Three-Dimensional Computer Graphics Using EGA or VGA Card

Hua Li, *Member, IEEE*

Abstract— **Scientific visualization is one of the major applications of computer graphics. It has been given increasing attention in the science and engineering communities. We developed an upper-level undergraduate 3D computer graphics course using the IBM PC with an enhanced graphics adapter (EGA) or video graphics array (VGA) card. Topics discussed in this course include three-dimensional viewing transformations, geometric transformations, animation, hidden line and surface removal, interpolations, and the shading techniques. A broad coverage of computer graphics techniques and the easy accessibility of the IBM PC to a large number of students are the major features of the course.**

I. Introduction

APPLICATIONS of computer graphics to different areas in engineering and sciences provide a useful tool to visualize complex phenomena and to comprehend large sets of data. Examples of such applications can be found in space science, numerical analysis, biomedical applications, robotics, and many others [6], [8], [9], [11], [13].

It has been predicted that the demand for computer graphics professionals will increase [3], [4], [17]. However, the high expense of sophisticated computer graphics equipment has been one of the prohibitive factors for some institutions wishing to offer computer graphics courses. We developed a high resolution three-dimensional computer graphics course by using the IBM PC with an enhanced graphics adaptor (EGA) or a video graphics array (VGA) card. This allows us to cover fairly large topics in computer graphics. It also provides an opportunity for many students to access graphics equipment and have hands-on experience in dealing with the topics in scientific visualization. About 25 students take the course per semester and five projects are assigned throughout the course.

II. Objectives

The objectives of this course are to provide students with hands-on experience in computer graphics and to furnish them with in-depth understanding of the basic topics in this field. These objectives are two-fold: one is on the theoretical aspect of algorithm analysis and the other is to highlight the implementation by programming assignments and projects. We expect that, by the end of the course, the students will be able to handle basic computer graphics techniques, to analyze and implement the algorithms of generating three-dimensional solid objects, shading, and perform hidden line removal.

The course is designed for the senior-level undergraduate engineering students. Most students have already had calculus and linear algebra as well as the concept of vector analysis. This mathematical background provides a basis for further discussion on transformations, curve interpolations, and shadings. From my experience, students are generally able to handle the topics well. The following topics were chosen for this course.

1) An introduction to computer graphics hardware is given. The architecture of graphics systems as well as the basic CRT techniques are briefly overviewed. The emphasis is placed on the features of VGA and EGA graphics cards to provide students knowledge to program on them.

2) The algorithms for drawing graphic primitives are discussed. The algorithms include simple DDA (digital differential analyzer) algorithms and integer DDA algorithms for drawing a straight line with the elimination of gaps [2]. The Bresenham's circle algorithm and an algorithm of drawing an ellipse are also covered [7]. Although the primitive-drawing algorithms have been implemented in hardware in sophisticated specialized graphics systems, we feel it is still necessary for students to understand the basic principles of the algorithms and to be able to implement them on those systems having no special hardware and software.

3) Transformations between two-dimensional and three-dimensional spaces are extensively studied since they form the foundation of three-dimensional computer graphics [1]. These include rotation, scaling, shearing, and translation. The viewing transformation from a world-coordinate system to a viewer-coordinate system, and the projection from the viewer-coordinate system to an image plane are the core topic in this category. We then discuss the three-dimensional rotation of a point about an arbitrary axis, reflection of a point about a given plane, and reflection of a known point about a straight line.

4) The Cohen–Sutherland's clipping [1] is discussed. Then flood-filling algorithms and scan-line algorithms [2] are covered. The technique for animations by switching frame buffers to display a sequence of preprocessed pictures is introduced to the class.

5) Hidden line and hidden surface removal algorithms [1], [2], [5], [7] then are provided. These algorithms are further emphasized by assigning a project.

6) Finally, curve interpolations using B-spline functions [1], [2], [5] are presented to the class. The topics on the interpolation are carefully chosen so that the students have no difficulty understanding the idea behind the mathematics.

Manuscript received January 1990; revised May 1990 and November 1990.

The author is with the Department of Computer Science, College of Engineering, Texas Tech University, Lubbock, TX 79409.

IEEE Log Number 9104606.

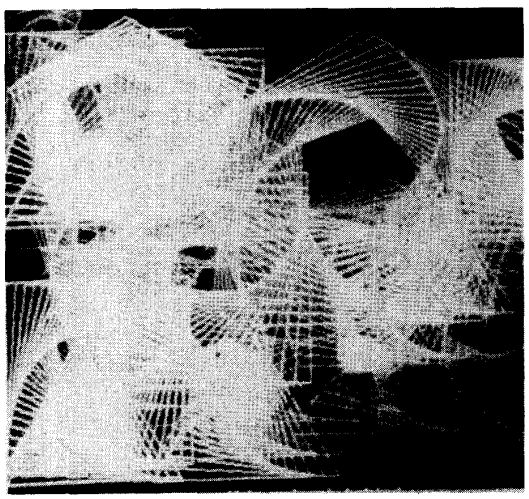


Fig. 1. The implementation of project 1. Note that each object is generated by using a vector formula with a factor k, where k is the number of vertices of a given polygon with $k = 3$, 4, and 5 for triangles, squares, and pentagons, respectively.

Fig. 2. Forest is generated by using two-dimensional transformations and recursion technique. This picture shows randomly positioned trees with different branch lengths and colors.

Phong's shading model [1] is introduced to the class. The technique of trading spatial resolution to color is given, which assigns different pixel patterns to a 3-by-3 template. Each pattern is defined to represent one level of a color intensity. This idea is borrowed from the "half-tone" technique, which has long been employed in the printing industry [1]. The technique is needed for an EGA or a VGA card where the number of colors is limited. Using this method, one color can be split into nine intensity levels.

The above topics basically form three categories: hardware, low-level computer graphics techniques, and high-level computer graphics techniques. The chosen topics in this class reflect the balance between hardware and software, as well as the balance between the theoretical aspects of the algorithms and their implementation.

III. Implementation

This course consists of lecture, projects, and homework assignments. We spend six weeks covering hardware and low-level computer graphics techniques, and six weeks covering the high-level computer graphics techniques. Several books are used as references: the book by Foley and Van Dam [5] is referenced for the topics of computer graphics hardware. The books by Berger and by Harrington [2], [7] are chosen as the references for the low-level computer graphics techniques. The book by Angell [1] is used to cover a fair amount of vector analysis and linear algebra topics which are carefully chosen for computer graphics applications, and the book by McGregor [12] gives an interesting treatment on the topic of generating three-dimensional data by decorative algorithms.

The software utilized in this course is TURBO C, version 1.5 or 2.0, which provides procedures for drawing graphics primitives. This software also allows students to write device drivers directly linked to MS assembly language procedures.

The project assignments are given to enhance the theoretical analysis of the algorithms. There are five projects assigned to each student during this course. They weight 30% of the whole course grade. Two midterm examinations weight 40%, and a final examination weights 30%. The implementation of the algorithms requires not only the understanding of the algorithms but also programming skills. The projects are listed below.

Project 1: Using vector analysis techniques to generate sets of different regular shaped two-dimensional objects. By "regular" shape, we mean the object having equal length on each side and equal internal angles, such as triangles, squares, pentagons, and so on. Each original object is reduced and rotated toward a certain direction. The iteration is then performed to generate a family of rotated objects of different sizes and colors. The whole computation can be defined by one compact vector equation.

Defining a polygon with a set of vertices $\{(x_i, y_i) | i = 1, 2, \cdots, k\}$, one can use a vector formula to describe an object reduction and rotation as:

$$(x_i^{l+1}, y_i^{l+1}) = (x_i^l, y_i^l) + \mu(x_{i+1}^l - x_i^l, y_{i+1}^l - y_i^l) \quad (1)$$

where the subscript i is used to denote each vertex of a given object, $i = 1, 2, \cdots, k$. If $i = k$ and $i + 1 > k$, then $i + 1 = 1$. The superscript l is used to denote the level of iteration. The constant μ is defined as $0 \leq \mu \leq 1$, which is used to define the rate of reduction. For example, if $\mu = 0.5$ then each side of the object is reduced to half. This μ is also related to the direction of the rotation. For μ less than 0.5, the rotation is toward the current reference point (x_i^l, y_i^l), otherwise the rotation is away from the current point. Equation (1) is, in fact, derived directly from a vector addition. Let us assume $\overline{a} = (x_i^l, y_i^l)$, $\overline{b} = (x_{i+1}^l - x_i^l, y_{i+1}^l - y_i^l)$, and $\overline{a}' = (x_i^{l+1}, y_i^{l+1})$, then (1) can be written as $\overline{a}' = \overline{a} + \mu(\overline{b} - \overline{a})$ for vertices i and $i + 1$ with reduction at level l. Following

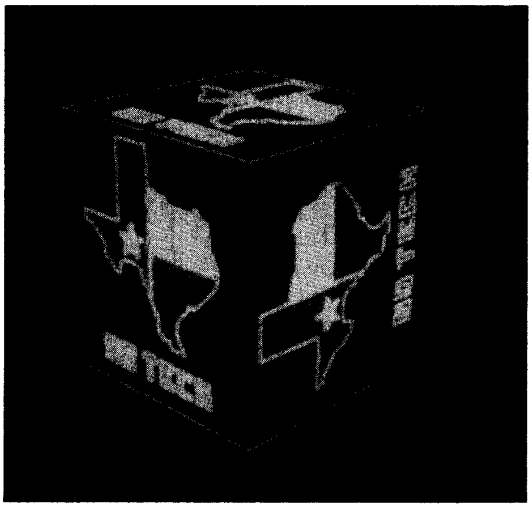


Fig. 3. The map of Texas is first digitized in project 3 and displayed as a two-dimensional picture. Then, in project 4, a linear decorative algorithm is utilized to place the picture on each surface of a cube as shown here.

the argument above, μ can also be expanded to the range greater than 1 to produce magnification.

Fig. 1 shows the implementation of this algorithm. Note the random position of objects and their different sizes.

Project 2: The second project is to use two-dimensional transformations, namely translation, rotation, and reduction, to generate a forest with many trees. The implementation of the project is given in Fig. 2. The technique to generate a tree is similar to that of generating a quad-coded image, which has a wide range of applications in computer graphics and image coding [14], [15]. This project is implemented by using recursion. The algorithm is given as follows.

1) Start from drawing the base branch (trunk) of the tree.

2) Perform translation of the branch and scale it down by a factor α $(0 < \alpha < 1)$ to generate a next-level major branch and draw it.

3) Perform rotation of the major branch obtained from 2) by $+\gamma$ and $-\gamma$ degrees to generate two subbranches and draw them (one subbranch is to the right and the other is to the left of the major branch).

4) Check if to stop; if not, then use each branch generated from 2) and 3) as the new major branch, go to step 2), and continue.

Note each branch of the tree having three subbranches, meaning each "parent" has "three children." This is the difference from quad-tree coding, where "four children" are expected for each "parent." The parameters of the tree, such as α and γ, can be randomly altered so that they are not identical at each different level. This can produce different-looking trees.

Project 3: This project requires students to generate their own data file using a digitizing technique. The file is used to draw a two-dimensional picture. This picture is then to be used later in project 4 to decorate a three-dimensional solid cube.

A map of Texas is chosen. The students digitize this map, then plot it with a caption GO TECH (go Texas Tech University) by using their own designed character font. One of the implementations, with the combination of a three-dimensional linear decorative algorithm, is given in Fig. 3. Two techniques have been taught to digitize a two-dimensional pattern. One is to use a digitizing device to directly get (x, y) coordinates from the map. The second is to use a paper-and-pencil method to hand measure (x, y) coordinates of the map. The hand-measure method is useful in designing different character fonts too, yet it is quite slow when the two-dimensional pattern has many details to be converted to points.

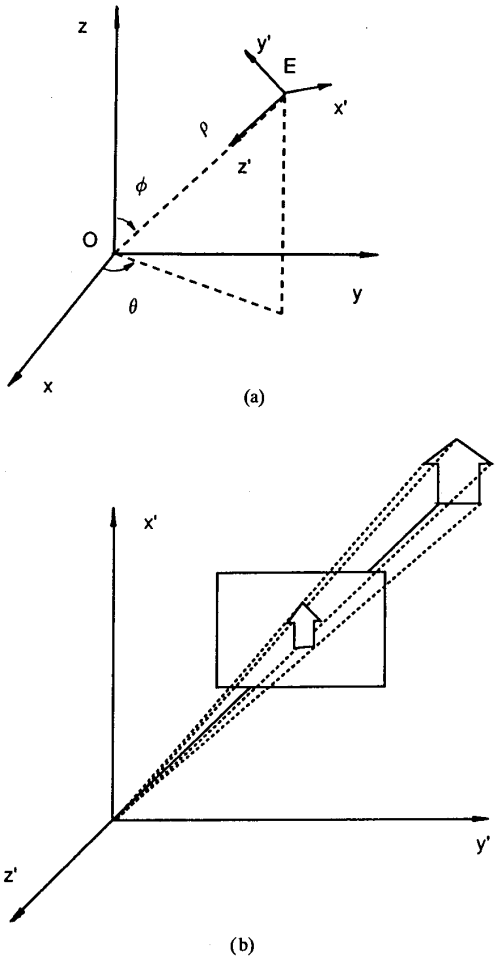


Fig. 4. (a) Illustrated here is the relation between the world-coordinate system and the viewer-coordinate system. Note how θ, ϕ, and ρ are defined. (b) The relation between the viewer-coordinate system and perspective projection.

Project 4: Upon completion of project 3, a solid three-dimensional cube is ready to be generated with each of its surfaces decorated. Then a two-dimensional Texas map is converted to the three-dimensional picture and is pasted on each surface of the cube. A transformation from two-dimensional space to three-dimensional space can be defined to accomplish this. For example, given below is the formula

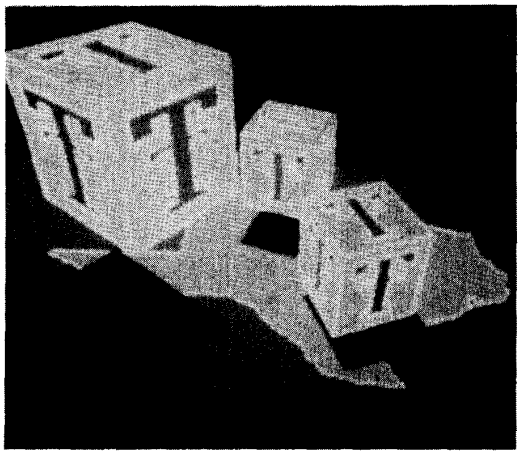


Fig. 5. A shading algorithm is utilized to generate this three-dimensional solid cube with each of its surfaces decorated by the Texas Tech University Logo. Note that the floor is also decorated by the map of Texas.

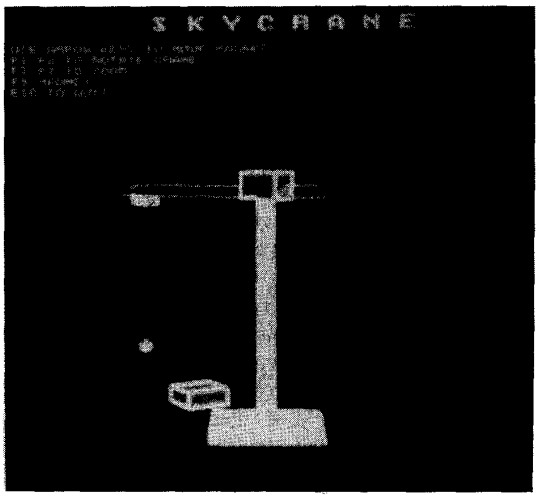


Fig. 6. This picture is generated for the demonstration of the interactive computer graphics technique. The user can modify the size of the object inside the picture and he may adjust the position of the skycrane in three-dimension and instruct the crane to pick up a load.

to paste a two-dimensional picture to a plane perpendicular to the y-axis in three-dimensional space:

$$z_w = y_{\text{old}}$$
$$x_w = x_{\text{old}}$$
$$y_w = C \tag{2}$$

where $(x_w, y_w, z_w)^t$ is a vertex in a three-dimensional world-coordinate system. Note that the above equations redefine each pixel of the two-dimensional picture to a new point in a three-dimensional world coordinate system. The constant C is chosen such that it is equal to the side length of the cube. Then the transformation from a world-coordinate system to a viewer-coordinate system is performed by:

$$(x_e, y_e, z_e, 1)^t = \mathbf{T}(x_w, y_w, z_w, 1)^t \tag{3}$$

where $(x_e, y_e, z_e)^t$ is a vertex of the picture in the viewer-coordinate system. The matrix $\mathbf{T}$ defines the transformation and is given as [1]:

$$\mathbf{T} = \begin{bmatrix} -\sin\theta & \cos\theta & 0 & 0 \\ -\cos\phi\cos\theta & -\cos\phi\sin\theta & \sin\phi & 0 \\ -\sin\phi\cos\theta & -\sin\phi\cos\theta & -\cos\phi & \rho \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where θ, ϕ, and ρ are illustrated in Fig. 4(a). Their physical meaning is illustrated graphically. Upon viewing the transformation, the perspective transformation is performed to produce a two-dimensional description of the three-dimensional object. This transformation is defined as:

$$x_p = x_e\left(\frac{D}{z_e}\right)$$
$$y_p = y_e\left(\frac{D}{z_e}\right) \tag{4}$$

where (x_p, y_p) is the vertex to be actually plotted on the screen. The physical meaning of D can be explained directly from the definition of the perspective projection. It is the distance from the frontal plane to the origin of the viewer-coordinate system as illustrated in Fig. 4(b). The longer the distance from the "eye" to the frontal plane, the shorter the distance from the frontal plane to the actual three-dimensional object. Hence, the bigger the picture on the display. Following the above transformations, a three-dimensional solid cube is generated with each of its visible sides decorated by the map of Texas. The implementation is given in Fig. 3.

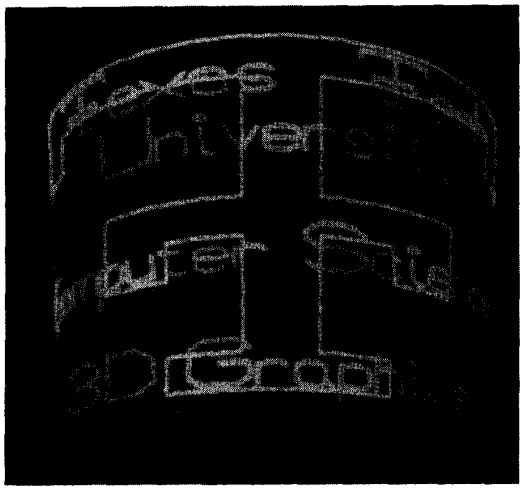
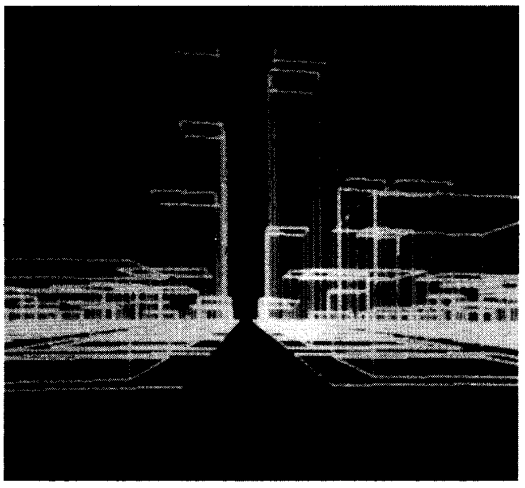
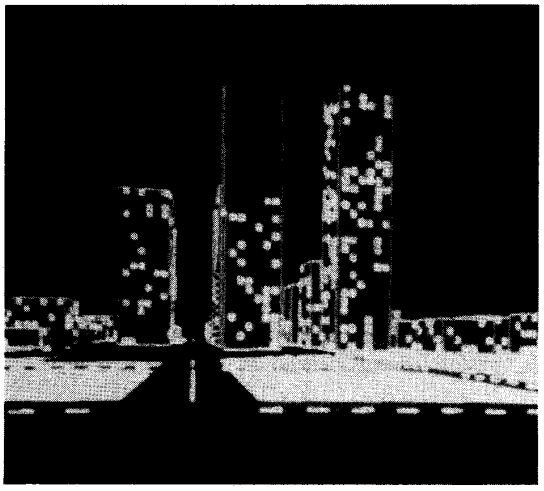


Fig. 7. Shown here is the plot of a three-dimensional nonlinear decorative algorithm.

Project 5: This last project requires students to apply several computer graphics techniques to build a relatively large project. The basic idea is to encourage students creative thinking. Students have to prepare a short proposal describing the design objective and outlining the implementation. Assistance is provided to the students in order to help them to identify an adequate and workable project. Upon approval of the proposal, students begin working on it. Supervision and technical guidance are provided throughout the project. Figs. 5–9 give some of the implementation results. Fig. 5



(a)



(b)

Fig. 8. Hidden surface removal algorithm is utilized to generate this three-dimensional city block with (a) wire-frame model; and (b) solid object model.

gives the picture generated by a shading algorithm. Fig. 6 shows a skycrane which can be operated interactively. It can rotate, its grabber can go up, down, or translate along the arm. It can even pick up a load. Fig. 7 shows the plot which utilizes a nonlinear decorative algorithm. Fig. 8 utilizes a hidden surface removal algorithm to generate a city block. A fault map of South America is produced to visualize the sources of earthquake. The result is shown in Fig. 9.

IV. Equipment

The computing facilities for this course are IBM PC's or compatibles with a 8086, 80286, or 80386 microprocessor. Most machines have one megabyte memory with EGA or VGA graphics cards. Some of our machines have a 20 megabyte hard disk.

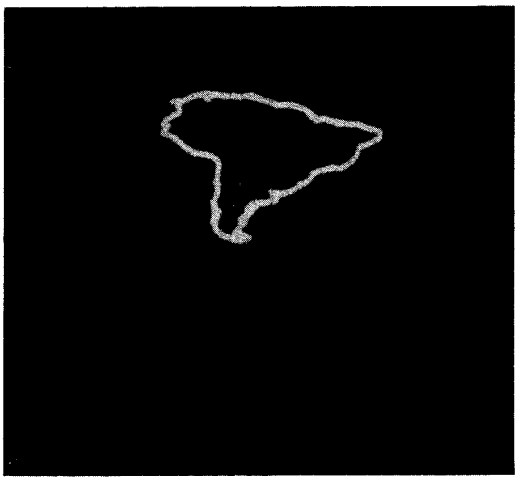


Fig. 9. Fault map of South America is generated to visualize the sources of earthquakes, where the red spots indicate the earthquake sources. A user can look at the fault map from any angle, even beneath the earth's surface by changing the viewing position.

EGA (Enhanced Graphics Adaptor) has become the IBM PC graphics standard since its introduction in late 1984. It is capable of producing a 640×350 resolution picture with 16 colors. BIOS (Basic Input Output System) supports software interrupt (INT 10H), which provides quite a handy way to program the EGA card. This video services interrupt gives the functions of setting cursor type and position, reading or writing pixels and/or characters, and initializing and manipulating video buffers [16]. With memory expansion, one can utilize up to eight pages of graphics memory on EGA [10]. Each page of the graphics can be displayed while the other is being modified. This allows viewing one video buffer and modifying the other independently. In project V of this course, this technique has been utilized to produce interactive computer graphics.

VGA was introduced in 1987 when IBM introduced PS/2 microcomputers. It improves EGA's graphics capabilities. The VGA has both high resolution mode (640×480) and low resolution mode (320×200). With the low resolution mode, 256 colors or 64 shades out of 262 144 possibilities may be manipulated simultaneously, which can be used to produce some realistic-looking images. VGA utilizes analog display rather than digital display used by EGA.

The high-level language TURBO C is used to program IBM PC and utilize EGA or VGA functions. This structured, modular, compiled language provides good portability. It can be used for a wide range of programming applications on different systems with little modifications. Using this language, one can also access to the IBM PC or PS/2 at the hardware level, which is necessary in certain cases where speed is concerned.

V. Summary

We developed a senior-level undergraduate course of 3D computer graphics. Topics discussed in this course include viewing transformation, hidden line removal, interpolations,

and basic shading techniques.. Five projects are designed to enhance the understanding of lecture material. The easy accessibility of the IBM PC to a large number of students and the broad coverage of computer graphic topics are the major features of this course.

ACKNOWLEDGMENT

The author would like to thank the referees for their comments and thank C. Young for lending her collections of graphics videotapes, which was helpful when this course was designed. Finally, the author would like to acknowledge his students, whose projects were used to prepare the camera-ready pictures for this paper.

REFERENCES

[1] I. O. Angell and G. Griffith, *High-Resolution Computer Graphics Using Pascal*. New York: Wiley, 1988.
[2] M. Berger, *Computer Graphics with PASCAL*. Menlo Park, CA: Benjamin/Cummings, 1986.
[3] W. Carlson, "Preparing for the future," Panel Sess. Part I, SIGGRAPH 1989 Panel Proc., pp. 295–304.
[4] T. A. DeFanti, M. Brown, and B. H. McGormick, "Visualization: Expanding scientific and engineering research opportunities," *Computer*, vol. 22, no. 8, pp. 12–25, Aug. 1989.
[5] J. D. Foley and A. Van Dam, *Fundamentals of Interactive Computer Graphics*. Reading, MA: Addison-Wesley, 1984.
[6] H. Fuchs, M. Levoy, and S. M. Pizer, "Interactive visualization of 3D medical data," *IEEE Trans. Comput.*, vol. 22, no. 8, pp. 46–51, Aug.1989.
[7] S. Harrington, *Computer Graphics–A Programming Approach*. New York: McGraw-Hill, 1987.
[8] J. Helman and L. Hesselink, "Representation and display of vector field topology in fluid flow data sets," *Computer*, vol. 22, no. 8, pp. 27–36, Aug. 1989.
[9] W. Hibbard and D. Santek, "Visualizing large data sets in the earth sciences," *Computer*, vol. 22, no. 8, pp. 53–57, Aug. 1989.
[10] B. D. Kliewer, *EGA/VGA: A Programmer's Reference Guide*. New York: McGraw-Hill, 1988.
[11] M. B. Long, K. Lyons, and J. K. Lam, "Acquisition and representation of 2D and 3D data from turbulent flows and flames," *Computer*, vol. 22, no. 8, pp. 39–45, Aug. 1989.
[12] J. McGregor and A. Watt, *The Art of Graphics for the IBM PC*. New York: Addison-Wesley, 1988.
[13] G. M. Nielson, "Guest editor's introduction: Visualization in scientific computing," *IEEE Trans. Comput.*, 1989.
[14] C. A. Shaffer and H. Samet, "Optimal quadtree construction algorithms," *Computer Vis. Graph. and Image Process.*, vol. 37, pp. 402–419, 1987.
[15] H. Samet and M. Tamminen, "Computing geometric properties of images represented by linear quadtrees," *IEEE Trans. Patt. Analy. Mach. Intell.*, vol. 7, no. 2, pp. 229–240, 1985.
[16] H. Thom, *The Programmer's PC Sourcebook* .Redmond, WA:Microsoft Press, 1988.
[17] M. Yanilmaz, "Preparing for the future," Panel Sess. Part III, SIGGRAPH 1989 Panel Proc., pp. 320–323.

Hua Li(S'86-M'87–S'88–M'89) received the B.S. degree in electrical engineering from Tientsin University, China, in 1982 and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Iowa, Iowa City, in 1984 and 1989, respectively.

He joined the Department of Computer Science, Texas Tech University, Lubbock, in 1989 as an Assistant Professor. His current research interests include computer vision and graphics, neural networks, and analog VLSI. He has published about 20 technical papers in IEEE Transactions, Technical Journals, and IEEE conferences.

Dr. Li is a member of the IEEE Computer Society and Upsilon Pi Epsilon.