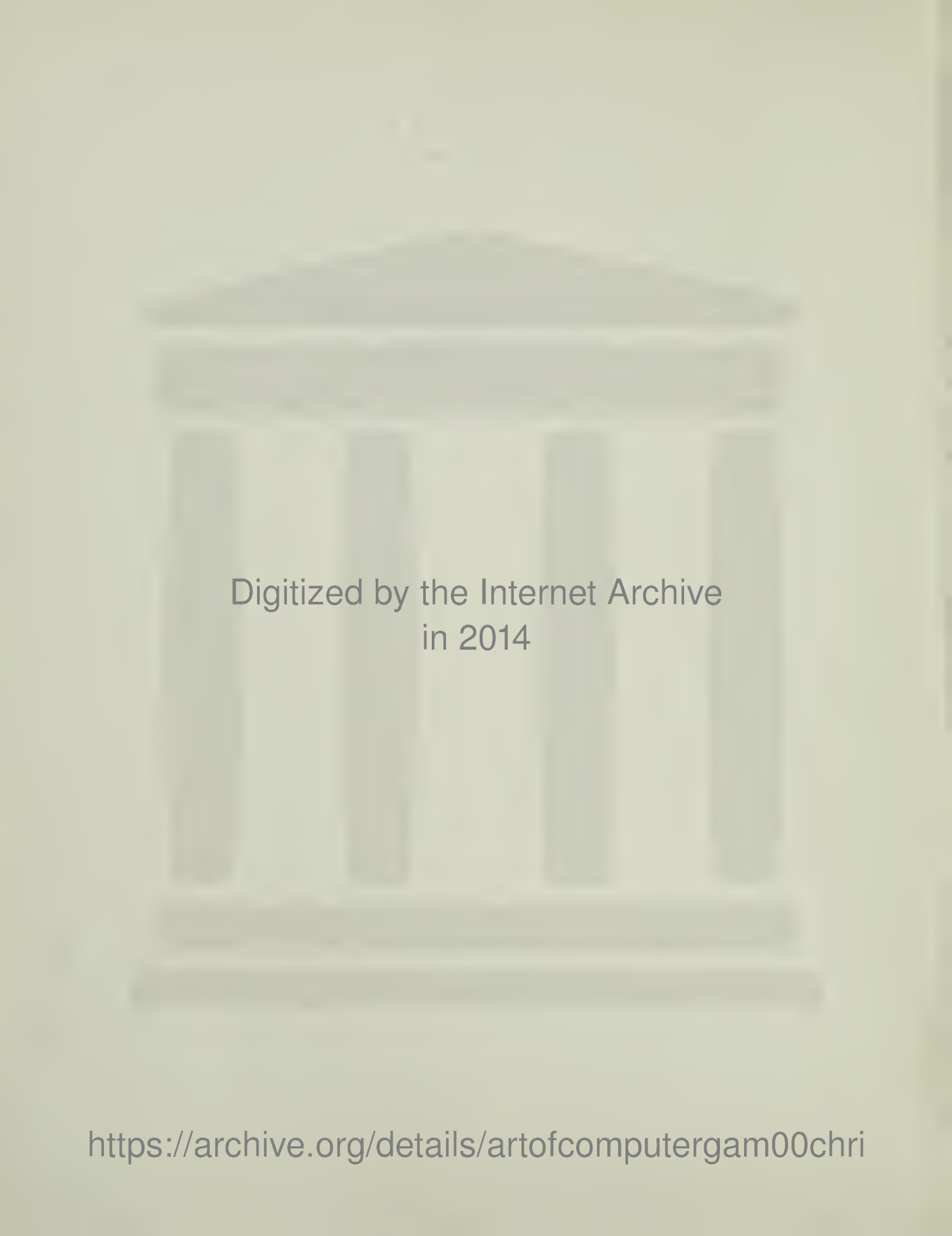
Osborne/McGraw-Hill

THE ART OF COMPUTER GAME DESIGN

REFLECTIONS OF A MASTER GAME DESIGNER

Chris Crawford

THE ART OF COMPUTER GAME DESIGN



THE ART OF COMPUTER GAME DESIGN

Chris Crawford

Osborne/McGraw-Hill
Berkeley, California

Published by
Osborne/McGraw-Hill
2600 Tenth Street
Berkeley, California 94710
U.S.A.

For information on translations and book distributors outside of the U.S.A., please write to Osborne/McGraw-Hill at the above address.

THE ART OF COMPUTER GAME DESIGN

1234567890 DODO 89876543

ISBN 0-88134-117-7

Judy Ziajka, Acquisitions Editor
Paul Jensen, Technical Editor
Richard Sanford, Copy Editor
Judy Wohlfrom, Text Design
Yashi Okita, Cover Design

TRADEMARKS

The capitalized trademarks are held by the following alphabetically listed companies:

PREPPIE!	Adventure International
TEMPEST	Atari, Inc.
MISSILE COMMAND	
RED BARON	
PONG	
STAR RAIDERS	
SPACE WAR	
ASTEROIDS	
CENTIPEDE	
BATTLEZONE	
CAVERNS OF MARS	
YAR'S REVENGE	
MAZE CRAZE	
DODGE 'EM	
BREAKOUT	
SUPERBREAKOUT	
CIRCUS ATARI	
WARLORDS	
AVALANCHE	
NIGHT DRIVER	
SUPERMAN	
HAUNTED HOUSE	
EASTERN FRONT 1941	
SCRAM	
ENERGY CZAR	
COMBAT	
EXCALIBUR	
ADVENTURE	

CRUSH, CRUMBLE, AND CHOMP	Automated Simulations
LEGIONNAIRE BLITZKRIEG WATERLOO AFRIKA KORPS TANKTICS	Avalon Hill
APPLE PANIC	Broderbund Software
GALAHAD AND THE HOLY GRAIL	Douglas Crockford
TEMPLE OF APSHAI	EPYX
MATCH RACER	Gebelli
DOG DAZE	Gray Chang
DEADLINE	Infocom
ROCKY'S BOOTS	The Learning Company
PAC-MAN	Namco
MONOPOLY	Parker Brothers
ALI BABA AND THE FORTY THIEVES	Quality Software
DOWNHILL	Mark Reid
CROSSFIRE JAWBREAKERS MOUSKATTACK THE WIZARD AND THE PRINCESS TIME ZONE	Sierra On-Line Systems
WAR IN THE EAST BATTLE FOR GERMANY RUSSIAN CIVIL WAR	Simulations Publications
COMPUTER BISMARK COMPUTER AMBUSH COMPUTER NAPOLEONICS	Strategic Simulations
CHICKEN	Synapse Software
DUNGEONS AND DRAGONS	TSR Hobbies
SPACE INVADERS	Taito America, Inc.

ACKNOWLEDGMENT

I am deeply indebted to Madeleine M. Gross for her painstaking and thorough criticisms of this book. In many cases she invested greater efforts into her criticisms than I had put into my original thoughts. She strove to restrain my wild hyperbole and place my arguments on a firmer foundation of rigorous logic. The logical consistency and reliability in this book I owe to her; the speculative flights of fancy must be laid at my doorstep.

ABOUT THE AUTHOR

Chris Crawford began amateur game designing in 1972,but it was not until 1978 that he offered his first game, TANKTICS, to the public. It became the first commercially available wargame. In 1979 Crawford joined Atari, Inc., as a game designer and began a prolific career. He is the author of ENERGY CZAR (1980), SCRAM (1981), TANKTICS (1978, 1981), EASTERN FRONT (1981, 1983), LEGIONNAIRE (1982), EXCALIBUR (in press), and GOSSIP (in press). In addition to game design, he is also the editor and primary author of *De Re Atari*, a tutorial guide to the Atari home computers, as well as the author of more than 20 magazine articles. Still in the vanguard of game design, he now leads the Games Research Group at Atari, Inc.

CONTENTS

PREFACE

The central premise of this book is that computer games constitute a new and poorly developed art form that holds great promise for both designers and players.

At first, this premise may seem laughable. How can SPACE INVADERS and PAC-MAN be classified as art? How can TEMPEST or MISSILE COMMAND compare with Beethoven's Fifth Symphony, Michaelangelo's *Pieta*, or Hemingway's *A Farewell to Arms?* Computer games are too trivial, too frivolous to be called art; they are idle recreation at best. So says the skeptic.

But we cannot relegate computer games to the cesspit of pop culture solely on the evidence of the current crop of games. The industry is too young and the situation too dynamic to dismiss computer games so easily. We must consider the potential, not the actuality. We must address fundamental aspects of computer games.

There are many definitions of art, few of which make much sense to the nonartist. What I am interested in is the way art evokes emotion through fantasy. The artist presents the audience with a set of sensory experiences that stimulates commonly shared fantasies and thus generates emotions. Art is made possible by the richness of the fantasy world we share. Art is nevertheless difficult because there are so many practical problems associated with stimulating fantasies deep inside another person's mind.

A major problem is that most art forms allow very little active audience participation. You sit quietly and listen to music that other people created and perform, or you stroll through museums and stare at pictures or statues other people made. You sit passively and read a novel, a poem, or a short story. In all these art forms, the role of the audience is passive. The artist does all the active work, makes the greatest emotional investment. The audience is expected to absorb quietly the fruits of the artist's labors.

This is not a criticism of art or artists. The technologies of most art forms preclude active audience participation. If every klutz jumped into the orchestra pit or pranced on the opera stage or slopped paint with Picasso, we might have some great parties but no art. It seems the curse of most forms of art that artists can say so much in their work and most people will hear so little because, locked in the role of passive audience, they cannot actively participate.

Enter the computer. Conceived long ago, born in war, reared as the servant of business, this now-adolescent technology has exploded out of the computer room and invaded shopping centers, pizza parlors, and homes. Popular characterizations of the computer alternate between the old image of the computer as an omniscient, cold-blooded, giant calculator and the new image of the computer as the purveyor of video thrills and 25-cent fixes.

Originally developed as a number-cruncher, the computer assumed a new personality when it was given graphics and sound capabilities. These capabilities made the computer more powerful: it could now communicate with human beings, not just in the cold and distant language of digits, but in the emotionally immediate and compelling language of images and sounds. With this ability came a new, previously undreamed-of possibility: the possibility of using the computer artistically as a medium for emotional communication. The computer game has emerged as the prime vehicle for this communication.

Unfortunately, the current generation of microcomputers cannot produce a sensory experience as rich as that produced by a symphony orchestra or a movie. This weakness is more than offset by a fundamental advantage over most art forms: a game is intrinsically participatory. The game designer has here a tool that is more subtly indirect than traditional art. In traditional art forms, the artist directly creates the experience that the audience encounters. Because this experience is carefully planned and executed, the audience must somehow be prevented from disturbing it. In a game, the designer creates not the experience itself but the conditions and rules under which the audience will create its own individualized experience. The demands on the game artist are greater than those on other artists, for the game artist must plan the experience indirectly, taking a special interest in the probable and possible actions and reactions of the audience. In turn, participation increases audience attention and heightens the intensity of the experience.

When we passively observe someone else's artistic presentation, we derive some emotional benefit, but when we actively participate in a game, we involve a portion of ourselves in the fantasy world of the game. This greater participation yields a commensurately greater return of emotional satisfaction. Indeed, participation is so important that many people derive greater satisfaction from participating in an amateur artistic effort than from merely observing a professional effort. For this reason, games, being intrinsically participatory, present the artist with a fantastic opportunity for reaching people.

Until now, games in general, and computer games in particular, have not been impressive as art forms. The computer games, especially, are downright puerile. This is because the technology of computer games has been in the hands of technologists, not artists. These people can write beautiful operating systems, languages, linking loaders, and other technological wonders, but artistic flair has generally been treated as if it were less important than technical prowess.

Another contributor to the fecklessness of current computer games is the timidity of the marketplace. Computers are new; the public is unfamiliar with them, and the manufacturers are hesitant to press the public too hard too fast. We in the industry, therefore, opt to build inhibited little games that evoke pathetically trivial emotions. Truly intense emotions or situations of pathos, ecstasy, majesty, rapture, catharsis, or tragedy intimidate us. We hide behind the idea that we are in the entertainment business, not in the art business, but that defense only betrays a profound misunderstanding of art. Art can be stuffy and elitist, but good art can also be a foot-stomping blast.

Fortunately, times are changing. Already we see a backlash against computer games. It expresses itself in many ways: in ordinances against the placement of arcade games in some areas, in statements by educators denouncing the games, and in more vigilant regulation of children's game activities by parents. This response is viewed anxiously by smaller-minded members of the industry. More visionary thinkers watch the backlash with eager interest. The American people are telling us something here, something very important, something important enough that they are willing to compromise their traditional reluctance to interfere with other people's business.

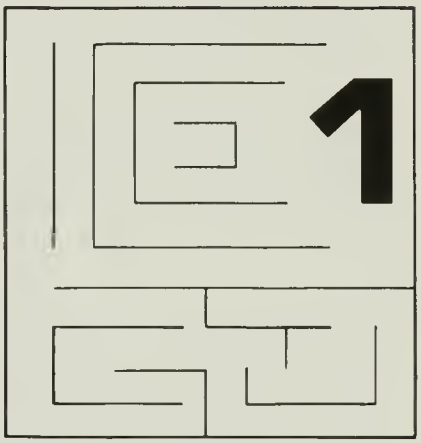
Although the arguments against video-game parlors presented in public debates normally focus on issues such as delinquency from school, large groups of rowdy teenagers, and so forth, the concerns

expressed privately reflect the distaste for the games, a feeling that the games are a waste of time. People are beginning to realize that the world of computer games is a vast wasteland.

Computer games are much like candy, comic books, and cartoons. They all provide intense or exaggerated experiences. Whether they use sugar, exclamation points, or animated explosions, the goal is the same: to provide intense experience. Children appreciate these activities because their novelty is still strong. Adults, jaded by years of experience, prefer diversions with greater subtlety and depth. We thus have literature, culinary arts, and movies as the adult counterparts to comic books, candy, and cartoons. Yet we have no adult counterpart to computer games. This deficit suggests great opportunities in computer game design.

This developing revolution in game design has nothing to do with the rapid hardware developments of the last few years. While technological improvements will surely continue, we are no longer hampered primarily by the limitations of the hardware. Our primary problem is that we have little theory on which to base our efforts. We don't really know what a game is or why people play games or what makes a game great. Real art through computer games is achievable, but it will never be achieved as long as we have no principles of aesthetics, no framework for criticism, and no model for development. New and better hardware will improve our games, but it will not guarantee our artistic successes any more than the development of orchestras guaranteed the appearance of Beethoven. We are a long way from a computer game comparable to a Shakespeare play, a Tchaikovsky symphony, or a Van Gogh self-portrait. Each of these artists stood on the shoulders of earlier artists who plunged into an unexplored world and mapped its territories so that later artists could build on their work and achieve greater things. We computer-game designers must put our shoulders together so that our successors may stand on top of them. This book is my contribution to that enterprise.

WHAT GAMES ARE AND WHY PEOPLE PLAY THEM



If we desire to understand games and game design, we must first define what we mean by the word *game*. We must also determine the fundamental characteristics of all games. After discussing some of the obstacles to this effort, I will briefly describe the principal classes of games. Then I will propose a set of attributes that characterize all games.

Games are a fundamental part of human life. The vocabulary of games has insinuated itself into our language and refers to activities that are not truly games. We "play along" with activities we find distasteful. We "play ball" with those who require our cooperation. We "play games" when we are insincere. A willing participant is "game for the enterprise." This broad penetration of gaming concepts into everyday experience presents us with two potential barriers to understanding games.

First, liberal use of gaming terms promotes an exaggerated idea of our own understanding of games. We fail to give the subject the careful and critical analysis that we reserve for more academic topics, and we blithely ignore the complexities of game design. Amateurs, whose only relevant skill is programming, undertake to design games with no further preparation than their own experience as game players. They overrate their own understanding of games and undercut their own potential for learning.

The second obstacle is ambiguity. We have applied the principles

and concepts of gaming so widely that we have diluted their original meanings. Game designers have no well-defined set of common terms with which to communicate. Discussions of game design frequently disintegrate into arguments over meanings of terms. To cut through the tangled undergrowth that has grown up around gaming, we shall need the bulldozer and the scalpel.

THE DEFINITION OF A GAME

Let us begin by stepping back for a moment and taking our bearings. I would like to take you on a brief tour of the universe of games, glancing briefly at each of the major regions. In the course of this tour, I hope to refresh your memory of games and make some simple points before digging into the serious analysis of fundamental game characteristics. I perceive four major categories of games: board games, card games, athletic games, and computer games.

Board Games

A board game consists of a playing surface divided into sectors populated by a set of movable pieces. In the most common arrangement, the pieces are directly controlled by the players, but the playing surface represents an environment beyond the player's direct control. Players maneuver their pieces across the playing surface in an effort to capture other players' pieces, reach an objective, gain control of territory, or acquire some valued commodity. The player's primary concern in these games is the analysis of geometrical relationships between the pieces.

Card Games

Card games use a set of 52 symbols generated from combinations of two factors: rank (13 values) and suit (4 values). Players may gain or lose possession of symbols either by random processes or by matching some combination allowed by the rules of the game. Each legal combination is assigned a victory value for final assessment of game results. Players must recognize both existing and potential combinations and estimate the probability of obtaining the cards necessary for completing

a combination. This probability must be weighed against the victory value of the combination. Because the number of combinations is very large, precise computation of the required probabilities exceeds the mental powers of almost all players, and the game becomes primarily an intuitive exercise. Thus the player's primary concern in these games is the analysis of combinations.

Athletic Games

Another traditional game form is the athletic game. These games emphasize physical more than mental prowess. The rules of the game strictly specify a precise set of actions that the player is either allowed or required to execute. Skillful use of the body is the player's primary concern.

We must be careful to distinguish between athletic games and athletic competitions. For example, a race is a competition, not a game. The difference between games and competitions emphasizes a fundamental element of all games. I distinguish the two by the degree of interaction among players. Theoretically, the runners in a race do not interact with each other. Each is racing only against the clock; the presence of other runners should be immaterial. In fact, the runners may interact psychologically, for the performance of one runner can affect the performance of the other runners. Furthermore, in some races a runner (or driver or pilot or captain) can physically interpose himself between the goal and another racer, thereby gaining an advantage. I conclude, however, that the simplest athletic contests, those in which each person strives to perform some task optimally without direct interaction with the other competitors, do not constitute games but competitions. A competition that does allow interaction is a game.

Computer Games

The next kind of game we will consider is the current fad in gaming and the subject of this book, the computer game. These games are played on five types of computers: expensive dedicated machines for the arcades ("coin-op" machines), inexpensive dedicated machines ("handhelds"), multi-program home game machines such as the ATARI 2600 and the ATARI 5200, personal computers, and large mainframe computers. The computer acts as opponent and referee in most of these games;

in many of them it also provides animated graphics. The most common form of computer game is the skill-and-action (S&A) game emphasizing hand-eye coordination. These S&A games are frequently violent. There are many other kinds of computer games: adventure games, fantasy role-playing games, and war games, to name a few. In our cursory overview, these other computer games are eclipsed by the sheer volume of the skill-and-action games.

This concludes our quick survey of the four categories of games. We shall return to the subject later to create a taxonomy of computer games and later still to analyze specific examples of games. We must now address the question that motivated this overview: what are the fundamental elements common to these games? I identify four common factors: representation, interaction, conflict, and safety.

REPRESENTATION

First, a game is a closed, formal system that subjectively represents a subset of reality. Let us examine each term of this statement carefully. By *closed* I mean that the game is complete and self-sufficient in its structure. The model world created by the game is internally complete; no reference need be made to agents outside the game. Some badly designed games fail to meet this requirement. They produce disputes over the rules by allowing situations to develop that the rules do not address. The players must then extend the rules to cover the new situation, and this causes arguments. A properly designed game precludes this possibility because the rules cover all contingencies encountered in the game.

Formal

By *formal* I mean only that the game has explicit rules. There are informal games in which the rules are loosely stated or deliberately left vague, but such games are not typical.

System

The term *system* is often misused, but in this case its application is quite appropriate: a game is a collection of parts that interact with each other, often in complex ways. It is a system.

Subjectively Represents

Representation is a coin with two faces: an objective face and a subjective face. The two faces are not mutually exclusive, for subjective reality springs from objective reality. In a game, these two faces merge, with emphasis on the subjective face. For example, when a player blasts hundreds of alien invaders, nobody believes that his recreation directly mirrors the objective world. However, the game may be a very real metaphor for the player's perception of the world.

I do not wish to sully my arguments with pop-psychological analyses of players giving vent to deep-seated aggressions at the arcades. Clearly, though, something more than the simple destruction of alien monsters is going on in the mind of the player. We need not concern ourselves with its exact meaning. For the moment, it is sufficient to realize that the player does perceive the game to represent something from his private fantasy world. Thus, a game represents something from subjective reality, not objective reality.

Games are objectively unreal because they do not physically recreate the situations they represent, yet those situations are subjectively real to the player. The agent that transforms an objectively unreal situation into a subjectively real one is human fantasy. Fantasy thus plays a vital role in any game. A game creates a fantasy representation, not a scientific model.

The distinction between objective representation and subjective representation is made clear by considering the differences between simulations and games. A simulation is a serious attempt to represent accurately a real phenomenon in another, more malleable form. A game is an artistically simplified representation of a phenomenon. The simulations designer simplifies reluctantly and only as a concession to material and intellectual limitations. The game designer simplifies deliberately in order to focus the player's attention on those factors the designer considers important.

The purposes of the two are fundamentally different. A simulation is created for computational or evaluative purposes; a game is created for educational or entertainment purposes. (There is a middle ground where training simulations blend into educational games.) Accuracy is the *sine qua non* of simulations; clarity the *sine qua non* of games. A simulation bears the same relationship to a game that a technical drawing bears to a painting. A game is not merely an inferior or partial

simulation, lacking the degree of detail that a simulation possesses; a game deliberately suppresses detail to accentuate the broader message that the designer wishes to present. Where a simulation is detailed, a game is stylized.

Consider, for example, the differences between a flight simulator program for the personal computer and the coin-op game RED BARON. Both programs concern flying an airplane; both operate on microcomputer systems. The flight simulator demonstrates many of the technical aspects of flying: stalls, rolls, and spins. RED BARON has none of these. Indeed, the aircraft that the player flies in RED BARON is quite unrealistic. It cannot be stalled, rolled, spun, or forced into the ground. When the stick is released, it automatically rights itself. It is incorrect to conclude from these observations that RED BARON is inferior to the flight simulator. RED BARON is not a game about realistic flying; it is a game about flying and shooting and avoiding being shot. The inclusion of technical details of flying would distract most players from the other aspects of the game. The designers of RED BARON quite correctly stripped out technical details of flight to focus the player's attention on combat. The absence of these technical details from RED BARON is not a liability but an asset, for it gives the game focus. Their absence from a flight simulator program would be a liability.

Subset of Reality

The last term I use is *subset of reality*. One aspect of this term (subset) is easily justified. Clearly, no game could include all of reality without being reality itself; a game must be, at most, a subset of reality. The choice of concerns within the subset provides focus to the game. A game that represents too large a subset of reality defies player comprehension and becomes almost indistinguishable from life itself, robbing the game of one of its most appealing features—its focus.

INTERACTION

Some media for representing reality are static. A painting or sculpture depicts a snapshot of reality frozen in time. Some media are dynamic; they show change with time. Movies, music, stories, and dance are dynamic in this way. They are able to represent changing reality

more richly than static representations like still photography. But the most fascinating thing about reality is how it changes, the intricate webwork of cause and effect by which all things are tied together. The best way to represent this webwork is to allow the audience to explore it fully—to let the audience generate causes and observe effects. Games provide this interactive experience, and it is a crucial factor of their appeal.

Games Versus Puzzles

One way to understand the interactive element of games is to contrast games with puzzles and other noninteractive challenges. Compare playing a cube puzzle with playing a game of tic-tac-toe. Compare the sport of high jumping with the game of basketball. The key difference that makes one activity a game and the other activity not a game is that one is interactive. A cube puzzle does not actively respond to a person's moves; a high jump bar does not acknowledge the jumper's efforts. Both tic-tac-toe and basketball involve opposing players who acknowledge and respond to one another's actions.

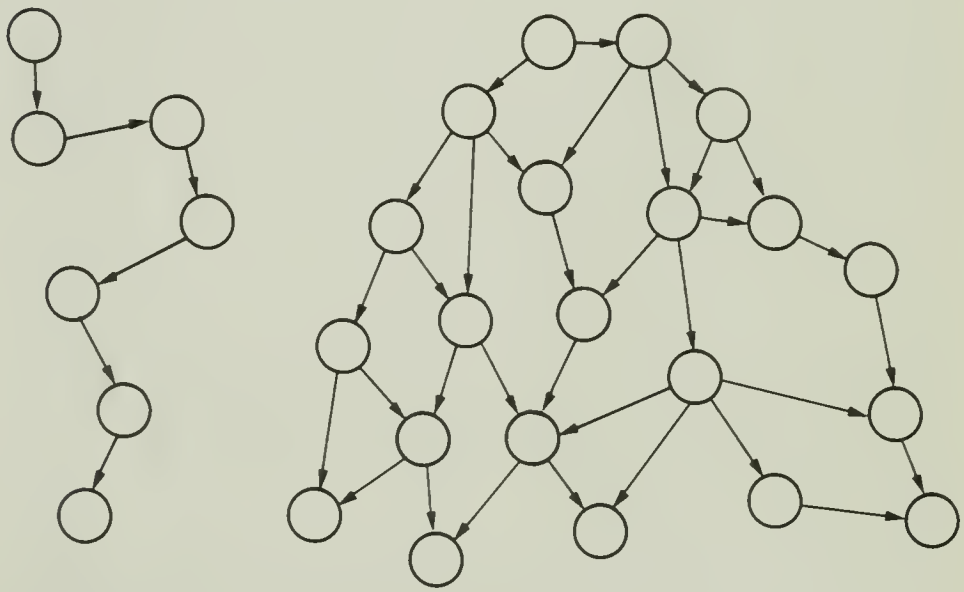
The difference between games and puzzles has little to do with mechanics; we can easily turn many puzzles and athletic challenges into games and vice versa. For example, chess, a game, has spawned a whole class of puzzles, the end-game problems. Games can include puzzles as subsets and many do. Most of the time such puzzles are a minor component of the overall game, because a game that emphasizes puzzles will rapidly lose its challenge once the puzzles have been solved.

Games Versus Stories

Another way to illustrate the role of interaction is to compare games with stories. A story presents a series of events in a time-sequence that suggests cause-and-effect relationships. These events are often deliberately fictitious. Indeed, the entire concept of fiction ("an untruth that is not a lie") is only valid because the facts presented in the fiction are secondary in importance. The cause-and-effect relationships suggested by the sequence of facts are the important part of the story. For example, we do not care whether Luke Skywalker and the Death Star really existed. Luke Skywalker was good and pure, and the Death Star was

evil, and Luke Skywalker destroyed the Death Star. The cause-and-effect relationship suggested by the story was that good overcomes evil. A story represents reality not through its facts per se, but through the cause-and-effect relationships suggested by the sequence of facts.

Games also attempt to represent reality. One important difference between games and stories is that a story presents its facts in an immutable sequence, while a game offers a branching tree of possible sequences and allows the player to make choices at each branch point and thus to create his own narrative. The audience of a story must infer causal relationships from a fixed sequence of events. The player of a game is encouraged to explore alternative sequences, contrapositives, and inversions.



A Story

A Game

While a story may be experienced differently with each rereading or retelling, the structure of its presentation each time is the same. The structure of a game's presentation, in contrast, can be different each time it is played. One might expect to play a game many times, trying different strategies each time until a representative subset of all the branches in the game-net has been explored.

This does not mean that games are intrinsically superior to stories. For although stories trace a fixed sequence of causal development, they do so with greater intricacy and detail than games. Detail can provide the texture, the feel of reality that makes a story compelling and sweeps the audience to some predestined conclusion. The game designer, on the other hand, creates a complex network of paths cunningly crafted to offer the player many choices. Stories enjoy another advantage over the current generation of computer games: the element of surprise. A good story might boast an array of interesting plot twists, lead us into a set of expectations, and then cleverly deflate those expectations. This process can be repeated many times during the course of the story.

Among computer games, only adventures provide this element of surprise. Unfortunately, adventure games can do so only by limiting the player's freedom of action in order to guarantee that the player will encounter the surprise under the proper circumstances. The truly exciting possibility offered by computer games is the prospect of formulating a plot twist in response to the player's actions instead of merely dragging him down a pre-ordained path. However, the ability to formulate surprise requires an ability to analyze the player's actions, deduce his expectations, and generate a believable plot twist that confutes his expectations without frustrating him. Artificial intelligence that advanced has yet to be created.

Games Versus Toys

Games lie between stories and toys in manipulability. Stories do not permit the audience to control the sequence of fictional events. Games allow the player to manipulate some of the facts of the fantasy, but the rules governing the fantasy remain fixed. A toy is much less constrained; the toy-user is free to manipulate it in any manner that strikes his fancy. The storyteller has direct creative control over his audience's experience; the game designer has indirect control; the toymaker has almost none.

Significance of Interaction

Interaction is important for several reasons. First, it injects a social or interpersonal element into the event. It transforms the challenge of the game from a technical one to an interpersonal one. Solving a cube

puzzle is a strictly technical operation; playing chess is an interpersonal operation. In the former, a player competes against the logic of the situation; in the latter, the player uses the logic of the situation to play against the opponent.

Second, interaction transforms a passive challenge into an active challenge. A puzzle will always present the player with exactly the same challenge, but a game opponent reacts to a player's actions and presents different challenges in each game. This difference has major emotional significance. The person solving the puzzle must somehow guess, deduce, or master the key trick built into the puzzle by the designer. The puzzle player is working against the puzzle (or its designer) to unmask its secret. Once the secret is known, the puzzle is no longer interesting.

The game player, by contrast, faces different challenges each time. Where a puzzle is dead, a game is alive; the game player must create a solution best suited to the personalities of both the player and the opponent. The key distinction between a game and a puzzle is the difference between creating your own solution and discovering the designer's solution. A game acknowledges the player's existence and reacts to the player's personality; a puzzle is a lifeless, unresponsive object.

Computer games seldom provide a human opponent, and so they lack the social element that other games offer. They can, however, present an illusory personality against which the player competes. This is one of the most exciting and least developed potentials of the computer as a game technology. Regardless of the computer's success or failure in synthesizing a social element, the computer can at least make the game a highly interactive experience for the player. It can react to the player's moves with speed and thoroughness.

Nature of Interaction

Interaction is not a binary quantity; it is a continuous quantity with a range of values. Puzzles have little or no interactiveness; games have more. This suggests that interactiveness is an index of "gaminess." Some games, such as blackjack, tag, or PONG provide very little interaction between the players (binary decision to stand or hit, running, and twisting paddle), although the players may wish to interact. These games do not allow players to invest much of themselves in the play or to react extensively with their opponents.

Other games, such as bridge, football, and LEGIONNAIRE allow a far richer interaction. Players can grapple with each other on several levels, making these games more exciting than the others. What is important about interaction is not its mechanics but its emotional significance. PONG is insipid because a player cannot express much personality through the medium of a bouncing ball. Bridge is better because it includes elements of teamwork, deception, and cooperation. I can better involve my personality in a game of bridge. Thus, degree of interaction provides a useful index of "gaminess."

CONFLICT

A third element of all games is conflict. Conflict arises naturally from the interaction in a game. The player is actively pursuing some goal, which obstacles prevent him from achieving easily. If the obstacles are passive or static, the challenge is a puzzle or an athletic challenge. If the obstacles are active or dynamic, if they purposefully respond to the player, the challenge is a game. However, active, responsive, purposeful obstacles require an intelligent agent. If that intelligent agent actively blocks the player's attempts to reach his goals, conflict between the player and the agent is inevitable. Conflict is fundamental to all games.

Some people shrink from this aspect of games. A number of attempts have been made to design games cleansed of conflict. Such attempts emphasize cooperative efforts rather than conflict. Few people seem to enjoy them.

Such attempts tend merely to shift the conflict rather than eliminate it. Members of a team, for example, can cooperate with each other in the team's conflict with another agent. This other agent could be another team, a human player, or a computer-simulated player. In all cases, the opponent must be perceived as having a persona. Without at least the illusion of purposeful reaction to the player's actions, the game collapses.

Our real-world conflicts are usually indirect, diffused over time, and all too frequently lack resolution. Seldom does a person achieve an outright victory in the conflicts of daily life. Because games are subjective representations of the real world, they focus attention on a particular aspect of the world by accentuating that aspect. Conflict in games usually tends to be (but need not always be) exaggerated to its most direct

and intense form—violence. Violence is not essential or fundamental to games, but is common in games because it is the most obvious and natural expression for conflict.

SAFETY

A game, then, is an artifice for providing experiences of conflict and danger while excluding their physical realizations. In short, a game is a safe way to experience reality. More accurately, the consequences of a game are always less harsh than the situations that the game models. A player can blast monsters all day and risk only a quarter, can amass huge financial empires and lose them in an hour without risking actual empires, or lead great armies into desperate battles on which hang the fate of nations, all without shedding a drop of blood. In a world of relentless cause and effect, of tragic connections and inevitable consequences, the dissociation of actions from consequences is a compelling feature of games.

Of course, games do have consequences. The penalties for losing a game can sometimes be a significant deterrent to game play. Gambling presents a real financial risk to the player and losing to another person often entails some loss of dignity. This may be one attraction of computer games: there is less shame in losing to a computer. True victory—the total destruction of the computer's forces—is acknowledged to be impossible in most of these games. This further lessens the shame of defeat, and the loser can keep coming back for more defeats without losing face. But our categories, as always, tend to blur. Poker, for example, is a game of bluffing; the key to success lies in convincing your opponent that you have better or worse cards than you really have. Because money is at stake, the players experience stresses that strain their ability to deceive their opponents.

I have been discussing a set of characteristics that defines what I mean by the term "game." Mostly, I have emphasized characteristics intrinsic to games themselves rather than the motives of players. Separation of game from player is artificial and misleading, for neither exists without the other. Why do people play games? What motivates them? What makes games fun? The answers to these questions are crucial to good game design.

WHY PEOPLE PLAY GAMES

One way to address the question of why people play games is to inquire into the history of games. Games now are too varied, too intricate, and too culturally involved to indicate the fulfillment of any single need. Perhaps the fundamentals of games would be more evident in their earliest incarnations. How far back must we go? If we wish to get back to the beginnings of games, we must go beyond the realms of the historian and archeologist into the realm of the paleontologist. Games are not a human invention.

The incidence of game-playing in animals is itself instructive; game-playing has been observed only in mammals and birds. Game play seems to be associated with that quality that we have clumsily attempted to quantify and describe in terms of brain size, intelligence, and ability to learn. When we find two lion cubs at the zoo wrestling with their mother or growling and clawing at each other, we laugh at the comedy. We say that the cubs are playing a game, that they are having fun, and that they are such fun-loving, carefree creatures. We are right on the first count: these cubs do indeed appear to be playing a kind of game. We certainly see in their behavior all four of the fundamental game attributes: representation, interaction, conflict, and safety. We may be right on the second count; who knows if lions can have fun? But we are wrong on the last count—these games are a deadly business.

I claim that the fundamental motivation for all game-playing is to learn. This is the original motivation for game-playing, and it surely retains much of its importance. Game playing is a safe way to learn. The desire to learn, however, need not be conscious. Indeed, it may well take the form of a vague predilection to play games. Other motivations have little to do with learning and may assume greater local importance than the ancestral motivation to learn. These other motivations include: fantasy, nose-thumbing, proving oneself, social lubrication, exercise, and the need for acknowledgment.

Fantasy

A very important motivation to play games is fantasy fulfillment. Most of us are confined to a world of asphalt, plastic, and paper. Like a movie, a book, or music, a game can transport us from the world that

oppresses us and create a fantasy world in which we can forget our problems. Games are potentially superior to other means of escape (movies, books, music) because games are participatory. Instead of merely watching a movie, reading a book, or listening to music, the player is actively involved in the game. Indeed, the player drives the game, controls it in a way that is quite impossible with passive fantasies. This need to escape, to fantasize, is certainly an important motivation in game playing.

Nose-Thumbing

A common function of games is to provide a means of overcoming social restrictions, at least in fantasy. Many games place the player in a role that would not be socially acceptable in real life, that of a pirate or a thief, for example. An excellent, although extreme, example is the game CRUSH, CRUMBLE, AND CHOMP by Automated Simulations. In this game the player is cast as a 1950-vintage monster going on a rampage through his favorite city. He stomps on police cars, crushes buildings, swats helicopters, and creates general mayhem. The illustration on the package shows a monster about to attack an IRS building as terrified citizens flee. This represents an extreme case of anti-social behavior made acceptable by the safety of the game.

Sometimes the player's role is socially acceptable, but the actions taken are discouraged in real life. MONOPOLY encourages players to engage in what the Federal Trade Commission delicately calls "predatory trade practices." War games encourage players to start and win wars. Some games address sexual matters, allowing players to indulge in make-believe behavior that they could never exhibit in the real world.

The most telling example of this nose-thumbing is found in the arcade games. These games emphasize violence and lots of it. One theme is almost universal in arcades: destroy somebody. The *coup de grace* is not delivered discreetly or elegantly. On the contrary, the victim is dispatched with the most colorful animated explosion possible. The violence is the whole point and purpose of the enterprise. Yet, even as we pander to distasteful drives, we delicately clothe them in less offensive garb. We never, never obliterate human beings; instead, we vaporize ugly space monsters. The monsters have perpetrated some

odious interstellar crime; the player is cast as the defender, the protector, or the avenger. To heighten the player's sense of urgency, the case is often presented that the game represents a time of extreme crisis ("THE FATE OF HUMANITY IS AT STAKE!!!!!"). This conveniently justifies the use of extreme violence, thereby allowing the player to have violence without guilt. The player can thumb his nose at social strictures and engage in violence and mass murder without risking censure. The game provides a safe way to thumb one's nose.

Proving Oneself

Another reason people play games is to demonstrate prowess. All games support this motivation to a greater or lesser degree. (Many game-playing communities sponsor tournaments or player ratings.) Arcade games exploit this motivation by recording and displaying the initials of the top-scoring players. Some players carry this to extremes. Their prime goal is not merely to win but to beat somebody, preferably somebody worth beating. Chess has an unusually high concentration of such sharks; so do war games. (A common question asked during a war game is "Are you playing for blood or for fun?") Such players normally prefer games that allow their skills to be brought to bear properly, so they tend toward games in which chance plays a minimal role. Despite the concentration of such players in games of deductive logic, almost all games have sharks preying on the playful opponents. When a shark plays for serious rewards (social dominance, for example) and takes serious risks of failure, the crucial element of safety is eliminated from the game. At this point the contest is no longer a game but a conflict.

Inasmuch as all games have the potential for being played in an overly competitive way, some people who are especially sensitive to the social risks of game-as-conflict refuse to play games, because they do not perceive the games to be safe. If they do play, they prefer to play games of pure chance, not so much to disable or discourage the shark as to create a situation in which winning is patently unrelated to prowess. If victory conditions are arbitrary, social risk is eliminated and safety is restored. In most games, of course, safety from social risk is conferred by the attitudes of the players, their willingness to say, "It's only a game."

Social Lubrication

Games are frequently used (especially by adults) as social lubricants. The game itself is of minor importance to the players; its real significance is that it provides a focus for an evening of socializing. Card games and some light board games serve this function. An obvious example of such a social lubricant game utilizes a large plastic gameboard about four feet square marked with colored spots. During each player's turn, a random process determines which of four appendages (arms or legs) is to be placed on which spot on the board. As the players contort to fulfill the requirements, they inevitably make physical contact with each other in innocent and foolishly humorous ways. This game, a true icebreaker, promotes social lubrication.

Exercise

Exercise is another common motivation to play games. The exercise can be mental or physical or a combination of both. In any case, the game is an entertaining way to stay in shape. Some players like to exercise their cognitive skills, others prefer to use intuition, and some prefer to exercise their athletic skills. All players need to exercise their skills at appropriate levels. A chess player will receive little stimulation from a game of tic-tac-toe. Similarly, a person who finds tic-tac-toe challenging will get little useful exercise from playing chess. These preferences separate players and route them to the different games available.

Need for Acknowledgment

We all need to be acknowledged, to be recognized by other people. We crave not merely an acknowledgment of our existence, but of our personalities. For example, when we meet a casual acquaintance, we usually get a perfunctory acknowledgment ("Hello there, Jones"). We are more gratified when the greeting in some way acknowledges us as individuals with special personalities and problems ("Hello there, Jones; is that knee still bothering you?").

This is one reason why interaction is so important to a game; it allows the two players to acknowledge each other. A truly excellent game allows us to involve a large portion of our personalities in our game-playing. Such a game allows me to play in a way that only I can.

My opponent must look beyond the playing pieces and acknowledge my cleverness, my rashness, my deviousness—my entire personality. When such a game ends, my opponent and I know each other better than we did before we sat down to play.

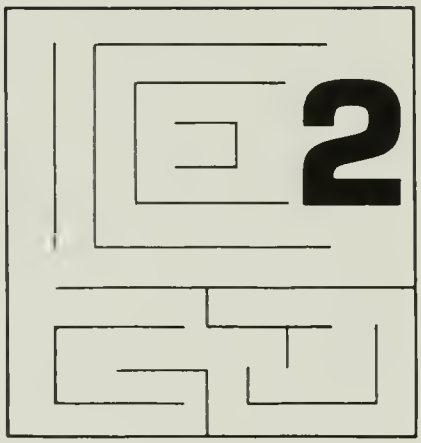
WHAT'S SPECIAL ABOUT THIS GAME

We should distinguish the question "Why do people play games?" from the question "What makes one game more fun than another?" Some things motiviate a person to play games in general; other things can motivate that person to select from among similar games, to prefer one game to another. For example, special effects—graphics, sound, and animation—can provide sensory support to a game fantasy and can distinguish a good game from a bad game. But we must not confuse their roles. Sensory gratification through special effects should provide crucial support, not be the central feature of the game. Sensory texture can enhance the impact of the fantasy created by the game or movie, though wonderful graphics or sound do not by themselves make good movies or games. But special effects could well be the reason why someone might prefer one game to another similar game.

So far I have discussed various reasons why people play games (and why they prefer one game to another) as if they were absolute categories of player attitudes and motivations, as if a game need only be designed to fit these categories to guarantee its success. Such is not the case. The response of a prospective player to any game depends heavily on that person's personality and tastes. And a person's tastes in games are not static. As a person changes, so do his tastes and motivations. Just as rock 'n roll, for example, was the entry point into the world of music for an entire generation, so will skill-and-action games be the entry point into the world of games for much of the population. Like early rock 'n roll, skill-and-action games have broad appeal and are easy to understand. As people's attitudes toward games become more sophisticated, their tastes will evolve along different paths, and their motivations will change. Like rock 'n roll, skill-and-action games will not go away. But they will change to reflect the evolving taste of the public. We can see this happening already. The early arcade games are tame pussycats compared to the rip-snorting, fire-breathing games of the 1980's. Had TEMPEST been released in 1977, it would have intimidated and repelled players.

Can we discover the personality traits and differences that currently determine people's preferences among games? Can we anticipate the evolution of taste? One way would be to observe and catalog groups of game-players and to identify the game traits valued by these groups. The youth of the computer game industry is an obstacle to this approach. We can at this time identify only two broad, vague, and overlapping groups of players: skill-and-action enthusiasts and players of cognitive games.

A TAXONOMY OF COMPUTER GAMES



Hundreds of computer games are commercially available on a variety of hardware configurations. Of this bewildering array, many are quite similar. Most have some unique design feature. Given this large sample of games, we can learn a great deal about game design by establishing a taxonomy of computer games. In another field of study, Charles Darwin's meticulous taxonomic work while on the Beagle led inevitably to his development of the theory of evolution. Similarly, a taxonomy of computer games should illuminate the common factors that link families of games and reveal critical differences among families and among members of families. A well-constructed taxonomy may suggest previously unexplored areas of game design. Most importantly, a taxonomy can reveal underlying principles of game design.

I will insist on an important qualification: I do not claim that the taxonomy I propose is the correct one nor will I accept the claim that any correct taxonomy can be formulated. A taxonomy is only a way of organizing a large number of related elements. Many taxonomies are admissible. Indeed, attempting to construct several alternative taxonomies may be a useful way to examine the common traits of computer games. I shall be content, however, to propose just one taxonomy. I divide computer games into two broad categories: skill-and-action (S&A) games (emphasizing perceptual and motor skills) and strategy or cognitive games (emphasizing cognitive effort). Each major category has several subcategories.

SKILL-AND-ACTION GAMES

This is easily the largest and most popular class of computer games. Indeed, most people associate all computer games with skill-and-action games. All arcade games are S&A games, and almost all games for the ATARI 2600 are S&A games. This class is characterized by real-time play, heavy emphasis on graphics and sound, and the use of joysticks or paddles rather than a keyboard. The primary skills demanded of the player are hand-eye coordination and fast reaction time.

I group skill-and-action games into six categories: combat games, maze games, sports games, paddle games, race games, and miscellaneous games.

Combat Games

Combat games all present a direct, violent confrontation. The human player must shoot and destroy the "bad guys" controlled by the computer. The challenge is to position oneself properly to avoid being hit by the enemy while shooting back. These immensely popular games are Atari's forte. The variations on this theme are distinguished mostly by the geometry of the situation or by the weaponry of the opponents.

STAR RAIDERS (Figure 2-1) and SPACEWAR can be compared in terms of geometry and weaponry. In both games the player flies

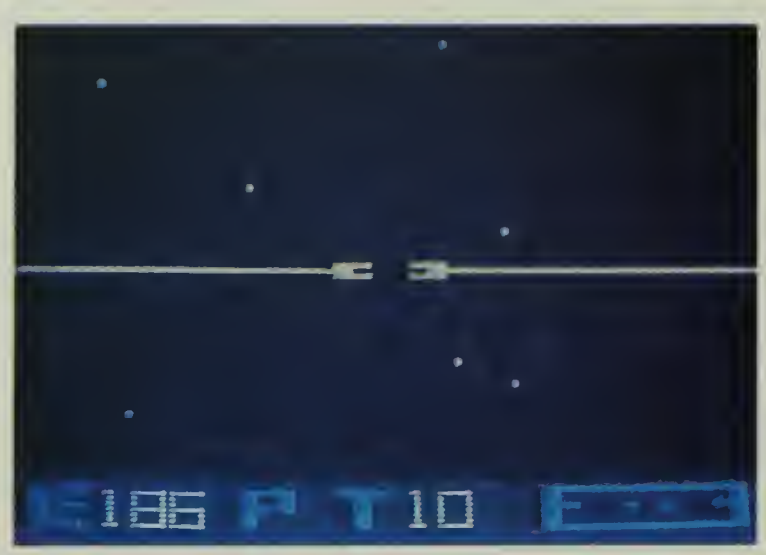


Figure 2-1. STAR RAIDERS

through space in a rocket ship and engages enemy spaceships in real-time cosmic dogfights. STAR RAIDERS presents the conflict in first-person geometry (the television screen shows the same scene the pilot would see). SPACEWAR uses similar weaponry and equipment with one crucial difference: the geometry of the game is third-person rather than first-person (the player sees her own and her opponent's spaceships from a distance). The different result is obvious to anyone who has played both games. The first-person game is more exciting and compelling than the third person game. Unfortunately, the first-person geometry is so technically difficult to execute that it has been implemented on only a few games. Most games use third-person geometry.

ASTEROIDS (Figure 2-2) is a "shoot-'em-up" game that uses the same space environment as STAR RAIDERS. The primary difference between the two games is the opposition. The enemy in ASTEROIDS is not a small number of intelligent opponents armed with weapons identical to the player's, but a hail of flying rocks armed only with the ability to collide destructively with the player.

MISSILE COMMAND (Figure 2-3) is a combat game with several interesting twists. First, the player must defend not only himself but also his cities from falling nuclear bombs. Second, the game is purely defensive because the player never has the opportunity to attack his

Figure 2-2. ASTEROIDS

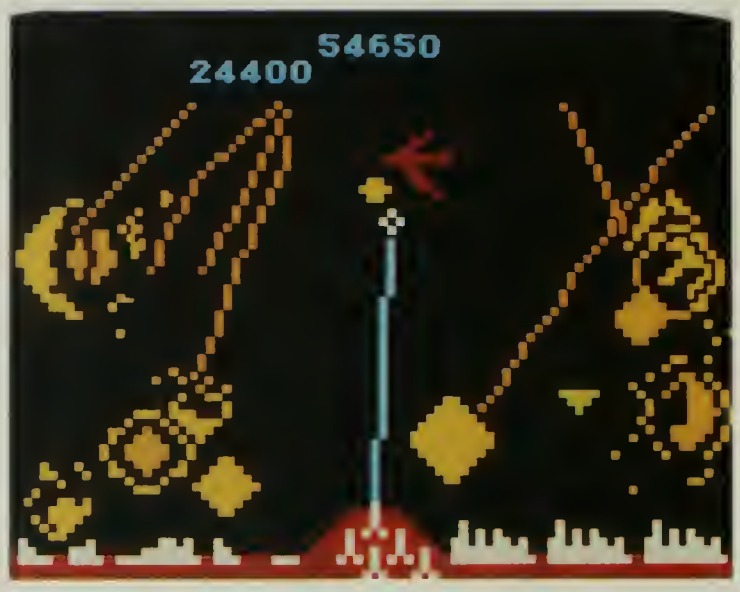


Figure 2-3. MISSILE COMMAND

enemy. Third, the shooting process in this game is slower than in other games because the missiles must fly to their targets before detonating. Because the time between firing and impact is so long, the player must plan his shots with greater foresight, lead his target, and make use of multiple explosions. Although this is a skill-and-action game, MISSILE COMMAND incorporates more strategic elements than many games in this category.

SPACE INVADERS (Figure 2-4) is one of the most successful of all combat games. It was one of the first smash hit games and contributed to the upsurge of popularity of all computer games that began in 1979. While STAR RAIDERS and ASTEROIDS give the player great mobility and MISSILE COMMAND gives him none, SPACE INVADERS gives the player limited mobility in one dimension only. As in ASTEROIDS, the player must face a multitude of stupid opponents who can win by merely touching the player (landing). In addition, as in STAR RAIDERS, the monsters also shoot back. The monsters in SPACE INVADERS march back and forth across the screen, slowly descending on the player. As the player kills more monsters, they march faster and faster. This gives the game a hypnotic, accelerating tempo. SPACE INVADERS is definitely a classic.

The success of SPACE INVADERS has spawned a series of copies

Figure 2-4. SPACE INVADERS

Figure 2-5. GALAXIAN

and derivatives to cash in on the success of the original game. Successful derivative games include, for example, GALAXIAN, CENTIPEDE, and TEMPEST. GALAXIAN (Figure 2-5) is a simple variation on SPACE INVADERS. Individual invaders "peel off" and attack the

player with more ferocity than the docile monsters of the original game. CENTIPEDE (Figure 2-6), another derivative of SPACE INVADERS, is different enough to be considered a new design, but the internal game structure is similar to the original. The invaders have been grouped into a segmented centipede. Their side-to-side motion is bounded not by the edges of the screen but by randomly scattered mushrooms. Numerous embellishments (spiders, fleas, and scorpions) enhance the game considerably. TEMPEST is a three-dimensional first-person derivative of SPACE INVADERS using vector graphics. The amount of design attention that SPACE INVADERS has attracted is a tribute to the game's originality, appeal, and durability.

There are many other combat games. BATTLEZONE and RED BARON are two first-person combat games utilizing vector displays. Others include CAVERNS OF MARS, YAR'S REVENGE, CROSSFIRE, and DEFENDER.

You may wonder why so many combat games are set in outer space. There are three reasons. First, space is easy to depict and animate with a computer—all the designer need do is draw a blank screen with a few white dots for stars. Second, space is unencumbered by the expectations of the players. Design problems can always be solved by concocting a

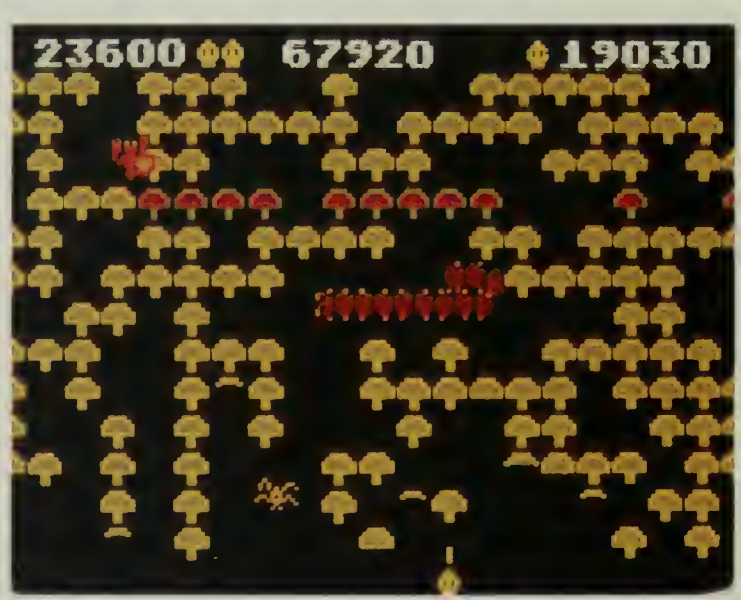


Figure 2-6. CENTIPEDE

super-duper zapper and nobody can object that it is unrealistic. Earthbound games force the designer to look reality squarely in the eye—such a tiresome burden for a creative mind. Third, because it is unfamiliar, space is an intrinsically fantastic environment that encourages suspension of disbelief.

Combat games have always been at the heart of computer gaming. Players never seem to tire of them, and it appears they will be around for a long time to come.

Maze Games

The second subgroup of S&A games is the set of maze games. PAC-MAN (Figure 2-7) is the most successful of these, although not the first. In maze games, the player must move through a labyrinth. Often, one or more "bad guys" pursue the player through the maze. Some maze games (MAZE CRAZE for the ATARI 2600 is a good example) require that the player make his way to an exit; others require the player to move through each part of the maze. DODGE 'EM is an early example of such a game. In either case, the number, speed, and intelligence of the pursuers determine the pace and difficulty of the game. PAC-MAN uses a carefully balanced combination of these factors. The pursuers are

Figure 2-7. PAC-MAN

just slightly slower than the human player, but their intelligence and number compensate for their lack of speed. The overall pace of the game makes it difficult for the player to analyze the positions of the five pieces in real time.

Any successful game is certain to give rise to copies and variations, and PAC-MAN is no exception. One of the first such games for the ATARI Home Computer System was the first edition of JAWBREAKERS. This game, now removed from the market, was structurally indistinguishable from PAC-MAN. The play of the game was identical. Cosmetically, there were a number of differences: the pursuers were faces instead of ghosts, the player was a set of teeth instead of a head with a mouth, the maze was laid out differently, and the sounds were different. This game provides a good example of the methods that can be used to copy games while attempting to minimize legal problems.

Another PAC-MAN derivative is MOUSKATTACK (Figure 2-8). Compared with PAC-MAN, this game shows some structural changes. The player is again pursued through a maze by four computer-controlled creatures, but the scenario contains a few embellishments.

Figure 2-8. MOUSKATTACK

Merely passing through every point in the maze is not enough; some points, randomly chosen by the computer, must be passed twice. In addition, the player can fight back against the pursuers by setting mousetraps. Finally, a very interesting two-player version allows both cooperative and competitive strategies. In MOUSKATTACK we see the basic structure of PAC-MAN extended to produce a distinct game.

Part of the appeal of maze games can be attributed to the cleanliness with which they encapsulate the branching structure that is a fundamental aspect of all games. You will recall that a game forms a tree structure with each branch point representing a decision made by the player. In a maze game, each branch point is neatly depicted by an intersection in the maze, and the options available to the player are visually presented as paths available at the intersection. A maze game thus presents a clear visual representation of the branching structure.

Even more fascinating is the looping structure possible with maze games. A player can return to an intersection in the maze many times. Yet, each time she does so, her options change because, meanwhile, the other maze-inhabitants have moved to different positions. In this way, a small number of displayed intersections can represent a huge number of branch-points in the game-tree. A revealing analogy can be drawn with a computer program in which a small number of program instructions, through looping and branching, can address a large number of specific cases.

Sports Games

These games model popular sports. They are anachronisms from the early days of computer game design when computer games had no separate identity. Lacking original ideas, designers turned to sports for models. This also served a marketing purpose: why should a conservative consumer buy a game with an alien title and subject. Better to offer him a game he is already familiar with: basketball, football, baseball, soccer, tennis, boxing, and others. All these games take liberties with their subject matter for the sake of playability. The most enjoyable aspects of these computer games have little to do with the real athletic game. Only by substantially altering the original games were the authors able to produce decent designs. Even so, sports games remain the wallflowers of computer gaming. I suspect that sports games will

not attract a great deal of design attention in the future. Now that computer games have a popular identity of their own, the need for recognizable game titles has diminished.

Paddle Games

I use the title *Paddle Games* to cover the PONG-based games. PONG is certainly one of the most successful and fertile of game designs and it has many grandchildren and great-grandchildren. The central element of the game, intercepting a projectile with a paddle-controlled piece, has been used in countless variations. The original PONG pitted two players in an electronic version of ping-pong, hence the name. BREAKOUT required a solitary player to chip away at a wall with the ball. The player received points for each brick destroyed. SUPERBREAKOUT (Figure 2-9) introduced variations on this theme with moving walls, extra balls, and other tricks. CIRCUS ATARI introduced parabolic trajectories and a complex moving wall of balloons. WARLORDS took the genre even further; as many as four players (one in each corner) defend brick castles against a projectile bounced around the field by their shield-paddles.

In all of these games, the player uses the ball as a weapon to batter; in other paddle games the player must only catch the ball or many

Figure 2-9. SUPERBREAKOUT

balls. AVALANCHE is one such game. The player is at the bottom of the screen under a number of falling rocks, and each one must be caught. The game becomes quite frantic as more and more rocks fall at an accelerating pace. Another game, CHICKEN, expands on this theme by replacing the rocks with eggs that hatch upon striking the ground, forcing the player-hen to jump over them as she moves about.

The paddle game system is a very simple one. Although I doubt that it has much development potential remaining, I am hesitant to pronounce such a durable old system dead.

Race Games

Some computer games are a straightforward race. Most of these games allow the player to move at constant speed but exact time penalties for failure to negotiate an assortment of hazards. Thus, a player in the APX skiing game DOWNHILL must avoid the trees and rocks; the player's score is based on her time to complete the course. MATCH RACER by Gebelli Software is a car-racing game with oil slicks and obstacles. NIGHT DRIVER, another car race, features a first-person view of the road. A problem with all of these games is that they are not true games but puzzles. There is no real interaction in a race between a player and her opponent, and it is difficult even to identify the opponent in these games.

A more involved variation on the race game is DOG DAZE by Grey Chang. This is a true game, not a puzzle. It presents a two-player competitive race with variable goals and asymmetric obstacles. Each player has a dog. Hydrants pop onto the screen in random locations. Each player must race to be the first to touch a hydrant and claim it as his own. Players touching hydrants owned by their opponents are temporarily paralyzed. The game has many interesting twists and turns without being overly complex and demonstrates that the race game can be a flexible vehicle of game design.

Miscellaneous Games

A number of games do not fit into my taxonomy very well. The first I will mention is DONKEY KONG (Figure 2-10), a game that looks a little like a race game with intelligent obstacles. PREPPIE! (Figure 2-11) is another game that defies classification in this taxonomy. It could be called a maze game with moving walls or obstacles, but only by

Figure 2-10. DONKEY KONG

Figure 2-11. PREPPIE!

straining the maze-game category. APPLE PANIC by Broderbund Software is like a maze game in some ways but like a combat game in others. The pace of the game is oddly slow. Although these games do not

fit the taxonomy neatly, I am content to wait for other developments before I create new categories or revise old ones.

STRATEGY GAMES

Strategy games comprise the second broad class of computer games. These games emphasize thinking rather than motor skills. Some S&A games do have a strategic element, but the major difference between strategy games and S&A games is that all skill-and-action games require some motor skills; strategy or cognitive games do not. Indeed, real-time play is rare in strategy games, although this is changing (LEGIONNAIRE from Avalon-Hill is a notable real-time strategy game). Strategy games typically require more time to play than S&A games. Strategy games, which are nonexistent in the arcades and rare on the ATARI 2600, are almost exclusively restricted to personal computers. I divide strategy games into six categories: adventures, D&D games, war games, games of chance, educational games, and interpersonal games.

Adventures

These games derive from one of the oldest computer games, ADVENTURE. In these games the adventurer must move through a complex world, accumulating tools and booty adequate for overcoming each obstacle, until finally reaching the treasure or goal. Scott Adams created the first set of adventures widely available for personal computers, and his software house (Adventure International) is built on those games. The Scott Adams games are pure text adventures that use only a small amount of memory and do not need disk drives. They are also readily transportable to different machines.

A short time later, Ken and Roberta Williams built On-Line Systems with THE WIZARD AND THE PRINCESS (Figure 2-12), an adventure that presented pictures of the scenes in which the adventurer found himself. The game itself was not particularly new; the use of graphics was the primary innovation. Both On-Line and Adventure International have expanded with more games using the systems they pioneered. Most of these derivatives are structurally similar to the originals, differing only in detail, polish, and size.

The next variation on the adventure theme was the giant adventure, of which there are several. TIME ZONE by On-Line Systems is one of

Figure 2-12. WIZARD AND THE PRINCESS

these. These games use multiple diskettes to link together a gigantic adventure. The player solves the puzzle in one environment then moves on to another environment on another disk. The games are structurally identical to earlier games; the only difference is magnitude. Giant adventures take many weeks of play to solve.

A new variation on the adventure game genre is DEADLINE, an interesting detective adventure. Its heritage as an adventure game is evident in the lack of graphics and the use of an excellent sentence parser. In DEADLINE, the player is a detective attempting to solve a murder. The game is played in a real-time mode, which makes it more challenging. The player searches not for treasure but for clues to solve the murder. This game shows the potential of the adventure in that the same system can be used, with storyline and goals altered, to appeal to a different audience.

One of the cleverest adventures ever designed is Warren Robinett's ADVENTURE on the ATARI 2600. This game follows the same basic format as all adventures, except that it uses absolutely no text. Instead, the player moves through a series of rooms represented by rather simple graphics. Although the graphics and input schemes are different, the basic feel of the adventure system has been retained. SUPERMAN, HAUNTED HOUSE, and GALAHAD AND THE HOLY GRAIL by Doug Crockford are all derivatives of this game.

Adventures are more similar to puzzles than to games. As discussed in Chapter One, puzzles are distinguished from games by the static obstacles they present to the player. Adventures present intricate obstacles that, once overcome, no longer challenge the player. It is true that some adventures approach games by incorporating obstacles such as hungry dragons that in some way react to the player. Nevertheless, they remain primarily puzzles.

D&D Games

D&D games trace a completely independent line of development. Fantasy role-playing was created by Gary Gygax with DUNGEONS AND DRAGONS, a complex noncomputer game of exploration, cooperation, and conflict set in a fairytale world of castles, dragons, sorcerers, and dwarves. In D&D, a group of players under the guidance of a "dungeonmaster" sets out to gather treasure. The game requires a minimum of hardware; players sit around a table and use little more than a pad of paper. The dungeonmaster sets the rules and referees the game. He has authority to adjudicate all conflicts, allowing complex systems to be created without the frustrations of complex rules. The atmosphere is loose and informal. For these reasons, D&D has become a popular game with endless variations and derivatives.

D&D first appeared in the mid-seventies. It didn't take long for people to realize that it had two serious limitations. First, the game required a group of players and a dungeonmaster, so it was impossible to play the game solitaire. Second, the game could sometimes become tedious, requiring lengthy computations and throwing of dice. Designers soon recognized that these problems could be solved with a microcomputer. The first company to make a D&D-style computer game available was Automated Simulations. Their TEMPLE OF APSHAI program (Figure 2-13) has been very successful, and they also market a number of other D&D-style games.

So far, however, few games have been marketed that truly capture the spirit of D&D. Most D&D players are young and don't have the money for such packages. Moreover, adventure games have slowly absorbed many of the ideas of the D&D games. It was once possible easily to distinguish an adventure from a D&D game. Adventures were pure text games; D&D games used some graphics. Adventures were

Figure 2-13. TEMPLE OF APSHAI

puzzles; D&D games were true games. Adventures were mostly nonviolent, but D&D games tended to be quite violent. Lately, we have seen adventures acquiring many of the traits of D&D games, so it is now harder to tell the difference between them.

An example of this phenomenon is ALI BABA AND THE FORTY THIEVES (Figure 2-14), a game with elements of both adventures and D&D games. The player must search through a large maze to find and rescue a princess, but on the way, he must fight monsters and thieves. The player, as Ali Baba, possesses personal characteristics (dexterity, speed) that are reminiscent of a D&D game, but he must explore the maze, as in an adventure. For these reasons, this game cannot be classified as either an adventure or a D&D game, but as a solid example of the merging of these two genres into a new class of games, the fantasy role-playing (FRP) games. This suggests that we will see more such games combining the "search and discover" aspects of adventure games with the "defeat opponents" aspects of D&D games.

War Games

A third class of strategy games is provided by the war games. Noncomputer war games have a long lineage. Commercial war gaming began in the 1880s with an American game using wooden blocks. The

Figure 2-14. ALI BABA AND THE FORTY THIEVES

British have long had a dedicated group of war gamers using miniature models of soldiers and very complex rules. Their games, called miniatures games, have grown in popularity and are now played in the USA. But the largest segment of war gamers in recent years has been the board gamers. This hobby was started in the late 1950s by Charles Roberts, who founded the Avalon-Hill Game Company and created such classic games of the sixties as BLITZKRIEG, WATERLOO, and AFRIKA KORPS. During the 1970s a new company, Simulations Publications, Inc., turned board war gaming into the most popular form of war game.

War games are easily the most complex and demanding of all games available to the public. Their rules books resemble contracts for corporate mergers, and their playing times often exceed three hours. War games have therefore proven to be very difficult to implement on the computer. We have, nevertheless, seen some entries in the field.

The computer war games available now fall into two distinct groups. The first is composed of direct conversions of conventional board games. COMPUTER BISMARK, COMPUTER AMBUSH, and COMPUTER NAPOLEONICS are examples of this group of games. They illustrate the folly of directly converting games from one form to another. They imitate successful and respected board games but are not as successful.

Figure 2-15. EASTERN FRONT

Because they attempt to replicate board games, they are slow and clumsy to play.

The second group of computer war games is less imitative. My own EASTERN FRONT 1941 (Figure 2-15) is generally considered to be the best of this lot, primarily because of its graphics and human engineering features. Many of the games in this category are experimental; therefore, the successes are outnumbered by the failures. Avalon-Hill's first entries into the computer war gaming arena were such experiments. My own TANKTICS, an early experiment, was once the most advanced commercially available war game. (It was the ONLY commercially available war game when I first released it in 1978.) It is now generally regarded as a mediocre game.

Computer war gaming is not a well-developed area. For the moment, it is too closely associated with board war gaming in the minds of the public and of most designers. Until it can shake free from the constraints of board games and establish its own identity, computer war gaming will evolve slowly.

Games of Chance

Games of chance have been played for thousands of years; their implementation onto computers is therefore not surprising. Games of chance are quite easy to program, so we have seen many versions of

craps, blackjack, and other such games. Despite their wide availability, these games have not proven very popular, most likely because they do not take advantage of the computer's strong points. Furthermore, they lose the advantages of their original technologies. These games demonstrate the folly of mindlessly transporting games from one medium to another.

Educational And Children's Games

The fifth category of strategy games is that of the educational games. Although all games are in some way educational, the games in this set are designed with explicit educational goals in mind. This group is not heavily populated as yet, perhaps because the people interested in educational uses of computers have not yet concentrated much attention on game design. The Thorne-EMI puzzles are good entries in this field, and APX sells a collection of simple children's games that have some educational value. Several of the classic computer games are educational: HANGMAN, HAMMURABI, and LUNAR LANDER are three noteworthy early games. SCRAM, a nuclear power plant simulation (Figure 2-16), and ENERGY CZAR, an energy economics simulation, are two of the more complex programs in the educational field. My favorite entry to date is ROCKY'S BOOTS, a children's game about

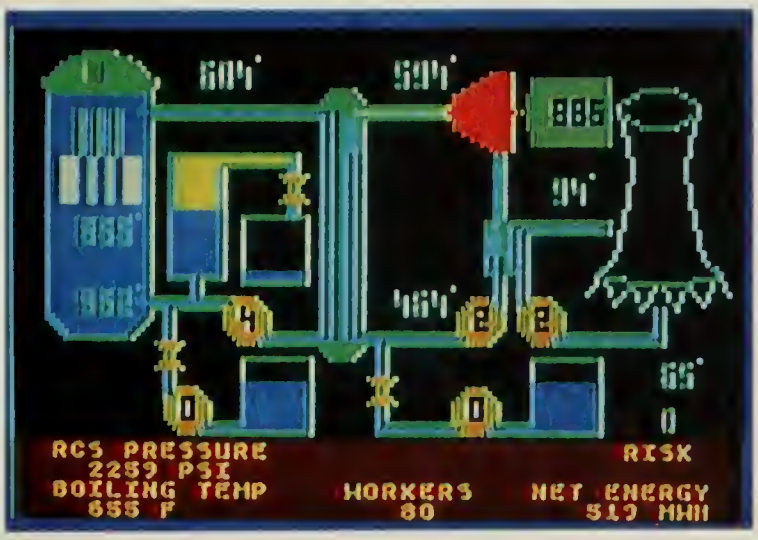


Figure 2-16. SCRAM

Boolean logic and digital circuits. The child assembles logic gates to create simulated logical machines. This game demonstrates the vast educational potential of computer games. Educators are becoming more aware of the motivating power of such games, and in time we can expect to see more entries of the caliber of ROCKY'S BOOTS.

Interpersonal Games

I have been exploring a class of games that focus on relationships between individuals or groups. In one such game, a player exchanges gossip with as many as seven other computer-controlled players. The topic of conversation is always feelings, positive or negative, expressed by one person for another. Clever responses increase popularity. Similar games could involve corporate politics, soap-operas, gothic romances, international diplomacy, and espionage. Although the category is undeveloped, it is promising because it addresses important fantasies. Other art forms devote attention to interpersonal relationships; it is only a matter of time before computer games follow a similar course.

CONCLUSIONS

This concludes my proposed taxonomy. Obviously, it has flaws, primarily because the criteria for division are a matter of historical happenstance. There is no fundamental reason why war games should be treated any differently from D&D games. Yet, both game systems evolved separately and are historically quite distinct. Similarly, the creation of an educational games category is my response to the efforts of educators to create games. My taxonomy is a patchwork because the set of available computer games is a patchwork. With the passage of time, market forces will assert themselves, and a more organized and consistent taxonomy will emerge.

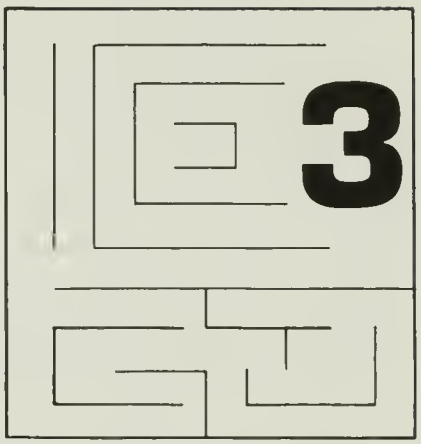
This taxonomy suggests a few observations about the state of game design with computers. For example, it should be obvious that there are few basic scenarios for skill-and-action games with one scenario per category. The archetypical game in each category spawned a family of imitators which incorporated variations and improvements. Moreover, the first game in each category was seldom the biggest moneymaker. Thus COMBAT led to SPACE INVADERS in the combat category,

DODGE 'EM prefigured PAC-MAN in the maze category, and PONG foreshadowed SUPERBREAKOUT in the paddle category.

Another observation arising from this taxonomy is that cognitive games are still underdeveloped compared to the S&A games. While S&A games fall into clear-cut categories, the categories of cognitive games are less satisfying and the distinctions between categories less clear. This ambiguity suggests much creative opportunity in the cognitive games field.

A taxonomy reflects the state of the material it attempts to organize. Computer game design is changing quickly. We would therefore expect the taxonomy presented here to become obsolete in a short time. New taxonomies must be created to reflect the changes in the marketplace in the next few years. For the present, however, the proposed taxonomy can provide us with an organized way to view the potpourri of games while suggesting new areas to explore.

THE COMPUTER AS A GAME TECHNOLOGY



Every art form is expressed through a physical medium. The control and manipulation of the medium is a technical problem the artist must master. Thus the sculptor must thoroughly understand the limitations of marble or bronze. The painter must understand the technology of paint and the behavior of light. The musician must be deeply skilled in the technology of creating sound. So too must computer game designers thoroughly understand their medium. The computer offers special possibilities and imposes special constraints on the designer. In this chapter I will discuss these possibilities and constraints. A few examples of simpler game technologies may help establish basic principles.

GAME TECHNOLOGIES

Cards involve a very simple set of physical equipment—52 pieces of cardboard imprinted on one side with a uniform pattern and on the other side with distinct symbols. The key traits of this equipment can be summarized as follows:

- There are many cards.
- Each card is unique.
- Each card possesses a numeric value.

- Each card possesses a suit, a two-bit value.
- The identity of a card can be selectively revealed.
- Each card is easily assignable to an owner.

These six characteristics fundamentally constrain the design of all card games. Each characteristic carries implications for game design with cards. Some things are easy to do with this technology; some are not. For example, games of probability are easily implemented with this technology, for numeric value and suit can be combined into many sets according to the laws of probability. The limitations on information created by the cards can be used to create games of guesswork and intuition. Indeed, one of the most intriguing of card games, poker, is based not so much on cold probability assessments as on the deceptions made possible by the limited information available in the game.

Like other technologies, cards also have weaknesses. For example, it would be tricky to design a card game for more than 52 players, because there are only 52 cards in one deck. It would also be very difficult to design a good skill-and-action game using cards as technology. Another design challenge would be to create a good athletic game using cards. Such games could be implemented with cards, but the results would probably be disappointing. Some things can be done well with cards and other things can't.

Another game technology, the board game, is somewhat more flexible than cards. Board games can be described but not rigorously defined. They use a large surface of paper or cardboard on which are printed various images, usually taking the form of a stylized map. Frequently the area represented on the map is divided into discrete regions by a regular geometric pattern (rectgrid or hexgrid), a segmented path to be traversed, an irregular division of regions, or a network of points connected by paths. The map itself remains the same throughout the game; players change the situation with a set of markers that can be moved about on the map. Sometimes a randomizing machine is used to determine outcomes of random processes; spinners or dice are most frequently used for this purpose. Sometimes cards from a special set are drawn to produce this randomizing function.

This technology has proven to be very successful for game designers. It easily accommodates groups of players and, with appropriate game design, can address a wide range of gaming situations. Chess is certainly the all-time classic board game. MONOPOLY, a successful early

board game, concerns real estate transactions. Other board games have addressed such disparate topics as life goals, murder mysteries, and race relations. The most ambitious modern board games are the war games. Among these are games with boards of 25 square feet, several thousand movable pieces, and a rules manual 50 pages long. A small industry has sprung up around these designs, complete with historical research, star designers, and jargon.

Board games provide a flexible and powerful technology for game designers. In recent years, however, design progress has stagnated. Many new board games look like cheap copies of MONOPOLY. War games, after showing a burst of creative energy in the sixties and seventies, have started to stagnate. Few fundamentally new ideas are being introduced; it may be that we have mined this vein to the limit.

What are the limitations of this technology? First and foremost, it is very difficult to maintain privileged information in a board game. All players can see the board and the position of all the markers. Second, the mechanics of handling all the pieces must be managed by the players. In some cases, as in the monster war games, this can become a sizable chore. For this reason, most board games are long affairs, frequently filling an evening. Short board games playable in twenty minutes or less are quite rare. Finally, should the pieces be disturbed, a board game would easily be ruined.

The central point of this discussion is that every game utilizes some technology and that each technology has strengths and weaknesses, things it can do well and things it does poorly. Astute game designers must fully grasp the strengths and weaknesses of the technologies they use. Let us now examine the computer as a game technology.

COMPUTERS

The most striking feature of the computer in a game context is responsiveness. Responsiveness is vital to the interactiveness so important to any game. The computer can respond to the human player in a variety of ways. If the action in a card game or board game starts to drag, the players have no choice but to plod through or take desperate measures. There is no reason why a computer game could not speed up on demand. It could change the length of the game or the degree of difficulty or the rules themselves. SPACE INVADERS for the ATARI

2600 provides an example. The player can select one- or two-player versions, visible or invisible invaders, stationary or moving shields, fast or slow bombs, and a variety of other options. In effect, the player chooses the rules by which to play and the game is responsive to that choice.

The computer is dynamic; it imposes little constancy on any element of the game. Board games, card games, and athletic games all have invariable parameters that constrain the designer. Once you have printed 100,000 game boards, it becomes very difficult to modify the map. We can't have 53-card stud; the decks aren't made that way. And should some miracle of science produce footballs that kick farther, we will not be able simply to extend football stadiums without spending many millions of dollars. The computer is far less restrictive. All of the game parameters are readily changed, even during the course of the game. There is nothing stopping us from creating a football game in which the goal post recedes from the visiting team. Territories in computer war games can be switched around the map more easily than we move a chair across the living room. This flexibility is of paramount importance to the game designer. As yet, it has been put to little use.

A second valuable feature is the computer's ability to function as game referee. All other game technologies demand that somebody take the time to handle the administrative responsibilities of the game. Athletic games are the most demanding; they require several impartial referees or umpires to administer the rules of the game and to settle disputes. Card games and board games require that the players also function as referees. This is seldom a problem with card games, but it can be a big problem with board games, especially the more complex war games. Rules disputes and administrative foul-ups are unavoidable liabilities of board games. The computer can eliminate all of these problems. It can administer the game, freeing the player to concentrate on playing. Also, the computer can implement complex arithmetic and logical rules. With other technologies, game rules must be overly simple because the humans using them cannot be trusted to perform simple numerical calculations. The computer eliminates this restriction.

In the original version of EASTERN FRONT 1941, for example, I was able to use exceptionally complex victory calculations. Most board-based war games about the eastern front in World War II assign victory points for captured cities and perhaps for casualties inflicted and sustained. More complex calculations recognizing the realities of the campaign are too tedious for human computation. The original EASTERN

FRONT 1941 was able to calculate not only cities captured and casualties inflicted and sustained, but also the eastward progress of every German unit as well as the westward resistance of every Russian unit. The game was thereby able to provide a more realistic and meaningful measure of the player's performance.

Another advantage of the computer is real-time play. Other game technologies have pauses and procedural delays while administrative matters are dealt with. The computer is so fast that it can handle administrative matters faster than human beings can play the game. This makes real-time games possible. Skill-and-action games are the direct result. The speed of the computer also eliminates the need for turn-sequencing so common in card games and board games.

An additional strength of computers for game design is their ability to provide an intelligent opponent. All other games require a human opponent (solitaire card games are actually puzzles rather than games). The greatest success so far has been with chess-playing games. Programs written for microcomputers can now play chess well enough to challenge most non-rated players. These games represent the best we have achieved to date in game artifical intelligence. Most computer games, being far less intelligent, rely on overwhelming numerical advantage to compensate for the superior intelligence of the human player. With time, we can expect to see more sophisticated algorithms that provide more intelligent play from the computer.

Another advantage of the computer is its ability to limit in a purposeful way the information given to the players. This capability can be valuable. Guessing a random number between one and ten, for example, is not a very interesting challenge, but guessing your opponent's resources based on your assessment of his or her actions and personality can be an extremely interesting exercise. When guesswork is included in the framework of a complex and only partially known system, the challenge facing the human player takes on a decidedly real quality.

Limited information provides another important bonus. It encourages the use of imagination. If we know all the pertinent facts, the problem becomes one of deduction. But if we know only a portion of the truth, our minds grope for an appropriate model on which to hang our projections. What model could be more appropriate than the reality the game attempts to re-create? We are forced by lack of information to imagine ourselves in the real-world predicament postulated by the game so that we may deal with the problems imposed by the game. In

the process, the illusion of reality is heightened. The game draws us into its fantasy world more effectively.

Another feature offered by computers is their ability to utilize data transfer over telephone lines for game play. The use of telecommunications makes possible game structures that are out of reach of other technologies. It allows us to create games with huge numbers of players. Until now, administrative problems have made it necessary to limit the number of players in any game. Six is a rough upper limit for a non-refereed game; twelve require several referees and twenty players or more require many referees. Obviously, games with hundreds of players face many administrative problems. Indeed, the logistic problems of assembling all the players are prohibitive. These problems are solved by computers linked by a telecommunications network. With this technology, it should be possible to design games tying together thousands of players scattered over the continent. Players could drift into and out of the game; with large numbers of players, the coming and going of individuals would not be detrimental.

Like any technology, computers also have weaknesses. The first, and most painful one, is the limited I/O capability of most computers. The computer itself may be supremely responsive, but if the human players can't tell it what they want or if they fail to understand the computer's response, the machine's effective responsiveness is nil. In other words, the computer must communicate its responsiveness to the human; it does so through I/O. Most output is expressed through graphics and sound; most input is accomplished through keyboard, joystick, and paddle.

Good graphics are hard to come by. Even the Atari Home Computer System, boasting the best graphics in the microcomputer world, has limitations that severely constrain the game designer. You simply cannot show all the graphic details you would like to show. For example, I suspect that few game boards could be duplicated on a single screen with this machine. The number of colors, the combination of text with high-resolution graphics, and the size of the board all make the task hopeless. It is possible to use a variety of tricks to produce something functionally similar to a given game board. We could reduce the number of colors displayed, dispense with text, and design an oversize display through which the user would scroll. EASTERN FRONT 1941 uses all these tricks and the result is quite usable, but the game wends a tortuous path around the graphics constraints of the computer.

Of course, the computer also boasts some graphics advantages. I have yet to see the board game that can show animation or change itself the way a computer game can. These sensory features can dramatically increase the impact of any game. So the graphics picture is not all bad.

Another I/O restriction comes from the input requirements. Input to the computer must come through the keyboard or the controllers. This can make things difficult for the game designer. For example, you can't say much with a joystick or keyboard. A joystick can mimic five fundamental words: *up*, *down*, *right*, *left*, and *button*. A keyboard can say more, but only through a lengthy and error-prone sequence of keystrokes. The player who wishes to express a meaningful communication to the computer must successfully enter a long and clumsy string of simple commands.

Input is made even more difficult by the indirectness of keyboards and joysticks. Little about such devices directly corresponds to real-world activities. Simple actions can become complicated with the computer. If I give you a bat and tell you that your goal is to hit the ball, you will have few problems divining that you should swing the bat at the ball. A computer baseball game is not so easy to figure out. Do you press H for "hit" or S for "swing" or B for "bat?" Do you press the START key or the joystick trigger? Perhaps you should swing the joystick by its cable at the ball displayed on the television screen.

After I/O, the second weakness of the personal computer is its single-user orientation. These machines were designed for one person to use while seated at a desk. If two people use it, they may be forced to exchange seats, a clumsy and distracting procedure. With joysticks or paddle controllers, the problem is diminished but not eliminated. Therefore, many computer games are played alone, leading to the accusation that the games are antisocial. A board game invites a group of people to sit around the table. A computer game encourages one player, accepts two, but discourages more.

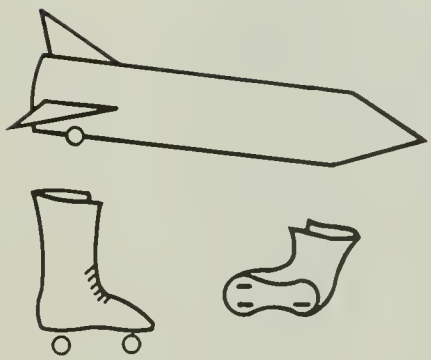
The final weakness of the computer is the requirement that it be programmed. No other game technology imposes so harsh a demand on the game designer. The board game designer can sketch an adequate board and construct simple playing pieces that serve quite effectively. When the time comes to produce the game, a professional can produce a quality version of the amateurish prototypes made by the designer. For this reason the designer need not be concerned with the technical aspects of game production.

The computer game designer does not have life so easy. The design must be implemented on the computer by programming. This is a tedious and difficult process, and it is not easily delegated, for the programming effort exerts a major influence over the design process. Implementing a design well is a major hurdle for any computer game designer.

DESIGN PRECEPTS FOR COMPUTER GAMES

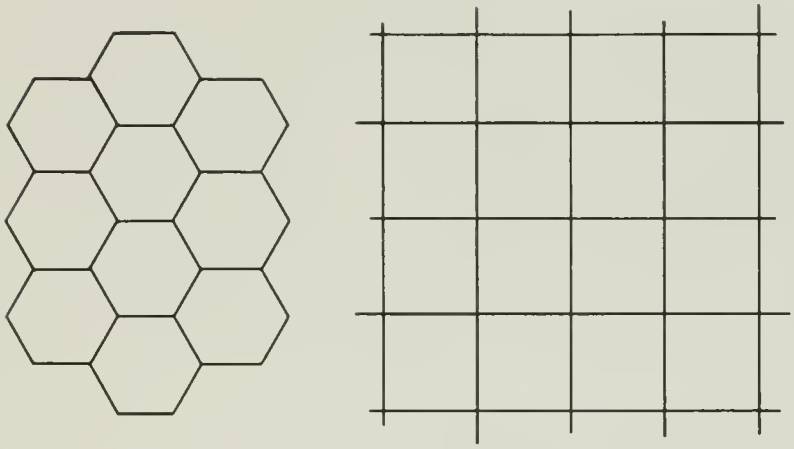
How do we translate an understanding of these strengths and weaknesses of the computer into a set of guidelines for game designers? These characteristics imply several precepts.

Precept #1: Go With the Grain



The first precept can be summarized: "Work with the grain of the machine, not against it." Too many game designers set out with unrealistic goals. They attempt to force the machine to perform tasks for which it is not well suited. This is not an excuse for lazy programming. But we must remember that the computer is the servant of the designer; the convenience of the computer is not the concern. The goal is to extract maximum performance from the computer, to make it work its best. We can only do this by making it perform functions it can perform well.

Case in Point: Hexgrids. An example of this principle might be illuminating. Board war games are traditionally executed on maps that use a hexgrid system to regulate movement and define positions. Hexgrids are preferable to rectgrids for several reasons. First, rectgrids have diagonals—two units can be diagonally adjacent. This situation can be very messy, and rules to cope with it are always burdensome and confusing. Hexgrids have no diagonals so they eliminate the problem. Second, hexgrids allow a player a choice of six directions in which to move, rectgrids only four. The greater range of choice allows the player more precise control over the movement and positioning of his or her pieces.



hexgrid rectgrid

It seems natural, therefore, that designers of computer war games would use hexgrids for their maps. Indeed, most computer war games do so—but it is a mistake. The hex does have advantages, but it imposes a penalty on computer war games that does not apply to board games. You can print anything you desire on a piece of paper, but the graphic display of the computer is not so accommodating.

A television display is fundamentally rectangular in its architecture. Horizontal lines are stacked in a vertical sequence. Such a display can easily represent rectangular shapes; hexagonal shapes don't work very well. To draw a hex, the program must draw four diagonal lines, each composed of a set of staggered dots. To make the hexgrid recognizable, the lines must be surrounded by an exclusion zone at least one pixel wide. This consumes a large portion of the screen if the hexes are small and dense. If they are larger, less screen area is consumed by the gridwork but fewer hexes can be shown on a single screen. Moreover, joysticks cannot be easily used with hexgrids because joysticks use rectangular geometry.

I do not wish to imply that hexgrids cannot be implemented on personal computer displays; on the contrary, they have already been used on many personal computers. The problems are that they are clumsy to display, lacking in graphic detail, and difficult to use. They just don't work smoothly. A topologically identical solution has been used in a few games: horizontally staggered rows of squares ("bricks") are used in

place of hexes. This system retains the flexibility of hexes while imposing fewer display problems, but it is difficult to use with a joystick.

For these reasons, I used rectgrids for EASTERN FRONT 1941. My decision was not based on laziness or unwillingness to tackle the problem of hexgrids. Indeed, I had already solved the problem with another game (TANKTICS) and could easily have transported the code. The experience I gained in working with TANKTICS convinced me that hexgrids weren't so important. The success of EASTERN FRONT 1941 seems to indicate that the lack of hexgrids need not impose a handicap.

Precept #2: Don't Transplant

One of the ogres of computer gamedom is the transplanted game. This is a game design, originally developed on another medium, which some misguided soul has seen fit to reincarnate on a computer. That this practice is widespread does not excuse its fundamental folly. The most generous reaction I can muster is the observation that we are in the early stages of computer game design; we have no sure guidelines and must rely on existing technologies to guide us. Someday these early transplanted games will seem as sophisticated as aircraft designs based on flapping wings.

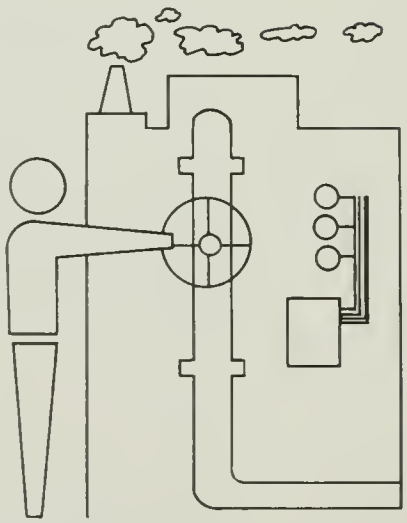
Why denounce transplanted games so vehemently? Because they are design bastards, the illegitimate children of two technologies that have nothing in common. Consider the worst example I have discovered so far, a computer craps game. The computer displays and rolls two dice for the player in a standard game of craps. The computer plays the game perfectly well, but the point is, why bother implementing on the computer a game that works perfectly well with another technology? A pair of dice can be had for less than a dollar. Indeed, a strong case can be made that the computer version is less successful than the original. Apparently, one of the appeals of the game of craps is the right of the player to shake the dice. Many players share the belief that a proper grip on the dice or speaking to them or perhaps kissing them will improve their luck. Thus, the player can maintain the illusion of control by participating rather than observing. The computer provides none of

this; the mathematics may be the same, but the fantasy and illusion aren't there.

In one way or another, every transplanted game, although it may gain a little, always loses more in the translation. Any game that succeeds in one technology does so because it is optimized to that technology; it takes maximum advantage of the strengths and avoids the weaknesses. The transplanted version uses the same design on a different set of strengths and weaknesses; it will almost certainly be an inferior product.

Any memorable work of art is as much a product of its vehicle of expression as it is a work of imagination. Shakespeare reads best in Elizabethan English; translation to modern English loses the linguistic panache we find so entertaining. The rhetoric of Isocrates, dull and drab in English, acquires a thrilling cadence in ancient Greek. Great books that touched our souls when we read them almost always disappoint us when we see their movie adaptations. Why should computer games be immune to this law of loss-on-translation?

Precept #3: Design Around the I/O

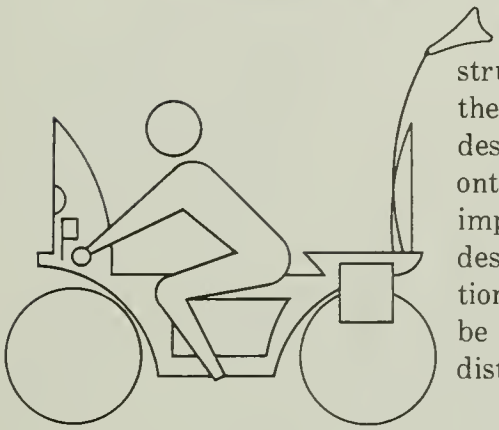


As I mentioned earlier, the computer's ability to calculate is a strength, but its I/O is a weakness. Thus, the primary limitation facing the computer game designer is not in the machine's ability to perform complex computations but in the I/O—moving the information between the computer and the human player. The game must be designed so that the information given to the player flows naturally and directly from the screen layout and sound output. I have seen far too many games with good game structures ruined by poor I/O structures. The user cannot appreciate the architectural beauties of a game buried in a confusing display structure. Even worse are games with poor input techniques, especially poor use of the keyboard. Most game players find keyboards difficult to use smoothly. Some difficulties can be challenging, but difficulties with keyboards generate only frustration. The implementation of games is dominated by the limitations of

I/O. What can and cannot be displayed, what can and cannot be input must decide the shape of the game.

A comparison of two of my own games provides an excellent example of the importance of I/O structures. EASTERN FRONT 1941 and TANKTICS are both war games dealing with World War II. Both provide reasonably intelligent opponents, complex detailed simulation, a rich variety of options, and thought-provoking strategic challenges. They differ primarily in their I/O. EASTERN FRONT 1941 was designed around its I/O; it provides clean, informative graphics and an easy joystick input system. By contrast, TANKTICS was designed around its game structure; its keyboard input system is clumsy and confusing and its alphanumeric screen display cryptic. EASTERN FRONT 1941 has been acclaimed by critics and has received awards. TANKTICS has been panned. The quality of a game's I/O structure is crucial to its success.

Precept #4: Keep It Clean



Many game designers fail to keep the overall structure of their game close to heart as they develop the details of the game structure. As they encounter design problems, they resort to quick patches grafted onto the main game structure without regard for the impact of such grafts on the overall cleanliness of the design. A game must have artistic unity to have emotional impact on its audience. Artistic unity can only be achieved by sticking to the theme and eschewing distracting details.

Any factors that do not contribute to the central theme of the game I call "dirt." The debilitating effect of dirt is seldom recognized, because dirt also endows a game with "color," the texture or feel that makes the game seem real. It is true that proper use of color will enhance a game. However, the game designer must realize that color is obtained at the price of a certain amount of dirt. The critical quantity then becomes the ratio of color to dirt. The designer wants the highest possible ratio, but sometimes, to increase the amount of color, must accept more dirt. In all cases, the inclusion of dirt in a game must be a conscious trade-off on the part of the designer, not

an accident produced by an attempt to resolve quickly some irritating problem.

Dirt most often arises from special-case rules that are applied rarely. For example, EASTERN FRONT 1941 has a number of such rules. The worst is the one forbidding Finnish units to attack. Inasmuch as there are only two Finnish units, this rule has little effect on the game as a whole, yet the player must still be aware of it. I had to put it in to solve a design problem: what's to stop the Finns from taking Leningrad all by themselves? But it clutters the game and the player's mind without adding much. A less dirty rule provides that Axis allies (Rumanian, Hungarian, and Italian units) fight with less determination than the Germans. There are six of these units in EASTERN FRONT 1941; thus, the rule is not quite so special a case and not quite so dirty. Another rule in EASTERN FRONT 1941 allows armored units to move faster than infantry units. Because the game has many armored units, this rule is not a particularly special case and is therefore not dirty.

In general, the narrower the range of application of a rule, the dirtier it is. My precept against dirt thus requires the designer to formulate rules that cover the entire game situation without recourse to special-case rules. In the perfect game design, each rule is applied universally. We can never achieve the perfect design, but we can and should strive to give each rule the widest possible application, which would require the player to consider the implications of each rule while making every decision in the game.

There is a school of game design that I derisively label the "humongous heap" school. Perpetrators of this approach design a game by selecting a simple structure and then piling onto it the largest possible jumble of special odds and ends (they call them "features"). These people design with a shovel instead of a chisel. They confuse magnitude with magnificence, intricacy with insight.

Precept #5: Store Less and Process More

The role of information storage in a computer is often misunderstood. A computer is not primarily an information storage device but an information processing device. Information storage is a necessary precondition for information processing, but it is not an end in itself.

Greater amounts of stored information permit greater processing, but if the processing capability is insufficient to realize the full potential of the storage, then that storage is wasted. The ideal program strikes a balance between storage and processing. Most game programs I have seen are long on storage and short on processing because data for storage—facts—are easier to obtain than is process-intensive material—program code. In taking the path of least resistance, most game designers end up going downhill.

Thus a game that stores huge quantities of static data is not making best use of the strengths of the computer. A game that emphasizes processing and treats information dynamically is more in tune with the machine. Relegate all static information to a rules book; paper and ink are still a better technology than personal computers for storing static information. Information that lies around and that must be dusted off before using has no place inside the microcomputer. As you look over your program listing, inspect each byte and ask yourself, "Am I getting my money's worth from this byte? Is it working hard for me, doing useful things frequently? Or is this a lazy byte that sits idle for hours and is used only rarely?" Fill your program with active bytes that do things, not with lazy bytes.

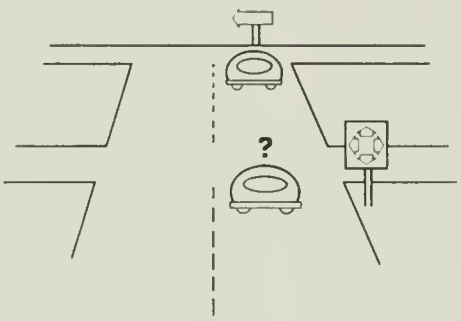
Lazy bytes are often associated with dirty rules. (They like to hang out together in sleazy pool halls.) Dirty rules are special cases that occur rarely. The bytes associated with them are seldom used, lazy bytes.

Another argument supporting this precept arises from more fundamental issues of game play. Interactiveness is central to game enjoyment. As mentioned earlier, the computer's plasticity makes it an intrinsically interactive device. Yet the potential of the computer can easily go unrealized if it is programmed poorly. A program emphasizing static data is not very dynamic. It is not plastic, hence not responsive or interactive. A process-intensive program, by contrast, is dynamic, plastic, responsive, and interactive. Therefore, store less and process more.

My final argument has more to do with games than with computers. You will remember from Chapter 1 that a game is distinguished from a story by the network of options it offers as opposed to the fixed thread of

a story. Much of the quality of a story is derived from the richness of the information it contains. A story is thus all information and no processing. A game derives its quality from the network of options it presents, options only accessible through the process-intensive aspects of the game. Games that are information-rich and process-poor are closer to stories than to the ideal game.

Precept #6: Branch Softly



A process-intensive game offers the player many branches from which to choose. These branches may be soft or hard. A hard branch is one whose characteristics are directly defined by the designer before the game is played. A soft branch is one whose characteristics are defined at the instant that the branch is encountered.

Consider, for example, an adventure game in which the player encounters a troll. If the player reads the book, the troll will allow that player to pass. If the player takes any other action, the troll will not allow the player to pass and the player's only recourse is to turn back. This is a hard branch; the designer specified the properties of the branch point during the design process.

Consider by contrast a game like EASTERN FRONT 1941. Each move by each unit has but four options: north, south, east, or west. Yet these four simple branches take on different meanings with each unit at each turn, because all the units are in motion. Thus it may be desirable to move one unit northward to attack an enemy unit; another unit might best be moved west to retreat from another enemy unit. Although the fundamental choices are always the same, the practical significance of each choice changes with the situation.

The difference between hard and soft branching is analogous to the difference between direct program commands and indirect ones. If we wish to adjust a variable in our program by adding a number to it, we might simply take the direct route and add a constant to the variable. However, if we are clever, we might add a variable quantity instead of a constant. This gives our program more flexibility and power because we can change the value of the additive value to suit whatever conditions we wish to address. This concept, called indirection, is fundamen-

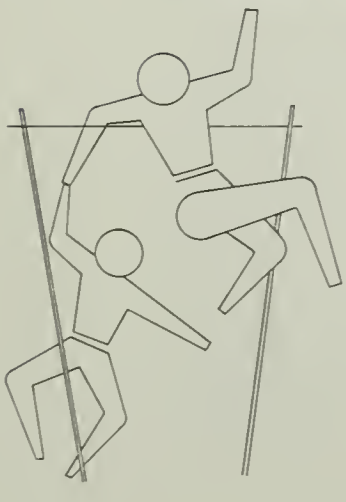
tal to all programming. Any program that does not use indirection is utterly useless.

Hard branching must be hand-wired by the game designer. This restricts the total amount of branching in the game to the amount the designer has time to hand-wire. The problem cannot be solved with high-capacity storage media like optical disks, for the limitation is not in the storage medium but in the designer's inability to anticipate every case and condition and to provide proper branches to deal with all allowed cases.

Soft branching, by contrast, allows the game designer to provide options for myriad cases and circumstances. This is because the designer need not address each case individually, but instead may deal with classes of cases. The members of each class share common traits yet are distinguished from each other by one or more variables. The designer must devise algorithms that respond to each case on the basis of the values of the variables.

The experienced programmer or game designer may dismiss this precept as an abstruse way of expressing an obvious principle. Yet this principle is so fundamental, and so often ignored, that it demands emphatic restatement. Designers should not use increased computer resources merely to heap more data into the machine in an attempt to specify all aspects of the game branching network. The real challenge is to develop more indirect algorithms that allow more deeply nested and widely interconnected branching. This is the essence of soft branching.

Precept #7: Maintain Unity of Design Effort



Games must be designed, but computers must be programmed. Both skills are rare and difficult to acquire, and their combination in one person is rarer still. For this reason, many people have attempted to form design teams consisting of a nontechical game designer and a nonartistic programmer. This system would work if either programming or game design were a straightforward process requiring few judicious trade-offs. The fact is that both programming and game design are desperately difficult activities demanding many painful choices. Teaming the two experts is rather like handcuffing a polevaulter to a

high jumper; the resultant disaster is the inevitable result of their conflicting styles.

More specifically, the designer/programmer team is bound to fail because the designer will make unrealistic demands on the programmer while failing to recognize golden opportunities arising during programming. For example, when I designed the game ENERGY CZAR (an energy-economics simulation game), I did not include an obviously desirable provision for recording the history of the player's actions. During the final stages of development, virtually everyone associated with the project suggested such a feature. From technical experience, I knew that it would require an excessive amount of memory. A nontechnical designer would have insisted upon the feature, only to face the disaster of a program too big to fit into its memory size.

Another example comes from EASTERN FRONT 1941. While writing the code for the calendar computations, I realized that a simple insertion would allow me to change color register values monthly. I took advantage of this opportunity to change the color of the trees every month. The improvement in the game is small, but it cost only 24 bytes to install, a cost-effective improvement. A nontechnical game designer would never have noticed the opportunity; neither would a nonartistic programmer.

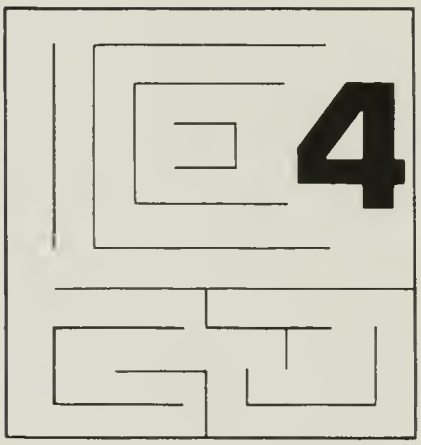
There is no easy way to produce good computer games. You must start with a good designer, an individual with artistic flair and a feel for people. That person must then learn to program. The opposite direction of development (from programmer to designer) will not work, for programmers are made but artists are born. When you find the rare individuals who are both designers and programmers, then you can hire subordinate designers and programmers and thus multiply their creative power. In the process, the subordinates will receive valuable training. In all cases, the creative process must be unified by a single mind. Committees are good for generating red tape, deferring decisions, and shirking responsibility, but they are useless when it comes to creative efforts.

CONCLUSION

In this chapter I have discussed the computer as a technology for game design. Discussions of computers and of their impact on society vary from the "gee whiz" school to the cynical school. The former group

sees a rosy future of countless triumphs wrought by the computer. The latter sees computers as a dehumanizing threat, a waste of time, or yet another vehicle for the expression of human perfidy. In this chapter, I have tried to present computers as one more technology like hammer and nails, clay and stone, paper and ink. Like any technology, computers can do some things well. And like any technology, they do some things poorly. The artist's role is to sidestep their weaknesses while using their strengths to greatest advantage.

THE GAME DESIGN SEQUENCE



Game design is primarily an artistic process, but, like all artistic activities, it is also a technical process. The game designer pursues grand artistic goals even while grinding through mountains of code. During the process of developing the game, the designer inhabits two apparently different worlds, the artistic and the technical. How does one manage the integration of such apparently dissimilar worlds? In short, how does one go about the process of designing a computer game? In previous chapters I have touched on some questions related to this process and established a few precepts. In this chapter I will suggest a procedure by which a computer game can be designed and programmed.

The procedure I will describe is based on my own experiences with game design and incorporates many of the practices that I use in designing a game. However, I have never used this procedure in a step-by-step fashion, nor do I recommend following the procedure exactly. Game design is far too complex an activity to be reducible to a formal procedure. Furthermore, the game designer's personality should dictate working habits. Most importantly, formal reliance on procedures is inimical to the creativity in game design. Because my experience in game design is primarily with personal computers, my suggestions may not be completely applicable to arcade game designers or home video game designers. Therefore, this procedure is not a normative formula but a set of suggested work habits that the prospective game designer might wish to assimilate. With these important qualifications in mind, let us proceed.

CHOOSE A GOAL AND A TOPIC

This vitally important step seems obvious, yet is ignored time and time again by game designers who set out with no clear intent. In my conversations with designers, I have often noticed indifference to the need for clear goals. Game designers sometimes admit trying to produce a "fun" game or an "exciting" game, but that is usually the extent of their thinking about goals.

A game must have a clearly defined goal, and it must be expressed in terms of the effect it will have on the player. It is not enough to declare that a game will be enjoyable, fun, exciting, or good; the goal must establish the fantasies that the game will support and the types of emotions it will engender in its audience. Since many games are in some way educational, the goal in these cases should establish what the player will learn. It is entirely appropriate for the game designer to ask how the game will enlighten its audience.

The importance of a goal does not become obvious until later in the design cycle. The crucial problems in game development with microcomputers are always problems of trade-offs. Everything that the game designer wants to do costs memory, and microcomputer memory is always in short supply. Thus, the designer must make trade-offs. Some game features can be included; some must be rejected. At two o'clock in the morning, when you face the awful decision of rejecting one of two highly desirable features, the only criterion for making this painful choice is the goal you have established for the game. If your goals are clear, your decision will be painful but obvious. If they are murky, you may well make the wrong choice, and whatever you choose, you will never know if your decision was correct.

How does one select a proper goal? There is no objective answer to this question. The selection of a goal is undeniably the most subjective process in the art of computer game design. This is your opportunity to express yourself; choose a goal in which you believe, a goal that expresses your sense of aesthetics, your world view. Honesty is an essential in this enterprise. If you select a goal to satisfy your audience but not your own taste, you will surely produce an anemic game. If you are true to yourself in selecting your goal, your game can be executed with an intensity that others will find compelling, whatever the nature of the game. If you are false to yourself, your game will necessarily be second-hand.

There are situations in which it is not quite possible to attain this artistic ideal. For example, I would not suggest that only immature, childish people should design games for children or that good "shoot-'em-up" games can only be done by shoot-'em-up personalities. The realities of the marketplace demand that such games be written, and it is better that they be written by mature professionals. But such emotionally indirect games will never have the psychological impact, the artistic power of games coming straight from the heart.

Once you have settled on your goal, you must select a topic. The topic is the means of expressing the goal, the environment in which the game will be played. It is the concrete collection of conditions and events through which the abstract goal is realized. For example, the goal of STAR RAIDERS apparently concerns the violent resolution of anger through skillful planning and dexterity. The topic is combat in space. The goal of EASTERN FRONT 1941 concerns the nature of modern war, especially the difference between firepower and effectiveness. The topic is the war between Russia and Germany.

Most game designers begin by selecting their topic with their goals subordinated to the topic. They commonly describe a game under development by its topic rather than its goal. When I tell other designers I am working on a game about leadership, I am met with quizzical expressions. Is it a space game, a war game, or a dungeon game, they wonder. They seem satisfied when I tell them it's a game about King Arthur. It is a serious mistake to subordinate the goal to the topic. Although your initial flash of inspiration may focus more on the topic than the goal, you must have the determination to take control of the design and impose your goal onto the topic rather than allow yourself to be swept away by the topic's momentum.

Selecting a good topic can be time-consuming. Each topic must be carefully examined for its potential to realize the goals. Many topics carry with them associations that may interfere with the goals of the game. My most recent game design effort, for example, uses the Arthurian legends as its topic. My goal in the game is to examine the nature of leadership, and I found the Arthurian legends to be a compelling vehicle. Unfortunately, these legends contain much male braggadocio, much vanquishing of opponents by brute force. This directly contradicts some of the points I want to make with the game, weakening the usefulness of the topic. But I find the legends so powerful and so malleable that I am willing to accept and work around this potential pitfall.

RESEARCH AND PREPARATION

With a goal and topic firmly in mind, the next step is to immerse yourself in the topic. Read everything you can; study all previous efforts related either to your goal or your topic. Which aspects of these earlier efforts appeal to you? Which disappoint or anger you? Make sure that you understand the mechanics of the environment your game will represent. Your game must have the authentic feel, the texture of the real world, and you can only achieve this if you firmly understand the environment of the game.

While researching EXCALIBUR, I studied the history of Britain during the period A.D. 400—700. I found little in the history books that was harmonious with my goal of exploring leadership, but in the Arthurian legends I found recurring themes more closely related to the goal. You may well find yourself adjusting your goals as you research your topic. Such a change of heart is an admission of poorly defined goals, but it reflects an honest willingness to adapt to the exigencies of the topic-environment. It is a departure from the ideal in which I have sinfully indulged myself many times.

During this phase it is critical that you commit little to paper and, above all, write no code! Take long walks as you contemplate your game. Cogitate. Meditate. Let the goal, the topic, and the facts gleaned from your research simmer together in your mind. Take your time with this phase; impatience now will lead to mistakes that can kill the game. I give myself at least three weeks to develop a game idea before proceeding to the next step. EXCALIBUR consumed several months in the meditation stage. During this time I kept my fidgeting hands busy writing an opening graphic display that had little to do with the final game.

During this phase you will generate a variety of specific implementation ideas. They will not all fit together neatly but will require much sorting and rearranging before they can be used. You should not wed yourself to any of them. A large collection of ideas is a useful resource. A laundry list of implementation ideas that must be included is a liability. Indulge yourself in creating implementation ideas but be prepared to winnow them ruthlessly during design.

For example, another designer and I recently designed a corporate politics game. During the research and preparation phase, we came up with a long list of clever ideas we wanted to include. We had agreed that

the game would have a feminist point of view without being preachy. We wanted to have a demanding boss, tough projects, deadlines, brownie points, one "male chauvinist pig," neutral males, neutral females, family and home obligations, mentors, and competition for the big promotion. We managed to include almost all of these ideas in the final design but were unable to integrate the important family elements. Every design we created failed. In the end, we had to discard this desirable element.

DESIGN PHASE

You now have a clear idea of the game's goals and topics, but you know nothing of its form. You are now ready to begin the concrete design phase. Your primary goal in this phase is to create the outlines of three interdependent structures: the I/O structure, the game structure, and the program structure. The I/O structure is the system that communicates information between the computer and the player. The game structure is the internal architecture of causal relationships that define the obstacles the player must overcome in the course of the game. The program structure is the organization of mainline code, subroutines, interrupts, and data that make up the entire program. All three structures must be created simultaneously, for they must work in concert. Decisions relating primarily to one structure must be checked for their impacts on the other structures.

I/O Structure

I prefer to start with the I/O structure because it is the most constraining of the three. I/O is the language of communication between the computer and the player. Like any language, it is the funnel through which we must squeeze the avalanche of thoughts, ideas, and feelings we seek to share with other human beings. I/O dictates what can and cannot be done with the game.

The computer has two basic means of output to the player: graphics on the screen and sound. In the future, we may see more exotic output devices, but for the moment, these are the two most common. Graphics are the more important of the two because human beings use vision more than hearing. Consequently, many game designers devote most of

their energy to designing quality displays. Indeed, some designers design the display first and let the game develop from the display, as extreme an example of goalless design as we could imagine.

Don't make the common mistake of creating cute graphics solely to show off. Graphics are there for only one reason: to communicate to the user forcefully and with feeling. Plan functional, meaningful graphics that convey the critical game information while supporting the fantasy of the game. Don't use graphics tricks as a crutch for bad game design. If the game is boring, no amount of graphic giftwrapping is going to fix it. Use of sound should follow the same rule: use it to tell the player what's going on in the game. The only place in which striking but uninformative graphics and sound can be useful is at the beginning of the game to establish the mood or tone.

Story boards (sequences of sketches used to plan video presentations) are a tempting graphics design tool, a well-developed technology from the film industry. They are not appropriate to games because story boards are intrinsically sequential. Games are not sequential; they are branching tree structures. The game designer who uses a sequential tool risks having designs made subtly sequential. The tool shapes the mind of the user; the freeway suggests that we drive wherever it takes us, not where we choose to go. Similarly, story boards impress their sequential character upon our games.

Devote special care to the input structure. It is the player's tactile contact with the game. People attach deep significance to touch. Have you ever noticed the tremendous importance programmers attach to the feel of a keyboard? Players will do the same thing with your game. The games JAWBREAKER and MOUSKATTACK are examples. In both games the joystick entry routine admits an unfortunate ambiguity when a diagonal move is made. This gives the player the impression the joystick is unresponsive. I have seen players slam down the joystick in frustration and swear they would never play the thing again. Remember this well as you plan your input structure: will it frustrate and anger your players?

The input structure raises a fundamental dilemma all game designers must face. A well-designed game allows the player to interact heavily with the opponent, to invest a great deal of his or her personality in the game. This requires that the game offer players a large number of meaningful options, enough that the players can express the nuances of their personalities through the choices they make. Yet decisions must be

input, and a large number of options seem to require an extensive and complicated input structure, which could be intimidating to the player. Our dilemma, then, is that an excellent game seems to require a huge input structure.

The dilemma is resolved by creatively designing a clean input structure that allows many options. This is not easy; many schemes must be considered and rejected before a satisfactory solution is found. Yet such a solution is often possible. In designing SCRAM, a nuclear power plant game, I faced this problem: how can a player control an entire nuclear power plant with only a joystick? At first glance, the task seems hopeless. Nevertheless, the solution I eventually discovered works very well. The player moves a cursor through the plant display. With the cursor adjacent to a piece of controllable equipment, the player presses the joystick button and pushes the stick up to turn on or increase power or down to turn off or decrease it. The system is simple and easily understood once the player has used it.

There is a general theoretical solution to the dilemma of option richness versus input cleanliness. I call this solution the "webwork." To design a webwork game, we start with a small number of pieces, then define a relationship that applies to all pairs of pieces. The set of relationships between pieces constitutes a webwork. The webwork can easily become quite complex, although few pieces are required to create it. In general, the number of pair relationships is equal to $N \times (N-1)$, where N is the number of pieces. Thus, four pieces can generate 12 pairings, 8 pieces can generate 56 pairings, and 16 pieces can generate 240 pairings. With fewer pieces to manipulate, the player faces fewer I/O problems but still enjoys a rich set of relationships in the game.

Backgammon illustrates the simplicity and power of webwork games. Backgammon uses only 30 pieces and 26 positions. The relationships between pieces are fairly simple and are expressed through the ability to move and bump. Yet on any given move, each piece has an offensive, defensive, blocking, or blocked relationship with most of the other pieces on the board. Almost every other board position in front of the piece can be reached, given the right die roll. It is no accident that the length of the playing area (24 steps) is exactly equal to the maximum die roll. In this way, all of the pieces are in range of each other, thereby maximizing the number of pair relationships.

Most webwork games rely on spatial webworks. These are easy to depict and easy for the player to visualize. Few games use non-spatial

webworks; my own GOSSIP is one such game. Curiously, though, GOSSIP uses a spatial webwork for its internal computations. This may imply either that game webworks are intrinsically spatial or that I cannot shake my mind free from spatial webworks.

The choice of input device is an important design decision. A good game designer should avoid the use of the keyboard and restrict input to a single simple device such as a joystick, paddle, or mouse. The value of these devices is not in any direct element of superiority over the keyboard but in the discipline they impose on the designer. Simple input devices go hand in hand with simple input structures.

The I/O structure is the most important of the three structures, for it is the face of the game that the player sees. It is the vehicle for interaction. It is also the most difficult of the three structures to design, demanding both sensitivity and complete technical mastery of the computer. Give it the care it deserves.

Game Structure

The central problem in designing the game structure is how to distill the fantasy of the goal and topic into a workable system. The game designer must identify some key element from the topic environment and build the game around that key. The key element must be central to the topic, representative or symbolic of the issues addressed in the game, manipulable, and understandable. For example, in EASTERN FRONT 1941, I started with the enormous complexity of modern warfare and extracted a key element: movement. Movement dictates the dispositions of the military units. Moving into an enemy's position initiates combat. Moving behind the enemy disrupts supplies and blocks the retreat route. Moving into a city can lead to its capture. Movement is not central to all aspects of war; it is, however, the key element through which many aspects of war are expressible. It is easily manipulable and immediately understandable.

GOSSIP presented a more difficult design challenge. This game addressed social relationships. The complexity of the subject matter and the intricate twists of human interaction suggested that the subject was beyond treatment as a game. After much thought I was able to isolate a key element: affinity. In one way or another, many of our social exchanges reduce to one of two declarations: a first-person statement of feeling ("I rather like Sandra") or a third-person statement ("Well, Tom

told me that he doesn't like Sandra one bit"). This key element encapsulates the grand array of human exchanges rather well. It is manipulable, quantifiable, and understandable. The isolation of the statement of affinity as the key element of human interaction made possible the game GOSSIP.

Manipulability assumes tremendous importance to the success of the game. The key element must be manipulable but in specific ways. It must be expressively manipulable; that is, it must allow the player self-expression, to do the things desired or necessary to experience the fantasy of the game. For example, in a combat game, shooting is almost always a key element. If the player's freedom to shoot is heavily restricted, the player cannot experience the fantasy. Similarly, manipulability must also be concise. To use the combat example again, if the player is required to declare the amount of gunpowder to be expended on each shot, he may find manipulability to be a hindrance to the game. Finally, the manipulability must be focused; the options from which the player chooses while manipulating the key element must be closely related. In GOSSIP, for example, the key element (statement of affinity) assumes a linear sequence of values from hatred through love. ENERGY CZAR violates this principle by requiring the player to choose from a large set of unrelated options. Menu structures and reliance on the keyboard can both result from unfocused key elements.

Many games employ multiple key elements. For example, most combat games include both movement and shooting. If both key elements are kept simple, or if one key element retains primacy, the game can be successful. However, too many key elements violating too many principles will rob the game of its focus.

Your main problem in creating the I/O structure is overcoming constraints; your primary concern in creating the game structure is realizing possibilities. Your work with the I/O structure defines the limitations on the structure of the game. You can take more liberties with the internal structure because the player will not directly encounter it. For example, in TANKTICS I developed a complex combat algorithm that realistically calculated the effects of armor-piercing shot. The complexity of this algorithm would have confused the player had I tried to explain it. But the player does not need to understand the internal algorithm; he need only experience its effects. I was not constrained to design a simple, understandable algorithm.

Concentrate on providing enough color to convey the authentic feel of reality. Keep your sense of proportion while adding details. It will do your game no good to provide exquisite detail in one area while overlooking the fundamental elements in another.

A common mistake many designers make is to pile too many features onto the game structure. In so doing, they create an overly intricate, dirty game. As I discussed in Chapter 3, dirt is undesirable; a game must be a clean structure, not a brushpile. Dirt creates another problem not mentioned in Chapter 3: it fouls the I/O structure. For example, the long-range scan feature of STAR RAIDERS does provide some additional capabilities, but it adds another keystroke the player must memorize. That's dirty input. Fortunately, this problem is overridden in STAR RAIDERS, because the fantasy puts the player at the controls of a starship. The intricacy of the control layout becomes a supporting element of the fantasy rather than a hindrance. In many games, you may be forced to surrender elements of the game structure to maintain the quality of the I/O structure. Nevertheless, you may be forced to change the I/O structure to incorporate a feature you are unwilling to abandon. If you do so, do not simply tack on a new command; rethink the entire I/O structure and modify it so that the new command is well integrated into the I/O structure.

Designing the game structure is an emotional experience very different from designing the I/O structure. While designing the I/O, the designer must sail a precarious path between the Scylla of expressive power and the Charybdis of expressive clarity while the storms of hardware limitations toss the design to and fro. While creating the game structure, the designer floats on a placid sea stretching flat to the horizon. The taunting challenge now is "Where do I go?"

Program Structure

The program structure is the third object of your design attentions. This structure is the vehicle which translates the I/O and game structures into a real product. One of the most important elements of program structure is the memory map. You must allocate chunks of memory for specific tasks. Without such safeguards, you may expend excessive quantities of memory on minor functions and have insufficient memory remaining for important tasks. Definitions of critical variables

and subroutines are also necessary. Finally, some documentation of program flow is important: use flowcharts, Warnier-Orr diagrams, or whatever suits your fancy. This book is not primarily concerned with programming. If you need guidance, consult a few of the many excellent books on program development.

Evaluation of the Design

You now have three structures in mind: the I/O structure, the game structure, and the program structure. You are satisfied that all three will work and that they are compatible with one another. The next step in the design phase is to evaluate the overall design for common flaws. The first and most important question is: does this design satisfy my goals? Does it do what I want it to do? Will the player experience what I want him to experience? If you are satisfied that the design does pass this crucial test, proceed to the next.

Examine the stability of the game structure. Remember that a game is a dynamic process. Are there any circumstances in which the game could slip out of control? For example, if the game uses money, could a situation arise in which the player could suddenly become ridiculously wealthy? In short, does the game structure guarantee reasonable upper and lower limits on all values? If not, reexamine the game structure carefully with an eye to structural changes. If you have no other alternative, you may be obliged to use brute force methods (for example, "IF MONEY > 10000 THEN MONEY = 10000").

Now probe the design for unanticipated shortcuts to victory. A player who can find a way to guarantee victory with little effort will not derive full benefit from the game. Ensure that all unintended shortcuts are blocked so that the player must go through the processes you intend. Any obstacles you place must be unobtrusive and reasonable. An example of obtrusive blocking comes from the game WAR IN THE EAST. This war game also deals with the Eastern Front in World War II. The Germans blitzed deep into Russia but ground to a halt before Moscow. To simulate this, the designers gave the Germans overwhelming superiority but also gave them a supply noose short enough to jerk them to a dead halt just outside Moscow. The effect was correct, but the means of achieving it were obvious and obtrusive.

The last and most crucial decision is whether to abort the game or to proceed. It should be made now, before you commit to programming the

game. Even if you abort now, you will still have learned a great deal and can say that the effort was worthwhile. A decision to give up at a later stage will entail a real loss, so give this option careful consideration. Abort if the game no longer excites you. Abort if you have doubts about its likelihood of success. Abort if you are unsure that you can successfully implement it. I have in my files nearly a hundred game ideas. Of these, I have explored at length some 30 to 40. Of these, all but eight were aborted in the design stage.

PRE-PROGRAMMING PHASE

If the game has survived this far, you are now ready to commit your ideas to paper. Until now, your documentation has been sketchy, more like notes and doodles. Now you are ready to prepare your complete documentation. First, commit all of your design ideas from the previous phase to paper. Define the I/O structure and the internal game structure. The documentation should emphasize the player's experience over technical considerations. Compare this first set of documents with your preliminary program structure notes. Adjust the program structure documents if necessary.

PROGRAMMING PHASE

This is the easiest of all the phases. Programming is straightforward and tedious work, primarily requiring attention to detail. Seldom has a game failed solely because the programmer lacked requisite programming skills. Games have sometimes failed to live up to their potential because the programmer did not expend enough effort or rushed the job or didn't bother to write in assembly language. But in few cases has talent or the lack of it been the crucial factor in programming a game; effort or the lack of it is most often responsible. If you place all your self-respect eggs in the programming basket, I suggest that you get out of game design and work in systems programming. Otherwise, write the code and debug it.

PLAYTESTING PHASE

Ideally, playtesting is a process that yields information used to polish and refine the game design. In practice, playtesting often reveals fundamental design and programming problems that require major efforts

to correct. Thus, playtesting is often interwoven with a certain amount of program debugging.

Sometimes playtesting reveals that the game is too flawed to save. A nonfatal, correctable flaw is usually a matter of insufficiency or excess: not enough color, too many pieces, not enough action, too much computation required of the player. A fatal flaw arises from a fundamental unforeseen conflict between two important elements of the game. You must have the courage to discard a game with a fatal flaw. Patching after the game is programmed can only achieve limited gains.

If playtesting reveals serious but not fatal problems, you must weigh your options very carefully. Do not succumb to the temptation to do a quick-and-dirty patch job. Many times the problem discovered in playtesting is really only a symptom of a more fundamental design flaw. Be analytical; determine the essence of the problem. Then take plenty of time to devise several solutions. Don't rush this process; sometimes the ideal solution comes from an unexpected angle. Choose a solution for its promise of faithfulness to your goals. Do not opt for the easiest solution but the one that best meets your goals.

While designing EASTERN FRONT 1941, for example, I ran into a severe problem: there were far too many units for the player to control easily. After wasting much time trying to devise ways to shrink the map or reduce the number of units, I eventually stumbled upon zones of control, a standard war gaming technique that extends the effective size of a unit by allowing that unit to inhibit enemy movement in squares (or hexes) adjacent to it. The inclusion of zones of control in the game not only solved the unit count problem but also made the logistics rules more significant and gave the game a richer set of strategies. I set out with the narrow goal of reducing the unit count, but I found an improvement with much broader implications.

If your initial design was well developed (or you are just plain lucky) the game will not encounter such crises. The problems you will face will be problems of polish. The little things that make a game go will be out of tune, and the game will move like a drunken dinosaur instead of like the lithe leopard you had envisioned. Tuning the game will take many weeks of work. It requires delicate adjustments of all the game factors; any other changes will only be a disruption. Therefore, defer final tuning work until the very end of the polishing stage.

There are actually two forms of playtesting. The first, which we considered earlier in this chapter, is your own playtesting done in the final

stages of debugging. The second form comes later when you turn over the game to other playtesters. The main difference between the two lies in the kinds of bugs exposed. Your own playtesting should reveal and eliminate all program bugs and many of the game bugs. The game you give to the playtesters should be free of program bugs; they should discover only bugs in the game structure. There is no reason to show an incomplete game to playtesters. In fact, you might destroy their objectivity by showing them a version of the game too early. But the time will come when you feel that the game is very close to completion and your own stock of ideas for improvements is dwindling. This is the time to show the game to a few select playtesters.

Playtesters must be selected with great care. You cannot simply grab a few friends and ask them what they think of the game. You need playtesters who have a deep familiarity with games, who can analyze and criticize your game from experience. Ideally, playtesters would also be game designers who would share your appreciation for the trade-offs essential to good game design. You should also know the playtesters well, both their personalities and their game tastes. You should never use more than five or six testers. A surplus only ensures that you will not be able to assess carefully the reaction of each.

A variety of other systems have been used for playtesting. Most rely on gathering large groups of "real people" and assessing their reactions to the game. I have little respect for such systems. Although they are scientific, objective, and democratic, they seldom yield useful design information, for consumers make lousy critics. Their suggestions too often are inane and impractical. They don't know enough about computers or games to give practical criticism. Such methods may work with detergent and shaving cream, but I doubt that any great movie, book, or song was created through market research of this kind. I will concede that such methods can prove to be a useful way to guide the mass production of cheap games by designers of limited talents, but this book is not written for them.

The playtesters will need a preliminary manual for the game. It need not be a finished product, any more than the game itself, but must include enough orientation information to get the playtester going. Make sure that there is enough in the manual so that the playtester doesn't waste time detailing problems that should be solved by the manual. Do not coach the playtester in advance; you will only destroy his or her objectivity. The playtester's first reaction to the game is your best

indication of the success of the manual. Let the tester experiment with the game for perhaps a week before you meet. Do not request lengthy written records of play performance; the playtester won't do it. Instead, include in the manual a few suggestions about potential problems that worry you. The most you should request in writing is a simple record of game options selected and subsequent scores.

Schedule a long interview after the playtester has had time to digest the game. Come to the interview with a set of standard questions that you ask all playtesters. Do not ask leading questions or solicit praise. Your job is to find flaws; accolades come later. Although it is more scientific to use a third person to conduct the interview (thereby assuring more honest answers), I prefer to get the information directly. I also encourage the playtester to criticize the game along with me and to suggest means of improving it.

Playtesters' criticisms are difficult to evaluate. Most must be rejected for a variety of reasons. Some are incompatible with your goals. Some are not achievable in the memory space. Some are reasonable but would require major software surgery incommensurate with the potential gains. Do not hesitate to reject 90% of the suggestions made. The remaining 10% are right; waste no time implementing them. How do you recognize the good 10%? This is the stuff of wisdom; I certainly don't know.

Polishing

The final stage of the design cycle is devoted to polishing the game. The polishing stage is actually concurrent with the later stages of playtesting and may involve several sessions with the playtesters. This stage is critical. The designer has been working on the game for a long time and the luster of the new design has worn off. It is now only a big job that should have been finished months ago. The playtesters love it, the publisher loves it and wants it right now, but the designer is sick of it. The urge to dump the project feels overpowering. Resist it; press on relentlessly and polish, polish, polish. Keep testing the game, fine-tuning, and adding tiny embellishments. Once it's out the door, it's gone forever.

Every game I have designed has followed the same pattern: I polished it until I never wanted to see it again. When at last I sent the game out, I rejoiced. I was free of it at last. Within a month, I was

regretting my impatience and wishing I could have a chance to clean up that one embarrassing bug I had never noticed. Within three months, my regret had turned to shame as I discovered or was told of more bugs. I have games out in the world whose authorship I hope never becomes widely known.

One of the final tasks you must perform before releasing the game is the preparation of a game manual. Manuals are given too little attention by almost everyone associated with computer games. This is a serious mistake, for the manual is a vital element in the overall game package. Computers have many limitations, and some of these can be overcome with a good manual. Much of the static information associated with a game can be presented in a manual. It is also an excellent place to add fantasy support elements like pictures and background stories. Finally, a well-written manual can clear up many of the misunderstandings that often arise during a game.

You should write your own manual for the game, no matter how poor a writer you are, and even if a professional writer will prepare the final manual. The attempt to write your own manual will increase your respect for the skills of the professional writer, making it more likely that you will have a productive relationship. Writing your own will also give you a feeling for the cleanliness of the game design—clumsy designs are hard to describe, clean ones easier. The manual is, after all, a rephrasing of the game design and thus allows one to access the game design from a fresh perspective. If a particular game feature requires cumbersome and tedious explanation, that should warn you that the game feature is itself cumbersome. Finally, your own manual will be a useful source document for the professional writer. You should be prepared for the writer to throw out your manual and start over. A good writer would rather create a new manual than polish an amateur's crude efforts. You must cater to the writer's needs, answering all questions as completely as possible. Only a close and supportive relationship between designer and writer can produce an excellent manual.

POST-MORTEM

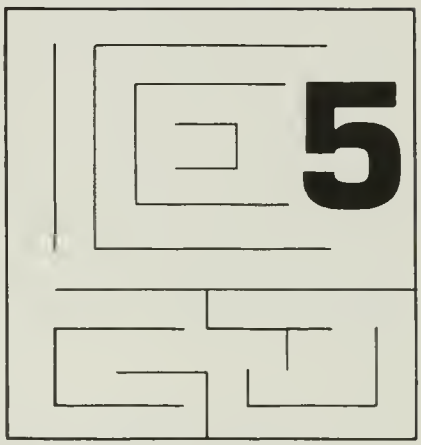
Once the program is out, brace yourself for the critics. They will get their filthy hands on your lovely game and do the most terrible things to it. They will play it without reading the rules. If it's a strategic game, they will castigate it for being unexciting; if it's an S&A game, they will

find it intellectually deficient. They will divine imaginary technical flaws and speculate incorrectly on the deep psychological hang-ups that led you to produce such a game. One critic of mine concluded that TANKTICS was obviously slapped together on a rush schedule. Actually, the time between first efforts and final publication was five years and two months. Another roasted ENERGY CZAR (an energy economics educational simulation) because it wasn't as exciting as his favorite arcade game.

Don't let these critics affect you. Although a few critics with the larger publications are quite thoughtful and worth paying attention to, most critics are far less qualified to criticize programs than you are to write them. You should pay heed only to views shared by three or more independent critics. Remember that even a good critic will roast you if your goal does not suit his or her personal taste.

The public is another matter. If they don't buy your game, you lose two ways: first, you or your employer make little money on the game, and second, you don't reach people with your message. The magnificence of your message doesn't matter—if nobody hears it, you have failed as an artist. One failure is nothing to worry about; every artist bombs occasionally. Two failures in a row are bad. Three should initiate a serious reconsideration of artistic values. Are you willing to be a noble and starving artist, or would you rather be a somewhat wealthier artisan?

DESIGN TECHNIQUES AND IDEALS



All artists develop their own special techniques for the execution of their art. The painter worries about brush strokes, mixing paint, and texture. The musical composer learns techniques of orchestration, timing, and counterpoint. The game designer also acquires a variety of specialized skills, techniques, and ideals. In this chapter I will describe some of the techniques that I use.

BALANCING SOLITAIRE GAMES

A solitaire game pits the human player against the computer. The computer and the human being are very different creatures. Human thought processes are diffuse, associative, and integrative; machine processes are direct, linear, and arithmetic. This creates a problem. A computer game is created for the benefit of the human player and is cast in the intellectual territory of the player, not of the computer. This places the computer at a disadvantage. Although the computer could easily defeat a person in games involving computation or sorting, such games would be of little interest to the human player. How do we design the game to challenge the human being? Four techniques are available: vast resources, artificial smarts, limited information, and pace.

Vast Resources

This is by far the most frequently used technique for balancing a game. The computer is provided with immense material resources that

it uses stupidly. These resources may consist of large numbers of opponents that operate with rudimentary intelligence. Many games use this ploy: SPACE INVADERS, MISSILE COMMAND, ASTEROIDS, CENTIPEDE, and TEMPEST. It is also possible to equip the computer with a small number of opponents that are more powerful than the human player's units, such as the supertanks in BATTLEZONE. The effect in both cases is the same: the human player's advantage in intelligence is offset by the computer's material advantage.

This approach has two benefits. First, it gives the conflict between the player and the computer a David-and-Goliath aspect. Most people would rather win as apparent underdog than as equal. Second, this approach is the easiest to implement. Providing artificial intelligence for the computer's players can be difficult, but repeating a process for many computer players requires little more than a simple programming loop. Of course, the ease of implementing this solution carries a disadvantage: everybody does it. Laziness and lack of determination account more for the prevalence of this technique than do game design considerations.

Artificial Smarts

The alternative to the use of sheer numbers is to provide the computer (as player) with intelligence adequate to meet the human player on equal terms. Unfortunately, artificial intelligence techniques are not sufficiently developed to be useful here. Tree-searching techniques have advanced far enough to allow us to produce fair chess, checkers, and Othello players. Any game that can be expressed in tree-searching terms can be handled with these techniques. Unfortunately, very few games are appropriate for this treatment.

An alternative is to develop special intelligence routines for each game. Since such routines are too primitive to be referred to as "artificial intelligence," I use the term "artificial smarts." This is the method I have used in TANKTICS, EASTERN FRONT 1941, and LEGIONNAIRE, with varying degrees of success. This strategy demands great effort from the game designer, for such special routines must be reasonable yet unpredictable.

Our first requirement of any artificial smarts system is that it produce reasonable behavior. The computer should not drive its tanks over cliffs, crash spaceships into each other, or bring targets to rest directly

in front of the player's guns. This requirement tempts us to list all possible stupid moves and to write code that tests for each and precludes it. This is the wrong approach, for the computer can demonstrate unanticipated creativity in the stupidity of its mistakes. A better (but more difficult) method is to create a general algorithm that obviates most absurd moves.

A second requirement of an artificial smarts routine is unpredictability. The player should never be able to anticipate the behavior of the computer, for this would shatter the illusion of intelligence and make victory much easier. This may seem to contradict the first requirement of reasonable behavior, for reasonable behavior tends to follow patterns that should be predictable.

The apparent contradiction can be resolved through a deeper understanding of the nature of interaction in a game. First, reaction to an opponent is in some ways a reflection of that opponent. A reasonable player tries to anticipate an opponent's moves by assessing the opponent's personality. Second, interactiveness is a mutual reaction—both players attempt to anticipate each other's moves. Third, this interactiveness is itself a measure of "gaminess."

A game becomes analogous to two mirrors facing each other with one player looking out from each mirror. A puzzle is analogous to two unreflective mirrors; the player sees a static, unresponsive image. A weakly interactive game is analogous to two mirrors being only partially reflective; each player can see and act at one or two levels of reflection. A perfectly interactive game is analogous to two perfectly reflective mirrors; the two players exchange places recursively in an endless tunnel of reflected anticipations. No matter how reasonable the behavior, the infinitely complex pattern of anticipation and counteranticipation defies prediction. It is reasonable yet unpredictable.

Unfortunately, a perfectly interactive game is beyond the scope of microcomputers, for if the computer is to anticipate human moves interactively, it must be able to assess the personality of its opponent—a hopeless task, as yet. For the moment, we must rely on more primitive guidelines. For example, my experience has been that algorithms are most predictable when they are "particular." By "particular" I mean that they place an emphasis on single elements of the overall game pattern. For example, in war games, algorithms like "determine the closest enemy unit and fire at it" are particular and yield predictable behavior.

But I have found that the best algorithms treat the greatest amount

of information in the broadest context. That is, they factor into their decision-making the largest number of considerations rather than focus on a small number of particular elements. To continue with the earlier example, a better algorithm might be "determine the enemy unit posing the greatest combination of threat and vulnerability (based on range, activity, facing, range to other friendly units, cover, and sighting); fire on unit if probability of kill exceeds probability of being killed."

How does one implement such principles into specific algorithms? I doubt that any all-purpose system can ever be devised. The best general solution I have found utilizes point systems, field analysis, and changes in the game structure.

Point Systems. First, I establish a point system for quantifying the value of each possible move. This is a time-honored technique used in many artificial intelligence systems. A great deal of thought must go into the point system. The first problem with it is one of dynamic range: the designer must ensure that the probability of two accessible moves, each accumulating a point value equal to the maximum value allowed by the word size (eight bits), approaches zero. In other words, we can't have two moves each scoring 255; we have no way to know which is truly the better move. This problem will diminish as 16-bit systems become more common.

A second problem with the point system is the balancing of factors against each other. In a hypothetical tank game, we could agree that climbing on top of a hill is good but that moving onto a road is good also. Which is better? If a hilltop position is worth 15 points, what is a road position worth? These questions are difficult to answer. They require a deep familiarity with the play of the game. Unfortunately, such familiarity is impossible with a game that has yet to be completed. The only alternatives are broad experience, intimate knowledge of the situation being represented, painstaking analysis, and lots of experimenting.

Field Analysis. A second element of my general approach to artificial smarts is the use of field analysis. This is only applicable to games involving spatial relationships. In such games the human player relies on pattern recognition to analyze positions and to plan moves. True pattern recognition on a human level is beyond the abilities of a microcomputer. Something approaching pattern recognition can be attained, however, through the use of field analysis. The key effort here is the

creation of a calculable field quantity that correctly expresses the critical information the computer needs to make a reasonable move.

For example, in several of my war games I have used safety and danger fields, which tell a unit how much safety or danger it faces. Danger is calculated by summing the quotients of enemy units' strengths divided by their ranges. Large, close units are very dangerous and small, distant units are only slightly dangerous. A similar calculation with friendly units yields a safety factor. By comparing the danger value with the safety value at its position, a unit can decide whether it should exhibit bold or timid behavior. Once this decision is made, the unit can measure the difference between danger and safety in each position into which it could move. If the unit is bold, it moves toward the danger; if it is timid, it moves away. Thus, the use of fields allows a unit to assess a spatial array of factors.

Changes in the Game Structure. Another technique for coping with artificial smarts problems is so simple it seems like cheating: change the game. If you can't come up with a good way to use a feature, you really have no choice but to delete it. For example, while designing TANKTICS, I encountered a problem with lakes. With a concave lake, the computer would drive its tanks to the shore, backup, and return to the shore. The concave lake was a trap for my artificial smarts algorithm. I wasted much time working on a smarter routine that would not be trapped by concave lakes but would still retain desirable economies of motion. After much effort I discovered the better solution: delete concave lakes from the map.

Ideally, the experienced game designer has enough intuitive feel for algorithms to sense intractable game factors and avoid them during the design stages. Most of us must discover these things the hard way and retrace our steps to modify the design. Exeriencing these disasters is part of what provides the intuition.

Coordination of Moves. A special problem is the coordination of moves with many different units under control of the computer. How is the computer to assure that the different units move in a coordinated way and that traffic jams don't develop? One way is to use a sequential planning system together with a simple test for the position of other units. Thus, unit =1 moves first, then =2, then =3, each one avoiding collisions.

I can assure you from my own experience that this system replaces collisions with the most frustrating traffic jams. A better way uses a

"virtual move system" in which each unit plans a virtual move on the basis of the virtual positions of all units. Here's how it works: we begin with an array of real positions of all computer units. Then we create an array of virtual positions and initialize all virtual values to the real values. Each unit plans its move, avoiding collisions with the virtual positions. Then it places its planned final position into the virtual array. Other units plan their moves. After all units have planned one virtual move, the process repeats, each unit planning its move on the basis of the interim virtual move array. This huge outer loop should be convergent. After a sufficient number of iterations, the routine terminates and the virtual positions form the basis of all moves made by the computer's units. This technique is useful for coordinating the moves of many units and preventing traffic jams.

Transitions Among Algorithms. No matter how good an algorithm is, its applicable range is limited. The odds are that a specific algorithm will work under relatively few conditions. A good game design must offer a broad range of conditions to be truly interesting. Thus the designer must frequently create a number of algorithms and switch from one to another as conditions change. The transition from one algorithm to another is fraught with peril because continuity must be maintained during the transition.

I remember well a frustrating experience with algorithm transitions in LEGIONNAIRE. The computer-barbarians had three algorithms: a "run for safety," an "approach to contact," and an "attack." Under certain conditions, a barbarian operating with the "approach to contact" algorithm would make a bold dash forward to make contact with the human player. Then the barbarian would make a transition to the "attack" algorithm which, unfortunately, would declare an attack unsafe. The barbarian would balk at the attack and convert to the "run for safety" algorithm which would then direct the barbarian to turn tail and run. The human player was treated to a spectacle of ferociously charging and frantically retreating barbarians, none of whom ever bothered to fight. I eventually gave up and redesigned the algorithms, merging them into a single "advance to attack" algorithm with no transitions.

Non-spatial Games. The artificial smarts techniques I have described so far are for use in games involving spatial relationships. Many games, however, are non-spatial, and other artificial smarts techniques are required. One of the most common types of non-spatial games uses

coupled differential equations to model complex systems. LUNAR LANDER, HAMMURABI, ENERGY CZAR, and SCRAM are all examples. The primary problem facing the designer of such games is not so much to defeat the human player as to model complex behavior. I advise the designer to be particularly careful with games involving large systems of coupled differential equations. HAMMURABI uses three coupled first-order differential equations, and most programmers find it manageable.

But the complexity of the problem increases enormously with the number of differential equations used. ENERGY CZAR used 48 differential equations, a feat made believable only by the fact that many constraints were imposed on them. In general, be wary of more than four coupled differential equations. If you must use many differential equations, try to use parallel ones in which the same fundamental equation is applied to each element of an array of values.

To help keep the system balanced, each differential equation should have a damping factor that must be empirically adjusted:

new value = old value + (driving factor/damping factor)

A small damping factor produces lively systems that bounce around wildly, a large one sluggish systems that change slowly. Unfortunately, recourse to simple damping factors can backfire when a negative feedback relationship exists between the new value and the driving force. In this case, large damping inhibits the negative feedback and one of the variables goes wild. The behavior of systems of differential equations is complex. I suggest that designers interested in these problems study the mathematics of overdamped, underdamped, and critically damped oscillatory systems. For more general information on solving systems of differential equations, a good textbook on numerical analysis will serve as a useful guide.

Conclusions Concerning Artificial Smarts

The application of all these methods may produce a fairly intelligent game, but expectations should not be too high. Even great effort is not enough to produce truly intelligent play. To date, none of my three efforts plays with intelligence that is adequate, by itself, to tackle a

human player. Indeed, they still require force ratios of at least two to one to compensate for the intelligence of the human player.

Limited Information

Another way to compensate for the computer's lack of intelligence is to limit the amount of information available to the human player. Without the information to process, the human player cannot apply superior processing power to the problem. This technique should not be used excessively, for this reduces the game to a game of chance. It can, nevertheless, equalize the odds. If the information is withheld in a reasonable context (the player must send out scouts), the restrictions on information seem natural.

Limited information provides a bonus: it can tease the player's imagination by suggesting without actually confirming. This only happens when the limitations on the information are artfully chosen. Random gaps in information are confusing and frustrating rather than tantalizing.

Pace

Another way to even the balance between human and computer is through the pace of the game. A player may be smart, but the computer is much faster at performing simple computations. If the pace is fast enough, the human player will not have enough time to apply superior processing skills. This is an easy technique to apply, so it comes as no surprise that it is heavily used by designers of skill-and-action games.

I do not encourage the use of pace as an equalizing agent. Increasing pace only succeeds by depriving the human player of the time needed to invest more imagination in the game. Without that investment, the game can never offer a rich interaction. Pace does for computer games what the one-night stand does for romance.

Summary

These four techniques for balancing computer games are never used in isolation. Every game uses some combination of the four. Most games rely primarily on pace and quantity for balance with very little intelligence or limited information. There is no reason why a game could not

use all four techniques. Indeed, this should make the game all the more successful, for by using elements from each method, the game would not have to strain the limitations of each. The designer must decide upon an appropriate balance for the goals of the particular game.

RELATIONSHIPS BETWEEN OPPONENTS

Every game establishes a relationship between opponents that each player strives to exploit to maximum advantage. The form of this relationship is fundamental to the game. It defines the interactions available to the players and sets the tone. To date, most computer games use very simple player-to-player relationships, thus limiting their range and depth. A deeper understanding of player-to-player relationships will lead to more interesting games.

Symmetric Relationships

The simplest architecture establishes a symmetric relationship between the two players, granting each the same strengths and weaknesses. Symmetric games offer the obviously desirable feature of being automatically balanced. They tend to be much easier to program because the same processes can be applied to each player. Finally, they are easier to learn and understand. Examples of symmetric games include COMBAT for the ATARI 2600, BASKETBALL, and DOG DAZE by Gray Chang.

Symmetric games suffer from a variety of weaknesses, the greatest of which is their relative simplicity. Any strategy that seems effective can and will be used by both sides simultaneously. In such a case, success results not from planning but from execution. The game turns on very fine details; chess provides an example—an advantage of but a single pawn can lead to victory.

Asymmetric Games

Because of the weaknesses of symmetric games, many games attempt to establish an asymmetric relationship between opponents. Each player has a unique combination of advantages and disadvantages which the game designer must somehow balance so that both sides have the same likelihood of victory, given equal levels of skill. The simplest

way of doing this is through the use of plastic asymmetry. These games are formally symmetric, but the players are allowed to select initial traits according to a set of restrictions. For example, in the Avalon-Hill board game WIZARD'S QUEST, the players are each allowed the same number of territories at the beginning of the game, but they choose their territories in sequence. Thus, what was initially a symmetric relationship (each person has N territories) becomes an asymmetric one (player A has one combination of N territories but player B has a different combination). The asymmetry is provided by the players themselves at the outset, so if the results are unbalanced, the player has no one else to blame.

Other games establish a more explicitly asymmetric relationship. Almost all solitaire computer games use an asymmetric relationship between the computer player and the human player because the computer cannot compete with human intelligence. The human player is given resources that allow her to bring her superior planning power to bear, and the computer gets resources that compensate for its inferior intelligence.

Triangularity

The advantage of asymmetry is that it allows the game designer to build nontransitive or triangular relationships into the game. Nontransitivity is a well-defined mathematical property. In the context of games, it is best illustrated by the rock-scissors-paper game. Two players play this game; each secretly selects one of the three pieces; they simultaneously announce and compare their choices. If both made the same choice, the result is a draw and the game is repeated. If they make different choices, then rock breaks scissors, scissors cut paper, and paper enfolds rock. This relationship, in which each component can defeat one other and can be defeated by one other, is a nontransitive relationship. That rock beats scissors and scissors beat paper does not mean that rock beats paper. Notice that this particular nontransitive relationship only produces clean results with three components. This is because each component relates only to two other components; it beats one and loses to the other. A rock-scissors-paper game with binary outcomes (win or lose) cannot be made with more than three components. One could be designed with multiple components if several levels of victory (perhaps using a point system) were admitted.

Nontransitivity is an interesting mathematical property, but it cannot be used to produce rich games as long as we adhere to the strict mathematical meaning of the term. The value of this discussion demands generalization of the principle into less well-defined areas. I use the term "triangular" to describe asymmetric relationships that extend the concepts of nontransitivity beyond its formal definition.

A simple example of a triangular relationship can be found in the game BATTLEZONE. When a saucer appears, the player can pursue the saucer instead of chasing an enemy tank. In such a case, there are three components: player, saucer, and enemy tank. The player pursues the saucer (side one of the triangle) and allows the enemy tank to pursue him unmolested (side two). The third side of the triangle (saucer to enemy tank) is not important to the human player—the computer maneuvers the saucer to entice the player into a poor position. This example is easy to understand because the triangularity assumes a spatial form within the game as well as a structural one.

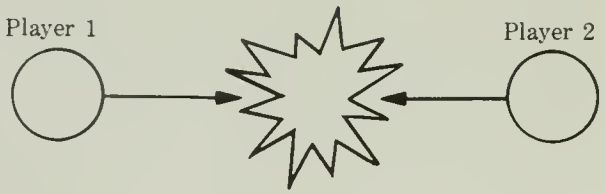
Triangularity is usually used with mixed offensive-defensive relationships. In most conflict games, regardless of the medium of conflict, there are offensive actions and defensive ones. Some games simply make one side the attacker and the other side the defender. This is risky business because it restricts the options available to each player. More entertaining are games that mix offensive and defensive strategies for each player. In this way, each player gets to attack and to defend. More importantly, players can trade off defensive needs against offensive opportunities. Triangular relationships develop from such situations.

The value of triangularity is its capacity for indirection. A binary relationship makes direct conflict unavoidable; players must approach and attack each other. These direct approaches are obvious and expected, and for this reason such games often degenerate into tedious exercises. A triangular relationship allows each player indirect methods of approach, leading to richer and subtler interaction.

Actors and Indirect Relationships

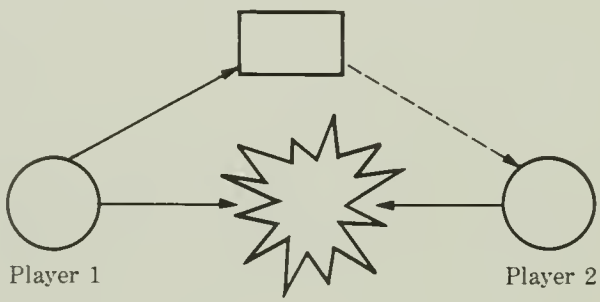
Indirection is the essential element triangularity provides in game design. Indirection is itself an important element to consider, and triangularity is only the most rudimentary expression of it. We can extend the concept of indirection further.

Most games provide a direct relationship between opponents, as shown in the following diagram:



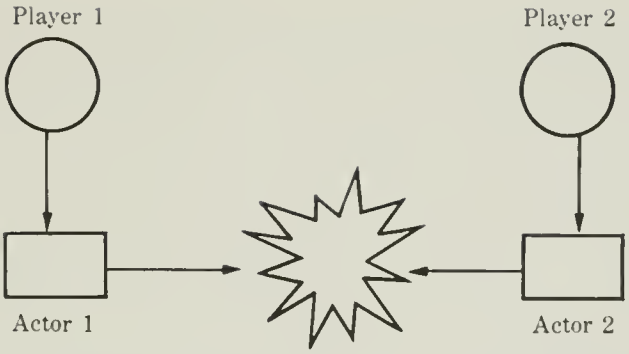
Since the opponent is the only obstacle facing the player, the simplest and most obvious resolution of the conflict is to destroy the opponent. This is why so many direct games are violent.

Triangularity, however, provides some indirection:



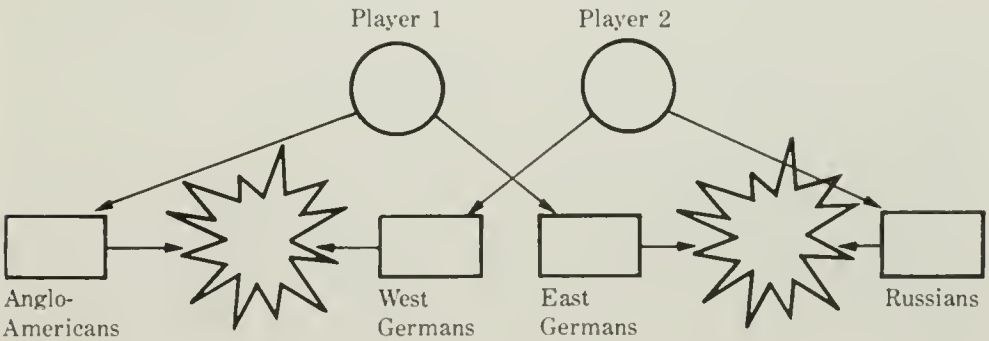
Using triangularity, each opponent can encounter the other through the third party. The third party can be a passive agent, a weakly active one, or a full-fledged player. However, it's difficult getting two people together for a game; therefore the third agent is often played by a computer-generated actor. An actor is not the same as an opponent. An actor follows a simple script. It has no guiding intelligence or purpose of its own. For example, the saucer in BATTLEZONE is an actor. Its function is distraction. Its script calls for it to drift around the battlefield without actively participating in the battle.

The actor concept allows us to understand a higher level of indirection.



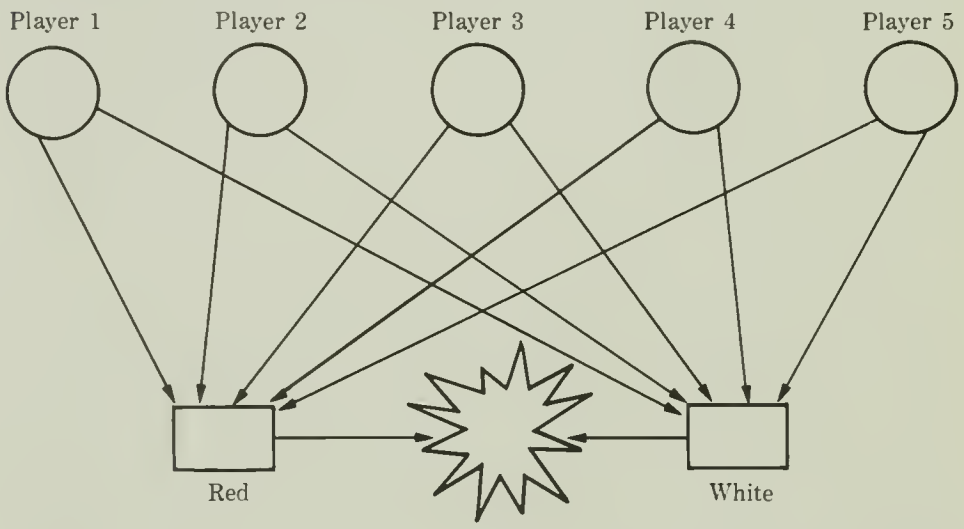
In this arrangement, the players do not battle each other directly; they control actors who engage in direct conflict. A good example of this scheme is shown in the game ROBOTWAR by Muse Software. In this game, each player controls a killer robot. The player writes a detailed script (a short program) for her robot. This script is used by the robot in a gladiatorial contest. The game thus removes the players from direct conflict and substitutes robot-actors as combatants. Each player is clearly identified with her own robot. But this form of indirection is unsuccessful because the conflict itself remains direct. Moreover, the player is removed from the conflict and forced to sit on the sidelines. This form of indirection is an unsuccessful transitional stage.

A higher level of indirection is exemplified by a clever board game design by Jim Dunnigan, BATTLE FOR GERMANY. This game concerns the invasion of Germany in 1945. This was obviously an uneven struggle, the Germans simultaneously fighting the Russians in the east and the Anglo-Americans in the west. Uneven struggles make frustrating games. Dunnigan's solution was to split both sides. One player controls the Russians and the west-front Germans; the other controls the Anglo-Americans and the east-front Germans. Thus, each player is both invader and defender.



Neither player identifies directly with the invaders or the defenders. The two combatants lose their identities and become actors.

The highest level of indirection I have seen is Dunnigan's RUSSIAN CIVIL WAR game. This board game concerns the civil war between the Reds and the Whites. Dunnigan's brilliant approach is to dissolve completely any identification between player and combatant. Each player receives some Red armies and some White armies. During the course of the game, a player uses his Red armies to attack and destroy other players' White armies. He also uses his White armies to destroy other players' Red armies. The end of the game comes when one side, Red or White, is annihilated. The winner is then the player most identifiable with the victorious army (with the largest number of loser's casualties and the smallest number of winner's casualties).



The indirection in this game is truly impressive. The combatants are in no way identifiable with any individual player until very late in the game. They are actors; Red and White battle without human qualities even though they are played by human players. There is only one limitation to this design: the system requires more than two players to work effectively. Nevertheless, such highly indirect player-to-player architectures provide many fascinating opportunities for game design.

Direct player-to-player relationships can only be applied to direct conflicts such as war, which tend to be violent and destructive. For this

reason, society discourages direct conflicts. Yet conflict remains in our lives, taking more subtle and indirect forms. We fight our real-world battles with smiles, distant allies, pressure, and cooperation. Games with direct player-to-player relationships cannot hope to address most examples of real human interaction. Only indirect games offer any possibility of successfully exploring the human condition.

SMOOTH LEARNING CURVES

As a player works with a game, steady and smooth improvement in score should result. Beginners should be able to make some progress, intermediate players should get average scores, and experienced players should earn high scores. A graph of a typical player's score as a function of time spent with the game should show a curve sloping smoothly and steadily upward. This is the most desirable case.

A variety of other learning curves can result and they can reveal a great deal about the game. If a game has a curve that is relatively flat, we say that the game is hard to learn. If the curve is steep, the game is easy. A sharp jump in the curve suggests that there is just one trick to the game, mastery of which guarantees complete mastery of the game. If the game has many sharp jumps, there are many tricks. A particularly bad case arises when the player's score falls or levels off midway through the learning experience, indicating that the game contains contradictory elements that confuse and distract. The ideal curve always slopes upward smoothly and steadily.

Games without smooth learning curves frustrate players by failing to provide them with reasonable opportunities for improving their scores. Players feel that the game is either too hard, too easy, or simply arbitrary. Games with smooth learning curves challenge players at all levels and encourage continued play by offering the prospect of new discoveries.

A smooth learning curve results from a game that provides a smooth progression from beginner's level to expert. This requires the game designer to create not one game but a series of related games. Each must be intrinsically interesting and challenging to the level of player for which it is targeted. Ideally, the progression is automatic; advanced features are brought in as the computer recognizes increasingly proficient play. More often, players must identify for the computer the level at which they desire to play.

THE ILLUSION OF WINNABILITY

Another important trait of any game is the illusion of winnability. If a game is to provide a continuing challenge to the player, it must also provide a continuing motivation to play. It must appear to all players, from beginner to expert, that the game can be won. Yet it must never be truly won or it will lose its appeal. This illusion is difficult to maintain. Some games maintain it for the expert but never achieve it for the beginner; these intimidate all but the most determined players. TEMPEST, for example, intimidates many because it appears to be unwinnable. The most successful game in this respect is PAC-MAN, which appears winnable to most players yet never quite is.

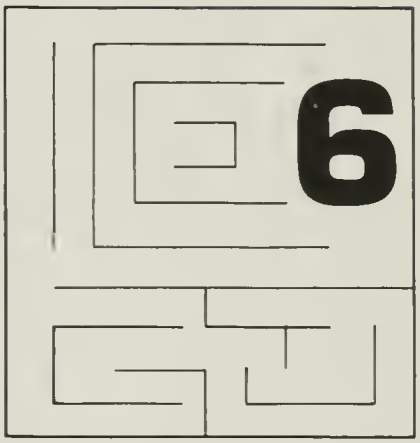
The most important factor in creating the illusion of winnability is the cleanliness of the game. A dirty game intimidates beginners with an excess of detail. The beginner never overcomes the inhibiting suspicion that somewhere in the game lurks a "gotcha." By contrast, a clean game encourages all players to experiment.

Successfully maintaining the illusion of winnability requires careful analysis of the source of player failure. In every game the player is expected to fail often. What trips her up? If the player believes that failure arises from some flaw in the game or its controls, she becomes frustrated and angry with an unfair and unwinnable situation. If the player believes that her failure arises from her own limitations but judges that the game requires superhuman performance, the player again rejects the game as unfair. But if the player can attribute her failures to her own correctable errors, she will play on in an effort to master the game. When the player fails, she should slap herself gently and say, "That was a silly mistake!"

SUMMARY

In this chapter I have discussed design methods and ideals I have used in developing several games. Methods and ideals together constitute the elusive element we call "technique." Technique is part of an artist's signature. When we listen to Beethoven's majestic Fifth Symphony or the rapturous Sixth or the ecstatic Ninth, we recognize in all the identifying stamp of Beethoven's masterful technique. If you would be a computer game designer, you must establish and develop your own technique.

DEVELOPMENT OF EXCALIBUR



In Chapter 4, I presented an idealized game design sequence. I attempted to describe a general method that included concepts developed in this book. It is, however, a sad truth that the practicality of the schemes we devise is too often inversely proportional to the idealism we embody. I have never designed a game in complete accordance with the system described in Chapter 4. My actual designs have followed rockier courses. In this chapter, I will describe the development of EXCALIBUR, a recent design. The contrast between the real process, uneven and error-laden, and the ideal process should help the reader bridge the gap between theory and practice.

BEGINNINGS

In December of 1981, I began working for Alan Kay in his new Corporate Research unit at Atari. Given total creative freedom, I resolved to devise a game worthy of the faith Dr. Kay had invested in me. I wanted this game to be grand and glorious, so lofty in its goals and challenging in its play that it would put all others to shame. Since marketing considerations were not important for the game, I resolved that it would run in a 48K disk-based environment. This afforded plenty of computer resources with which to work.

My background is in war games, and so I naturally thought in terms of a war game. War is the most extreme expression of human conflict, the greatest human evil, and the highest tragedy of our species. It

therefore seems to be an obvious starting point for a serious artist. I wanted to break away from the conventional treatment of war in games, which either glorified war as an expression of misconceived heroism, or trivialized it as a fascinating intellectual exercise. I wanted a game that placed war in a meaningful context. It would include war as an option that must sometimes be exercised, but not frivolously. I wanted a game that warmongers would inevitably lose, because I believe that peaceful strategies are often the most practical ones. This game would address statecraft and focus on leadership. Another fundamental goal was that the game would actually consist of a number of games linked together. This would allow me to show policy, statecraft, and war on several levels, from the most strategic and indirect to the most tactical and direct.

My next task was to determine the fantasy context for the game. I narrowed the possibilities to two: a game about the USA after a major nuclear war and a game about Britain in the Dark Ages after the collapse of Roman authority. Both contexts reflect societies attempting to reorganize themselves after a calamity. The first fantasy was too gruesome for my purposes. Furthermore, the second fantasy context was shrouded in the fascinating legends of King Arthur. I chose the Arthurian context.

The player in this game would be King Arthur, and his goal would be to unify Britain and bring peace to the troubled land. The challenge of the game would arise from the unwillingness of the other kings to submit to Arthur's sovereignty. The player would be required to use various techniques to establish his authority, only one of which would be military action. Indeed, I resolved that overuse of military methods would brutalize the nation and result in endless insurrections and anarchy. With these noble goals established, I began serious design work on the game.

Early Work (January—April 1982)

I first turned to the question, what is leadership? The answer would be central to the game. It was necessary to me to determine the essence of leadership on a national level then translate it into a form managable in a game. I needed to extract the central decisions of leadership and design a form for expressing them. The military aspects of leadership were the most obvious and easiest to work with. I would have had no difficulty designing a game in which the player must make correct military decisions. But this was not satisfactory to me. I wanted to address

wider issues: the social, diplomatic, and interpersonal aspects of leadership. How was I to represent and manipulate these factors in the course of the game? The problem vexed me for months.

I quickly grew impatient struggling with such fundamental problems. The child in me wanted immediate gratification, and so, to satisfy these impulses, I wrote the title and ending scenes for the game. These were not crucial to the structure of the game, but they gave me an opportunity to explore some interesting graphics techniques without compromising the integrity of my design. The ending scene posed some interesting problems: The sword Excalibur twirls through the air above a lake, falls into a hand that rises abruptly out of the water to catch it, then recedes beneath the waves. I spent much time trying to add the lonely sound of wind whistling against the blade of the sword, but I was never able to obtain satisfactory results. I therefore decided to try accompanying the title and ending scenes with some appropriate music. I chose as my two prime candidates a passage from Siegfried's death and funeral in Wagner's *Siegfried* and a portion of Dvorak's Seventh Symphony.

At this time I also determined the fundamental structure of the game. There were to be four nested games. The first, CAMELOT, would concern Arthur's activities within his castle. These would include management of his own kingdom, conduct of diplomacy, and preparation of the army. The second game module, BRITAIN, would allow Arthur to travel around the island of Great Britain with his army and engage in strategic military activity. The third game module, BATTLE, would allow Arthur to engage enemy armies in direct conflict. If Arthur himself managed to encounter an enemy king on the battlefield, he would then enter the fourth module, JOUST. This last module was intended to be a simple skill-and-action game in which Arthur would attempt to unhorse his opponent. The game would use a full first-person view of an advancing horseman, lance leveled, the whole scene bouncing with the galloping of Arthur's own horse. I entertained myself by devising clever graphics algorithms that would generate true three-dimensional first-person graphics. After I had expended a great deal of effort, though, I realized the the JOUST game would take only a few seconds to play and would not provide much challenge.

I started over with a new idea: a swordfight game. The first problem I faced was, how can I simulate the motion of a sword using joystick commands? I found a yardstick and spent hours in my living room

swinging it, trying to discover some pattern that could be represented cleanly with a joystick. My difficulties arose from the fact that the motion of a sword in a swordfight is very complex, and a joystick simply cannot express all the intricacies of the motion. I eventually devised a reasonable system. The side-to-side motion of the joystick controlled the angle of attack of the sword, from a horizontal swing from the left, through a vertical swing over the player's head, to a horizontal swing from the right. Moving the joystick backward swung the sword back in preparation for the stroke; forward motion of the joystick caused the sword to strike.

This problem solved, I began work on some new graphics routines to show an opposing swordsman in first-person graphics. This proved to be a very difficult task. I eventually gave up on the swordfight game for many of the same reasons that had led me to abandon the joust game. Besides, I didn't want Arthur to be able to hack his way to victory. If swordfights cannot assure success, what's the point of having them in the game?

By now it was March. I began work on the BRITAIN module. It consisted of a scrolling map with a number of embellishments. I had earlier designed scrolling maps in EASTERN FRONT 1941 and LEGIONNAIRE, so implementing this module was easy for me. Because I had lots more memory for this game, I decided to make a gigantic scrolling map. I finished with a large 6K map of Great Britain.

Slowly the game design was taking shape in my mind, but a fundamental question remained unanswered: was this to be a historical game or a fictional game? That is, was this a game about Britain in the sixth century A.D. or was this a game about King Arthur? I read every book I could find about both subjects. This research led me to conclude that Britain in the sixth century was a chaotic and depressing place. The native Celts were defending their homeland against the invading Anglo-Saxons landing on the eastern coast of the island. For two centuries the Anglo-Saxons slowly pushed the Celts westward. King Arthur was actually a Celtic general who led a brief counteroffensive against the Anglo-Saxons, winning the battle of Mount Badon and halting the Anglo-Saxon offensive for about 50 years. But Arthur's success brought only a brief respite; in the end, the Celts lost their struggle. Thus, the historical record did not provide what I needed; a society struggling to reorganize itself. Instead, the story of Britain in the Dark Ages was that of one people being driven out relentlessly by another.

Yet from the dreams of the vanquished arose the legends of the conquering King Arthur, legends that endured through the ages, transforming themselves to suit the needs of the storyteller. As I read the many incarnations of these legends, I was impressed by their great flexibility. Each artist who recited them impressed a different character upon them. They worked equally well as religious inspiration, ribald tales, or expositions of the chivalric ideal. Even Mark Twain employed them to make his characteristic blistering social commentary.

A major turning point in the design process came when I saw the movie EXCALIBUR. This magnificent film beautifully captures the best elements of the Arthurian legends, yet makes its own statement. I watched it again and again, reveling in the richness of the tale. This movie shamed me. I realized I had been compromising the important artistic issues in my game in order to indulge myself with cute graphics. I rededicated myself to the high artistic goals I had set earlier. I also knew that I could not realize them alone; I needed help. I enlisted the aid of Larry Summers and hired Valerie Atkinson to help me. With new determination, we set to work.

The Long Haul (May—December 1982)

Here is where we stood in May 1982: I had established the broad design but had left many details unfinished. Several disparate chunks of code had been written but did not fit together at all. There was no overall design document. Faced with so many things to do, I foolishly opted to finish some of the easier minor elements. I wrote the CALIG module that draws Gothic characters on the screen. Valerie set to work preparing the bit map tables for the routine. Larry worked on finishing the title scene by adding music and dissolve routines. These projects, never intended to be more than flashy window-dressing, unfortunately consumed nearly two months.

In June we began work on the CAMELOT module, with Valerie taking primary programming responsibility. This module was actually a set of illustrated menus. Each room (menu) presented four options described by a single-word entry. A vertical band allowed the player to move his crown-cursor to the menu selection. To the right of the vertical band we placed a graphics window for displaying some critical bit of information. For example, in the Round Table Room, we showed a circle depicting the table itself and a set of shields representing the Knights of

the Round Table. Their positions in the room indicated their social relationships. In the Treasury Room we had intended to show piles of coins, but were forced to delete that feature later to show more detailed economic data. We had also intended to use a kernelled display that would have allowed more color on the screen but gave up on that idea, for it would have consumed too much execution time.

As Valerie set to work, I began on the social game associated with the Round Table. I plunged into the task without realizing the magnitude of my tasks. I wanted to produce a small game that would require Arthur to manage a social group. I quickly realized that the most interesting features of the situation were not the radial relationships (relationships between Arthur and the other knights) but the circumferential relationships among the knights. Although Arthur would be forced to deal with the knights radially, relationships among the knights could be the deciding factors.

I found this system fascinating and worked intensively with it. I developed a set of algorithms that modeled group behavior interestingly. I was so pleased with the algorithms that I threw together a short BASIC program to develop a stand-alone game. This game seemed very promising; particularly impressive was my wife's reaction. A woman who takes a dim view of silly games, she took an instant liking to this one. Surprised and gratified that I had finally produced something she could enjoy, I decided to pursue the new game, originally a study for EXCALIBUR, as a completely new project. Aric Wilmunder was hired to execute the design, called GOSSIP.

In July, we entered a long and frustrating period of slow progress. I began devoting more of my time to writing this book. Other duties distracted me further. Larry and Valerie continued their work, making the best of a weak situation. For months they slowly built on the system we had created, fleshing out the skeleton I had briefly described. In the process, many inconsistencies and deficiencies appeared because I had so little time to devote to the project, I did a lot of spontaneous designing. In our regular weekly meetings, my partners would present me with the latest design flaw they had uncovered. Having no clear memories of previous decisions, I would patch together an ad hoc solution. My intuitions were fairly good and many times these deplorable techniques worked. However, many of my on-the-fly patches wrought havoc with the overall design. Poor Valerie installed features in the CAMELOT module only to have them stripped out then later reinstalled.

Our records of this period indicate much wasted effort. We had intended that the Treasury Room in Camelot would be illustrated with piles of coins indicating quantities of wealth. We expended many hours designing coin-drawing routines. In the end, we realized that we didn't have enough screen space to show these piles of coins and needed to show more detailed economic data, so we had to use simple numbers drawn onto the screen. Indeed, the list of things we designed, programmed, and later dropped is revealing evidence of my own failure to plan ahead. The list includes declarations of war (dropped, but later reincarnated as "Attack"), alliances, sieges, demands for tribute, armies moving across Britain, and a host of minor patches.

Six months were consumed in this muddle. They were not a total loss, however; indeed, much progress was made. Larry completed the economics processing, the BRITAIN module, disk swapping modules, a routine for presentation of diplomatic news, and a number of major consolidations of the ever-burgeoning code. Valerie expanded the CAMELOT module, linking it to all the new features and making it the largest and most complex module in the entire game. Yet all of this could have been completed in half the time had I been more organized and devoted more energy to the project. By Christmas everybody was tired, demoralized, and despairing that the project would ever be completed. Those were dark days indeed.

Renewed Effort (January—April 1983)

In January 1983, EXCALIBUR returned to its rightful place as my highest priority project. I plunged into it with a cold determination to finish this game and get it out the door. Gone were the grand inspirations of 1982, the misty-eyed vision of a truly grandiose game. In their place was a bitter resolve. I met long and frequently with Larry and Valerie. Ruthlessly I slashed at the design, ripping out vaguely-defined or nonessential sections. The discipline I had hoped to escape by using vast computer resources was forced upon me by my inability to complete the project.

At home, I worked on the artificial intelligence routines for the knights in the Round Table Room. This required a few weeks. Then I tackled the BATTLE module. During February and March, I wrote, debugged, and playtested it. I was possessed, driven to complete the game by a self-imposed deadline of April 1. My records indicate that I avaraged 300 bytes of debugged code per day; industry averages are

75-100 bytes per day. Larry and Valerie were caught up in the frenzy. They worked furiously integrating all the pieces of the program and resolving the myriad inconsistencies produced. Entire modules handling Merlin's room, economics, vassalage, tithes, and swapping code were designed, coded, and debugged.

Despite this, we failed to make our April 1 deadline and extended it to April 15. Even this new deadline became impossible. Nevertheless, we made April 15 an important milestone—all coding would be completed by this date.

The first two weeks of April were consumed in a wild orgy of effort. Meeting every day, sometimes for four hours at a stretch, we hammered out what was undoubtedly the toughest part of the entire design: the artificial intelligence algorithms.

I had reserved this task for last, for the AI routines would reflect every aspect of the design. The design must therefore have been complete, and all variables precisely defined, before AI algorithms could be designed. Moreover, the creation of the AI routines would freeze the design, since significant design changes after AI was done could ruin the entire AI system.

The AI for EXCALIBUR was the most difficult I had ever attempted. It needed to include the personalities of the various kings as well as economic, military, and geometric factors. The system we developed uses intermediate variables to express concepts such as the amount of military authority a king has, his prestige as an economic manager, and his popularity. Personality traits factored into the algorithm include ambition, stupidity, and defensiveness.

Final Work (May—June 1983)

We almost met our goal of having all code written by April 15. The remaining code was quite trivial. We all took a break for two weeks. In May we began the final work on EXCALIBUR. Larry and Valerie began searching for and eliminating all the bugs in the program.

We managed to clear out most of these bugs by June 1. We then went into another period of intensive effort, playtesting the game many times, exposing countless additional tiny bugs. Each one was relentlessly tracked down, cornered, and eliminated. We met three times a week to discuss the bugs we had found and the progress we were making. We also made numerous adjustments in the design to improve the overall play.

During June, I wrote the short novel that accompanies the game. This novella was not part of the original design. I only thought of it in April; realizing that the manual would be very large, I decided that I might as well go all out and write a novel to explain the rules. My original intention was that the novel would explain the entire game and that no other documents would accompany it. It took me one day to prepare an outline that addressed all aspects of the game. I then wrote a chapter a day, two days a week. It was a most enjoyable writing task.

Our goal was to deliver a working program to APX by their deadline of June 29. We knew that the game we delivered would not be the final version, but it had to be good enough to ship. We made that deadline by the skin of our teeth. I had to tear the program out of Valerie's hands. She couldn't stop making "just one last change."

During July, we finished the program. First we playtested the game several times, making notes. We then prepared a list of all changes we wanted to make. The list came to 25 items, most of them minor, though a few entailed considerable work. We even had a few design changes, although they were minor, in the list. We swore to one another that this list was the final one, that no new entries would be added.

The playtest reports started coming in. We realized that the playtesters were having some problems understanding the game. I had to add a great deal of crutch documentation to help people who were unwilling to draw inferences from the novella. And I had to endure a certain amount of exasperating negotiation with the publisher, who (as always) was less interested in innovation than sales and, as always, failed to see the connection between the two.

Our list of changes grew to 36 items. We dealt with most of them and started a new list, which eventually grew to 16 items. I wrote two more drafts of the novella, expanding it considerably. We played the game countless times. Larry became expert at thinking up absurdly extreme tests for the game, just to make sure that it would not break down under unforeseen playing circumstances. He verified that the player could perform the most idiotic actions without upsetting the game.

Our final delivery date was Friday, July 29. On Thursday, July 28, we met and resolved the final issues. One last bug remained. Larry fixed it and we all spent Thursday night playtesting it individually. We each found a tiny bug in the economics routines. Larry came in to fix it at 6:30 Friday morning. After fixing it, he decided to print a copy of the program. The printer ran out of paper just before the last page. When

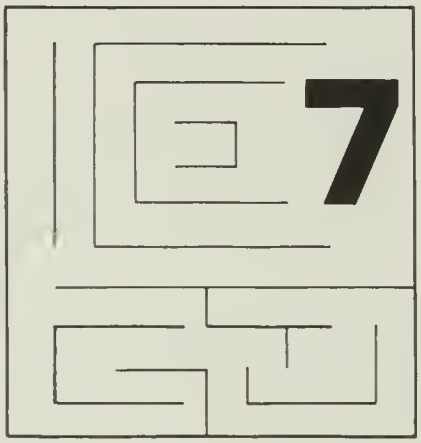
it ran out, the printer told the computer to stop and wait. The computer waited for a while and gave up, aborting the entire process just before completion. It had written the final version of the game onto a floppy diskette, but dropped the last few bytes because of the printer failure.

Unaware of all this, Larry collected his printout and diskette with the fatally missing final bytes. When the rest of us arrived, Larry presented the final version. My sense of caution dictated that we playtest this version before delivering it. The program crashed—it would not run. We smiled nervously: It had to be just a bad diskette. We recopied the game onto a fresh diskette; that crashed. Perhaps it was the disk drive. We used another drive. Still the program crashed. We looked at each other in dismay. Final delivery was four hours away. The program would not even run. And we hadn't the faintest idea why.

With false confidence we set to work analyzing the changes Larry made. Several ideas came to mind. With each one, we laughed and slapped ourselves. "Of course, that's it!" We made the change, but the bug remained. The smiles of calm and confidence were giving away to quiet desperation.

One o'clock came and we were no closer. Our desperation was turning to outright panic. We pulled out the logic analyzer and traced the program flow. We detected the missing section of the program and incorrectly guessed that some other portion of the program was destroying this section. We wasted half an hour looking for the offending overwriter. Then, around two o'clock, the answer came to Larry. He started giggling. The tension was too much. I started giggling, without knowing why. Larry tried to explain the sequence of events leading up to the bug. His explanation took five minutes; his giggling escalated to laughter and finally, guffaws. By the time I understood this improbable, ridiculous mistake, we were both staggering about the lab, crashing into equipment, laughing helplessly.

THE FUTURE OF COMPUTER GAMES



In this book, I have explored computer games from a number of perspectives. I have stated my claim that computer games constitute an untapped art form. Implicit in this claim is the hope that this art form will someday be developed to its potential. Unfortunately, history confirms the fears of cynics more often than the hopes of dreamers. Therefore, I must separate hopes from predictions. Where are computer games going? How will they change in the years to come? Will we see them emerge as a significant art form? A number of divergent trends are now apparent, but analysis of these trends is complicated by conflicting interpretations of the current state of computer game design. I shall begin by addressing the most commonly cited interpretations and proceed to the framework I prefer.

FAD OR FIXTURE?

The first and most important question concerns the very survival of the computer game industry. One school of thought maintains that computer games are merely a fad, a temporary infatuation that will quickly pass when their novelty wears off. Proponents of this view compare the computer game to other fads that swept society with equal force. Eventually, these people say, computer games will go the way of the hula hoop.

This opinion is breezily rejected by all members of the industry, but I fear that such confidence is little more than the Titanic syndrome—

confidence that arises from magnitude. They tend to extrapolate blindly into the future the astounding growth rates we have experienced in the past. It is certainly hard to believe doomsayers when the curve of growth slopes upward so steeply. Few industry optimists, however, can provide justification for their predictions. Just because industry sales doubled in 1982, it does not mean they will double in 1983 or 1984. Indeed, it cannot continue to double annually much longer; if it did, Atari alone would need only eleven years to engulf the entire Gross National Product like some monstrous Pac-Man.

Furthermore, size generates negative forces that will certainly reduce the growth rate. In the simple days of the seventies, when computer games numbered in the thousands rather than the millions, nobody much cared about their social effects because computer games were a minor component of our society. But now they are everywhere. They are such a powerful force that we now have a backlash developing against computer games, with ordinances being enacted against arcades all over the country. Parents are beginning to restrict their children's access to the games. Editorials warn against the dire effects of playing the games. Already several preliminary studies have been undertaken to determine the effects of computer games on children. So far, the still speculative conclusions have been mildly favorable, but the day will certainly come when the numbers game we call research will produce a blockbuster report demonstrating that computer games cause cancer in laboratory rats.

Bigger critters than Atari have bit the dust; bigger industries than ours have shrivelled and died. Size and past success are no guarantee of permanence. We need substantive reasons for confidence in the future rather than simple extrapolations from past history. I am convinced that such reasons for optimism exist; my full argument will come later in this chapter. For now let me say that computer games satisfy a fundamental desire for active recreation and so are assured of a bright future.

THE TECHNOLOGICAL EXTRAPOLATION

The most commonly cited future for computer games is projected in terms of technological extrapolation. Advocates of this school point to the steady march of technology and the rapid improvements that we

have seen in computer hardware. They then extrapolate these trends directly to project a future populated by supercomputers with fabulous games full of unbelievable graphics and sensational experiences. These people emphasize technological factors as the primary agents of change. They claim that the major breakthroughs will come with bigger and faster processors, megabytes of RAM, new languages, and better display hardware. Holography, trackballs, laserdisks, body sensors—these are the coin of the realm among the technological extrapolators.

I cast a cold eye on such predictions. This is the same line of thought that in the late sixties predicted ever-larger, ever-faster mainframes as the primary path of development in the computer industry for the seventies. Computers did indeed become larger in that decade, but the development of larger mainframes was not the dominant event of the seventies; the maturation of minicomputers and the genesis of microcomputers were. The extrapolators never foresaw the coming of microcomputers, because micros didn't fit into their "bigger and better" extrapolations.

I do not deny that technology will improve. It will. The real issue is whether or not technological limitations are the primary constraints on the game designer. I do not deny that these limitations do impose severe constraints on all computer games, and I readily acknowledge that technological advances will remove many of these constraints. Certainly technological immaturity—the weakness of current 8-bit, 64K, 1 mhz systems—is a crippling limitation. Yet I maintain that artistic immaturity is an even more crippling one.

Consider two extreme hypothetical futures. The first has no technological development and the second no artistic development. In the first world, I am stuck with an Atari 800 as my sole medium for game design. This does not worry me too much; I could explore the possibilities of this machine for five or ten years before beginning to feel trapped. The second world, though, is a bleak place indeed. I am doomed to write ever fancier variations on STAR RAIDERS and BREAKOUT, with more colorful explosions, snazzier sounds, and three-dimensional photon torpedos, but never anything new or different. I would feel trapped immediately.

Probably, neither of these futures will come to pass; we will have both technological development and artistic development. Yet we must remember that technological development, while entirely desirable, will never be the driving force, the engine of change for computer games.

Artistic maturation will be the dynamo that powers the computer game industry.

The relative values of technological development and artistic maturity are made clear by a comparison of modern movies with silent movies. Modern cinema boasts enormous technological advantages—sound, color, and fabulous special effects. When used with skill and artistry, the new technologies are indeed magnificent, but all these advantages cannot make up for a lack of artistic quality. The computer-graphics blockbuster TRON compares poorly with any of Charlie Chaplin's movies. If Chaplin could do so much with black and white film and no sound, why cannot we do good work with 8 bits and 48K?

MY ASSESSMENT: TECHNOLOGICAL REVOLUTION

To explain my own assessment of the situation, I must describe how I view technological revolutions. The first technological revolution I will discuss is the transportation revolution that swept American society in the first half of the twentieth century. The automobile was invented in the late 1800s; by the turn of the century it was available as a consumer product. Many problems plagued the automobile, however. It was expensive and unreliable. It lacked support services such as service stations and appropriate roads to make it truly practical. It required considerable skill and dedication to operate. Furthermore, it was unnecessary. American culture had developed quite successfully without it, so there was little need for it. Thus, the automobile was not a practical tool but a plaything of the wealthy.

With the passage of time, however, these problems lessened. Mass production lowered cost and increased reliability. More service stations and better roads became available. More automobiles were purchased. By the late twenties, the automobile was a common fixture in American life.

A third stage became obvious in the 1950s. The automobile changed the face of American society. Commuting became practical. Housing patterns began to change. Drive-in restaurants and theaters became common. Urban sprawl ensued. The technology changed the society.

As the automobile changed American society, however, so too did society change the automobile. Originally designed as a device to transport people and property quickly, safely, and reliably, the automobile

became a vehicle for self-expression, a recreational device, and ultimately, an end in itself. Could Henry Ford have anticipated dune buggies, vans with waterbeds, low-riders, and naked-lady hood ornaments? I doubt it.

Let me summarize the stages that occurred in this transportation revolution. First, the technology was initially desirable to only a small part of the public. With time, production and support conditions improved and the technology conquered society. Then it began to change society. In the process, society began to change the technology. The direction of this change was away from pragmatic needs and toward recreational ones.

Let us now examine the second great revolution of this century, the entertainment revolution sparked by television. When television first became available in the late 1940s, it was expensive, unreliable, and lacked sufficient software (programs) to make it anything more than a toy for the wealthy. With time, these deficiencies were overcome. Television became cheaper, more reliable, offered more programming, and swept society with great force. In the process, television dramatically changed the lifestyle of the American people. Nighttime entertainment was now readily available. Leisure time activities changed accordingly. But the public worked its will on television. The medium evolved from "visible radio"—a means of presenting lectures, plays, and speeches—into a medium with its own personality. Thus, the same four stages outlined for the automobile occurred with television: pioneer, conquest, transformation of society by the technology, and transformation of the technology by society.

The same sequence of stages is occurring with computers. At the moment, personal computers are still expensive, unreliable, hard to use, and lack software. But the situation is changing rapidly: prices are falling, machines are becoming friendlier, and software availability improves daily. All observers agree that personal computers will take society by storm. The only differences of opinion concern magnitude. Will 1990 see 5 million computers in American homes or 10 million or 20 million? No one knows but everyone agrees that the figure will be large.

We can therefore expect that personal computers will change the face of American society. We can expect that networking will allow more Americans to participate in economic activities from the home, decreasing transportation problems and accelerating the pace of eco-

nomic life. Ease of manipulating information will make information more important in our society. Our financial system will become less dependent upon currency. Our lives will be changed by these machines.

But we ourselves will not be changed. The computer will change our habits and our leisure time, but it will not change our personalities. We are still the same people who built the pyramids, fought the Crusades, and colonized the New World. Our analysis of the two revolutions leads us to expect that the relationship between society and the computer will be one of reciprocal transformation. We further expect that the nature of this transformation will be a shift from the pragmatic toward the recreational, from the functional to the frivolous. This leads us to suspect that games may be the primary vehicle for society to work its will on computers.

Ten years ago, even five years ago, this suggestion would have seemed ridiculous. Computers were the awesome creatures of humanity's cleverness, the intelligent progeny of the machine age. Computers were perceived to be powerful, endlessly capable, and more than a little fearsome. Most people's only concern was whether computers would be humanity's slave or its master. The possibility that they might be a playmate never crossed anyone's mind.

We were wrong. The computer game has already established itself as a primary computer application. Consider, for example, the number of existing computer games. The most reproduced computer program in history, the one with more copies in existence than any other program is undoubtedly COMBAT, the game cartridge supplied with every ATARI 2600. Millions of copies of this cartridge have been distributed. Perhaps this measure is unfair because nobody buys the program by itself. Then consider PAC-MAN, ASTEROIDS, SPACE INVADERS, and MISSILE COMMAND, each of which has sold millions of copies. Indeed, were we to compile a "Top Forty" list of the best-selling programs of all time, I very much doubt that any serious piece of software would make the list. Games dominate.

Perhaps numbers alone do not adequately measure social significance. Let's try an economic comparison. Visicalc, the most successful serious package for personal computers, has sold approximately 400,000 copies for $200 each. That amounts to $80 million gross. By contrast, if Atari sells 5 million copies of PAC-MAN at $30 each, that's $150 million. And that's just one title; there are many other games generating large sales figures.

Thus games already constitute a primary use of computer technology. They have established themselves in the world of computers. In the accelerated eighties, the fourth stage (transformation of technology by society) is already upon us even as the second phase (conquest) is beginning.

THE NATURE OF CHANGE

Games are the vehicle by which society will change the computer. How will the games themselves be changed by society? We can expect three factors to affect games: the mass market, the flowering of heterogeneity, and the evolution of taste. In some ways, these processes work against one another.

The Mass Market

As computer games become a mass market commodity, they will fall prey to the homogenizing forces of the market. Emphasis will not be on originality or creativity, but on adhering to time-honored formulas. Just as movies and television fell prey to the formulas of sex and violence, cops and robbers, sitcoms, and the other hackneyed creations, so too will games be victimized by the tyranny of the mass market. We will see the same game over and over in new clothing. My guess is that we are already caught in the grip of this force, producing little more than variations on a single theme: "Blast the monsters!" This has sold well, so we stick with it.

This cynical view of the market must be qualified. The mass market is occasionally capable of sustaining quality. Hollywood may grind out an army of soulless clones, but now and then something interesting is produced. When this happens, the mass market often responds favorably. 2001—A SPACE ODYSSEY, STAR WARS, and RAIDERS OF THE LOST ARK are examples of original, creative ideas produced successfully for the mass market. Just because an idea works in the mass market, it is not necessarily junk.

The Flowering of Heterogeneity

The games market differs from the movie and television markets in that it is less centralized and has fewer economies of scale. In this respect, it is closer to the book and record markets. For this reason, I

expect the games market to exhibit more heterogeneity and less obeisance to mass taste.

I therefore expect a host of "baby markets" in the wake of the mass market. While baby markets will never be as lucrative as the mass market, they perform two valuable services. First, they provide a testing ground for new ideas that, if successful, will be devoured by the voracious mass market. Second, baby markets will always provide a haven for the refugees from mediocrity and a playground for those whose tastes transcend the average.

Why have baby markets not developed very much? I answer with a story. Suppose that you were the first astronaut to land on a newly discovered planet and that you found a civilization there in every way the equal of ours, but for a single exception: they had no literature. No novels, no poetry, no children's books, no textbooks, no magazines. However, they did have comic books. On further study, you discovered the reason for this oddity. Reading was a new discovery only recently popularized by teenagers and shunned by the majority of adults who felt intimidated by this skill. Thus, literature was used by teenagers to explore their own fantasies and interests: confronting authority, violent resolution of conflict, and so forth. Hence comic books. Could you recognize the seeds of the future in this situation? Would you not expect the flowering of other forms of literature as the kids grow and develop new interests? Would the potential for novels, short stories, poetry, and other genres not be inherent in the situation you found?

So it is with computer games. Until now the province of teenage males, these games are bursting into all of society. To this point they have catered to the fantasies of twisted computer "nerds," but they will soon blossom into much richer explorations of fantasies. We will have country-western games, gothic romance games, soap opera games, comedy games, X-rated games, accountant games, and snob games. The society that invented the hot tub, CB radio, and the dune buggy will have no reservations about impressing its character on computer games.

Eventually, games will be recognized as a serious art form, although the exploration of games as an art form will be restricted to a small fraction of the total design activity. Most of the effort will always be devoted to serving popular taste. Yet this tiny group of games artists will be responsible for creating the future classics of games, the games that will endure.

The Evolution of Taste

Heterogeneity in the computer game market will not flower abruptly. It will evolve as the tastes of the audience evolve. Taste, whether in food, music, or computer games, takes time to mature. The novice prefers the most direct, intense experience available. Thus children and adolescents enjoy candy, junk food, and rock music. But human beings are a fickle lot. The simple intensity of our youthful tastes ensures that we will someday become bored with them. We will seek foods that have more subtle and interesting tastes and music that expresses a wider range of emotions.

A critical element in the evolution of taste is the development of a lexicon. Every communications medium requires a lexicon, a vocabulary of associations that allows it to communicate meaningfully. The richer the vocabulary, the greater the potential for expressiveness. Music requires an array of sounds that audiences can experience as meaningful. Even cinema, blessed with the capacity for directly displaying human gesture and facial expression, has evolved its own special visual lexicon that expands its communicative power.

Computer games must develop a lexicon before they can support the inevitable flowering of heterogeneity. This lexicon will be unlike any other for it must be interactive. It will not be enough to create images that convey feelings. Somehow we must allow the expression of human emotion from computer to human and vice versa. The artistic challenge is vast, yet we have managed to charge stone, pigment, taut string, and celluloid with feeling. We will succeed with silicon.

CONCLUSIONS

I see a future in which computer games are a major recreational activity. I see a mass market of computer games not very different from what we now have, complete with blockbuster games, spinoff games, remake games, and tired complaints that computer games constitute a vast wasteland. I even have a term for such games—cyberschlock. I also see a much more exciting literature of computer games exploring almost all spheres of human fantasy. Collectively, these baby market games will probably be more important as a social force than the homogenized clones of the mass market, though individual games in this category will never enjoy the economic success of the mass market games.

By 1985, software stores will be as common as record stores; by 1990, they will be as common as bookstores. Entering the software store, you will be confronted by racks and racks of games with serious software occupying a smaller portion of the floorspace. Just as in a bookstore or record store, you will see aisles devoted to particular tastes in games. You will perhaps browse through collections of cowboy games as your companion explores space games. Perhaps you will look for the latest product of your favorite author, all of whose works are collected in alphabetical order. On the walls you will see posters announcing new smash hit games by software superstars. After evaluating a number of games, you will make your choices and purchase them. Then you'll go out to the parking lot and discover that some idiot has dented the fender of your car. Some things never change.

INDEX

Other Osborne/McGraw-Hill Publications

An Introduction to Microcomputers: Volume 0—The Beginner's Book, 3rd Edition
An Introduction to Microcomputers: Volume 1—Basic Concepts, 2nd Edition
Osborne 4 & 8-Bit Microprocessor Handbook
Osborne 16-Bit Microprocessor Handbook
8080A/8085 Assembly Language Programming
6800 Assembly Language Programming
Z80® Assembly Language Programming
6502 Assembly Language Programming
Z8000® Assembly Language Programming
6809 Assembly Language Programming
68000 Assembly Language Programming
Running Wild—The Next Industrial Revolution
The 8086 Book
PET® Personal Computer Guide
CBM™ Professional Computer Guide
Osborne CP/M® User Guide, 2nd Edition
Apple II® User's Guide, 2nd Edition
Some Common BASIC Programs
Some Common Pascal Programs
Practical BASIC Programs
Practical Pascal Programs
Science and Engineering Programs—Apple II® Edition
A User Guide to the UNIX™ System
PET® Fun and Games
Commodore 64® Fun and Games
Assembly Language Programming for the Apple II®
VisiCalc®: Home and Office Companion
Discover FORTH
Your ATARI™ Computer
Wordstar® Made Easy, 2nd Edition
Armchair BASIC
Data Base Management Systems
The HHC™ User Guide
VIC 20™ User Guide
6502 Assembly Language Subroutines
Z80® Assembly Language Subroutines
8080/8085 Assembly Language Subroutines
The VisiCalc® Program Made Easy
Your IBM® PC: A Guide to the IBM® Personal Computers
Your Commodore 64™
The Programmer's CP/M® Handbook
The Home Computer Software Guide
Your IBM® PC Made Easy
Graphics Primer for the IBM® PC
The MBASIC® Handbook
Advanced Pascal Programming Techniques
54 VisiCalc® Programs
54 SuperCalc® Programs
Using dBASE II®

THE ART OF COMPUTER GAME DESIGN

REFLECTIONS OF A MASTER GAME DESIGNER

Chris Crawford, mastermind of some of Atari, Inc.'s most fascinating games, including Eastern Front™ and Excalibur,™ shares his insights into the creation of computer games.

Viewing the computer game as a promising new art form, unique in its capability to interact with its audience, Crawford, the manager of research for Atari, Inc., offers guidelines for creative game development.

The Art of Computer Game Design emphasizes the artistic dimension of computer games, revealing computer game design as a creative process rather than a merely technical one.

The Art of Computer Game Design is informative reading for any computer game enthusiast and an essential book for all those involved in computer game design.

Eastern Front and Excalibur are trademarks of Atari, Inc.

ISBN 0-88134-117-7

McGraw Hill