GRAPHIC INTRODUCTION TO PROGRAMMING

Yehuda E. Kalay

GRAPHIC INTRODUCTION TO PROGRAMMING

Yehuda E. Kalay
School of Architecture and Environmental Design
State University of New York
Buffalo, New York

A Volume in a Series entitled Principles of Computer-Aided Design

A WILEY-INTERSCIENCE PUBLICATION
JOHN WILEY & SONS
New York / Chichester / Brisbane / Toronto / Singapore

Library of Congress Cataloging in Publication Data:

Kalay, Yehuda E.
Graphic introduction to programming.
Bibliography: p.
Includes index.
1. Electronic digital computers—Programming.
2. Computer graphics. I. Title.
QA76.6.K3416 1987 620'.00425'0285 86-26746
ISBN 0-471-85389-5

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

SERIES PREFACE

Computers have made their debut as design tools in many engineering disciplines, providing designers with flexible means to represent design products. In that capacity they have already demonstrated their utility in improving the productivity and the quality of the production end of the design process, much like word processors have improved the production of documents in offices.

A growing number of researchers, developers, and users of computer-aided design (CAD) systems, however, have reached the conclusion that the utility of computers in design is far from what it might become, if their use were extended over the design process as a whole, including the complete, accurate, and efficient representation of the designed artifacts and the processes that are employed in their conception and creation.

This series of five books is intended to help realize the potential of CAD, through the introduction—in an integrative form—of principles, methodologies, and practices that underlie CAD. It is intended to be used by people who wish to engage in the process of research, development, and maintenance of the new generation of CAD systems. Since the current cadre of people involved with R&D in CAD is relatively small, the series assumes that the reader has no particular knowledge of the field and thus it can be used as a text for beginners. The topics it introduces progress, nevertheless, rapidly toward the frontiers of CAD and thus make the series useful as a text for advanced courses and as reference for professionals.

The first volume, *Graphic Introduction to Programming,* introduces the basic concepts of computing that are needed to master the tool—namely, the computer. These concepts include programming, structured problem solving, and interactive computer graphics. Programming is the means through which computers are instructed how to perform the desired tasks. Structured problem solving, through algorithm design and analysis, encourages a disciplined approach to the design process as a whole and to computer-aided design software development in particular. Given the established manner of communicating design between humans, and because of the ease with which graphic and pictorial information can be disseminated compared to text and numerals, computer graphics have become the standard means of interaction with CAD systems. These three topics have been integrated and are covered through learning the programming language Pascal. This language has certain features that qualify it for this task, which include ease of comprehension, enforcing good programming practices, availability on most computers, and provision of dynamic memory management facilities.

The second volume, *Modeling Objects and Environments,* introduces the concepts of modeling real-world phenomena in the computer's memory. It covers a set of methods and techniques for representing the physical environment as symbol structures that are understood and can be manipulated by computers. The book includes the study of data structures that are particularly suitable for representing two- and three-dimensional artifacts, and operating on these structures in a way that preserves their semantic integrity. The book covers the formative and other attributes of objects, such as topology, geometry, transformations, assemblies, and general database query/update operations.

The third volume, *Modeling Design Processes,* is concerned with the representation of design as a problem-solving process. The two essential components of CAD covered include the generation and the evaluation of alternative design solutions, represented through the modeling concepts introduced in the second volume. The study of solution generation covers all design phases, including preconcept, conceptual, and detailed design. It is introduced through techniques such as knowledge representation and search strategies and relies on selected practices from artificial intelligence, database theory, linear and dynamic programming, operations research, and optimization theory. All of these topics are covered in sufficient detail for the comprehension of the central topics in the book, though of course they do not attempt to be a substitute for formal texts in their respective disciplines. The theoretical treatment of the topics covered by the book is complemented by examples written in PROLOG, a programming language that has some powerful features useful for logic representation and manipulation. For the benefit of readers who are not familiar with PROLOG, an appendix introduces the basics of this language.

The fourth volume, *Performance Measurement Applications,* presents an assortment of task-specific applications that draw upon the model of the designed artifact and provide the designer and the knowledge-based CAD system with a variety of evaluative, simulative, and tabulative measures of the artifacts' expected performances. Written by noted experts in their respective disciplines, the applications present both state-of-the-art knowledge in specific fields relating to CAD and an integrative approach to employing computers as design consultants.

The fifth volume, *Computability of Design,* completes the series by reviewing the key topics that were presented, and by raising the fundamental issues of design computation: How can design processes be mathematically modeled? What is design knowledge, and how can it be computed? How can the design process as a whole be computed? Alternatively, how can particular design tasks be computationally assisted? This volume is based on a symposium that was held in Buffalo, New York, in December 1986, where noted experts discussed the feasibility, utility, and desirability of various approaches to the computation of design.

The series as a whole is concerned with the principles of CAD, rather than with an exhaustive survey of techniques, practices, and existing systems. Its goal is to educate students, researchers, and professionals, who will develop the CAD systems of the future and who will maintain them. It is intended to be used by designers in many engineering disciplines, rather than by computer scientists alone. It may, however, be of considerable interest to computer scientists too, by exposing them to the computational concerns of design professionals.

It is recommended that the books be used in sequence, but this is not a prerequisite for their utility. Neither is adherence to the programming languages Pascal and PROLOG, which are used to exemplify the theory. It is the concepts and principles presented in the series that are of primary importance, and they transcend the technologies and techniques used for their implementation.

YEHUDA E. KALAY

Buffalo, New York
February 1987

PREFACE

Computers are rapidly changing the methods and practices used by most engineering disciplines for the design and manufacture of physical artifacts. Consequently, there is a growing need for trained people who can contribute to the development of computer-aided design (CAD) software, as well as a need for people who can use that software effectively. This volume introduces the reader to the principles upon which computer-aided design software is founded.

In general, design can be considered a process of search for an optimal or satisfactory solution to a stated problem. The process requires a means for generating solutions and a way of evaluating solutions according to some criteria. Unlike manual design processes, computer-aided design involves the *representation* of design solutions in the computer's memory. In addition, it involves the issue of the human–machine *interface.*

The purpose of this first volume in the series is to serve as an introduction to the principles of computer-aided design and as a textbook for an introductory course in programming and interactive computer graphics. It attempts to unify in a single volume the fundamentals of structured programming, the syntax of programming in Pascal, and the principles of interactive computer graphics that are applied in displaying and manipulating two-dimensional images. The programming language Pascal has been chosen because it is well suited for computer-aided design and computer graphics, it is easy to learn, and it has a rigorous structure that induces good programming practices and facilitates structured problem solving. Since the major concern of this book is CAD, rather than programming, it addresses only those features of Pascal that have been deemed relevant. It is recommended, therefore, that this text be accompanied by the *Pascal Manual and Report* by Jensen and Wirth (published by Springer-Verlag), which can serve as a reference and provide additional explanations where needed.

Computer graphics, however, are usually considered to require a much higher level of computer literacy than programming, and their study is, therefore, typically deferred until the student has mastered that skill. Hence the study of computer programming is usually divorced from any graphic applications. Programming examples and exercises are based, instead, on numerical and text manipulation problems.

For students and professionals in disciplines such as architecture and engineering, which emphasize graphic communication, deferring graphics in favor of numerical problem solving is undesirable and unnecessary. Most, if not all, of the concepts of computer programming can be illustrated much more effectively and intuitively with graphic examples. The concept of iteration, for instance, can be easily illustrated through the drawing of an approximate circle made of many small straight line segments, and recursion is the hallmark of the famous Sierpiski curve. Furthermore, the satisfaction a student derives from the ability to generate pictures on a CRT screen, practically from day one, may enhance the learning process and improve retention of knowledge. The complexity and the significance of these pictures gradually increase through the book, building up to a simple yet effective drafting system that embodies everything the student has learned through this process. The exercises unify the components of the learning process into one whole.

The introduction to graphics in this book is by no means a substitute for a full text on computer graphics. Its intent is merely to introduce the student to the concepts of image generation and manipulation. The interested reader is advised to consult the Bibliography for other references on this subject.

From the student's point of view, this book is a step-by-step introduction to the design and development of computer tools for interactive drafting. Starting with simple line drawing, the student will gain the skills needed to write software that is capable of highly interactive entry, manipulation, storage, and display of two-dimensional images on a graphic display terminal. The book should be read sequentially, as each chapter builds on the knowledge acquired in the preceding ones.

The book consists of four parts, preceded by an Introduction and followed by the Appendices. Each part addresses a single subject through several chapters. The Introduction considers notions such as problem solving and programming and surveys briefly the history of computer-aided design. The first part deals with the basic concepts of programming, including sequential instruction processing and the important roles played by names and values. The second part introduces control structures, including repetition constructs, decision constructs, and partitioning constructs (procedures and functions). The third part develops the skills for data structuring, through such topics as constants, enumerated (user-defined) types, arrays, records, and lists. The fourth part deals with the manipulation of text—within the program, its communication to the terminal, and to files—and contains a conclusion. Appendices on Pascal keywords and graphic display drivers appear at the end of the book.

Each chapter in the book is organized in three parts:

1. A theoretical problem and its logical solution.

2. Pascal syntax to encode the solution.
3. Graphic application and examples.

To facilitate portability of the graphics package, the display drivers are presented as a set of external procedures, to be loaded together with each program. The drivers may be implemented in various ways; one implementation that will run on most TEKTRONIX-emulating hardware is described in Appendix B.

The programming examples in this book were written in Berkley Pascal Version 1.1, under the Berkley UNIX 4.3 operating system.

Books of this nature are not developed in a vacuum. They require a nurturing and supportive environment, where ideas are generated, discussed, and tested. The environment that supported the development of the methodologies and techniques that are presented in this book has been the Computer-Aided Design and Graphics Laboratory of the School of Architecture and Environmental Design at the State University of New York at Buffalo. I am indebted to my colleagues and students who criticized, tested, and otherwise helped in shaping the methodologies presented here. I am also grateful to the departmental, school, and the university administrations for providing the much needed support and encouragement for undertaking this task.

The production of the book itself owes much to C. Harvey Lichtblau, who edited this volume and improved its readability and consistency; and to L. Donald Steiger for his continuous effort and thoroughness in the production of this difficult manuscript. Many thanks to both.

YEHUDA E. KALAY

Buffalo, New York
February 1987

CONTENTS

INTRODUCTION

People have always striven to modify and adapt the environment they inhabit so it will provide better shelter, food, and recreation. At first people made progress only by accidental discoveries, then by purposeful trial and error. Although progress by trial and error was slow and costly, it led to the development of some sophisticated tools and environments, ranging from clocks to farm machinery and from Gothic cathedrals to planned cities.

The growing complexity of tools and environments required, nevertheless, a different approach to their development. This approach was to enable tool developers to comprehend (a) the multitude of components of an artifact or building working in concert and (b) their expected impact on the environment. The sixteenth century is noted for the invention of engineering and architectural drawings, along with their use as a means to represent and communicate hypothetical adaptations of tools and environments. This separation of the search for a solution from its physical realization allowed not only the development of much more complex artifacts than ever possible before, but also the selective implementation of only those solutions that seemed most suitable, at a considerable reduction in development cost and time. It has also created a new discipline, called *design*. Its practitioners were required to be capable of technical innovation and of predicting the future performance of their creations. Since both innovation and prediction are based on creativity and judgment (two of the most distinguished human traits), designers have attained the status reserved for artists. Yet, this has also been the source of many products and environments that failed to meet their designers' expectation.

The inherent uncertainties associated with creativity and judgment make the development of new artifacts and environments difficult and undermine the education of new design professionals. The attempt to understand how designers work and how to teach design principles to students is the subject of inquiry of a relatively new discipline known as *design theory*. It has also guided the search for tools that will help designers accomplish their tasks more effectively and predictably.

The advent of digital computers now provides design theorists and tool builders with the opportunity to make significant progress on both fronts. On the one hand, it provides us with a machine that can mimic, under certain conditions, human thought processes and thus serve as an extrospective examiner of these processes. On the other hand, it can assume some of the responsibilities of human designers and speed up some others. Yet, even though design theory has made much progress since computers have been introduced as its main tool of inquiry, we are still far from really understanding how designers work.

Nevertheless, some advances have been made. We now have at least a general understanding of the design process, portrayed as a hierarchical, goal-directed problem-solving activity. Furthermore, we are able to simulate environments that do not yet exist physically (which is what the design process is all about) and test their expected behavior. The computer allows us to view potential environments and explore their properties, just as the telescope allows astronomers to explore distant galaxies that are not visible to the unaided eye.

Computer-Aided Design (CAD) is, then, a utilization of the computer's abilities to simulate both the design process itself and the artifacts being designed. The computer can be made to assume different roles in the design process and thereby either relieve the human designer from certain responsibilities or aid the designer in making decisions along the way.

In order to identify the various roles computers can play in the design of physical artifacts, it is useful first to look at design as a problem-solving process whose goal is to find a way to change the current "state of the world" into one that is better suited for some purpose.

☐ THE PROBLEM-SOLVING MODEL OF DESIGN

Let us construct a model of design as a problem-solving process. The basis for this model is the assumption that for any problem we may define a solution space, which is the domain that includes all the possible solutions to the problem. Consider for example, the game of chess: At any point in the game there is a finite number of permissible moves. The chess player must attempt to evaluate these moves and choose the one most likely to lead to winning the game. In general, problem solving can be viewed as a process of searching through alternative solutions within the solution space in order to find one that meets certain criteria. The word *search* is used here in a metaphoric sense to describe a process of seeking and evaluating alternative solutions.

In this vein, a consumer looking to buy a car can also be said to be searching a "solution space" that consists of all the car makes that fulfill his needs and that he can afford. In contrast to this explicit solution space, the solution space searched by a design process is *implicit* since design problems usually involve seeking a solution that does not exist yet. This raises the question of how potential solutions may be produced for evaluation. The evaluation part must, therefore, be complemented by a solution-generating part—that is, a procedure that synthesizes potential solutions to the problem. If the solution-generating part is successful, the potential solutions will include at least one that complies with all the constraints of the design problem, achieves all its goals, and is technically and economically feasible.

Computers can be employed in various roles in this design model. They can be made to perform both the generation of potential solutions and their evaluation, a method which, when iterated, may eventually produce a solution to the given problem. In this case the human designer is completely excluded from the design process, and his role is limited to formulating the problem and presenting it to the computer for solution.

Alternatively, if we do not wish to or know how to make the computer generate potential solutions, we can include the human designer in the synthesis/analysis cycle as the generator of potential solutions and use the computer only for certain evaluations. The problem that arises in this case is, how will the designer communicate his generated potential solutions to the computer for analysis?

If we do not wish to bother with such communication problems, or if we cannot instruct the computer how to evaluate the potential solutions (e.g., their aesthetic qualities), then we can limit the role of the computer to a representational medium that replaces traditional paper-and-pencil drawing. The ability of the computer to represent the designed artifact in its three-dimensional form is far superior to the conventional method of representation by two-dimensional projections, such as the floorplans and elevations of a building. Three-dimensional representation allows the computer to test for interference between the pipes and ducts of the building or detect collisions between moving parts of a machine. It enables the computer to generate perspective, axonometric, and other views of the designed artifact automatically; and it allows the designer to "walk through" the building and get an impression of it, or to simulate the operation of a machine such as the tool path of a numerically controlled milling machine.

Finally, the computer can be used to generate shop drawings automatically, along with the many schedules and parts lists that accompany them. The computer can also be used to manage the design and production process as a whole, such as by identifying the critical path in a sequence of activities.

☐ BRIEF HISTORY OF COMPUTER-AIDED DESIGN

Computers were first introduced in the design process in the late 1950s as aids in performing the many calculations involved in analysis of engineered products. Computers calculated the structural behavior of artifacts, fluid flow, and thermal conductivity.

The early computer-aided design systems were aimed at improving the accuracy of various predictive models, using the computer's ability to handle many numbers very quickly. Little or no attention was given to the issue of data entry; it was assumed that design was performed in its traditional manner, except that at specific points in its progress some numerical quantities, representing the data for a set of equations, were taken off drawings and fed manually to the computer for specific processing.

Over the years, the tasks that could be processed by computers have multiplied and have been refined. Having learned how to solve many specific tasks, the developers of computer-aided design systems turned their attention to the growing problems of data collection and entry.

Then a new development occurred in the field of computing—a development that was crucial in bringing computer-aided design to its present state. This development was the realization that computers are capable of representing images as well as text and numbers. A new field, called *computer graphics*, was conceived at MIT in the late 1950s. It was, however, the work, of Steven Coons and his colleagues at MIT in the early 1960s that put computer graphics on the track that led to its current important status.

Coons developed a scenario that has guided most of the research and development in computer-aided design so far. According to his plan, design was to be done entirely on an interactive graphic terminal, using the computer to represent and manipulate the designed artifact. Since the entire artifact was already represented in the computer's memory, it could be used to extract the data required for a particular analysis automatically. Such analysis could be initiated by the designer or by the computer itself, and the results could be presented to the designer almost instantaneously and in the most convenient form. Upon completion of the design, the computer would be instructed to produce drawings, parts lists, and other production documentation, including fabrication instructions for numerically controlled machine tools. The computer, according to this scenario, becomes the engineer's "workstation," at which he can develop a product from start to finish.

The only design field in which Coons' scenario has been fully implemented so far is Electrical Engineering. More specifically, the design and fabrication of integrated circuits is now entirely computer-aided. This development owes as much to the work of Mead and Conway (who formalized integrated circuits design check rules) as to the many developments in computer-aided design, computer graphics, and computer science in general. In this capacity, computers are now used to interpret high-level design schemas, automatically generate and place electronic components, connect them, and simulate their performance.

Other fields of design are still a long way from realizing Coons' scenario. In architecture and engineering computers are limited, by and large, to the representation and analysis of design solutions developed manually. Automatic generation of design solutions has, so far, been limited to very well-defined problems. Furthermore, computer-aided integration of decisions made by many designers collaborating on one project is nonexistent. Yet, thanks to the work done by many researchers, Coons' scenario may be closer to realization in these "hard" design fields as well. Some examples of these advances include Nicholas Negroponte's and his MIT "Architecture Machine" group's proposed partnership approach to design, in which only those components of the process most suitable for automation are computerized, while others are left for the designer to perform. Charles Eastman and his Carnegie-Mellon University (CMU) "GLIDE" group proposed some models for representing the semantics, as well as the syntax, of large engineering and architectural artifacts and how their complex interrelationships could be managed automatically. William Mitchell and his UCLA colleagues demonstrated how the rules that underlie certain well-known architectural styles can be written down and reused, through a formalism known as "shape grammars." Steven Fenves and his colleagues at CMU demonstrated how certain decision-making processes that are employed in civil engineering can be made explicit and embedded in "expert systems" that mimic the behavior of human engineers.

All these developments, and many more, will advance the state of the art of CAD and will lead to a new generation of engineering workstations that will contain specific knowledge about the tasks for which they are employed, as well as the designed artifacts themselves. Thus they may become an active partner in the design process, rather than mere drafting and analysis aids. They may be able to check the progress of the design, suggest alternative courses of action, and guarantee that some predefined rules of consistency are preserved. Such systems will aid the *designer*, rather than the draftsperson, at a much higher level in the design process hierarchy than the current systems do and allow him to concentrate more fully on the critical design issues themselves.