
1070   LOCATE I+8,6
1071   PRINT USING "#####.##";
1072   REG(I,1);REG(I,2);REG(I,3);

1080   NEXT
1090 FOR I=23 TO 25: LOCATE I,1
1091   PRINT
     "                                    "
1092   NEXT
1110 RETURN
1200 LOCATE 17,10: PRINT
     "CR-calculation"
1210 LOCATE 18,1: PRINT "CHANGE"
1220 LOCATE 18,10: PRINT " 2-Size"
1230 LOCATE 19,10: PRINT " 3-Y0"
1240 LOCATE 20,10:
       PRINT " 4-Steps number"
1250 LOCATE 21,10:
       PRINT " 5-Branches number"
1260 LOCATE 22,10 :
       PRINT " C-Some value"
1270 RETURN
1300 LOCATE KH1%+8,KH2%*8+6
1301 PRINT CHR$(17);

1302 RETURN
1310 LOCATE KH1%+8,KH2%*8+6
1311 PRINT "  "

1312 RETURN
1350 FOR I%=0 TO NMAX
1360   A(I%)=REG(I%,1):
     D0(I%)=REG(I%,2)
1361             AL(I%)=REG(I%,3)
1362             NEXT
1370 RETURN
```

Стирание предыдущих значений

Печать текущих значений параметров ветвей

Стирание рабочего поля меню управления работой алгоритма

Подпрограмма печати меню управления работой алгоритма

Подпрограмма рисования указателя - стрелки (курсора) в таблице параметров ветвей в позиции KH1,KH2

Подпрограмма стирания указателя - стрелки (курсора) в таблице параметров ветвей в позиции KH1,KH2

Подпрограмма передачи параметров ветвей из таблицы параметров в рабочие массивы

ПРОГРАММА GENETIC (ГЕНЕТИКА)

Представим себе некоторое пространство, населенное абстрактными существами, которые условно можно поделить на «родителей» и «сыновей» (или «дочерей», или «сыно-дочерей», в общем «детей», так как таких потомков может быть много разных типов — столько, сколько Вы зададите). Программа запрашивает сама, какие правила развития электронных компьютерных существ Вы выберете для игры (см. текст программы ГЕНЕТИКА). Есть в игре стандартные правила, которые поясняет табл. 1. Из таблицы видно, что разных родителей в игре предусмотрено только три разных вида: 1, 2 и 3; задано также, какое число «сыновей» может быть у каждого сочетания видов «родителей». В следующей колонке указывается вид «сыновей», которые могут появиться у данной пары «родителей»; естественно, вид задается для каждой пары и может быть только 1, 2 или 3. В последней колонке таблицы для каждой пары «родителей» задается вероятность того, что на следующем временно́м шаге бывшие «сыновья», а теперь уже сами «родители», погибнут и поэтому, естественно, не смогут участвовать в появлении новых «сыновей».

При использовании стандартных правил программа при запуске сразу запрашивает длину исходной популяции, которая должна быть целым числом. После того как задана длина исходной популяции, необходимо на каждую

Таблица 1

Таблица исходных данных к программе ГЕНЕТИКА при использовании стандартных правил

Вид «родителя»		Число «сыновей»	Вид «сыновей»	Вероятность гибели «сыновей»
1-20	2-20			
1	1	4	1	0,6
1	2	1	3	0,55
1	3	2	2	0,5
2	1	1	3	0,55
2	2	4	2	0,4
2	3	3	3	0,3
3	1	2	2	0,5
3	2	3	3	0,3
3	3	4	3	0,2

единицу этой длины задать вид «родителей» (1, 2 или 3). После этого программа начинает работу и выдает на экран изображения каждого поколения, возникающего на всех последующих временных интервалах вслед за исходным положением (рис. 34).

Текст программы ГЕНЕТИКА (GENETIC)

```
1 DIM A(100),B(100),M(3,3),F(9),P(9)     Задание рабочих массивов
2 KEY OFF: SCREEN 0                      Переход в цифровой режим работы
                                         дисплея
3 M(1,1)=4: P(1)=.6                      Задание стандартных правил:
4 M(2,2)=4: P(2)=.4
5 M(3,3)=4: P(3)=.2                      количество потомков у родителей
6 M(1,3)=4: P(4)=.5                      типов (M)
7 M(3,1)=4: P(5)=.5                      вероятностей "смерти" (P)
8 M(2,3)=4: P(6)=.3
9 M(2,1)=4: P(7)=.55
10 M(1,2)=4: P(8)=.55
11 M(3,2)=4: P(9)=.3
12 F(1)=1: F(2)=2: F(3)=3
13 F(4)=2: F(5)=2: F(6)=3                возможного типа (цвета) потомка
14 F(7)=3: F(8)=3: F(9)=3                (F)
18 PRINT "Standart rules" INPUT A$       Определение правил работы:
19 IF A$="y" THEN GOTO 90                если стандартные (y), то 90
20 FOR I=1 TO 3                          ____ Ввод нестандартных правил
30   FOR J=1 TO 3                        формирования поколений
40     PRINT I;"*";J;" suns";:           кол-во потомков для I,J
       INPUT M(I,J)                      (I,J=1,2,3)
50   PRINT "Type of suns and prob.":     тип (цвет)
                 L=(I-1)*3+J             и
60     INPUT F(L),P(L)                   вероятность "смерти" потомка
70     NEXT J
80   NEXT I
90 PRINT "lenght": INPUT L               Определение длины исходной
                                         популяции
100 PRINT "Initial popularity"
110 FOR I=1 TO L                         Задание типов (цветов) клеток
120  INPUT B(L)                          исходной популяции
130 NEXT I
135 R=1: SCREEN 1: KEY OFF:              Задание номера поколения .
                 COLOR 8,1: CLS          Переход в графический режим.
140 M=50-INT(L/2)                        Определение центра популяции на
                                         экране
```

```
150 FOR I=1 TO L                    ---------- Рисование популяции с
                                    номером R
160 A(M+I)=B(I): X1=(M+I-1)*3       Перепись типов (цветов) в рабочий
    Y1=(R-1)*3:                     массив. Вычисление координат
                                    клеток поколения на экране.
162 LINE (X1,Y1)-(X1+3,Y1+3),B(I), BF Рисование клетки I цветом B(I).
163 B(I)=0                          Подготовка к определению
                                    следующего поколения.
170 NEXT I                          ----------
180 L1=1                            ---Формирование нового поколения
190 FOR I=1 TO 99                   В поколении не более 100 клеток
200 I1=A(I): Y1=A(I+1)
210 IF I1=0 OR Y1=0 THEN B(L1)=0:   Если предки "мертвы" - потомок
                GOTO 230            "мертв"
205 IF L1>100 THEN L1=100: GOTO 235
220 GOSUB 380                       Вызов подпрограммы 380
230 NEXT I----------------------------
235 L=L1                            Переопределение кол-ва клеток в
                                    поколении
240 S=0                             _ Определение "невырожденности"
250 FOR I=1 TO L                    популяции
260 IF (B(I)=A(I) THEN S=S+1        она "вырождена" если потомки
270 NEXT I                          _____ совпадают с предками
280 IF L=S THEN GOTO 350            Если вырождена то - 350
290 R=R+1                           Переход к новому поколению
300 IF R=60 THEN GOTO 330           Если 60 поколений (размер экрана)
                                    то 330
310 GOTO 140                        На 140 (продолжение работы)
330 LOCATE 24,1:                    Продолжать работу с данной
    PRINT "NEXT SCREEN";:INPUT A$   популяцией (следующий экран)
340 IF A$="y" THEN GOTO 135         Если да (y), то 135
350 LOCATE 24,15: PRINT "TRY AGAIN": Продолжать работу с новой
                INPUT A$            популяцией
360 IF A$="y" THEN GOTO 2 ELSE STOP Если да (y), то 2, иначе конец.
380 FOR J=1 TO M(I1,Y1)             ------Подпрограмма определения
382 JJ=(I1-1)*3+Y1                  потомков их M(I1,Y1).
383 IF L1>100 THEN GOTO 420         Проверка ограничения на размер
                                    популяции
385 RR=RND: IF RR<P(JJ) THEN R1=0:  Определение "жив/мертв" для
                GOTO 397            потомка."Мертв" на 397.
390 R1=INT(F(JJ)*RND)+1             "Жив" - определение его типа
                                    (цвета)
397 B(L1)=R1                        переход к следующей паре
400 L1=L1+1                         родителей
410 NEXT J
420 RETURN                          ----------
```

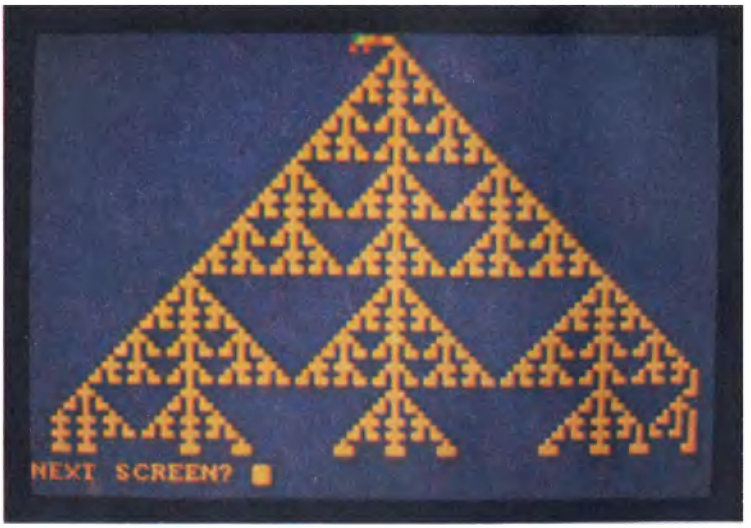


Рис. 34. Результаты работы программы ГЕНЕТИКА (CENETIC)

Если Вам не нравятся стандартные правила создания исходной популяции и правила появления следующих поколений, то Вы можете предложить программе свои правила, стоит только ответить на запрос программы «Стандартные правила?» — «N» — нет. В этом случае программа предложит вам самим заполнить всю таблицу (табл. 1), которая изначально будет пустой. Таблица должна заполняться по строкам: в первых двух колонках строки проставьте коды «родителей», затем число «сыновей» у этих «родителей», вид сыновей и, наконец, вероятность их исчезновения в следующем поколении. После заполнения таблицы нужно ответить на вопросы, как и в случае стандартных правил.

Итак, Вы рассмотрели много разных программ. Теперь предлагаем Вам проверить свои силы в самостоятельном построении игровой программы.

НЕБОЛЬШОЕ УПРАЖНЕНИЕ

Сейчас мы представим Вам программу, которая может «мысленно» завязывать и развязывать узлы на веревке. Мысленно — потому, что у персонального компьютера нет возможности, как у человека, проделать это руками.

Рис. 35. Изображение простейшего узла

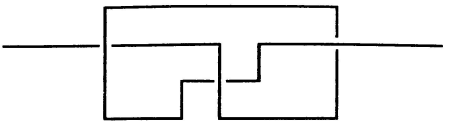


Рис. 36. Изображение сложного узла

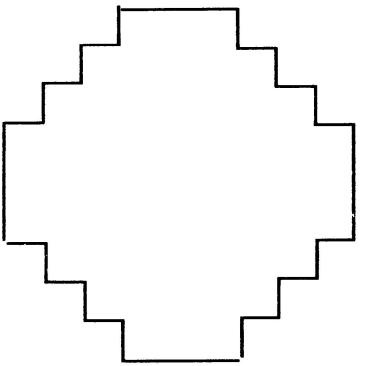


Рис. 37. Изображение кольца — тривиального узла

Например, узел, показанный на рис 35, может завязать каждый Это, пожалуй, самый простой узел, и завязывать или развязывать его можно даже мысленно Но если посмотреть на изображение узла, показанного на рис 36, то сразу и не скажешь, что будет, если растянуть веревку в кольцо (рис 37), завяжется ли на ней узел

Так вот, программа осуществляет «мысленное» растягивание кольца и определяет, завяжется на кольце узел или нет. Для этого ей нужно нарисовать изображение веревки, перепутанной определенным образом, как показано на рис. 35. Затем персональный компьютер проанализирует это изображение и скажет, получатся ли на веревке узлы.

Таким образом, перед программой, когда она запускается в работу командой RUN (выполнить), ставится задача: определить, можно ли перевести, не разрывая, нарисованную Вами веревку в кольцо.

Как же программа делает то, что мысленно не может сделать человек. Дело тут вот в чем. Математики придумали такое понятие, как «инвариант узла». Инвариант узла — это алгебраическое выражение, значение которого не меняется, как бы ни запутывали узел. Поэтому доста-

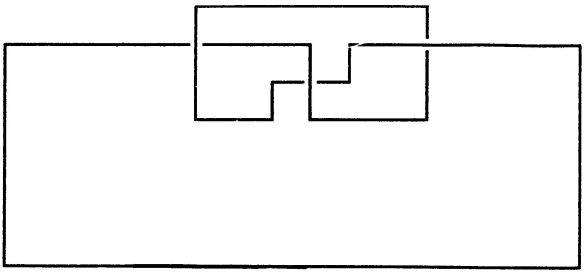


Рис. 38. Изображение простого, явно завязывающегося узла

точно определить инвариант для изображенного Вами узла и сравнить его со значениями инвариантов, найденными для узлов, вошедших в таблицу, которая вычислена заранее. Оказалось, что самыми удобными для вычислений являются многочлены Александера:

$\Delta(t)=1$ — для тривиального узла (рис. 38);
$\Delta(t)=t^2-t+1$ — для трилистника (рис. 39);

$\Delta(t)=t^2-3t+1$ — для восьмерки (рис. 40) и т. д., где Δ — обозначение полинома от аргумента t; t — аргумент (произвольная переменная).

Получается, что каждый узел характеризуется не отдельным числом, а алгебраическим выражением, в котором есть некоторая переменная, не имеющая специального смысла. Она введена только для удобства.

Вы можете поиграть с программой, задавая ей различные запутанные узлы любой сложности, она всегда опре-

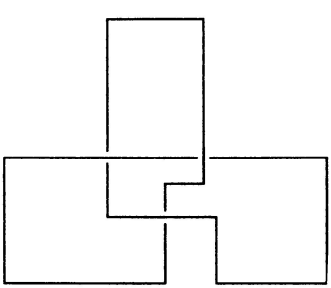


Рис. 39. Узел «трилистник»

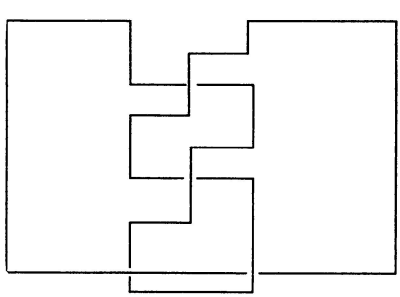


Рис. 40. Узел «восьмерка»

делит, к какому типу принадлежит нарисованный Вами узел. Конечно, Вы тоже сможете это сделать, если умеете вычислять для узла полином Александера. Поэтому всего лишь умение находить для конкретного узла этот полином дает в Ваши руки способность распутывать любые узлы, Гордиев узел, кстати, тоже.

Здесь специально не приводится текст программы, так как ничто так не развивает мозг и не приносит творческого удовлетворения, как поиск алгоритма и программы, позволяющей заполнить промежуток между исходными данными поставленной задачи и известным конечным результатом.

Авторы надеются, что приведенные тексты программ игр, а также пояснения к ним помогут заинтересованному читателю создать свою собственную концепцию игровой программы, которая может быть успешно реализована и имеет шанс завоевать популярность в компьютерном мире. При реализации приведенных программ на доступных читателю персональных компьютерах могут возникнуть некоторые сложности, источником которых является различие в диалектах языка программирования. Может быть, необходимо будет изменить ряд операторов программ, в основном в тех случаях, когда эти операторы отражают аппаратные особенности используемого персонального компьютера.

Приведенные в книге программы были реализованы на персональном компьютере типа AT в операционной среде DOS 3.0 и на диалекте Microsoft Quick BASIC 2.0.

Необходимо подчеркнуть, что «оформление» игры часто требует нестандартных средств языка, поэтому популярность игры сильно зависит от мобильности программы, т. е. возможности переносить ее с одной машины на другую. А это, в свою очередь, зависит от степени выхода операторов программы за «ядро» языка, мало отличающееся в различных диалектах.

Создание любой программы — творческий процесс. Каким является этот процесс при создании обучающих и тренирующих игр? Ответ мы попытаемся дать в следующей главе

6.
ТВОРЧЕСТВО, ЧТО ЭТО ТАКОЕ!

«Пусть познает каждый творец, что не плоды, а цветы его творчества нужны миру не результат, а творческий процесс»

Константин Эрберг «Цель творчества» 1919 г

Постоянный зуд познавательного синдрома во все времена заставлял человека соревноваться и сравнивать себя с объектами окружающего мира. Естественно, не мог он пройти и мимо исследования своей творческой деятельности... *«Сравнивая творческую деятельность с инстинктивной созидательной деятельностью животного, я прежде всего вижу, что животное — не творец. Природа поставила бобра, паука в такие условия, что он должен строить, должен ткать свою паутину. Бобр без строительных наклонностей — не бобр и паук без геометрических тенденций — не паук: иначе они осуждены на вымирание...*

Таким образом, строительная деятельность животного обусловлена властью природы, творческая же деятельность человека стремится от этой власти освободиться... Животное должно строить, Человек хочет творить...»

Итак, если творчество — это то, что свойственно только человеку, то почему все чаще и чаще слова «компьютер» и «творчество» используются совместно (например, в книге Д. Мичи и Р. Джонстона «Компьютер — творец»). Ответом могут служить следующие цитаты (Шерри Теркл, «Компьютер и Человек»):

1) *«До появления компьютера нашими ближайшими родичами в этом мире казались животные, создания смертные, хотя и не сознающие своей смертности. Сейчас на это место претендуют компьютеры, способные общаться друг с другом, обладающие как бы психологией и задатками разума. Дети, играющие теперь с компьютерами, пытаясь определить самих себя, ищут качества, отличающие их не от животных, а от компьютеров. Раньше мы были разумными животными, теперь сделались эмоциональными компьютерами, чувствующими машинами...»*,

2) *«Компьютеры — это не просто экраны, на которые проецируется личность, они уже сделались фактором, формирующим новое поколение. Для взрослых и для детей, увлеченных электронными играми, для всех, кто с помощью компьютеров манипулирует словами, информацией, зрительными образами, в особенности же тех, кто учится программировать, компьютеры становятся фактором, формирующим их личность, их „я".*

То есть в настоящее время заметно смещение акцента с темы диспута „могут ли машины мыслить?" на тему „могут ли компьютеры творить"».

Ответ на этот вопрос вытекает из исследования творческого процесса. Наличие у человека творческого начала вовсе не противоречит признанию за человеческим мозгом способности к анализу и упорядочению. Психологи установили, что человеку свойственны два вида мышления: конвергентное, которое систематически и строго стремится к получению ответа, и дивергентное, устремленное одновременно в разных направлениях, стремящееся не к заданной цели, а к поискам новых путей. Конвергентное мышление несостоятельно, когда существующая модель не охватывает решаемую задачу, создание новой модели требует дивергентного мышления. Итак, один вид мышления устанавливает причинно-следственные связи и приводит к предсказуемым выводам на уже известной основе. Другой вид — дивергентное мышление — приводит в ряде случаев к качественным изменениям исходной модели (например, отличия модели мира Эйнштейна от модели мира Ньютона).

По-видимому именно дивергентные механизмы мышления позволяют устранить существенные ограничения формальной конечности аппарата моделирования, которые следуют из известной теоремы Геделя о неполноте.

Гедель исследовал систему формальной арифметики при помощи созданного им метода кодировки Его исследование привело к следующему результату Когда системе кодировки предъявляется для доказательства формула, которая представляет собой метавысказывание о собственной недоказуемости, возникает противоречивая ситуация, приводящая к выводу о том, что если формальная арифметика не противоречива, то она не полна А если она противоречива, тогда ее теоремы теряют всякую ценность, поскольку в этом случае показывается, что можно доказать любую наперед заданную теорему и ее отри цание также

И цель творчества — в том, чтобы научиться отключать здравый смысл и включать воображение — дивергентное мышление.

В этом смысле компьютер — неоценимый помощник, аккумулирующий и усиливающий воображение. Более того, можно разработать специфические программы, порождающие на дисплее компьютера фантасмагорические изображения, которые не удается прогнозировать заранее и класс которых может быть установлен лишь последовательным экспериментированием.

Ранее мы уже рассмотрели части таких программ, которые были созданы А. И. Алексеевым и В. А. Ерохиным. Следующая программа разработана С. Н. Мысько. Она реализована на перспективном для персональных компьютеров языке С. В ее основе лежит простая идея проверки условия «ЕСЛИ ... ТО», которая, кстати, лежит и в основе разработок интерактивных информационных систем типа экспертных систем и баз данных. Здесь приведен фрагмент текста программы LOP.C на языке С, на рис. 41 показаны результаты ее работы. Каждый, кому доступна работа на персональном компьютере, может самостоятельно проверить построение программ, создающих на основе строгих логических формальных правил почти непредсказуемые картины фантастического и вместе с тем очень реального компьютерного мира.

Фрагмент текста программы LOP.C, написанной на языке С

```
mmpe2
/*-----------------------------------*/
int_try_rules ( num_of_rules, p_rule0 )

int num_of_rules;                    Функция последовательно прове-
struct rule dclr frame *p_rule0;     ряет условия правил-продукций
                                     и, в случае их выполнения,
{                                    инициализирует соответствующие
int i, fire_flag= 0;                 действия из правила.
struct rule dclr frame *p_rule;
int  swtch,  mode=0;

     marker( xx, yy );
for ( i=1, p_rule= p_rule0; i < num_of_rules;
                  i++, p_rule++)
     {
```

```
      if ( (*(p_rule ->p_cond)) ( p_rule->
                  p_cond_argm) == TRUE )
        {
        (*(p_rule->p actn)) ( p_rule->
                  p_actn_argm);
        fire_flag= 1;
      readlocator( &xx, &yy, &swtch );
      while( !(swtch==130 :: swtch==2) )
      {
      readlocator( &xx, &yy, &swtch );
      marker( xx, yy );
      }
startgraphics( &mode );
      setipal ( &xx, &yy );
corrent_context= (xx%300)*(yy%200);
        }
      }
return (fire_flag);
}
/*-------------------------------------*/
/*-----------------------------------------*/
actn_part_rule_001 ( p_argm )          Функция выполняет действия
char *p_argm;                          правила № 1
{
int *p_corrent_context, i;
int x1, y1, x2, y2;
int cx= 160, cy= 100;
int modem=8;
p_corrent_content= (int *) p_argm;
(*p_corrent_context)-=3;
x1= cx- *p_corrent_context ;
x2= cx+ *p_corrent_context ;
y1= cy- *p_corrent_context ;
y2= cy+ *p_corrent_context ;

movabs ( &xx, &yy );
setcolor ( p_corrent_context ) ;
for (i= -30; i (=30; i+=3 )
{
      x1= x1+i; x2= x2+1;
      y1= y1 +i/2; y2= y2 +i/2;
      box( &x1, &y1, &x2, &y2 );
fcir ( &x1 );
lnabs ( &x1, &y2 );
```

```
}
moveto ( &xx, &yy, p_mem, &modem );
return (1);
}
/*-------------------------------------*/
/*-----------------------------------------*/
actn part rule oo2 ( p_argm )          Функция выполняет действия
char *p_argm;                          правила № 2
{
int *p_corrent_context, i, s;
int x1, y1, x2, y2;
int cx= 160, cy= 100, col= 1;
int modem= 6;

p_corrent_context= (int *) p_argm;
(*p_corrent_context)+=4;
x1= cx- *p_corrent_context;
x2= cx+ *p_corrent_context;
y1= cy- *p_corrent_context;
y2= cy+ *p_corrent_context;

s=1; setxor( &s );
setcolor ( &col );
for ( i= 0; i <= (xx%300); i+=4)
{
      x1= x1 - i/2; x2= x2 + i/2;
      y1= y1 - i/3; y2= y2 + i/3;
cx= xx+x1; cy= yy-y1;
movabs ( &cx, &cy );
fcir ( &i );
lnabs ( &x1, &y2 );
x1%=300; x2%=300; y1%=200; y2%=200
 bar( &x1, &y1, &x2, &y2 );
}
movefrom ( &x1, &y1, & x2, &y2, p_mem );
moveto ( &xx, &yy, p_mem, &modem );
s=0; setxor ( &s );
        return (1);
}
/*-------------------------------------*/
/*-----------------------------------------*/
actn_part_rule_003 ( p_argm )          Функция выполняет действия
char *p_argm;                          правила № 3
{
```

```
int *p_corrent_context, i, s;
int x1, y1, x2, y2;
int cx= 160, cy= 100, col= 2;
int modem= 8;

p_corrent_context= (int *) p_argm;
(*p_corrent_context)+=7;
x1= cx- *p_corrent_context;
x2= cx+ *p_corrent_context;
y1= cy- *p_corrent_context;
y2= cy+ *p_corrent_context;

setcolor (&col);
s=1; setxor ( &s );
for ( i= 0; i (= 70; i+=3)
{
      x1= x1 - i/2; x2= y2 + i/2;
      y1= y1 - i/4; y2= x2 + i/4;
cx= xx%320; cy= yy%200 - y1%200;
movabs ( &cx, &cy );
fcir ( &i );
bar ( &xx, &yy, &x2, &y2 );
movabs ( &xx, &yy );
x2%=300; y2%=yy;
lnabs ( &x2, &y2 );
fcir ( &i );
}
moveto ( &xx, &yy, p_mem, &modem );
s=0; setxor ( &s );
return (1)
}
/*---------------------------------*/

/*-----------------------------------------*/
actn_part_rule_004 (p_argm)          Функция выполн
char *p_argm;                        правила № 4
{
int *p_corrent_context, i, s;
int x1, y1, x2, y2, mode=0;
int cx= 160, cy= 100, col= 3;
int modem= 4;

p_corrent_context= (int *) p_argm;
(*p_corrent_context)*=4;
x1= cx- *p_corrent_context;
```

```
x2= cx+ *p_corrent_context;
y1= cy- *p_corrent_context;
y2= cy+ *p_corrent_context;

s=1; setxor( &s);
setcolor ( &col );
for ( i= 0; i (= 40; i+=2)
{
      x1= x1 - i x2= x2 + i;
      y1= y1 - i/2; y2= y2 + i/2;
cx=(cx + xx)%320; cy=(cy + yy)%200;
movabs ( &cx, &cy );
fcir ( &i );
startraphics( &mode );
      setipal ( &xx, &i );
movabs ( &cx, &cy );
fcir ( &i );
bar( &x1, &y1, &x2, &y2 );
}
movefrom ( &x1, &y1, &x2, &y2, p_mem );
s=0; setxor( &s );
return (1);
}
/*-------------------------------------*/
/*-------------------------------------*/
cond_part_rule_001 (p_argm )          Условие правила №: 1
char *p_argm;
{
int *p_corrent_context;

p_corrent_context= (int *) p_argm;

if (   ((*p_corrent_context % 2) == 0) ::
      ((*p_corrent_context % 5) == 0) ::
      ((*p_corrent_context % 7) == 0) ::
      ((*p_corrent_context % 13) == 0)
      return (TRUE); else return (FALSE);
}
/*-------------------------------------*/
/*-------------------------------------*/
cond_part_rule_002 (p_argm )          Условие правила №: 2
char *p_argm;
{
int *p_corrent_context;
```

```
p_corrent_context= (int *) p_argm;

if (  ((*p_corrent_context % 3) == 0) ::
      ((*p_corrent_context % 5) == 0) ::
      ((*p_corrent_context % 11) == 0) ::
      return (TRUE); else return (FALSE);
}
/*-------------------------------------*/
/*-----------------------------------------*/
cond_part_rule_003 (p_argm )          Условие правила №: 3
char *p_argm;
{
int *p_corrent_context;

p_corrent_context= (int *) p_argm;
if (  ((*p_corrent_context % 17) == 0) ::
      ((*p_corrent_context % 19) == 0) ::
         ((*p_corrent_context % 23) == 0)
      return (TRUE); else return (FALSE);
}
/*-------------------------------------*/
/*-----------------------------------------*/
cond_part_rule_004 (p_argm )          Условие правила №: 4
char *p_argm;
{
int *p_corrent_context;

p_corrent_context= (int *) p_argm;

if (  ((*p_corrent_context - 1) == 0) ::
      return (TRUE); else return (FALSE);
}
/*-------------------------------------*/
```

Анализ структур и форм реализации на персональных машинах различных компьютерных игр позволяет сделать некоторые обобщения, которые мы предлагаем тем, кто хочет попробовать свои силы в создании подобных игр.

При разработке различных программ для персональных компьютеров мы хотели разобраться в вопросе влияния обучения в игровой форме на саму идею игры и ее структуру и, наоборот, в вопросе влияния игрового компонента на состав и структуру обучающих программ.

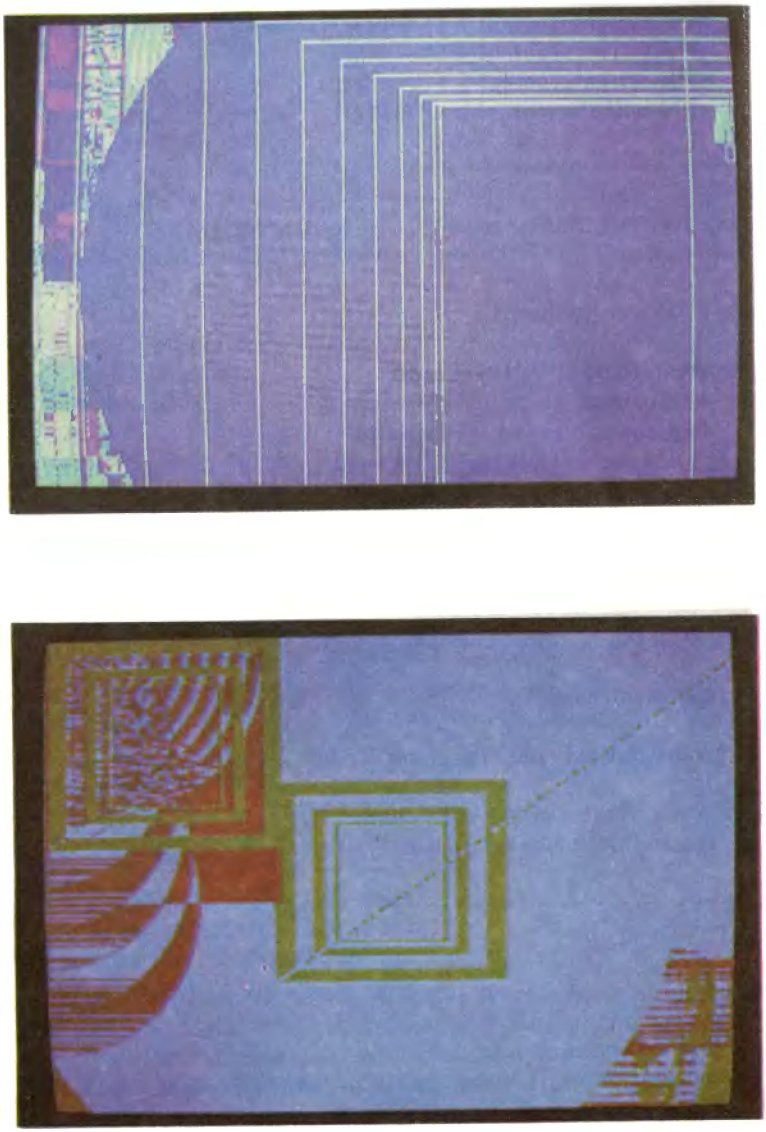
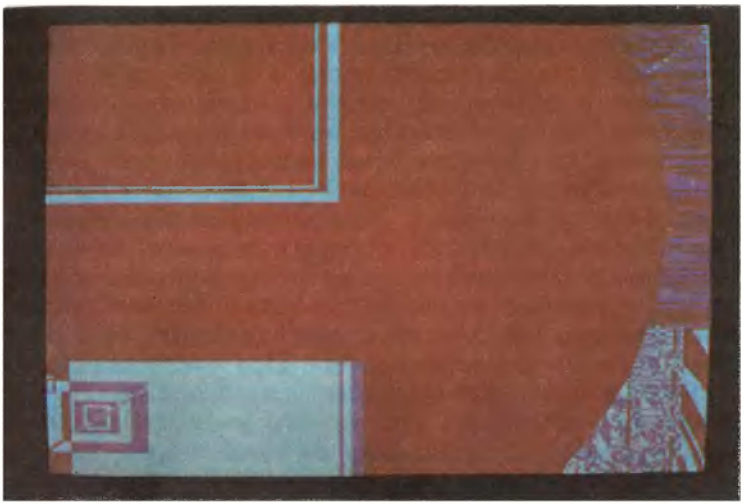
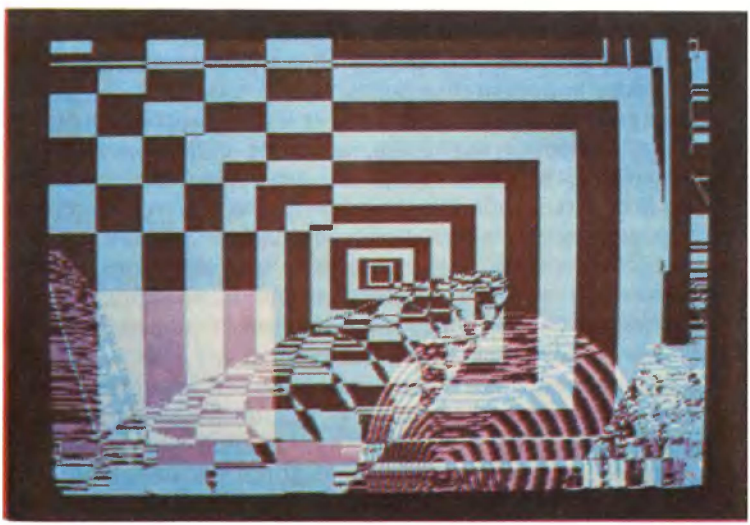


Рис. 41. Результаты работы





программы LOP.C на языке C

Нет необходимости еще раз подчеркивать, что обучение с использованием ЭВМ очень популярно в настоящее время. Хочется отметить, что если игра является хорошей (по общему признанию всех, кто играл в нее), то она зачастую является настолько же хорошим обучающим упражнением. С другой стороны, хорошее обучающее упражнение, в результате однократного или многократного выполнения которого достигается цель обучения, всегда близко к игре по своей форме, темпу выполнения и эмоциональному содержанию. Упражнения на языке Бэйсик, к которым чаще всего сводятся программы обучения языку программирования, обычно служат для повторения пройденного материала и проверки его усвоения, но в выполнении упражнений часто нет ничего увлекательного и это резко снижает их эффективность. При работе на персональном компьютере внимание привлекает не столько конкретная программа, сколько сам компьютер. Дело в том, что, работая с персональным компьютером, Вы учитесь подчинять своей воле и разуму произвольные образы на экране компьютера, можете наблюдать, как они воплощаются на экране, корректировать их форму, изменять цвет, формировать тени и освещение, тем самым создавать свой собственный компьютерный мир. Использование программы как игры постоянно обращается к Вашей инициативе и комфортно «заставляет» Вас запоминать, а иногда и создавать новое знание.

Придумать игру, разработать ее правила, создать интересно работающий алгоритм, а затем синтезировать по нему программу — все это очень непростые вопросы, которые все же необходимо решить, чтобы игра стала по-настоящему увлекательной и тем самым в значительной степени обучающей. Надо сказать, что желание играть в какую-либо игру в высшей степени индивидуально, и никакое внешнее давление не может гарантировать появление увлеченности игрой.

Чем же помогают игры обучению и помогают ли? Да, помогают и прежде всего — подниматься по крутым ступеням познания. Приобретение знаний — это обычно тяжелый рутинный труд. Вообще дорога к познанию открыта для каждого и, чтобы продвигаться к цели, необходимо и достаточно упорно идти по ней. Путь этот может оказаться длинным, и в этом случае познание окажется очень утомительным. Но если приложить усилия, то всегда можно дойти до цели. Когда на дороге к знаниям

встречаются разнообразные препятствия или, например, полностью меняется «рельеф учебной местности», то уже недостаточно просто шагать по дороге к знанию, а нужно перепрыгнуть, обогнуть препятствие или найти другое средство передвижения. Именно в таких случаях очень помогают обучающие компьютерные игры.

Играть — значит согласиться на определенные изменения. В ролевой игре ее участники добровольно отказываются от своих личных мнений, предрассудков и эгоистических интересов, чтобы временно (хотя бы и временно!) стать на позицию другого. И вообще в целом каждая игра представляет собой временное забывание привычных взаимоотношений, способ выражения эмоций и суждений. А такие перемены, эта готовность изменить свое мышление и есть как раз основа творчества, открытия и познания!

Использование игр в обучении позволяет сказать, что в процессе тесного игрового взаимодействия с компьютером мы лучше осознаем и познаем самих себя. Персональные компьютеры, таким образом, являются мощным средством индивидуального использования, и поэтому компьютерные обучающие игры создают идеальные возможности для поддержки и развития наглядно-образного мышления. Кроме того, игра с компьютером помогает нам самостоятельно исследовать взаимодействие и функционирование различных объектов реального мира через их моделирование и имитацию работы.

Можно утверждать, что персональные компьютеры, как и телевидение несколько десятков лет назад, представляют собой новое значительное явление в обществе. В связи с быстрым ростом числа производимых компьютеров, их широким распространением во всех областях человеческой деятельности возникают проблемы их эффективного использования в обучении. Компьютерные обучающие игры позволяют смягчить процесс усвоения новых знаний, сделать такой процесс интересным и увлекательным и тем самым создать прочный фундамент для активизации творчества и познания.

ОГЛАВЛЕНИЕ

НАУЧНО ПОПУЛЯРНОЕ ИЗДАНИЕ

Александров Виктор Васильевич,
Алексеев Андрей Игоревич,
Семенков Александр Иванович

ЭВМ: игра и творчество

Редактор *В М Рошаль*
Художественный редактор *С С Венедиктов*
Технический редактор *А И Казаков*
Корректор *А И Лавриненко*
Обложка художника *В И Коломейцева*

ИБ № 5868

Сдано в набор 16 09 88 Подписано в печать 10 07 89 М-29109 Формат 84×108$^{1}/_{32}$. Бумага офсетная № 1 Гарнитура литературная Печать офсетная Усл печ л 6,72 Усл кр-отт 27,88 Уч-изд л 7,5 Тираж 50 000 экз Заказ 1384 Цена 60 коп

Ленинградское отделение ордена Трудового Красного Знамени издательства «Машиностроение» 191065, Ленинград, ул Дзержинского, 10
Предприятие малообъемной книги дважды ордена Трудового Красного Знамени Ленинградского производственного объединения «Типография им Ивана Федорова» Союзполиграфпрома Государственного комитета СССР по делам издательств, полиграфии и книжной торговли 192007, Ленинград, Боровая ул , 51

60 коп.

ЭВМ: игра и творчество

Мир ЭВМ... Таинственный и притягательный, недоступный и такой знакомый. Как войти в него, как сделать его привычным и простым? Помочь в этом могут компьютерные игры. С их помощью легко, непринужденно Вы войдете в компьютерные города и леса, сможете дружески общаться с жителями заэкранного мира, и он станет для Вас понятным и управляемым. С помощью игры и через нее Вы начнете творить свой мир...

Предлагаемая книга требует от читателя некоторых усилий и тренинга мозговой деятельности. Игра на ЭВМ — не самоцель, а творческий процесс, развивающий интеллект и интенсифицирующий наши способности к познанию окружающего мира.

МАШИНОСТРОЕНИЕ