5-1986

Computer Simulations of Biological Growth Patterns: Tree Modeling Success and Applications

Robert L. Nagel Jr.
Utah State University

COMPUTER SIMULATIONS OF BIOLOGICAL GROWTH PATTERNS: TREE MODELING SUCCESS AND APPLICATIONS

by

Robert L. Nagel Jr.

A report submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF LANDSCAPE ARCHITECTURE

(Plan B)

Approved:

UTAH STATE UNIVERSITY

Logan, Utah

1986

ACKNOWLEDGMENTS

I wish to express my appreciation to those whose help was invaluable in preparing the work that follows. Professor John Nicholson provided the initial impetus for this topic and worked as my major professor. His encouragement kept me going on track when success appeared elusive, and worked to secure financial aid when times were lean. Above all, his enthusiasm and confidence served to stimulate my own energy. I am also grateful to committee members Larry Wegkamp and Dr. Larre Egbert. Their guidance and criticism sustained me through the many areas of mist and uncertainty.

Appreciation and credit are due On Khong Lie, a graduate student in computer science for his programming work (Appendix B) and tutoring regarding the PS-300 terminal.

As for my friends, with whom I have shared a major portion of my life at U.S.U., they have taken the special association of friendship seriously. They have been as friends should be; understanding, supportive, patient, and sympathetic.

Finally, I wish to thank my parents for their continuous love and support all along the way.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

The purpose of this investigation was to explore the depiction of trees in three dimensions on a microcomputer. While the use of computer-aided design in landscape architecture is increasing, imagery for plant materials is found to be at a more or less symbolic level. The literature concerning previous inquiries into the mechanisms of tree growth and differentiation provide a good deal of information ranging from physiological basics to sophisticated structural and mathematical growth models. This forms the basis from which programming work proceeded.

In this context, the body of work reported here emphasizes the development of a programming methodology for achieving better tree images, rather than the sophistication of the images themselves. A major goal in this effort was simplicity in the resulting algorithms. This is significant in both minimizing use of computer memory, and in aiding the transfer of the algorithms to other devices and uses. Discussed are the developmental steps taken from an initial tree model requiring a digitizing tablet and the internal storage of coordinates, to a tree model in which machine memory and algorithm complexity are minimized.

The methodology deemed most useful is that of storing the trees as a general set of rules for image generation, rather than a lengthy data file for each tree. The operational value of this process is intrinsic to future applications; whether six discrete tree types are to be used or sixty types, the computer is working

with the same amount of "data" -- the tree generation algorithm. Further applications of this approach could offer savings in both storage requirements and data input for a variety of complex graphic images.

(108 pages)

CHAPTER I

INTRODUCTION

Computers are becoming a common tool in the design fields; from architecture, commercial art, and entertainment to the manufacture of electronics, automobiles, and industrial machinery. Techniques for displaying three-dimensional objects have been developed extensively in these areas, and make screen transformations relatively straightforward.

COMPUTER-AIDED DESIGN and the DESIGN ELEMENTS

One of the important attributes of computer-aided design, or CAD, is its ability to relieve people of repetitive procedures so more time can be spent on creative tasks (Milliken 1983, p.43). Architects using CAD systems rely heavily on this feature in designing buildings. A structural module need only be drawn once, and the computer can redraw it at any scale wherever needed to build up a design (Fullenwider and Lefever 1981, p.22). This ability could also be exploited readily in landscape architecture where plant materials are one of the major elements utilized in the design process.

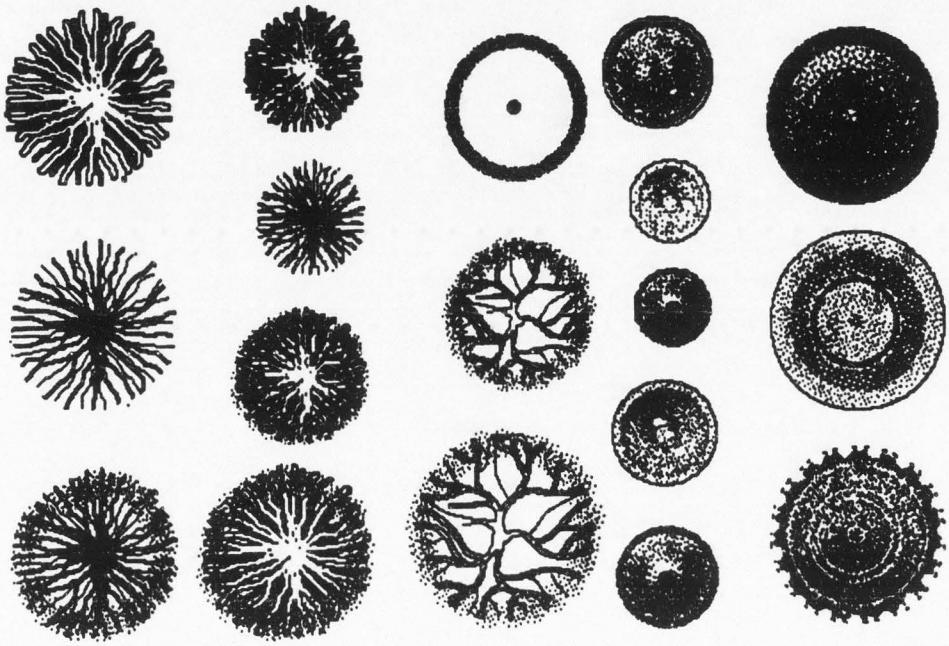
Past applications of CAD have dealt with engineered or "hard" design elements -- buildings and their components, machined metal parts, integrated circuits, etc. The

practice of landscape architecture encompasses not only hard elements, but also "soft" design elements such as trees, shrubs, and groundcovers, as well as landforms. Computer application in this area falls short of its potential when it comes to CAD, though. Most CAD programs used in landscape architecture are derived from Architecture-based systems. The result is a more sophisticated portrayal of structural design elements while plant materials tend to remain at a more primitive, symbolic level (figure 1).

Since trees and other plants assume such a dominant role in landscape design, it becomes desirable to depict them as more than flat images in a computerized environment. Understanding the mechanisms involved in how a plant achieves its form constitutes a good first step in developing more sophisticated graphic images.

TREE GROWTH INVESTIGATION and LANDSCAPE ARCHITECTURE

Much work has been done over the years in investigating biological growth patterns, from the branching of certain red algae filaments (Lindenmayer 1968), to general expressions of form in both plants and animals (Rashevsky 1943, and Cohen 1967). As research delves deeper into the dynamics of growth patterns, the expression of form in plants and animals is becoming more quantifiable, and computers have consequently become a



(Hayden Software Inc.)

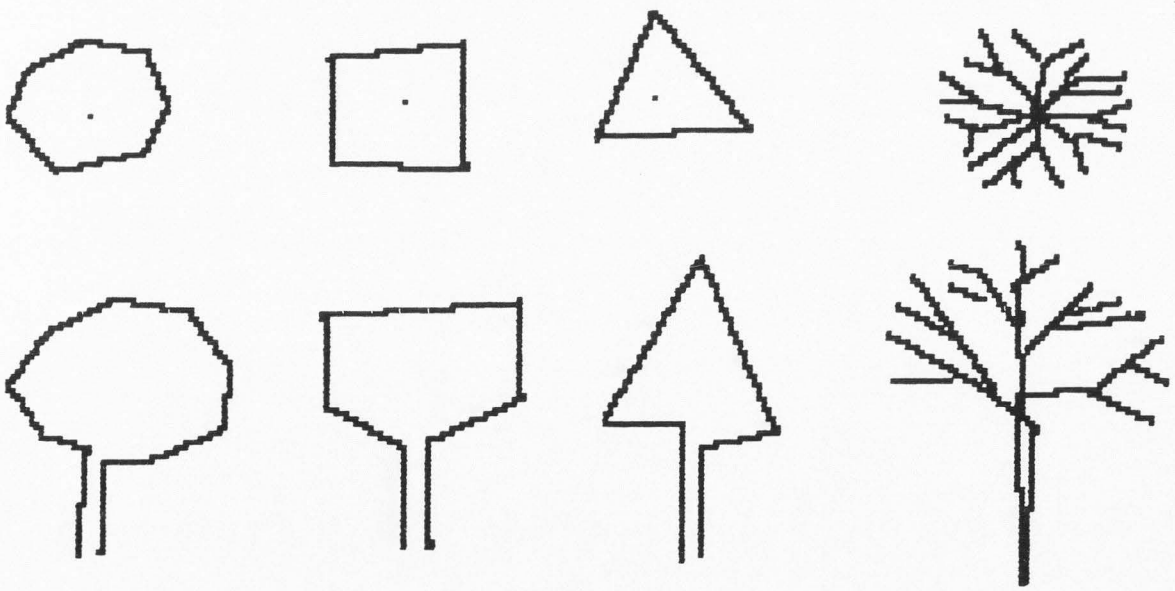


Figure 1. Examples of some symbols used to denote trees via computer systems ("Sketch" 1982, S. Ervin)

prominent tool in their study. There are a great number of factors affecting the form that a particular organism exhibits, with researchers tending to stress genetics, biophysics, and environment to greater or lesser degrees (Rashevsky 1943, Cohen 1967). Computer scientists have also worked along somewhat similar lines of investigation. Jack Fisher and Hisao Honda have worked with a specific tree species in determining the effects of branching angles on maximum effective leaf area through the use of computer simulation (Honda 1971, Fisher and Honda 1977, and 1979). Altering the variables in this computer model results in variations in tree form. The structure of this type of tree is a natural pattern generated by branching out in two directions from the terminal end of each previously generated branch. Two other researchers have recently reported on work they have carried out with nearly the same type of modeling proceedures (Aono and Kunii, 1984). Fisher and Honda's work is attractive in their desire to describe the smallest number of factors having the greatest effects on the overall form of their trees. This contributes directly to the simplicity of the algorithms involved. While the work of Fisher and Honda was directed toward determining tree crown geometry and its applications in forestry and horticulture, Aono and Kunii oriented their work more toward realistic graphics, achieveing their realism via a large number of control parameters (Ibid).

Landscape architecture stands to gain from contact with

this type of research, specifically where the dipiction of plant materials in design is concerned. Techniques and information generated in the fields of theoretical biology and computer science need to be investigated with respect to current needs in landscape architecture. Being a profession emphasizing implementation rather than pure research, landscape architecture must maintain communication with fields that can offer new approaches to problems. Landscape architects rely on research and data collection carried out by geologists, sociologists, soil scientists, structural engineers, biologists, hydrologists, computer scientists, and many others. Research pertaining to growth patterns of trees and the modeling of these patterns by computer offers the possibility of new techniques for the design professional.

METHODOLOGY DEVELOPMENT

One method for displaying a tree on the graphics screen is to enter all the coordinate points representing a given tree shape (either via the keyboard or digitizing tablet). Due to the large amount of time and memory involved, digitizing an entire tree is not a feasible undertaking. This results in a representation of one tree, making it an inefficient procedure when a range of tree types is required, and does not exploit the capabilities of the computer. More generalized techniques are needed to allow for flexibility and speed.

Rotation Model. Initial work began with rotating a tree profile about its central axis. This results in a structure not unlike that of an eggbeater or wire whisk used in cooking (figure 2). The tree shown in figure 2 was created by entering the original data via the Apple computer's digitizing tablet in the form of a tree profile (figure 3). This profile determines the shape of the resulting tree after rotation, and is capable of displaying both general tree outline and density -- the number of profiles comprising the canopy (the program to ficilitate digitizing and rotation is in Appendix A).

The next stage was digitizing a series of branches along the length of the tree trunk in place of the tree profile. Through the use of the same rotation subroutine used above, a tree composed of a radially arranged set of branches would result. This represented a net increase in digitizing time and complexity at a point when the work of other researchers came to light who's techniques offer a more complex tree structure without the necessity of digitizing.

Branching Model. Two researchers, Jack Fisher and Hsio Honda provide a more promising alternative to digitizing a tree: give the computer a set of rules and guidelines for constructing the tree. These rules constitute the tree data file instead of a long list of vector coordinates.

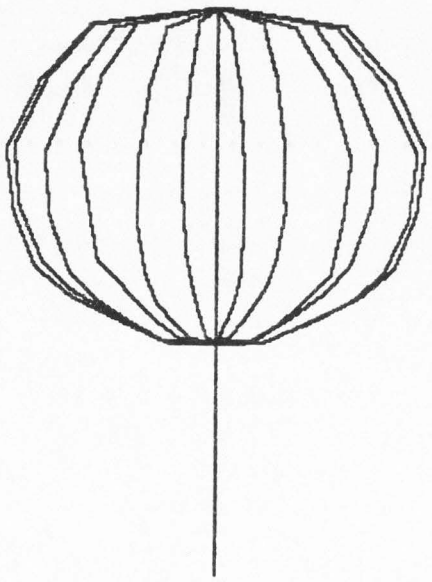


Figure 2. Whisk-type tree structure generated from rotation of canopy profile around vertical axis.

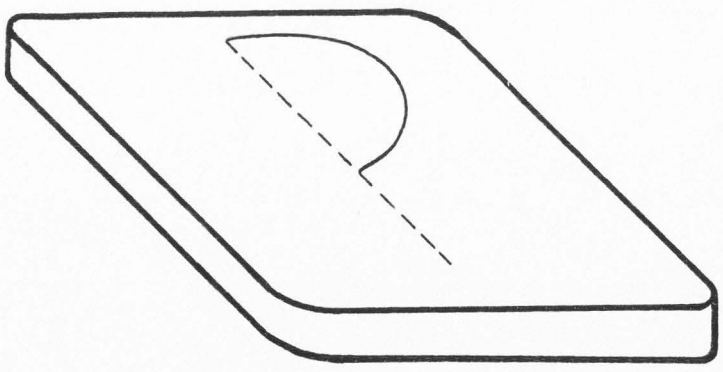


Figure 3. Whisk tree canopy profile as entered on digitizing tablet.

Fisher and Honda applied biological growth research to a simulation model that generates tree stuctures governed by a small set of parameters. As these growth parameters are altered, the final form of the tree also changes.

Honda's general equations (1971, p.334) for calculating branch coordinates are applied using a recursive type of algorithm to carry out the branching process. As the program progresses from the central tree axis outward, each branch shortens in length relative to that of it's predecessor according to a pre-set ratio. Two basic parameters, branching angle and branch length, are utilized to vary the way in which each tree exhibits its final form (figure 4).

The physical structure of this type of tree significantly parallels a relatively new area of mathematics; fractal geometry, which will be further discussed later. As a subject of many recent publications and conferences it is a concept that has greatly aided in unifying the understanding of complex physical systems (Pandey 1984). A major fractal type is that of recursively generated structures exhibiting the property of self-similarity found in this tree model. While not a central avenue of inquiry, fractal geometry lends a theoretical approach compatable with the evolution of this branching model.

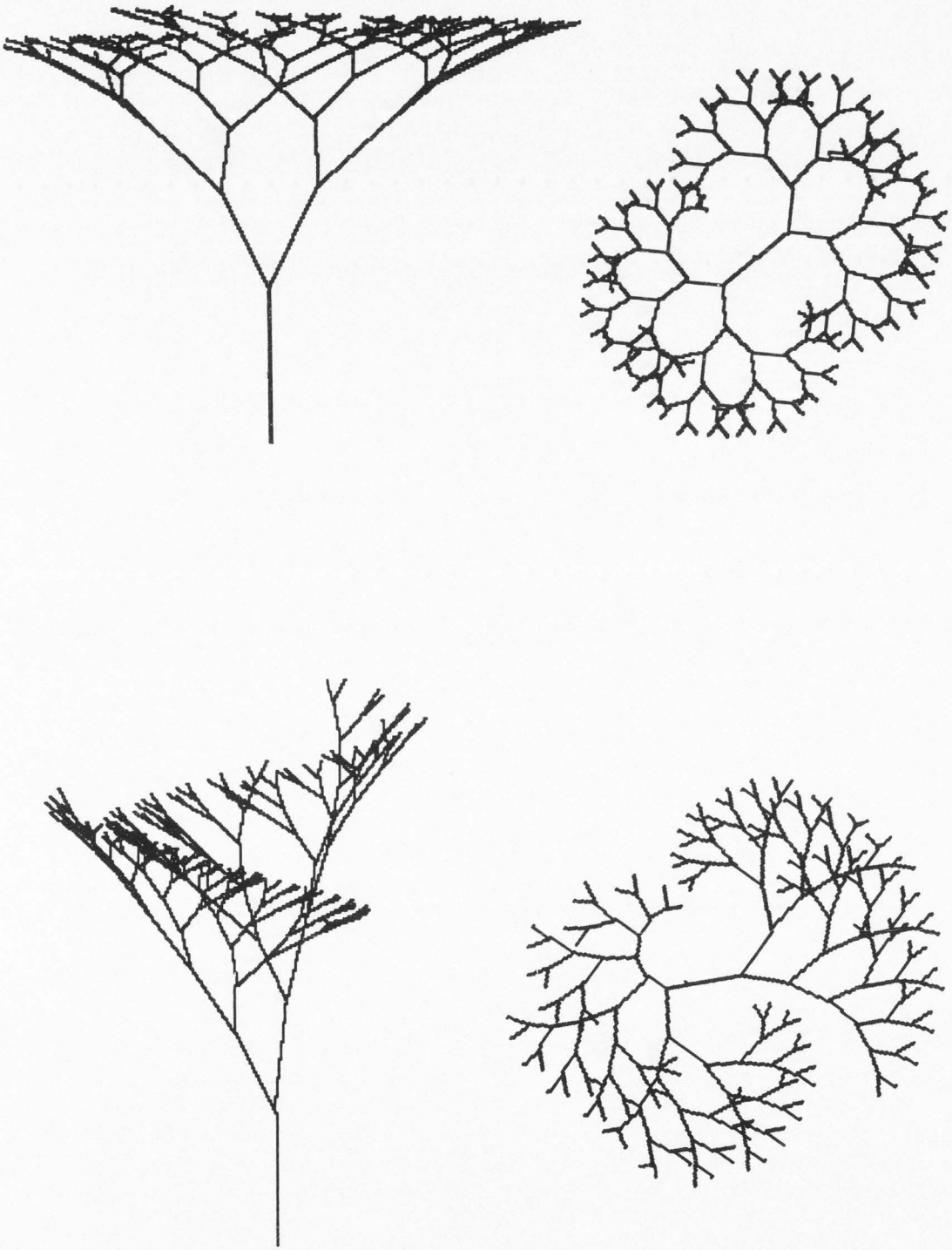


Figure 4. Top and side views illustrating change in tree form as program variables are altered.

GOALS

This thesis focuses on the generation and dipiction of plant materials in three dimensions on a computer, with major emphasis on the methods used for the generation of trees. Major goals identified as guides are:

* use of a minimum of machine memory
* portability of algorithms to other devices
* simplicity of governing variables
* output capable of display on other devices

A detailed tree structure that requires little memory is advantageous to its portability. This is not just because developmental work takes place on a microcomputer where memory can be a limitation, but more importantly because tree generation is an adjunct to the many other elements that occupy memory in a CAD system. Also important is developing techniques that can be generalized. The majority of this research carried out for this thesis takes place on an Apple II+ microcomputer, but the intent is for machine independent algorithms transferable to a wide range of computing devices. Another factor which compliments the above is simplicity. The more variables that come into play the more cumbersome the resulting program, and as more and more factors affect the overall tree shape the harder it becomes to isolate the effects of individual changes. The less complicated the

algorithms can be made then, the more easily they can be adapted to other uses and environments.

This paper describes the research and procedures applied to produce computer-generated tree structures, and addresses future possibilities. A basic question to be answered is whether adequate tree representations can be generated on a microcomputer while keeping complexity and memory requirements to a minimum. The conceptual approach involved in generating these trees is of great importance not only as documentation, but also in that it can point to new avenues for refinement. Also addressed is how these tree structures fit into current and future CAD applications, their advantages, and their limitations.

CHAPTER II

LITERATURE REVIEW

INTRODUCTION

The purpose of this chapter is to familiarize the reader with research and applications consequential to the three-dimensional depiction of trees. This will provide a background helpful in understanding concepts introduced later, as well as establishing a context for possible future applications. Computer-aided design will be discussed briefly as a vehicle for expressing design ideas and in illustration of its increasing usefulness. Next, its applications in both architectural and landscape architectural practice establishes its current status in these professions, and as the impetus for beginning the research for this thesis. Finally, a discussion of factors determining the physical form of organisms, and trees in particular, will provide the theoretical underpinning for development of tree simulations.

COMPUTER-AIDED DESIGN

The field of design, whether as applied to automobiles, integrated circuits, or to one's physical surroundings, has undergone its own technological revolution. The use of computers has greatly enhanced the speed of analyzing and manipulating designs. Computer-aided design has even made possible the testing of certain design schemes before they

are actually built.

Computer-aided drafting usually goes hand-in-hand with computer-aided design, or CAD, since it is a logical means for entering a design into the computer, and for hard-copy display. Once the design is in a computerized data format, it is very convenient to allow computer control of the drafting process. CAD is also often coupled with computer-aided manufacture (CAM). CAM is the industrial continuation of computerization, sometimes utilizing robotics in manufacturing. For the purposes of this thesis however, CAD will be discussed as a single entity.

For all practical purposes, a particular design (car, floor plan, machine part) exists only in the electronic memory of the CAD system. As changes are made in the object, it is only this memory that is modified. Looking at the object via the computer screen is much like looking into the designer's imagination, since attributes of the design can be rapidly explored and changed. It is only when the design has been finalized on the computer that time and money need to be spent in drafting a set of plans and/or building a prototype.

History. CAD had its beginnings about 20 years ago in a doctoral dissertation by Ivan E. Sutherland at MIT (Teicholz 1983). He introduced a CAD drafting program called "sketchpad." It's use was limited to specialized hardware at first, but six years later commercial versions of the system were marketed by Compuvision Corporation. It was a system

that automated two-dimensional production drafting. Since then the CAD/CAM industry has grown (by 60 to 70 percent yearly in the late 1970's) (Ibid). CAD systems are already in use in many engineering and manufacturing fields and have been for quite some time (figure 5). One need only look at the automobile or electronics industry to see state of the art CAD in use. While software development hasn't quite kept pace, hardware developments have greatly increased the speed of manipulating three-dimensional objects and removing hidden surfaces. Color graphics is also becoming a common feature, while costs are decreasing (Ibid).

Computer-aided design has found ready application in a number of fields. It has shown itself especially suited for electronics design, engineering design, automobile and aircraft design, mapping, modeling and simulation, highway planning, architectural design, and the layout of publications among many other applications (Sutherland 1970).

DESIGN APPLICATIONS in ARCHITECTURE

It is in this state of technological activity that CAD is beginning to gain a foot-hold in environmental design. In comparison with landscape architects, architects have tended to lead the way in the use of CAD. But they also have had to face some problems along the way. Computerization brings up many questions about whether or not the investment will pay off, and how to make an intelligent decision on procuring the

proper hardware and software. There is the fear of loss of key personnel and creativity, along with the time that must be devoted just for the office to "come up to speed" with a system. Long term investments require careful planning to make them pay off. It can be assumed that the effective life of a particular technical methodology will be much shorter than the productive career of its user (Steinitz 1982). All the technical information required in order to make intelligent decisions is one reason that Barry Milliken feels that "the [Architecture] profession will not adopt computerized techniques as quickly as some think. Some firms will move faster than others, but the over-all process will be more evolutionary than revolutionary." (Milliken 1983,p.41)

Even so, CAD is an integral part of practice in a number of architectural firms. Benefits include increased efficiency and productivity, retention of competitive costs, and improved quality and accuracy. Its continued use will produce dimensionally accurate, legible construction plans, promote standardization in the representation of components, and speed up the process of making changes throughout the design (Fullenwider and Lefever 1981).

In Ivan Sutherland's words, "The objective of most computer graphics programs is easily stated: to represent objects of some sort and to provide a means for manipulating them." (1970, p.65) This clearly states the basics of CAD as used in architectural design. A structure can be represented

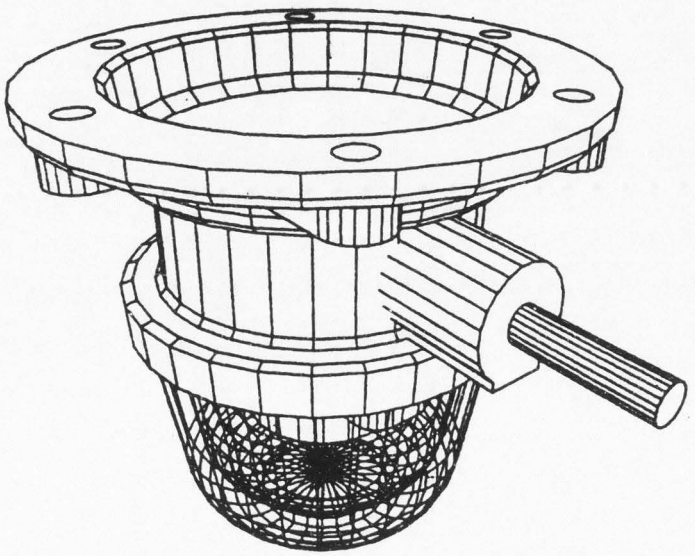


Figure 5. Plot industrial component designed on CAD system (CAD/CAM Digest 1982, cover).

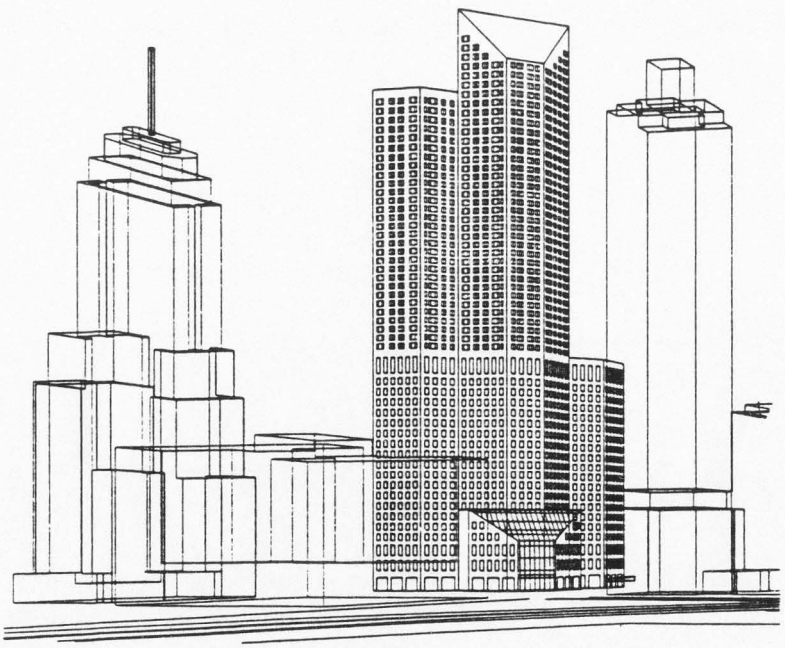


Figure 6. Proposed structure surrounded gy existing buildings as depicted on CAD system (Architectural Record 1980, p.87).

on the screen and quickly viewed from another angle or modified in some way. But most systems in use go beyond these basics.

Processing Speed. Through the use of layered drawing techniques different levels of information can be dealt with. For example, a good architectural CAD system can handle a multitude of design information; from a basic site plan, floor plans, wiring and plumbing plans, to interior furnishings. Throughout all this the computer can keep track of materials, costs, and labor needs. A basic structural change in a support column could require changes in all the wiring, plumbing, heating, and other mechanical elements that interact with the column. Posessing all the original data, the computer can quickly make necessary changes all the way through each level of information (Fullenwider and Lefever 1981). Changes in one design unit can affect the specifications for the entire structure, causing expensive design changes and redrafting, especially when time is short. A competent CAD system can adjust the entire structure to these changes in a short time, and allow the architect/client to view it. Likewise, repetitive designs can be drawn by the computer at great time savings. A skyscraper with many similar floors would entail drafting each floor plan individually, while a CAD system would require entering only the differences in each floor (Whitted 1984).

Simulation. Simulation also plays an important role in commercial applications. The construction of a building can be simulated in order to determine the most economical construction process and material use, avoiding costly readjustments once construction has actually commenced. Modification of often-repeated components before construction can save costly custom fitting later (Business Week 1982).

The ability to simulate three-dimensional objects on the computer is also a great aid in presenting designs to clients. A 3-D perspective view of a building can point out design conflicts that may not be readily apparent in two dimensions. This view can also make a design much easier for a client or other interested party to understand, ensuring clearer communications with the designer. The architect and client can simulate a tour through, around, or over a proposed project without ever leaving the computer terminal (Fullenwider and Lefever 1981) (figure 6).

These new abilities imparted by computers have made an economic impact on architectural practice in both man-hours and design fees. "With the sophisticated and easy-to-use CAD systems just now hitting the market, a growing number of architects are able to design buildings as much as 10 times faster than they could draw them manually. The computer has also enabled these pioneers to cut their design fees by 10% to 60% and increase their profit margins at the same time." (Business Week 1982, p.134) A three million dollar building that would have taken three months to produce plans for

manually can be drafted out in a matters of hours by one architect with a computerized building-component data base (Ibid). Of $237 million expected to be spent on computer equipment in 1983 and 1984, it is projected that $25 million of it will go toward CAD systems. This in part, is because systems that cost over $200,000 in the 1970s are now under half that price (Stasiowski 1982, Dietsch 1982).

Small Device Limitations and Applications. As widely used as microcomputers are today, they are not yet quite suitable to most mainstream CAD operations. The use of a large computer gives a designer the most in terms of memory and speed, but also represents a cost factor too great for many small to medium firms to afford. As one moves down from the main-frame and minicomputers to the microcomputers, not only do costs sharply decrease, but the ability to handle large-scale jobs with their vast amounts of data become increasingly difficult (Stitt 1982). The usual application of CAD in architecture is in creating, storing, and manipulating images of buildings and other objects. "This type of work, especially in three dimensions, requires high-powered equipment and extensive memory. As far as I can determine, CAD is still not widely used by smaller architectural firms." (Ibid, p.49) What is left for most microcomputer applications are the usual office management operations (word processing, accounting, billing, etc.); keeping track of materials, building code data, construction

schedules, and design checklists; as well as maintaining libraries of construction details, indexes, legends, and notations.

As microcomputer systems have become more sophisticated and their use more pervasive, software vendors are beginning to apply minicomputer-based systems to microcomputers. Additionally, conversion programs are being offered that will translate data directly from a microcomputer to a larger CAD system such as those by Computervision, Intergraph Corp., and Autotrol Corp. (Rouse 1984). More than thirty vendors are currently offering personal computer-based drafting systems and software, but very few with three-dimensional capabilities. Of those, Cubicom Corp. also has a solid modeler for the IBM-PC, rather than the usual wireframe display (Ibid).

COMPUTER APPLICATIONS in

LANDSCAPE ARCHITECTURE

Microcomputers are being employed in the landscape office most often for routine business procedures and word processing, tasks which they fulfill sucessfully in terms of speed and equipment costs. The application of microcomputers for tasks specific to landscape architecture has already begun, though it has been approached with some hesitancy along the way. Start-up costs, and a percieved loss of control over the design process are two primary concerns (Clay 1980).

Architects have tended to lead the way in the use of computers for design applications. As landscape architects face a rapidly changing world of computerization, there are still hurdles to overcome. E. Bruce MacDougall, in his book "Microcomputers in Landscape Architecture", emphasizes that the introduction of a computer is quite different than any other piece of new office equipment. The computer can very well change the organization of the office, from proceedures to personnel. The landscape architect may feel at a disadvantage in computer expertise, risking not only loss of direct control over proceedures, but out-and-out failure in the office (MacDougall 1983).

But the decreasing prices of hardware have enhanced the attractiveness of handing over more office activities to a computer: information management, accounting, word processing, cost estimation, technical calculations, etc.

The development of software directly applicable to landscape architecture is also helping to integrate the computer into a full range of office proceedures.

Applied Uses. While some software merely acts as an information base for inventories and cost accounting, other programs specifically aid in selecting plants to fit certain criteria. A sort routine produces a list of available plants compatable with on-site conditions and design needs.

Software written for the engineering and construction fields finds ready application in landscape construction. Programs are used to size lumber and compute stress loads for decks and other wooden structures. Others provide data for earthmoving operations -- grading and cut and fill calculations. The planning of new roads is aided by road alignment programs, some of which provide only the construction data necessary to lay out the road while others show a generalized view of the completed roadway on the screen (Breeden 1984).

Part of a landscape architect's work entails planning, and for some the major portion of their practice. There are many areas where computers are making a significant contribution in both analysis and mapping. Calculations involved in plotting shadow patterns and sun angles for solar orientation involves gathering and interpreting data from a number of complex charts. A properly programmed

computer can complete these operations in a matter of seconds, giving the planner an idea of how the sun will affect a certain site or building throughout the year (MacDougall 1983). This is especially helpful in designing closely spaced, energy conserving uses.

Much of land planning and analysis involves the manipulation of mapped data. Impact analyses, suitability analyses, and other regional planning procedures draw on a large number of map attributes. The output of most of these analyses is also in a mapped format. Soils, topography, vegetation cover, hydrology, wildlife, socio-economic and weather data, transportation networks, etc. come into play in a thorough environmental analysis. The speedy manipulation of spatial data, and its output in a graphic format is a task particularly suitable to computers. For example, one such program written at Utah State University by Prof. John Nicholson, "Microsieve", is designed as a flexible aid in making land use decisions. The program operates on the principle of comparing data maps in various ways and combinations. It can be used to either formulate an overall land use plan, or to identify areas either suitable and unsuitable for a particular land use activity (figure 7).

Other programs of this type may focus more on economic or social type data, or may specialize in one type of analysis. Visual Analysis, for example, relies mainly on topographic information to determine what can and cannot be

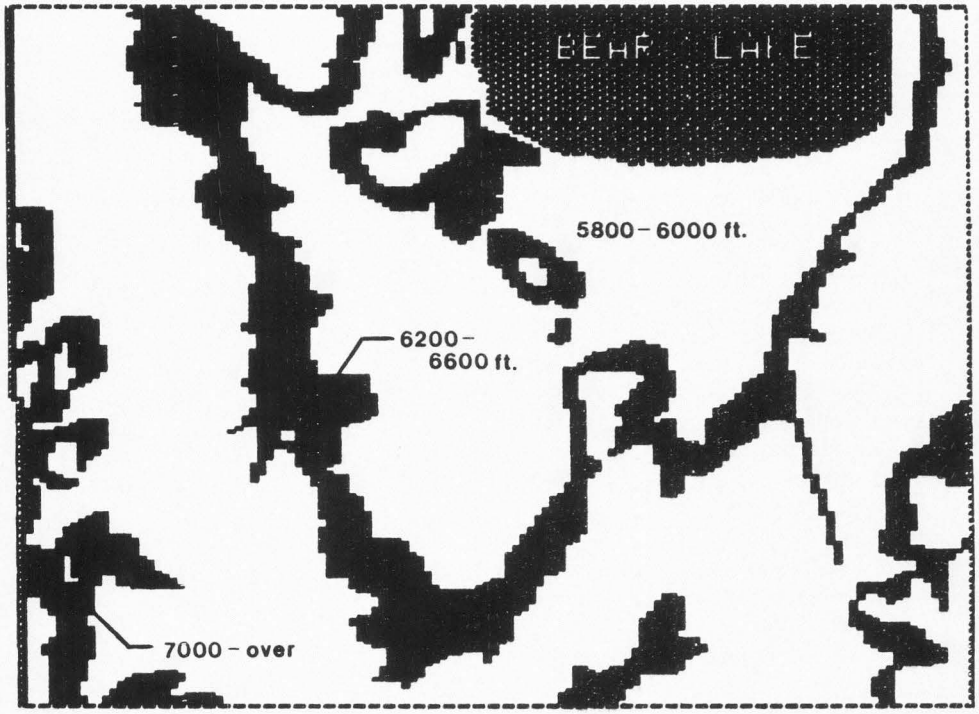


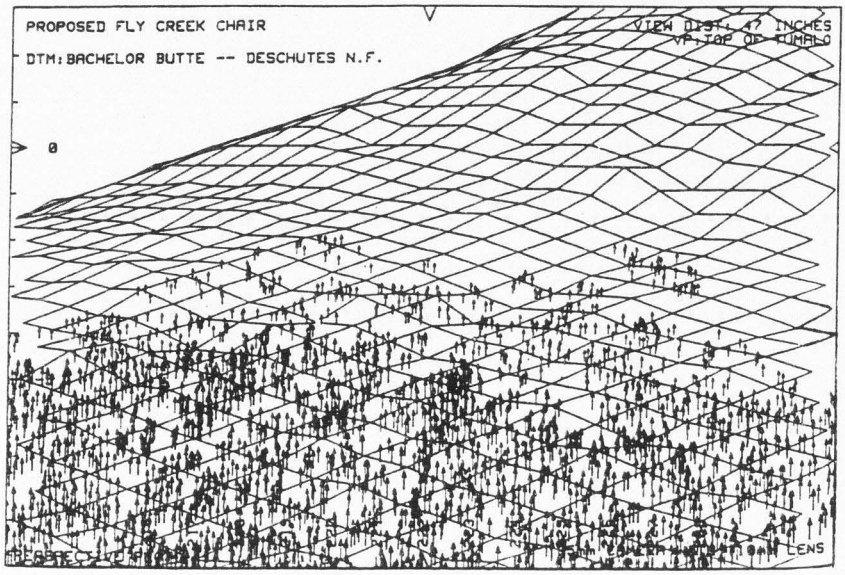
Figure 7. Example of "Micro-sieve data map displaying elevation data used in evaluating existing or proposed land use activities.

seen from a particular location. The U.S. Forest Service, Bureau of Land Management, and Soil Concervation Service are three government agencies that rely on several visual analysis programs to aid in implementing management objectives (figure 8). These visual resource management systems are used both to conduct visibility studies and to simulate changes in a particular landscape (Evans 1984).

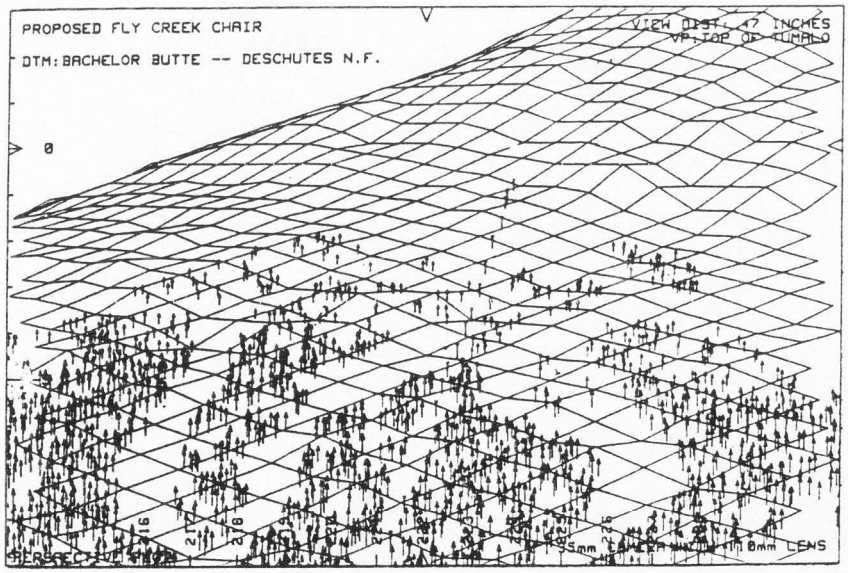
An adjunct to computer mapping is data collected via remoting sensing apparatus. Spectral data collected by high-flying aircraft or NASA's Earth Resources Technology Satellite (Landsat) can be combined with a computerized geographic information system to be used in agricultural land use classification, forest inventories, world-wide crop forcasts, urban studies, energy and mineral exploration, snowpack measurement, pollution detection, and strip mine monitoring among an expanding number of others (Killpack 1982). Many of these types of remote-data prove useful to landscape planning in regional as well as small scale projects.

Education of landscape students in computer applications entails a basic familiarity with computer languages and programming, along with hands-on experience. Most of the applications mentioned above are employed in education, depending on availability of programs and hardware to the school.

Actual application of CAD in landscape architecture is still at a rather early level. The high (though dropping)



view of area before proposed development



view of area after proposed development

Figure 8. Computer generated views of visual impact of proposed ski area (Nickerson and Arneson 1981, p.738).

costs of systems, and small (though growing) number of programs specific to landscape architecture, coupled with the fact that design is not the most cost effective area for computer applications in most firms has kept progress slow (Wagner and Mileaf 1983, p.43). Revision and analysis of structural designs in the architecture office have shown to be more productive in computer time than the actual start-from-scratch design process itself (Ibid).

TREE GROWTH INVESTIGATIONS and SIMULATIONS

As the use of CAD systems becomes more commonplace in landscape architectural practice, a more sophisticated method for representing trees will become necessary. In a landscape design the architectural elements are one of the site characteristics that need to be integrated into the total design. One of the landscape architect's tools for this purpose is trees. In pursuit of realism for graphic tree representations, some very important concepts come from the biological sciences.

Research History. Early natural philosophers such a Plato, Aristotle, Aquinus, and Goethe believed that form is a fundamental force in nature (Hapgood 1982). As science has become more pragmatic in its approach, the forces responsible for natural form have come under scrutiny. These forces and their effects were explored in a 1917 book

"On Growth and Form", written by British naturalist and mathematician D'Arcy Thompson. His main precept was that biological form is a direct result of the physical forces acting upon an organism. For instance, he suggested that the shape of a honeycomb results more from the strain and tension of close packing than from a predetermined design lodged in the mind of the bee. Thompson "was convinced that the form of a creature was a window into the world of forces in which it lived" (Hapgood 1982, p.51).

In 1943, N. Rashevsky published "Outline of a New Mathematical Approach to General Biology: I". In it he went beyond Thompson by quantifying the forces he saw through Thompson's "window". He worked to develop a mathematical theory of lever-propelled metabolizing systems to explain the locomotion of organisms. His thrust was that one could "express the essentially discontinuous properties of the organic form (no extremities, two extremities, six extremities) by a set of continuously varying parameters" (Rashevesky 1943, p.46).

Factors contributing to form and strength in trees were the work of another researcher, I. Opatowski. He dealt mostly with structural strength, elastic stability, height limitations, and stability -- in both trunk and branches (1944). His work, that of Rashevsky, and the work of others along these lines began to define structural principles and limitations (trunk and branch diameter and length, primary, secondary, and tertiary branch masses,

etc.).

Growth Mechanisms. The study of tree growth patterns is firmly based in plant morphology, physiology, and biochemistry. A thorough understanding of the mechanisms involved can be gained from a proper text and would be out of place in this discussion, but basically the terminal bud (apical meristem) controls stem growth and consequently height. Branches develop from axillary buds along the stem, and the location of these buds determines the location of branches (figure 9). Also, various hormones in the plant mediate this development (auxins, gibberillins, and cytokinins). The major ones controlling growth are the auxins (McMahon 1975). Auxin moves from the site where it is systhesized to certain target tissues, and these parts react to its presence with alterations in growth, development, or metabolism (Bidwell 1979). Auxin has been found to generally move from the tip to the base of a plant. In the instance of apical dominance (growth dominance of the growing tip), auxin moves downward from the growing tip and inhibits lateral bud growth (Ibid). This effect functions to a greater or lesser extent in most plants.

Growth In Trees. The apical meristem is a major auxin site in trees. As the meristematic tissue divides and grows its cells differentiate into various types of tissue, and the

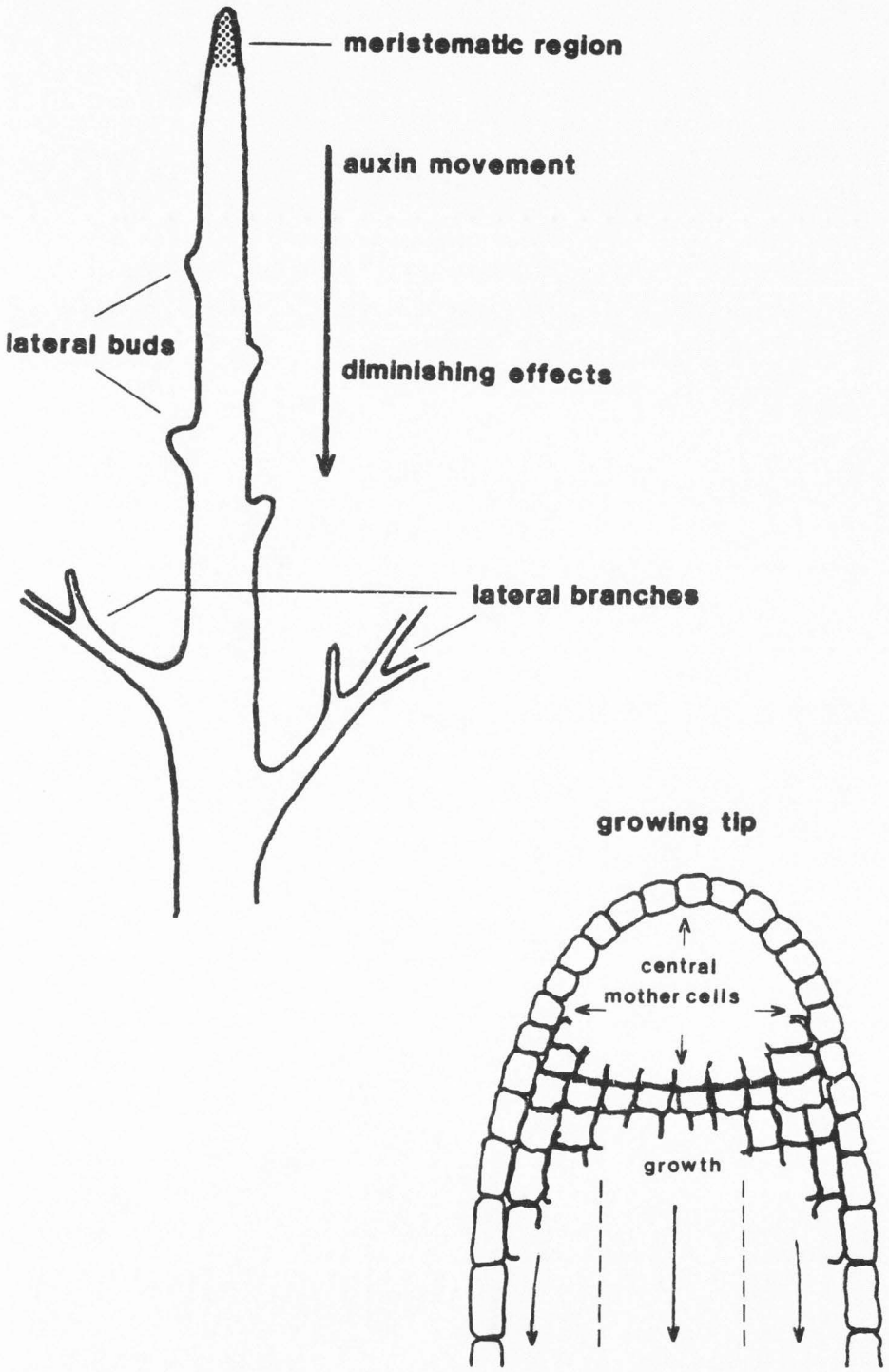


Figure 9. Schematic of apical meristem. Inhibitory effects on lateral bud growth diminish with distance from growing apex (adapted from A. Fahn, 1975).

axis of the stem (be it the trunk or branch) continues to elongate. With distance from the apical meristem (and thus diminished inhibiting influence of auxin), lateral buds can begin development into side branches or leaves. One can begin to see why there is a definite structural heirarchy in natural branching patterns. Both terminal and axillary buds either produce growth or lie dormant depending on the movement of auxins and inhibitors in the plant. Sunlight and gravity are the two dominant outside forces that affect auxin movement, and thus growth. Plants grow toward light (phototropism) and away from the pull of gravity (geotropism). Other tropisms exist, but do not have as much effect on growth and form as do the two above (Jacobs 1979).

Complexity. It is these forces which affect tree form that are the basic building blocks of an effective simulation, just as an understanding of materials and engineering is basic to an industrial CAD system. In attempting to understand growth patterns as fully as possible an overwhelming number of factors affecting a living system must be dealt with. Also, as more factors are explained, new ones appear that need answers (Rashevsky 1943). In the process of selecting pertinent growth parameters the accuracy of a particular tree growth model may be compromised. Using the minimum amount of information required to specify a certain process involves more than

enough information to make analysis quite confusing (Cohen 1967). Many times, model parameters must be based on averages of those observed in nature. While these parameters are exact in their simulated action, in nature they are often less distinct due to the many other mitigating forces at work (Fisher and Honda 1977). "Both the strength and weakness of the analytical approach to a complex problem is the necessity to simplify the system and to study only the most basic and important of its characteristics. There can be some argument as to which of the characteristics . . . are the most important" (Paltridge 1973, p.112).

Computer Models. Scientific investigations have always relied on current technology as a source of tools. Computers are an excellent example. Their speed and computational abilities suit them to very precise, orderly, analytical methodologies. Just as CAD systems help clarify and analyze jet engines, office buildings, or automobile components, the automatic plotting of experimental results has been useful in visualizing ambiguous natural processes. "For the chemist, computer graphics is probably the only reasonable method for investigating the interaction between large molecules . . . A chemist may synthesize a new molecule graphically before synthesizing it chemically, or may interactively construct an existing molecule whose properties are known but whose structure is not well

understood." (Whitted 1982, p.772)

Likewise, investigations into tree growth patterns entail a large number of variables whose interactions are quite complicated. Understanding these variables and how they interact to produce physical form is a topic of several researchers who have pursued computer modeling of the entire tree. The problem is one of overall form, and how to generate it as a recognizable tree. In his first publication, Hsiao Honda asked, "what information about the form does the gene store in it and through what process is its information represented as the form?" (1971, p.331) He went on to make a point for simplicity, one of the goals of this thesis, "the two interesting problems of the form, how to recognize and how to develop the form might be sublimated to the more general problem . . . how to describe economically the form or how to pull out the essence from miscellaneous information about the form." (Ibid, p.332)

In Honda's initial work (1971) and in subsequent work with Jack Fisher (1977, 1979), realistic tree structures were produced. Their work was aimed primarily at modeling maximum effective leaf area (figure 10). The tropical tree Terminalia catappa L. was used as a model due to its regularly repeating branching units. Throughout their work, relatively few parameters directing branching patterns were used. They found that the branching angles of T. catappa and thirty-one other tree species were very

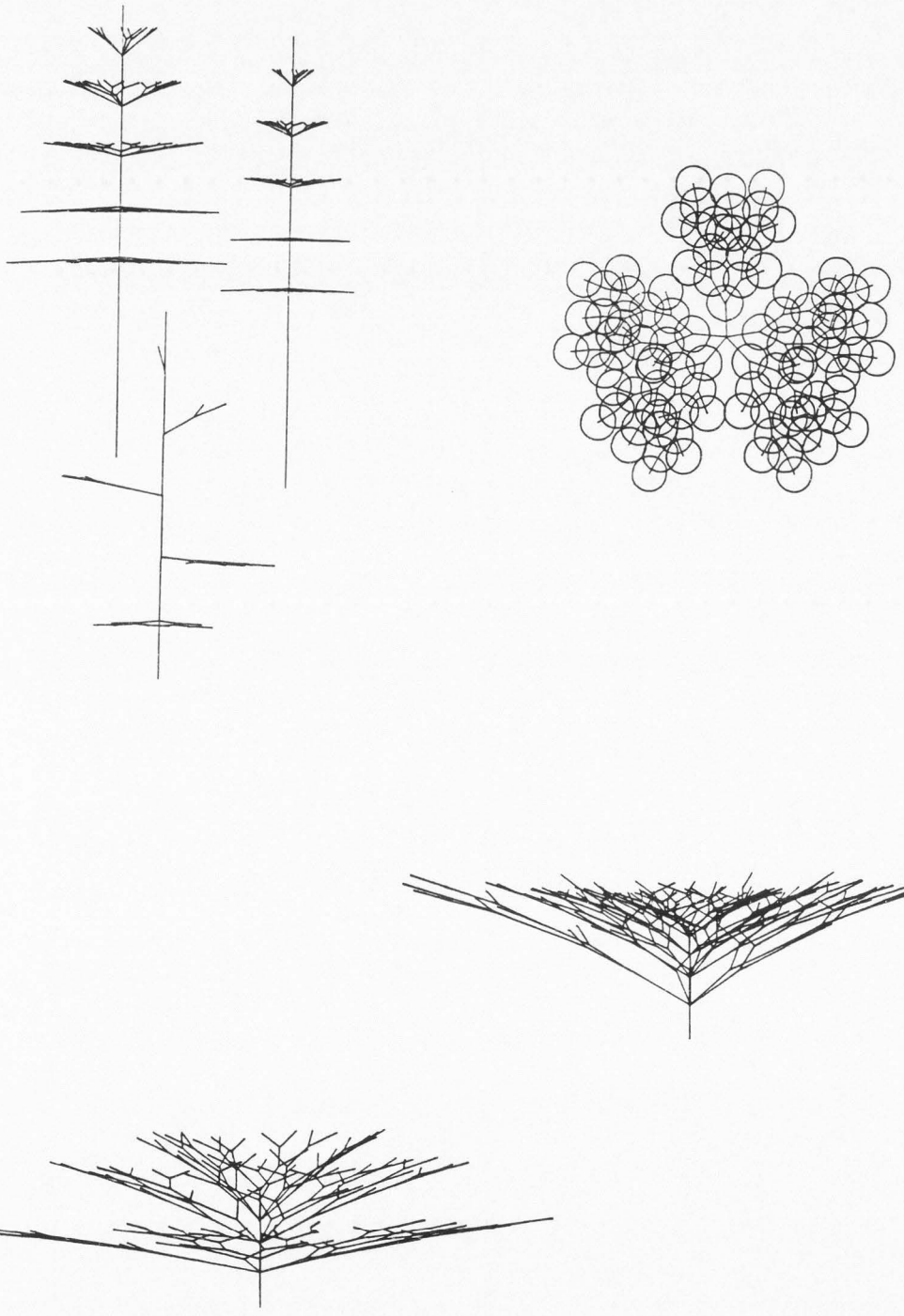


Figure 10. Branching pattern simulations by Fisher and Honda in modeling effective leaf area in Terminalia catappa (1977, p.382-3).

similar to theoretical angles producing maximum effective leaf area (Ibid 1979).

Fisher and Honda worked with branching angle and branch length ratios as variables, using the following assumptions:

1) branches were considered straight and without any girth

2) a mother-branch gives rise to two daughter branches at each branching step

3) daughter-branch length is a shortened ratio of mother-branch length

4) the daughter-branches fork in the plane of the mother-branch, and at the same vertical gradient as the mother-branch

Additionally, these branching rules were based only on averages observed in real trees (particularly T. catappa) and so cannot account for individual genetic changes or those induced by environment.

Further work by Fisher and Honda introduced a variation to the original model. Branches were whorled about the central leader at various nodes along its length. Several new parameters were introduced for this variation, but subsequent branching away from the trunk continued in the fashion of the earlier tree models described above. Fisher and Honda also made mention of the potential practicality of their work in horticulture and forestry. By pinpointing one or several parameters governing final tree form, these

characteristics might be identified in immature trees, avoiding the need to wait until they make themselves evident at or near maturity. Not only could general tree appearance possibly be predicted, but also reproductive capacity and vigor as a function of effective leaf area (Fisher and Honda 1977).

The most recent example of tree modeling at such levels of realism is reported by Masaki Aono and Tosiyasu Kunii of the University of Tokyo (1984). They approach the representation of botanical trees by first defining them in terms of distinct grammatical rules, and then as geometric objects. A major thrust of this research is a flexible model to more closely represent specific plants. The effects of wind, sunlight, gravity, and growth stage are also expressed as governing parameters. In fact, Aono and Kunii carry this type of graphic representation to a very sophisticated level, incorporating leaves, shadows, shading, and three-dimensional transformations in the graphic image.

Their approach is quite interesting not only for their quality of output, but also in that they deal with "irregularity and fuzziness, as exemplified by the algorithms for fractal surfaces" while describing objects ". . . that are inherently regular and deterministic throughout their life cycle" (Ibid, p.10). The concepts they introduce apply also to other structural problems. Just as a tree is in itself a study in optimal spatial arrangement for efficient solar interception, many other complex organizational

problems dealing with packing and spacing (housing units, circuit diagrams, etc.) are mentioned as possible application areas for this model (Ibid). The partameters employed in this model include; color, lateral branching behavior, girth, shadow and leaf arrangement, growth level, and many others. The importance of this growth model to this discussion lies mainly in the sophistication of its graphic output, and not the complex nature of its generation process and control parameters.

CHAPTER III

PROGRAMMING METHODOLOGY

INTRODUCTION

Present CAD systems offer quite sophisticated depictions of structural elements, yet the degree of realism for plant materials is still at a somewhat symbolic level. Extending the advantages of graphic realism to landscaping elements can only increase the level of communication and understanding between design professionals and their clients. The objective of the research reported in this thesis is to investigate methods for improving the representation of trees for three-dimensional display.

CONCEPTUAL APPROACH and GOALS

Emphasis is placed more on the methodologies developed rather than on the generated image. Though the appearance of the trees is a reflection of the way in which they are generated, how they are produced is a prime consideration. The work undertaken here recognizes that what is currently seen on the screen is basically transitory in contrast to the methodology driving it. It is expected and hoped that the trees themselves would be modified and improved upon in the future. The methodology then, strives to attain a degree of flexibility coupled with versatility in the resulting algorithms.

Guiding the development of workable algorithms are several basic goals discussed below that are deemed appropriate to the scale and scope of this thesis. Factors involved in arriving at these goals include; the desire to see plant materials addressed within a CAD system with as much realism as architectural elements have been; the variety of computer hardware in use in the design fields; and foremost, the recognition that development of algorithms must allow their adaptation to differing environments (program structure thus becomes secondary to program logic). Some of these goals are quite distinct from the objective of generating a satisfactory screen image. While involved in the actual structure of the tree, these goals place emphasis on the methodologies developed and how they go about their work.

Memory. An important goal is the economic utilization of the computer's active memory. The usual trade-off in this respect is that of image complexity for memory size (though memory capacity is rapidly increasing in microcomputers). A CAD system is comprised of a number of interrelated elements, each of which demands a certain amount of both permanent and active memory space. Simple tree symbols fit this organization due to their rudimentary structure. Detailed, three-dimensional trees could defeat their own value if their data structure is memory intensive. Storing an entire tree as a list of coordinate points is not feasible in this

context. However, representing a tree as a set of generation rules instead is quite practical. For example, the Boeing Aerospace Company, in developing complex terrain imaging techniques for battlefield simulations has created trees "syntactically". That is, by defining three starting points and seven rules for tree structure (Elson 1981). While their trees differ in structure and appearance from those reported here, the same type of data methodology is very important: minimizing memory requirements without compromising the complexity (detail) of the tree. Less memory requirements mean more flexibility in applying algorithms to other uses where storage space could be at a premium. This relates directly to what follows.

Portability. Another goal is to develop techniques that can be generalized to other machines. Practically all programming work in this investigation took place on an Apple II+ microcomputer because of availability, but the portability of the algorithms to other devices and uses is the desired result. Further, starting with a microcomputer aids easier translation to other machines of the same size or larger, rather than if trying to adapt algorithms written on a large mainframe computer to smaller devices.

Simplicity. A third goal is simplicity. While a less cumbersome program is easier both to understand and to adapt to other computers, another effect is just as important for

another reason. Part of this work is directed at determining variability in generated tree forms. As more and more factors affect overall tree shape, the effects of individual changes become more and more ambiguous. Simplicity in this case preserves a more direct cause and effect pathway.

Display of the resulting tree structures is also desired in order to give a quick check of the program's progress as it runs, and to make it possible to catch gross computational errors, as well as to determine the branching sequence under the influence of differing variables. Different viewing angles would also facilitate analysis. In this case the Apple screen is the primary display device, and it allows the operator to view the tree as it is being generated.

The Apple screen also allows front, side, and top views of the tree to be displayed. But other display terminals offer more versitility. Greater image resolution is afforded, for example, on devices such as the Apple MacIntosh, Televideo, Ramtek color graphics, and Evans and Sutherland terminals. The Evans and Sutherland PS-300 in particular allows a real-time detailed inspection of the tree's branching pattern while also allowing manipulation of the tree about it's X, Y, and Z axes (rotation, translation, and scaling). Data derived from tree generation on the Apple was transferred to the host computer for the PS-300 (VAX-11/780) for three-dimensional display.

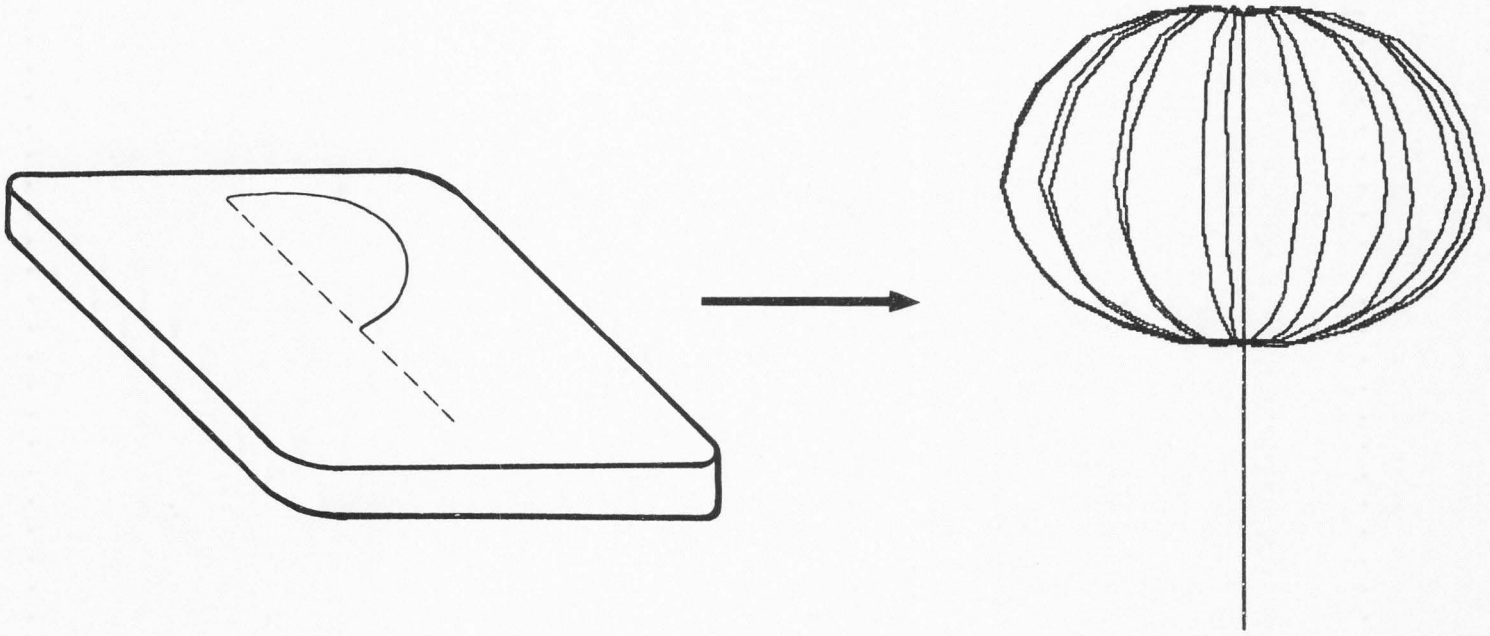
CHAPTER IV

RESULTS AND DISCUSSION

In order to improve the realism of tree images, they must be given some of the graphic attributes currently available in architectural images. The volume of space they occupy must be accounted for; height, depth, and width are preliminary characteristics contributing to realism. Adding density, a recognizable shape, plus a branching structure can also give the viewer a definite sense of realism.

WHISK TREES

The first programming step began with WHISK TREE. This produced a wire-frame image resembling a wire whisk as used in cooking (hence the term). Figure 11 shows the basic structure which consists of a number of "ribs" configured radially around a central axis (the trunk). As a first step in depicting trees in three-dimensional space, WHISK TREE started at a rather symbolic level in comparison to later trees. As such it does not have a true branching pattern but rather defines the shape and volume of trees. It shouldn't be assumed that this tree's utility has been rendered obsolete by subsequent tree models. It's unique way of representing trees could be much more appropriate under certain graphic conditions than a more dendritic tree pattern. It also served as a necessary starting point for interfacing output with the Evans and Sutherland PS-300



Figures 11 and 12. Wire-frame tree structure generated by Whisk Tree program (right). Tree canopy profile entered via digitizing tablet (left) which will be rotated about its central axis to generate a three-dimensional image.

terminal.

Logic. The program logic calls for digitizing only one of the ribs rather than the entire tree. This is done via the digitizer, and the rib thus digitized represents the profile of the tree canopy. Digitized profiles can serve as templates for creating different trees, and these profiles thus constitute short data lists to be used by the program's set of rules (rotation equations) to build a tree (figure 12).

The Whisk Tree program builds the tree by rotating this profile around the trunk. By recording the profile at certain angles from the origin, any number of ribs may be symmetrically arranged to form the canopy. A user selected density factor is also built into the program (Appendix A). By choosing the number of ribs, the density of the tree canopy can be varied (figure 13).

As each coordinate point (of X, Y, and Z values) is generated it is stored in an array. At the completion of the calculations, this array is used to display the tree on the screen one rib at a time. By storing the canopy ribs in an array, the coordinate points can be retrieved in a format compatable with either the Apple's screen plotting logic, or with that of the Evans and Sutherland. By assigning the tree a file name, the user can opt to save a particular tree as a data file, which is structured to function as a data file for the Evans and Sutherland (Evans and Sutherland Corp. Version

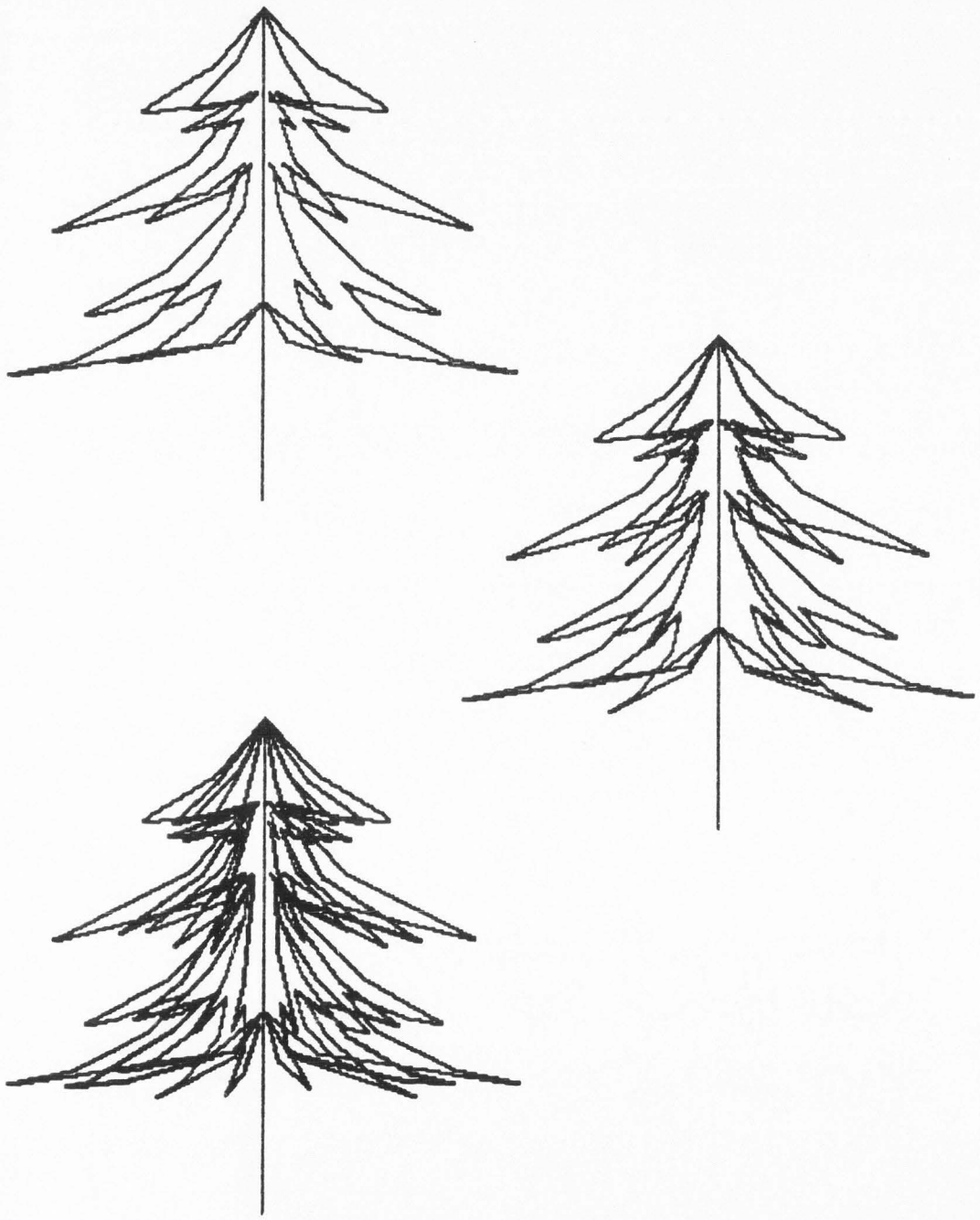


Figure 13. Density of the tree canopy as a function of the number of "ribs" arranged around the central axis.

P3.VØ1).

Mathematics. To achieve the three-dimensional spherical tree canopy, the basic trigonometric operation is the rotation of a point about a center.

It is helpful here to begin with a description of the tree canopy profile with which the digitizing program (Appendix A) is working. It is a curve, of some particular shape comprised of a series of points (figure 12). As the digitizing pen is moved across the digitizing tablet, it records coordinate points at regular intervals. A slow pen movement results in a large number of points along the profile curve (and consequently longer rotation computation time), and a faster pen movement results in fewer points. It is these points, not the entire curve itself, that the program rotates around the tree axis.

The rotation of each point employs the following general equations:

$$X(J)=R \times COS(I)$$

$$Z(J)=R \times SIN(I)$$

Where X(J) and Z(J) are the X, Z coordinates of the new point. The Y coordinate, denoting the vertical axis (height) is not involved in the rotation operation since it is constant for each point. R is the distance the point lies from the tree axis (circle radius); and I is the angle of

rotation from one rib position to the next (this angle depends on how many ribs have been selected, dividing 360 degrees into that many arcs -- see figure 14). The radius (R) remains constant for each point as it rotates about the axis, but varies from point to point on the profile. The angle (I) also remains constant for all points in the entire canopy. The X and Z coordinates are stored separately in two-dimensional arrays, while the Y coordinate is stored in a one-dimensional array.

Array Manipulation. Each point on the profile is rotated through the entire 360 degrees of the canopy before the next point is dealt with. The points generated during the rotation of an original profile point are stored across each row of the array, column by column. With each new profile point the program drops down one row in the array (figure 15). Retrieval for display occures in the opposite manner; each rib consists of a separate array column (the number of column elements corresponds to the number of points recorded in the profile curve, while the number of row elements corresponds to the number of ribs occuring about the axis).

To this point, the X and Z axes have been discussed. The Y axis isn't involved in the rotation calculations. In the case of WHISK TREE the Y axis is considered to define the height dimension of the tree. This was done in keeping with general screen setups where X is the horizontal axis, Y is the vertical axis, and Z is apparant depth (into the screen).

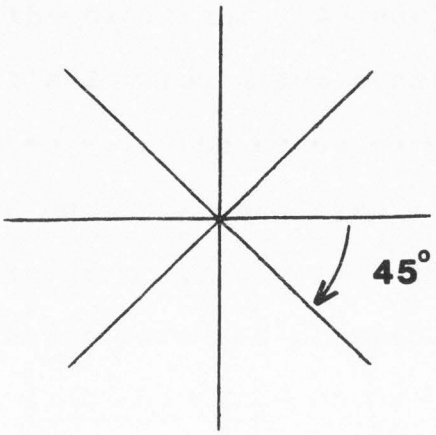


Figure 14. Top-view of Whisk tree showing arrangement of ribs around axis. In this case, eight ribs 45° apart.

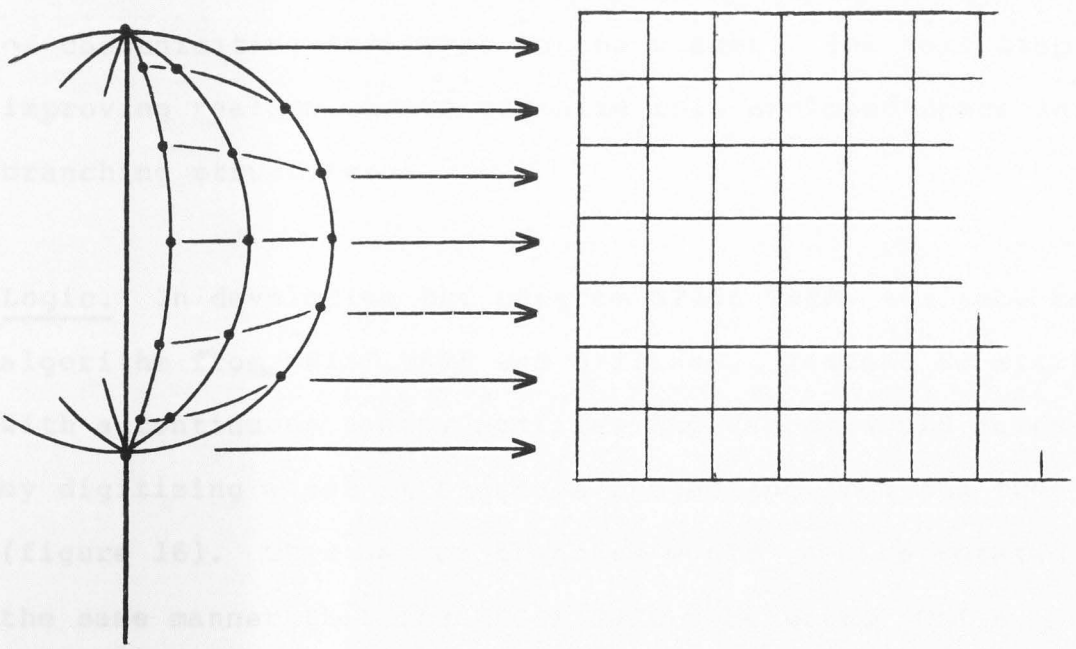


Figure 15. Side-view of tree canopy. The values generated during the rotation of each point in the profile are stored across the array.

The Y value is different for each set of points in the canopy and represents the relative position of the original profile point as read from the digitizer. As each point is rotated about the center, it's Y value stays constant. The value of this axis was included with the other two axes before storage in the array. The algorithms were also modified to show different views of the same tree (side, front, and plan views) by altering which axes are plotted on the Apple screen.

STICK TREES

The logic as developed for the program WHISK TREE offered one way of depicting trees by defining the volume of space occupied: shape, size, and surface density are major methods of communicating tree type to the viewer. The next step in improving realism was to organize this enclosed space into branching structures.

Logic. In developing the program STICK TREE, the same basic algorithm from WHISK TREE was utilized. Instead of starting with a continuous canopy profile, the canopy would be defined by digitizing a set of branches projecting from the tree axis (figure 16). This set of branches would then be rotated in the same manner that the WHISK TREE ribs were. While basically a variation of WHISK TREE (substituting a data list of branches rather than a profile), STICK TREE offered more in the way of realism and flexibility.

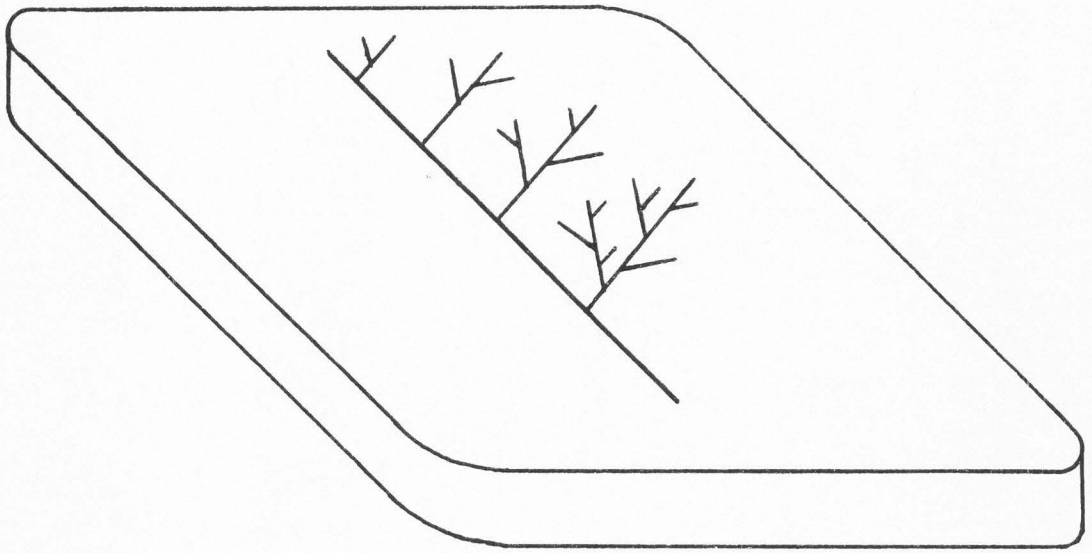


Figure 16. Stick Tree canopy profile as entered on digitizing tablet.

First, replacing ribs with branches gives the viewer an object more closely resembling real trees. Second, since the user is digitizing each individual branch, more detail and variability could be added. As with the previous tree model though, each rib would still be identical to the others around the tree axis.

As a successor to WHISK TREE, this program added more time to the digitizing process though it promised a more realistic looking tree. Extra user input was required to indicate where each branch ended and a new one began. Also, the branching structure had to be worked out on paper ahead of time, and each point (branch origin, branching point, and branch tip) had to be digitized in the proper order. While this is feasible from a programming standpoint, the process was getting further from the goal of simplicity. The relatively simple canopy profile as a starting point for tree generation was becoming much more complex and cumbersome.

Before the work on this tree model reached completion, the work of Hsio Honda and Jack B. Fisher came to light in the literature (Honda 1971, Fisher and Honda 1977, 1979). Their research offers a more streamlined approach than the course the STICK TREE program was taking.

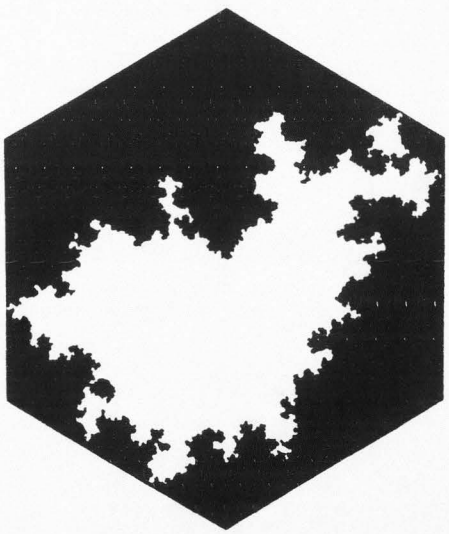
FRACTAL TREES

The work reported by Fisher and Honda (1971, 1977, 1979) represents a major step in constructing a tree solely from a set of rules without requiring an initial data list. The

only other input needed are several user defined control parameters.

The name "FRACTAL TREES" comes from the term FRACTAL coined by mathematician Benoit Mandelbrot in 1975 (Mandelbrot 1977). Fractals are a family of shapes that are hard to analyze mathematically. For example, figure 17 shows structures that seem to be randomly patterned, yet each one is structured in such a way that a small part resembles the whole. Fractals offer a way to describe natural patterns and processes in a quantitative manner (Mandelbrot 1983). The tree generation method introduced in this section represents some important characteristics of fractal geometry. Discussion of fractals at this point is necessary to the reader's understanding of the implications of FRACTAL TREE concepts, and to promote possible future consideration of an area of inquiry rapidly being applied in many theoretical fields. Fractals introduce possibilities that will be examined in the concluding chapter. This discussion of fractals serves as an overview to both illustrate their relationship with these trees and establish grounds for further discussion. The major concept covered here is the idea of self-similarity.

Benoit Mandelbrot's interest in fractals grew out of his work 25 to 30 years ago with random perturbations, or noise, in data transmissions by telephone. He found a way to describe these chance fluctuations as fractal sets (McDermott 1983). A good example of the fractal characteristic of



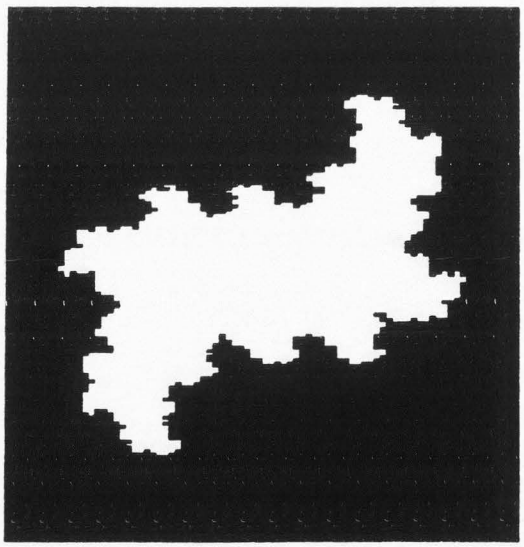
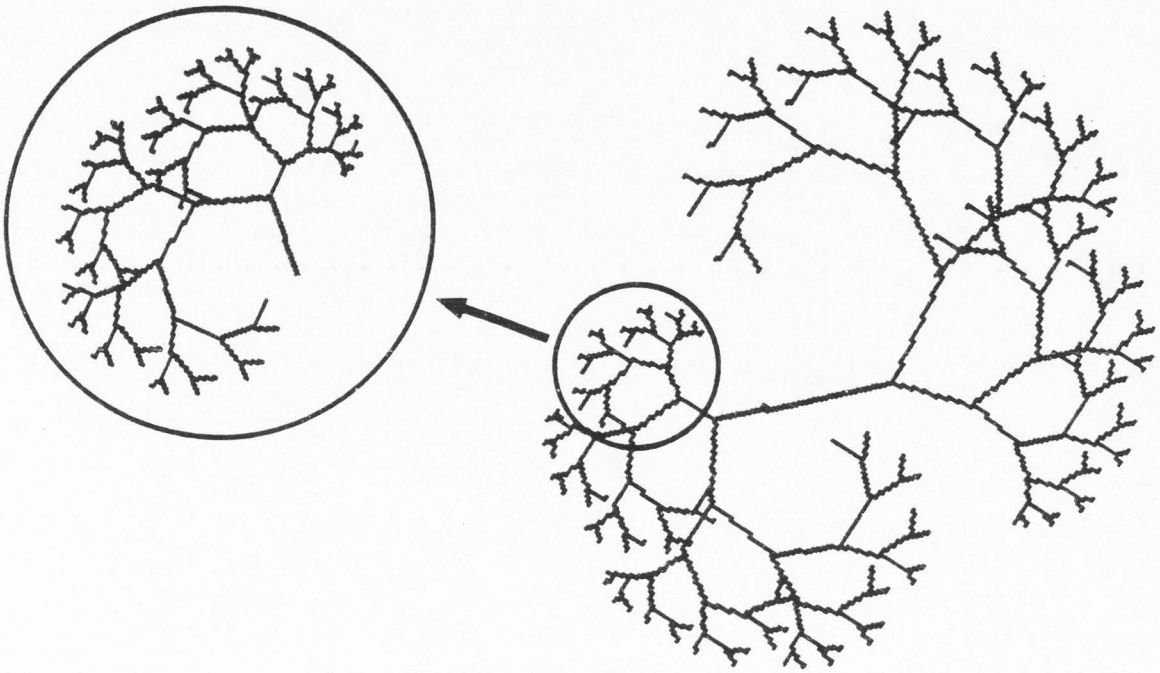


Figure 17. Fractal shapes of apparent random structure, yet with a definite structure in which smaller and smaller fragments resemble the whole (McDermott 1983, p.110, Mandelbrot 1983, p.231, Sorensen 1984).

self-similarity comes directly from the FRACTAL TREE structure itself. In his book Fractals: Form, Chance, and Dimension, Mandelbrot (1977, p.112) wrote of fractals, "Magnify one again and again and more detail always emerges. Just as a twig resembles a branch and a branch resembles the tree, each part of a fractal is like the whole." The same tendency can be observed in generated tree structures (figure 18). Whether looking at the second branching step or the twentieth, the same pattern is evident. The overall pattern of the tree is reflected in successively smaller subdivisions. If the tree generation is carried out until the smallest branches along the periphery are discernible as only a "fuzzy" border, magnification of one portion of this fringe would again reveal the same basic structure of the entire tree. In looking at just one isolated section of the pattern, it would be difficult to determine the magnification level of that particular view.

Logic. FRACTAL TREE is based on a recursive-type algorithm which "grows" a branching structure outward from an origin. The structure consists of daughter-branch segments (of lengths R_1 and R_2) which arise from a mother-branch segment, and diverge at angles θ_1 and θ_2 ($THETA_1$ and $THETA_2$). As growth progresses, each mother-branch produces two daughter-branches (figure 19). Each of these daughter-branches becomes in turn a mother-branch, giving rise to two more daughter-branches of it's own. This growth



Top view of fractal tree generated by using $\theta_1 = -45$, $\theta_2 = 20$, $R_1 = .76$, and $R_2 = .6$

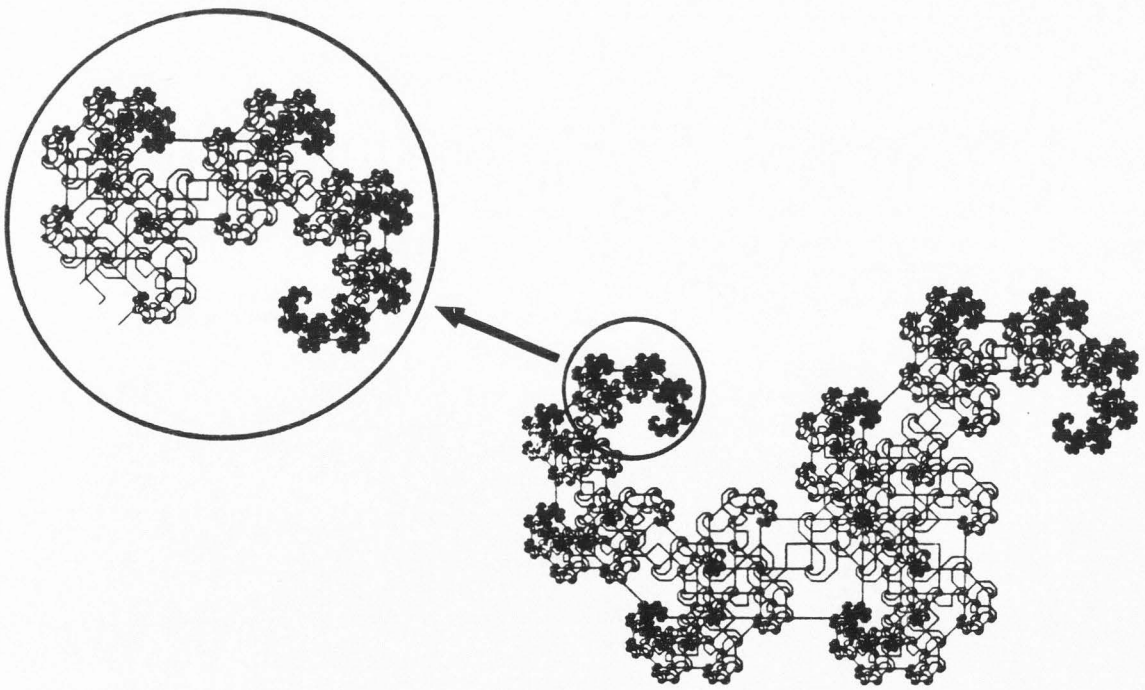


Figure 18. The fractal characteristic of self-similarity as exhibited by the tree structure (top) and fractal shape (bottom), in which smaller elements mirror the overall structure (McDermott 1983, p.117).

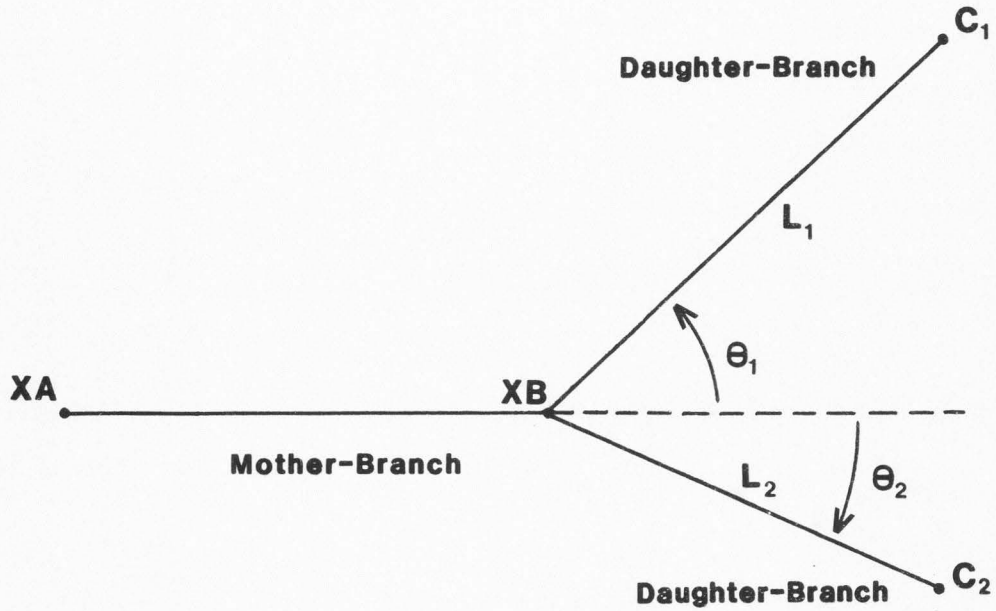


Figure 19. Schematic branching process. Mother-branch (XA-XB) gives rise to Daughter-branches ($XB-C_1$, $XB-C_2$) of lengths L_1 and L_2. Divergence angles of θ_1 and θ_2 are measured from projected axis of Mother-branch.

is produced by Honda's general equations (1971), and is governed by R_1,R_2 and θ_1,θ_2.

One of the above parameters is represented by the variable θ. This is the angle of divergence of a daughter-branch from the direction of the mother-branch. Each mother-branch gives rise to two daughter-branches, each of which can have different values for θ. For example, if θ_1 is -20 deg. and θ_2 is +20 deg., the total angular displacement between them is 40 deg. (figure 20). Should both divergence angles be the same (ie. -35 deg.), one will plot on top of the other giving the impression of only one branch (figure 21).

The other user-defined parameter is the length variable R. R defines the ratio of daughter-branch length to mother-branch length. An R value of 0.5 indicates a daughter-branch one-half the length of the it's mother-branch. In this case, as branching proceeds away from the origin, each branch will be shortened by one-half of it's predecessor's length. As with θ, there are also two values for branch length. R_1 and R_2 allow independent ratios to be applied to each of the two daughter-branches. Giving the value of zero to either R_1 or R_2 causes one branch to always be of length zero, producing the same apparent effect as making θ_1 and θ_2 identical (figure 21).

Mathematics. Hsio Honda gives three equations used in the branching process (Honda 1971, p.334):

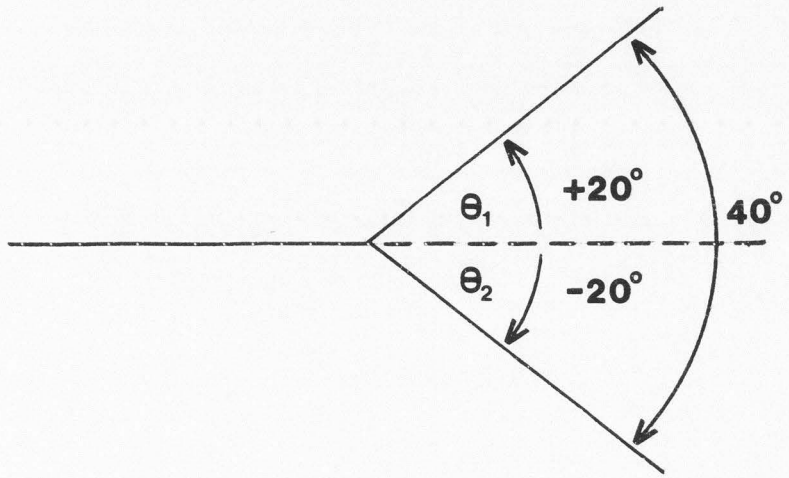


Figure 20. If θ_1 and θ_2 are of opposite signs (-20° and +20°) their values are added to form the total angle between the two Daughter-branches.

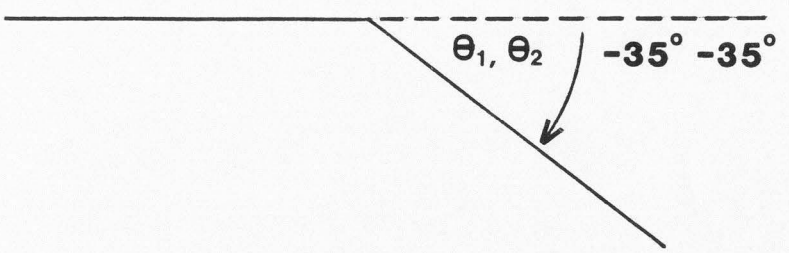


Figure 21. If both θ_1 and θ_2 are of the same sign and displacement (-35°), both Daughter-branches plot in the same position giving the impression of only one branch.

$$X=XB+R(UxCOS\theta-LxVxSIN\theta)/\sqrt{U^2+X^2}$$

$$Y=YB+R(VxCOS\theta+LxVxSIN\theta)/\sqrt{U^2+V^2}$$

$$Z=ZB+RxWxCOS\theta$$

Where:

$$U=XB-XA$$

$$V=YB-YA$$

$$W=ZB-ZA$$

And:

$$L=\sqrt{U^2+V^2+W^2}$$

θ again is the angle of divergence, and R is the branch length ratio. X, Y, and Z represent the three-dimensional coordinates for each generated point. U, V, and W define the displacement of the point in each axis, while these are then used to calculate L, which is the length of the new branch. Figure 22 illustrates the variables XA and XB (or YA,YB or ZA,ZB depending on which axis is being dealt with). The line XA - XB defines the length of the mother-branch (in the X axis). D_1, D_2, D_3, and D_4 can be used to designate the end points of daughter-branches XB - C_1 and XB - C_2. Point C_1 is generated using the above equations with θ_1 and R_1. Point C_2 is then generated using θ_2 and R_2. In the production of the next generation of branches, XB - C_1 becomes the new mother segment and C_1 - D_1 would be one of its daughter-branches.

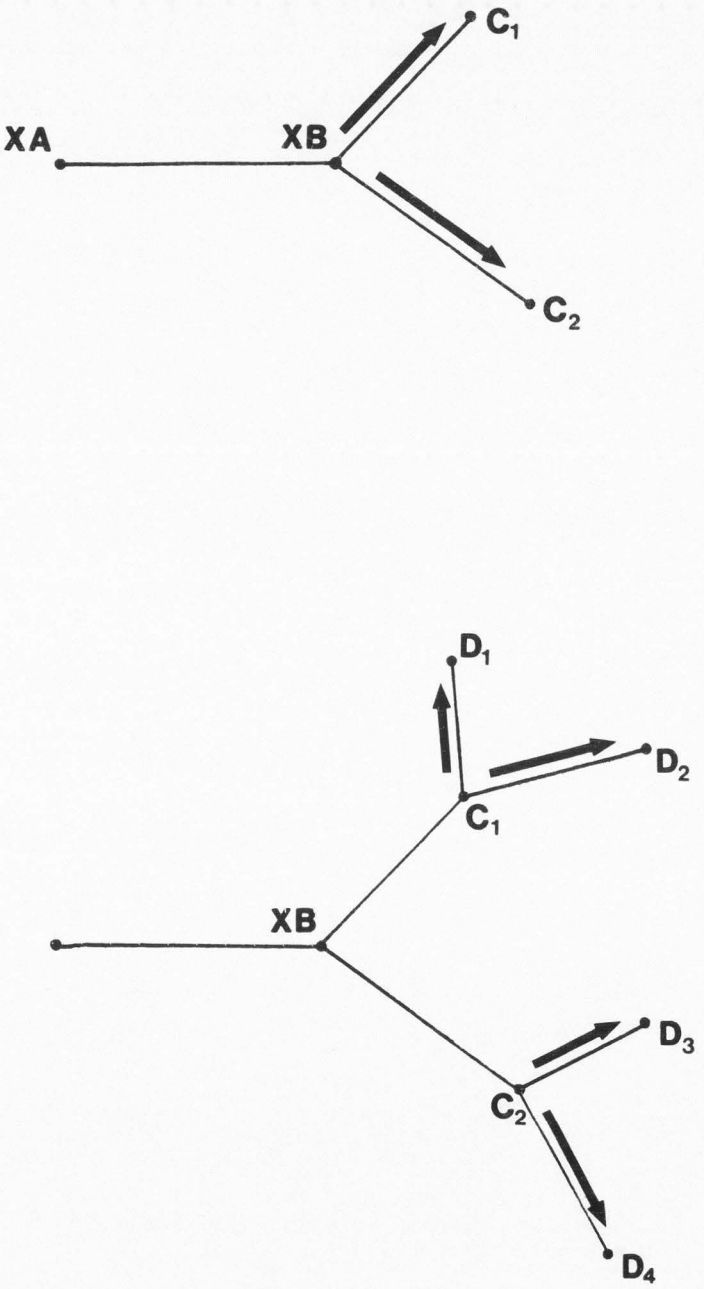


Figure 22. Progression of points in branching process. XA-XB produces C_1 and C_2. XB-C_1 produces D_1 and D_2, while XB-C_2 produces D_3 and D_4.

The same happens with the segment XB - C_2, and so forth through the tree-building process.

Array Manipulation. A problem encountered with this type of branching process is that as branching continues, the number of branches generated increases as a geometric progression. Eight mother-branches produce sixteen daughter-branches, which in turn produce thirty-two more branches. Assuming that this program is to represent a set of tree generating rules rather than produce a long data list, there is no need to save the coordinates of all branch segments produced. The program must preserve only those points necessary to generate the next set of branches. Once used they are no longer necessary, and the space they occupied in memory can be used for succeeding coordinates. This is accomplished by toggling between two rows of an array as each generation of branch coordinates is produced. With each toggle operation, the number of occupied array elements doubles (figure 23).

This process continues until the array is completely full. One benefit of this method is that since the terminal branch tips on the tree won't be used to produce another set of branches, there need not be any array space dimensioned for them. Thus, the program can produce as many terminal branches as desired, as long as they are plotted on the screen. This offers a chance to further alter the final appearance of the tree by modifying the density of the branch tips.

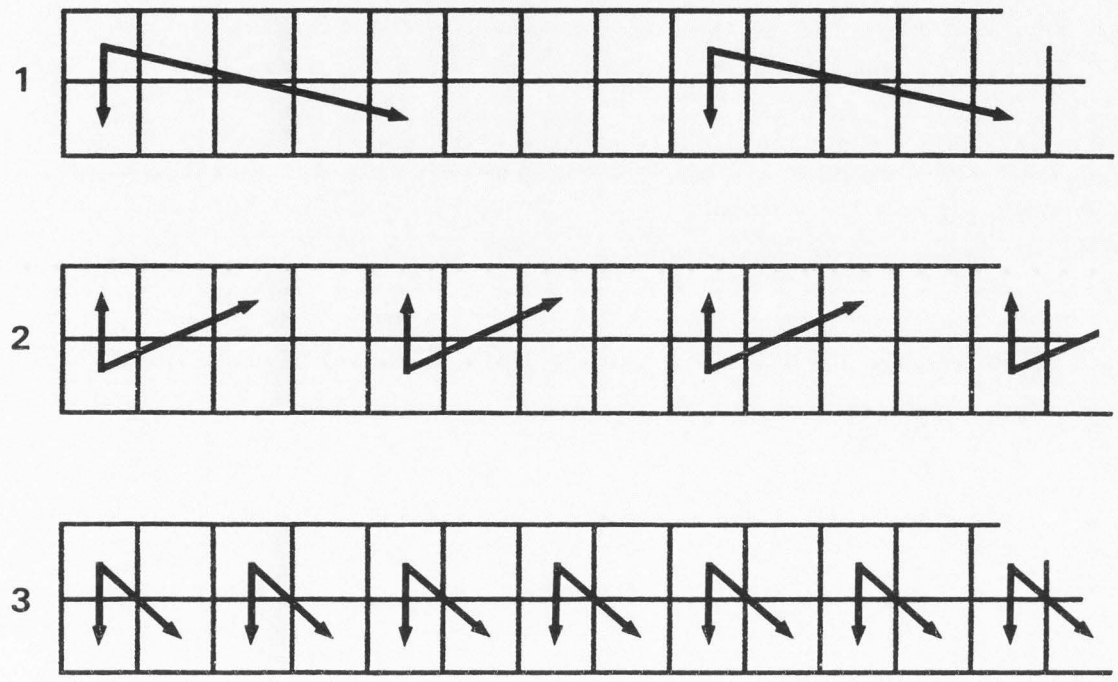


Figure 23. Array toggling process wherein the same two array rows are used each time. With each cycle the number of occupied array elements doubles as the process moves back and forth between rows.

Figure 24. Temporary storage of initial point used in the branching process. "A" is replaced by "B" which will then be replaced by "C" and so on.

Another aspect of the array toggling process is that in the equations it takes two points (A and B) to produce a third (C). The line A - B corresponds to the mother-branch, and the line B - C corresponds to one of the daughter-branchs (figure 24). Only the coordinates represented by B and C are stored in the working array. The value for A is stored in another temporary memory location. When B is then used with C to produce D, B is stored in this temporary area until the next generation occurs, at which point it is replaced again. The process was handled in this manner because it is less complicated to toggle two array rows rather than three.

As each branch segment is generated, it is plotted directly on the computer screen. In the case of Evans and Sutherland display, it is a simple matter to have the program save the coordinates in a disk file at the same time. Transfer to the University's VAX-11/780 mainframe is accomplished via phone line. Once stored in the VAX, this data file can be accessed for display on the Evans and Sutherland graphics terminal. Whereas the FRACTAL TREE program has the flexibility to display the trees in either the X,Y plane, the Y,Z plane, or the X,Z plane on the Apple screen, the Evans and Sutherland can rotate and move all three planes in real time with considerable image resolution. It handles three dimensional objects in such a way as to give the viewer the impression that the object projects back into the screen (this it accomplishes by lessening the light intensity of points and lines as their apparent distance from

the viewer becomes greater).

CHAPTER 5

CONCLUSIONS AND APPLICATIONS

INTRODUCTION

Computer-aided design has gained a firm foothold in the design fields for simulating the physical environment. It facilitates the use of a wide range of design elements, including plant materials to a certain degree. This thesis has been concerned with the somewhat primitive nature of plant images available with CAD systems. The line of inquiry that has been pursued here centers on one aspect of the CAD process; the depiction of trees in three-dimensions on a computer, and attempts to address it within the context and vocabulary of landscape architecture. Questions and possibilities outside the direct objective of this study have also been raised, and will be discussed according to merit.

The major premise operating in this thesis is the importance of developing a working methodology for tree depiction. Basic to understanding this methodology is the context of its approach to the problem. Originating as it does in the realm of landscape architecture, and not in plant morphology or computer science, it reflects the role of landscape architecture as a field concerned with implementation, one that pulls together the research of many people in different disciplines. In this capacity a valuable aspect of this work quite apart from it's tree generation conponent is that it compiles research and information from

disciplines not usually in contact with each other. This chapter draws on this information for conclusions as well as in formulating future application possibilities.

CONCLUSIONS

The scope of this thesis is limited to developing a methodological framework within which three-dimensional tree images can be generated. This methodology derives its functional value from the goals of simplicity, portability, and conservation of memory (Chapter 3). It's basic working objective is that of driving a tree generation process involving a minimum of time and extraneous information. It functions in that capacity by overcoming the usual need for storing long data lists. This is important in integrating these trees into a CAD system where many other elements compete for space. While memory size is becoming much less of a limitation for microcomputers, the many kinds used for CAD still impose a concern for storage space. The type of recursive algorithm developed here operates with low memory requirements. This feature also promotes the portability of the algorithms to other devices, another major goal. Considering that the hardware CAD systems can operate on vary in size and capabilities, features that limit the algorithm's use to one type of machine must be avoided. The tree generation algorithms, being written on an Apple II+ microcomputer can translate to comparable microcomputers as well as to main-frames. If originally written on larger

machines, transfer in the other direction would be considerably harder.

The goal of algorithm simplicity is also addressed within this methodology. While this is a factor in both low memory requirements and algorithm portability, it also relates directly to the continuing developmental aspect of this work. Limiting the number of variables which act to produce the physical appearance of the trees makes their influence on form more obvious. Structural variety can be more easily analyzed under these conditions. It is also noted that no methodological approach can be absolute in its usage. A certain amount of flexibility must be maintained as requirements and applications change. This is especially true when dealing with the present state of computer technology.

With a major emphasis placed on methodology itself, programming expertise was not stressed beyond the limits of the Basic language. A concerted emphasis on actual programming techniques is seen as a necessary component to serious applications work. An assembled machine language could considerably speed the present tree generation process.

APPLICATIONS

Methodology. The involvement of landscape architects with CAD systems is increasing, and an enhancement of the representation of trees in this respect will likely occur over time. A useful methodology for handling this can

promote a smoother transition. The work of Fisher and Honda (1971, 1977, 1979) and that of Aono and Kunii (1984) in producing graphically sophisticated trees can have a significant impact on CAD systems, not only in their ability to utilize realistic trees in the design process, but in broadening their area of application. The tree algorithm's place in a CAD system becomes one of generating an image when needed without concerning itself with the retention of much data (a technique that could be valuable in other situations where a calculation process flows in the same general manner). A scenario might be that of a user developing a landscape design for a building complex. Assuming the building structure already exists in the CAD system, the landscape architect would proceed by defining the design with walkways, paving patterns, the placement of walls, benches, planting areas, etc. At the point of placing trees in the plan, the system would access a tree generation algorithm. By defining a few parameters, various trees could be generated on the plan wherever desired. Adding or moving a particular tree type would again mean generation via the same parameters. A tree is simply generated each time one is needed.

The operational value of this process is intrinsic to any future applications: whether six separate tree types are to be used or sixty separate types, the CAD system is working with the same amount of "data" -- the tree generation algorithm. The trees, by occupying space only in the

system's screen buffer don't change the storage requirements, giving the user a good deal of latitude in terms of quantity and complexity of tree forms.

Whisk Trees. The whisk-type tree structure discribed in Chapter IV was discussed as an intermediate step in producing a tree image. These trees can be applicable to design situations where their form would integrate well with the graphic style being used. While a digitizer is necessary in their generation, the entire tree needn't be stored. The canopy profile, coupled with its rotation routine can be considered to constitute a set of rules for tree generation. To streamline this set of rules, it may be desirable to standardize the number of points digitized along the canopy profile, no matter what its shape. This would avoid the large number of points observed in complicated profiles. A number of different tree types could be represented by a series of these stored profiles.

Other changes could be made in the way these trees are constructed to enhance their variability. For example, the existing longitudinal ribs could be augmented with latitudinal rings around the circumference of the canopy (figure 25). A display of this type would only require connecting the points in the canopy in a different manner. A better impression of branches might also be achieved by plotting radiating "spokes" from the tree trunk out to each point in the canopy (figure 26). This may offer a better

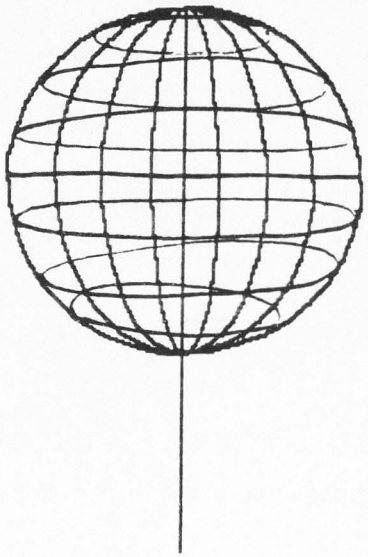


Figure 25. Example of addition of latitudinal rings to existing structure of Whisk tree.

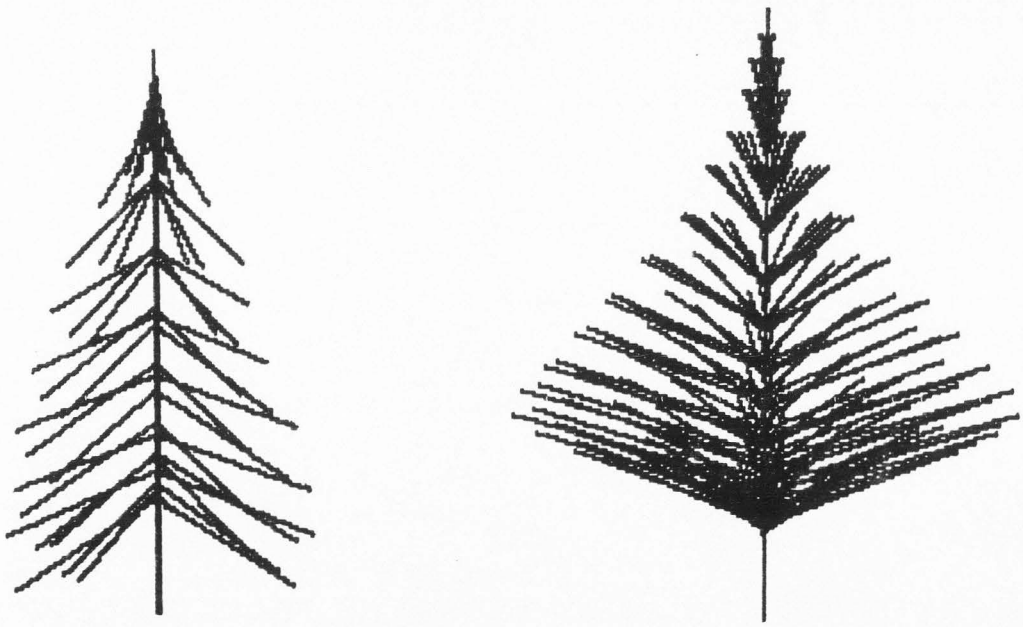


Figure 26. Example of alternate plotting procedure. Plotting lines from central axis out to points at edge of canopy. (See Appendix D).

method in modeling conifers such as pine, fir, and spruce trees .

Stick trees. The type of structure refered to as Stick Tree was not carried to completion as a tree model. It's main attribute is that it could represent very unique branching structures since these structures are entirely user-defined. However, the intensive digitizing process necessary is a major drawback to its practicality. At this point it isn't seen as a viable line of pursuit.

Fractal trees. With no need for digitizing, and the modest number of control parameters, this model offers the most promise for integrating trees into a CAD system. The methodology as presented could also be readily applied to other plant materials whose branching structure is an identifying characteristic.

The number of branching operations at each node is one area for modification where the density and character of the tree could be altered considerably. With an improvement in generation speed, a much fuller tree could be achieved. The number of daughter-branches arising from each mother-branch could be increased, decreased, or held constant as branching proceeds away from the trunk. For example , the impression of needle-like leaves could be created by ending each terminal branch with not two daughter-branches but perhaps ten or fifteen. It could also be possible through additional

generation routines to introduce a shape resembling a leaf at the branch tips.

For a more dense appearance, a side and front view of the tree could be plotted simultaneously at the same point on the screen. Additionally, the introduction of random factors could further vary the resulting branching patterns. A modification of this sort could generate random values for either branch length (R), divergence angle (θ), or both.

A characteristic of plants that landscape architects must deal with is growth. The effect of a planting design at maturity can be quite different than when first installed. The ability to view trees in the design at various stages of growth would be a useful modification. A currently available program, LA CAD (Itame, Gimblett, Brooks 1984), offers the option of incorporating plant growth, though in only two-dimensions (in plan view). The planting design can be viewed at specified intervals over time, and the visual and functional effects evaluated. The Evans and Sutherland PS-300 discribed earlier could also simulate the above with it's ability to scale the size of images. The use of a color terminal opens additional possibilities for adding to realism by using color as an identification characteristic. The major limitation to be considered here would be in further complicating the algorithms.

Fractal geometry has been discussed in Chapter four in it's application to the trees developed here. A fractal shape repeats itself at different scales, and can be

described mathematically (Sorensen 1984). As shown earlier, these trees exhibit the fractal characteristic of self-similarity. Their physical location in the landscape could also become a function of fractal geometry. This might be an area of application in vhrual analysis. Nickerson and Arneson (1981) have discussed the role of computers in simulating visual impacts of proposed changes in the landscape (see chapter two). Further, fractal geometry has been applied to vegetation patterns for ecorxstem modeling in the Okefenoke Swamp of Georgia (Peterson 1983). Just as Rashevsky (1943) employed mathematics to quantify animal locomotion (Chapter two), fractals offer potential in simulating not only the physical form of trees and other plants, but also their distrubution patterns in nature.

Taking the application of fractal concepts further in this context, it may be possible to fractally describe the predominant topographic forms present in the landscape as well (mountains, cliffs and plateaus, rolling hills, etc.). Similar applications can already be found in the generation of synthetic scenes for computer graphics and the film industry (McDermott 1983).

Further work with the present methodology could possibly result in an even more compact algorithm format, and more variable tree images. The basic methodological approach of substituting a set of generation rules for a long data file facilitates future applications. The concept of quickly generating a tree image whenever and wherever needed stands

as a major step in this direction, whether or not it is applied in it's present form.

REFERENCES CITED

Aono, Masaki and Kunii, Tosiyasu L. 1984. "Botanical Tree Image Generation." IEEE Computer Graphics and Applications May: p.10

Architectural Record 1980. "SOM's Computer Approach." Mid-August:84-87

Bidwell, Roger C. S. 1979. Plant Physiology. New York: MacMillan Publishing Co.

Breeden, J. Brooks 1984. "ROADEO." Landscape Architecture May/June:96-98

Business Week 1982. "CAD Redraws the Architect's Job." 15 June: 134-136

Clay, Grady 1980. "Towards a Computerized Practice." Landscape Architecture November: 601

Cohen, Dan 1967. "Computer Simulations of Biological Pattern Generation Processes." Nature 21, October: 246

Dietsch, Deborah 1982. "The First Interiors CAD/CAM Survey." Interiors November: 12

Elson, Benjamin 1981. "Boeing Studies New Imagery Technique." Aviation Week and Space Technology 14, December: 78

Evans, Brun 1984. "Computers In Landscape Practice." Landscape design June: 43

Fisher, Jack B. and Honda, Hisao 1979. "Branch Geometry and Effective Leaf Area: A Study of Terminalia -- Branching Pattern. 2. Survey of Real trees." American Journal of Botany July:645-655

Ibid 1977. "Computer Simulation of Branching and Geometry in Terminalia (Combretaceae), A Tropical Tree." Botanical Gazette December: 377-384.

Fullenwider, Donald R. and Lefever, James P. 1981. "Computer Graphics and the Practice of Architecture." IEEE Computer Graphics and Applications October: p.19.

Hapgood, Fred 1982. "Seekers of the Golden Form." Science Digest May: 46

Honda, Hisao 1971. "Description of the Form of Trees By the Parameters of the Tree-Like Body: Effects of the

Branching Angle and the Branch Length on the Shape of the Tree-Like Body." Journal of Theoretical Biology 31: 331-338.

Jacobs, William P. 1979. Plant Hormones and Plant Development. London: Cambridge University Press.

Killpack, Charles 1982. "LANDSAT: Innovating Landscape Architecture." Landscape Architecture Technical Information Series 6 April

Lindenmayer, Aristid 1968. "Mathematical Models for Cellular Interactions in Development." Journal of Theoretical Biology 18: 300-315.

MacDougall, E. Bruce 1983. Microcomputers In Landscape Architecture. New York: Elsevier Press.

Mandelbrot, Benout B. 1983. The Fractal Geometry of Nature. New York: W.H. Freeman and Company.

Ibid 1977. Fractals: Form, Chance, and Dimension. New York: W.H. Freeman and Company.

McDermott, Jeanne 1983. "Geometric Forms Known As Fractals Find Sense In Chaos." Smithsonian Magazine December: 110

McMahon, Thomas A. 1975. "The Mechanical Design of Trees." Scientific American July: 93

Milliken, Barry 1983. "The Personal Challenge of CAD." Architectural Record March: p.41.

Nickerson, Devon and Arneson, Mary 1981. "The Preview -- Quicker By Computer." Landscape Architecture November:738

Opatowski, I. 1944a. "Part I: The Trunk." Bulletin of Mathematical Biophysics 6: 113-118.

Ibid 1944b. "Part II: The Primary Branches." Bulletin of Mathematical Biophysics 6: 153-156.

Paltridge, G.W. 1973. "On the Shape of Trees." Journal of Theoretical Biology 38: 111-137.

Pandey, R.B. 1984. "The Notion of a Strange Fractal Phenomenon -- Disorder Induced Walk." Journal of Physics A: Mathematics and General 17: 1551-1554

Peterson, Ivars 1983. "Fractal For Modeling Ecosystems." Science News 11, June: 381

PS 300 User's Manual - Graphics Firmware Version P3.V01. 1982. Salt Lake City, Utah: Evans and Sutherland Computer Corporation.

Rashevsky, N. 1943. "Outline of a New Mathematical Approach to General Biology: I." Bulletin of Mathematical Biophysics 5: 33-47.

Ibid 1943b. "On the Form of Plants and Animals." Bulletin of Mathematical Biophysics 5: 69-73

Rouse, Nancy E. 1984. "Drafting On Personal Computers." Machine Design 23, August: 88

Stasiowski, Frank A. 1982. "Computer-Aided Drafting." Progressive Architecture May: 183-187

Steinitz, Carl 1982. "Landscape Planning and Computer Technology: A Developmental (and Personal) Overview." Unpublished paper presented at international Symposium on Landscape Information Systems, Bonn-Bad Godesberg, Germany 9, March

Stitt, Fred A. 1982. "Computer For the Smaller Office." Architectural Record February: 47-51

Sutherland, Ivan E. 1970. "Computer Displays." Scientific American June: 56-81

Teicholz, Eric 1983. "Can Computer Aided Drafting Be Effective and Affordable for the Small Firm?" Architectural Record February: 37

Whitted, Turner 1982. "Some Recent Advances In Computer Graphics." Science 12, February: 767-774

Wagner, Walter and Mileaf, Harry 1983. "Round Table." Architectural Record May: 39-53

APPENDICES

Appendix A

APPLE Program Listings

```
10  REM ------WHISK TREES ------
20  REM            2/25/83
30  REM          BOB NAGEL
35  REM ======================
36  REM     MAIN PROGRAM
37  REM ======================


40 P = 0
50  HOME : VTAB 8
60  PRINT "     THIS PROGRAM ACCE
     PTS A"
70  PRINT "     TREE PROFILE FROM
      THE"
80  PRINT "     GRAPHICS TABLET.
      IT"
90  PRINT "     THEN ROTATES THE
     PROFILE"
100  PRINT "     360 DEG. TO PROD
     UCE A"
110  PRINT "     3-D VECTOR FILE.
       TREE"
120  PRINT "     DENSITY IS DETER
     MINED BY"
130  PRINT "     USER CHOICE (NO.
      OF 'RIBS')."
140  VTAB 20: INPUT "     PRESS R
     ETURN TO BEGIN";M$
150  REM ......................
160 D$ =  CHR$ (4): REM CTRL-D
170  DIM K(70),L(70)
180  GOSUB 4000
190  GOSUB 2000
200  DIM X(B),Y(B)
210  GOSUB 3000
220 X1 = X(1):Y1 = Y(1):X2 = X(1)

230  GOSUB 5000
240  HOME : VTAB 21: GOSUB 1000
250  DIM Z(PTS)
260  FOR J = 1 TO PTS
270 Z(J) = U + 10
280  NEXT J
290 Z1 = U + 10:Z2 = U + 10
300  GOSUB 6000
310  HOME : VTAB 22
320 APP = 1
330  HOME : TEXT : VTAB 4
340  PRINT "           TREE DENSI
     TY
350  VTAB 10: PRINT "      USE A
     NUMBER FROM 1 TO 5": PRINT
360  VTAB 13: PRINT ".........
     ......................."
```

```
370  VTAB 15: PRINT "    1       2
          3       4       5"
380  PRINT
390  PRINT " LEAST
             MOST"
400  PRINT " DENSE
             DENSE"
410  PRINT "......................
     ..............."
420  INPUT " => ";G
430  IF G > 0 AND G < 11 THEN  GOTO
     450
440  HOME : VTAB 6: PRINT "      *
     * NOT IN NUMBER RANGE  **": PRINT
     : GOTO 340
450 N = 0
460  HOME
470  VTAB 18: PRINT "      ROTATIN
     G CANOPY AROUND TRUNK..."
480  FOR J = 6 TO 18 STEP 3
490 N = N + 1
500  IF G <  > N THEN  NEXT J
510 S = J: GOSUB 8000
520  HOME : VTAB 12: PRINT "
        TREE CONSTRUCTED."
530  PRINT : PRINT "         STORI
     NG ON DISC..."
540  GOSUB 7000
550  PRINT D$;"RENAME";F$;",";F$;
     "-";POIN
560  PRINT D$;"OPEN NAME"
570  PRINT D$;"WRITE NAME"
580  PRINT F$;"-";POIN
590  PRINT D$;"CLOSE NAME"
600  HOME : VTAB 12: PRINT "
     STORED AS:  ";F$;"-";POIN
610  VTAB 16
620  PRINT : PRINT "      PLOTTING
      PERSPECTIVE..."
630  PRINT D$;"RUN PLOT TREE"
650  REM ======================
651  REM      SUBROUTINES
652  REM ======================


1000  REM --------------------
1001  REM       FILE NAME?
1002  REM --------------------

1010  PRINT "          NAME FOR TH
     IS TREE FILE?"
1015  VTAB 23: PRINT "
       -----------"
1020  VTAB 22: INPUT "
```

```
      ";F$
1030  RETURN
1111  REM


2000  REM --------------------
2001  REM         AXIS
2002  REM --------------------

2010  HOME : VTAB 12: PRINT "
      PRESS PEN AT BASE OF TRUNK.
     ."
2020  PRINT D$;"PR#5": PRINT "T1,
     F,C,Q"
2030  PRINT D$;"IN#5": INPUT D,E,
     P
2040 D = D - 310:E = E - 980
2050 P =  ABS (P): IF P >  = 10 THEN
     P = P - 10
2060  IF P = 2 THEN  PRINT "
       ** ENTERED **"
2070 Y2 =  INT (E * .04)
2080  POKE  - 16368,0: PRINT : PRINT
     D$;"IN#0"
2090  RETURN
2222  REM

3000  REM ----------------------
3001  REM  REMOVE DUPLICATE PTS
3002  REM ----------------------

3010 X(1) = K(1):Y(1) = L(1)
3020 U = X(1):PTS = 1
3030  FOR J = 2 TO B
3040  IF K(J) > U THEN U = K(J)
3050  IF L(J) = L(J - 1) THEN  GOTO
     3090
3055  IF K(J) < X(1) THEN  GOTO 3
     090
3060  IF J > B THEN  GOTO 3100
3070 PTS = PTS + 1
3080 X(PTS) = K(J):Y(PTS) = L(J)
3090  NEXT J
3100 PTS = PTS + 1:X(PTS) = X(1):
     Y(PTS) = Y(PTS - 1)
3110  RETURN
3333  REM


4000  REM --------------------
4001  REM    CANOPY PROFILE
4002  REM --------------------

4010  HOME : VTAB 12: PRINT "
```

```
      ENTER CANOPY PROFILE..."
4020 B = 0
4030  PRINT D$;"PR#5": PRINT "T1,
     F,C,Q"
4040  PRINT D$;"IN#5": INPUT K,L,
     P
4050 K = K - 310:L = L - 980
4055  PRINT K,L
4060 P =  ABS (P): IF P >  = 10 THEN
     P = P - 10
4080  IF P = 2 THEN  PRINT "
       *****"
4090  IF P = 2 THEN B = B + 1
4100  IF P = 2 THEN K(B) = K
4110  IF P = 2 THEN L(B) = L
4120  IF P = 0 THEN B = B + 1
4130  IF P = 0 THEN K(B) = K
4140  IF P = 0 THEN L(B) = L
4150  IF P = 1 THEN  GOTO 4190
4160  PRINT : PRINT D$;"IN#0": PRINT

4170  FOR PAUSE = 1 TO 10: NEXT P
     AUSE
4180  POKE  - 16368,0: GOTO 4040
4190  POKE  - 16368,0: PRINT D$;"
     IN#0"
4200  FOR J = 1 TO B
4210 K(J) =  INT (K(J) * .05):L(J
     ) =  INT (L(J) * .04)
4220  NEXT J
4230  RETURN
4444  REM


5000  REM --------------------
5001  REM  PLOT TREE PROFILE
5002  REM --------------------

5010  HGR : HCOLOR= 3
5020  HPLOT X1,Y1 TO X2,Y2
5030  VTAB 21: PRINT "          V
     ERTICAL TREE AXIS"
5040  FOR PAUSE = 1 TO 500: NEXT
     PAUSE
5050  HOME : VTAB 21: PRINT "
                TREE PROFILE"
5060  HPLOT X(1),Y(1)
5070  FOR J = 2 TO PTS
5080  HPLOT  TO X(J),Y(J)
5090  NEXT J
5100  RETURN
5555  REM
```

```
6000  REM --------------------
6001  REM  SAVE AXIS & PROFILE
6002  REM --------------------

6010  HOME : VTAB 21: PRINT "
            SAVING AXIS..."
6020  PRINT D$;"OPEN";F$
6030  PRINT D$;"WRITE";F$
6040  PRINT "P ";X1;",";Y1;",";Z1

6050  PRINT "L ";X2;",";Y2;",";Z2

6060  PRINT D$;"CLOSE";F$
6070  RETURN
6666  REM


7000  REM --------------------
7001  REM     SAVE RIBS
7002  REM --------------------

7020 POIN = 0
7050  PRINT D$;"APPEND";F$
7060  PRINT D$;"WRITE";F$
7070  FOR J = 1 TO S
7080  PRINT "P ";X(1);",";Y(1);",
     ";Z(1)
7085 POIN = POIN + 1
7090  FOR I = 2 TO PTS - 1
7092  PRINT "L ";F(I,J);",";Y(I);
     ",";G(I,J)
7093 POIN = POIN + 1
7094  NEXT I
7100  PRINT "L ";X(PTS);",";Y(PT)
     ;",";Z(PTS)
7105 POIN = POIN + 1
7110  NEXT J
7120  PRINT D$;"CLOSE";F$: RETURN

7777  REM


8000  REM --------------------
8001  REM    ROTATE PROFILE
8002  REM --------------------

8010 T = S / 2:PI = 3.14159
8020  DIM F(PTS,S),G(PTS,S)
8030 CS =  COS (PI / T):SS =  SIN
     (PI / T)
8040  FOR I = 2 TO PTS - 1
8045 XX = X(I) - X(1):ZZ = 0:CX =
     X(1):CZ = U + 10
8050  FOR J = 1 TO S
```

```
8070 SX = XX + CX:SZ = CZ + ZZ
8090 F(I,J) =  INT (SX):G(I,J) =
     INT (SZ)
8100 XN = XX * CS - ZZ * SS:ZZ =
     XX * SS + ZZ * CS:XX = XN
8105  NEXT J
8120  NEXT I
8130  RETURN
8888  REM
```

```
2  REM       1/20/84
4  REM  ***** FRACTAL TREE ****
6  REM       BOB NAGEL

7  REM ======================
8  REM       MAIN PROGRAM
9  REM ======================


10 K = 128
15  DIM ARR(2,K,3): REM (ROW,COL,
     DEPTH)
18  DIM MA(3),MB(3),MC(3)
19  GOSUB 3000
20 S = K
21  REM ----GIVEN VALUES---
22 XA = 139:XB = 140:YA = 104:YB =
     105:ZA = 5:ZB = 48
30 ROW = 1:NROW = 2
37 U = XB - XA:V = YB - YA:W = ZB
      - ZA
38 L =  SQR (U ^ 2 + V ^ 2 + W ^
     2)
40  HGR2
44  HPLOT XA,(191 - ZA) TO XB,(19
     1 - ZB)
50  GOSUB 1000
60  IF S < 2 THEN 175
100  GOSUB 2000
110  REM -----TOGGLE-----
120  IF ROW = 1 THEN 140
130  IF ROW = 2 THEN ROW = 1:NROW
      = 2: GOTO 160
140 ROW = 2:NROW = 1
150  REM -----DIVIDE S-----
160 S = S / 2
163  REM -----FLIP ARRAY-----
170  GOTO 60
175  VTAB 22: PRINT "THETA 1 = ";
     T1;"   THETA 2 = ";T2
176  PRINT "   R1 = ";RA;"    R2
     = ";RB
178  PRINT "": REM  BELL
180  END
185 :
189  REM ======================
190  REM       SUBROUTINES
191  REM ======================



1000  REM --------------------
1001  REM      START ARRAY
1002  REM --------------------
```

```
1010 ARR(ROW,0,1) = XB
1012 ARR(ROW,0,2) = YB
1014 ARR(ROW,0,3) = ZB
1020 ARR(NROW,0,1) = XB + RA * (U
      * COS (TA) - (L * V * SIN
     (TA) / SQR ((U * U) + (V *
     V))))
1022 ARR(NROW,0,2) = YB + RA * (V
      * COS (TA) + (L * U * SIN
     (TA) / SQR ((U * U) + (V *
     V))))
1024 ARR(NROW,0,3) = ZB + RA * W *
      COS (TA)
1030 ARR(NROW,S / 2,1) = XB + RB *
     (U * COS (TB) - (L * V * SIN
     (TB) / SQR ((U * U) + (V *
     V))))
1032 ARR(NROW,S / 2,2) = YB + RB *
     (V * COS (TB) + (L * U * SIN
     (TB) / SQR ((U * U) + (V *
     V))))
1034 ARR(NROW,S / 2,3) = ZB + RB *
     W * COS (TB)
1080  RETURN
1090  REM ----------------------


2000  REM --------------------
2001  REM  ARRAY MANIPULATE
2002  REM --------------------

2010  FOR X = 0 TO (K - 1) STEP S

2020  REM -----SET MEMORY---
2030 MA(1) = ARR(ROW,X,1)
2032 MA(2) = ARR(ROW,X,2)
2034 MA(3) = ARR(ROW,X,3)
2040 MB(1) = ARR(NROW,X,1)
2042 MB(2) = ARR(NROW,X,2)
2044 MB(3) = ARR(NROW,X,3)
2050 MC(1) = ARR(NROW,X + S / 2,1
     )
2051 MC(2) = ARR(NROW,X + S / 2,2
     )
2052 MC(3) = ARR(NROW,X + S / 2,3
     )
2053  GOSUB 4000
2054 :
2055  REM ---SET U,V,W,L---
2056 UB = MB(1) - MA(1):VB = MB(2
     ) - MA(2):WB = MB(3) - MA(3)

2057 UC = MC(1) - MA(1):VC = MC(2
```

```
     ) - MA(2):WC = MC(3) - MA(3)

2058 LB =  SQR ((UB * UB) + (VB *
     VB) + (WB * WB))
2059 LC =  SQR ((UC * UC) + (VC *
     VC) + (WC * WC))
2060  REM

2062  REM ---SET VARIABLES---
2065  REM ---STORE RESULTS---

2069  REM ---A=>B,THETA 1---
2070 L = LB:U = UB:V = VB:W = WB
2071 R = RA:TH = TA
2072 XB = MB(1):YB = MB(2):ZB = M
     B(3)
2073  GOSUB 5000: REM =EQUATION=
2074 ARR(ROW,X,1) = XX:ARR(ROW,X,
     2) = Y:ARR(ROW,X,3) = Z
2075  REM - - - - - - - - - - - -

2079  REM ---A=>B,THETA 2---
2080 L = LB:U = UB:V = VB:W = WB
2081 R = RB:TH = TB
2082 XB = MB(1):YB = MB(2):ZB = M
     B(3)
2083  GOSUB 5000: REM =EQUATION=
2084 ARR(ROW,X + S / 4,1) = XX:AR
     R(ROW,X + S / 4,2) = Y:ARR(R
     OW,X + S / 4,3) = Z
2085  REM - - - - - - - - - - - -

2089  REM ---A=>C,THETA 1---
2090 L = LC:U = UC:V = VC:W = WC
2091 R = RA:TH = TA
2092 XB = MC(1):YB = MC(2):ZB = M
     C(3)
2093  GOSUB 5000: REM =EQUATION=
2094 ARR(ROW,X + S / 2,1) = XX:AR
     R(ROW,X + S / 2,2) = Y:ARR(R
     OW,X + S / 2,3) = Z
2095  REM - - - - - - - - - - - -

2099  REM ---A=>C,THETA 2---
2100 L = LC:U = UC:V = VC:W = WC
2101 R = RB:TH = TB
2102 XB = MC(1):YB = MC(2):ZB = M
     C(3)
2103  GOSUB 5000: REM =EQUATION=
2104 ARR(ROW,X + S / 2 + S / 4,1)
      = XX:ARR(ROW,X + S / 2 + S /
     4,2) = Y:ARR(ROW,X + S / 2 +
     S / 4,3) = Z
2105  REM - - - - - - - - - - - -
```

```
2110  NEXT X
2120  RETURN
2130  REM ----------------------


3000  REM --------------------
3001  REM     INPUT CONTROLS
3002  REM --------------------

3005  TEXT
3006  HOME : VTAB 7
3007  PRINT "     'THETA 1 & 2' =>
      BRANCHING ANGLES"
3008  PRINT : PRINT "     'R1 & R2
     ' => BRANCH LENGTH RATIOS"
3009  PRINT "     ----------------
     -----------------"
3010  PRINT : PRINT : PRINT
3020  INPUT "ENTER THETA 1 => ";T
     A:T1 = TA
3030  INPUT "ENTER THETA 2 => ";T
     B:T2 = TB
3040  PRINT : INPUT "      ENTER R
     1 => ";RA
3050  INPUT "      ENTER R2 => ";R
     B
3055 SS = 3.14159 / 180
3060 TA = TA * SS:TB = TB * SS
3070  RETURN
3080  REM -------------------


4000  REM --------------------
4001  REM     PLOT SUBUNITS
4002  REM --------------------

4004  IF MA(3) < 1 THEN 4030
4005  IF MA(1) > 279 THEN 4030
4006  IF MA(3) > 190 THEN 4030
4007  IF MB(1) > 279 THEN MB(1) =
     279
4008  IF MC(1) > 279 THEN MC(1) =
     279
4009  IF MB(3) > 190 THEN MB(3) =
     190
4010  IF MC(3) > 190 THEN MC(3) =
     190
4012  IF MB(2) < 1 THEN MB(2) = 1

4013  IF MC(3) < 1 THEN MC(3) = 1

4014  IF MB(1) < 1 THEN MB(1) = 1
```

```
4015  IF MC(1) < 1 THEN MC(1) = 1

4019  HPLOT MA(1),(191 - MA(3)) TO
     MB(1),(191 - MB(3))
4020  HPLOT MA(1),(191 - MA(3)) TO
     MC(1),(191 - MC(3))
4030  RETURN
4040  REM --------------------


5000  REM --------------------
5001  REM        COMPUTE
5002  REM --------------------

5010 XX = XB + R * (U *  COS (TH)
      - (L * V *  SIN (TH) /  SQR
     (U * U + V * V)))
5020 Y = YB + R * (V *  COS (TH) +
     (L * U *  SIN (TH) /  SQR (U
      * U + V * V)))
5030 Z = ZB + R * W *  COS (TH)
5040  RETURN
5050  REM --------------------
```

```
5  REM  ==== PLOT TREE ====
10  REM   ----- FEB. 10/83 -----
20  REM      PLOTS A VECTOR FILE
30  REM  IN PERSPECTIVE ON SCREEN

70  TEXT
80 D$ =  CHR$ (4): REM - CTRL-D -

90  HOME : VTAB 12
100  PRINT "     FILE TO BE DISPL
     AYED?"
105  PRINT
110  PRINT "ENTER ENTIRE NAME (EG
     .NAME-12)."
115  PRINT : INPUT "     => ";F$
120  FOR I = 1 TO  LEN (F$)
130  IF  MID$ (F$,I,1) <  > "-" THEN
      NEXT I
140 I = I + 1
150 PTS =  VAL ( MID$ (F$,I))
160  DIM A$(PTS),B$(PTS),X(PTS),Y
     (PTS),Z(PTS)
170  HOME : VTAB 12
180  PRINT "       READING ";F$;"
      FROM DISC."
190  PRINT D$;"OPEN";F$
200  PRINT D$;"READ";F$
210  FOR J = 1 TO PTS
220  INPUT " ";A$(J),Y(J),Z(J)
230  NEXT J
240  PRINT D$;"CLOSE";F$
260  FOR J = 1 TO PTS
270 B$(J) =  LEFT$ (A$(J),1)
280 X(J) =  VAL ( MID$ (A$(J),3))

290  NEXT J
300  HOME : VTAB 12
310  PRINT "     "F$;" STORED IN
     MEMORY"
320  VTAB 15: PRINT "     PLOT ";
     F$;" ON SCREEN? ('Y')"
340  INPUT "     => ";R$
350  IF R$ = "Y" THEN  GOTO 360
355  GOTO 550
360  REM  --- 3-D PERSPECTIVE ---

370  HGR : HCOLOR= 7: HOME
380 ZV =  - 1000
390  FOR J = 1 TO PTS
400 A = X(J) - 5
410 B = Y(J) - 10
420 D = Z(J) - ZV
430 Q =  SQR (A * A + B * B + D *
     D)
```

```
440 UX = A / Q
450 UY = B / Q
460 UZ = D / Q
470 QH = Q * ( - ZV) / D
480 XH = 5 + UX * QH
490 YH = 10 + UY * QH
500  IF B$(J) = "L" THEN 530
510  HPLOT XH,YH
520  GOTO 540
530  HPLOT  TO XH,YH
540  NEXT J
550  HOME : VTAB 21
560  PRINT "    TYPE 'RUN' TO VIE
     W ANOTHER FILE."
570  END
```

```
1  REM :::VECTOR SCALE:::
2  REM ::::::4/1/83::::::

3  REM    SCALES WHISK TREES
4  REM    FOR PS-300 SCREEN.
5  REM      BOB NAGEL
6  REM  ....................


9 D$ =  CHR$ (4): REM  CTRL-D
10  HOME : VTAB 12
20  PRINT "FILE NAME?"
30  INPUT F$
40  FOR I = 1 TO  LEN (F$)
50  IF  MID$ (F$,I,1) <  > "-" THEN
      NEXT I
60 I = I + 1
70 PTS =  VAL ( MID$ (F$,I))
75  DIM A$(PTS),Y(PTS),Z(PTS),O(P
    TS)
80 XMAX = 0:YMAX = 0
85  DIM M(PTS),N(PTS),B$(PTS),X(P
    TS)
90  PRINT D$;"OPEN";F$
100  PRINT D$;"READ";F$
110  GOSUB 1000
150  PRINT D$;"CLOSE";F$
160  REM  P/L NOW IN B$
165  PRINT " MAXIMUM X = ";XMAX
166  PRINT " MAXIMUM Y = ";YMAX
170  PRINT "  SCALING X,Y,Z"
180  GOSUB 2000
190  REM    SCALED IN M,N,O
195  PRINT D$;"MON C,I,O"
200  PRINT D$;"OPEN ALT";F$
210  PRINT D$;"WRITE ALT";F$
215  PRINT "BOX := VECTOR"; CHR$
    (95);"LIST ITEMIZED"
220  FOR I = 1 TO PTS
240  PRINT B$(I);" ";M(I);",";N(I
    );",";O(I)
250  NEXT I
255  PRINT ";"
260  PRINT D$;"CLOSE ALT";F$
265  PRINT D$;"NOMON C,I,O"
270  END
500  REM .......................

510  REM       SUBROUTINES
520  REM .......................
530  REM

1000  REM ---FIND XMAX & YMAX---
1010  FOR I = 1 TO PTS
```

```
1020  INPUT A$(I),Y(I),Z(I)
1030 B$(I) =  LEFT$ (A$(I),1)
1040 X(I) =  VAL ( MID$ (A$(I),3)
     )
1050  IF X(I) > XMAX THEN XMAX =
     X(I)
1060  IF Y(I) > YMAX THEN YMAX =
     Y(I)
1070  NEXT I
1080  RETURN
1999  REM ......................

2000  REM ----SCALE X,Y,Z----

2010  FOR I = 1 TO PTS
2020 M(I) = (X(I) * 10.0) / XMAX
2030 N(I) = (Y(I) * 25.0) / YMAX
2035 O(I) = (Z(I) * 10.0) / XMAX
2090  NEXT I
2100  RETURN
```

Appendix B

PS-300 Program Listings

```
  10  (               PROGRAM          : DRIVER.300                  )
  20  (               Programmer       : On Khong Lie                )
  30
  40
 100  §---------------------------MAIN PROGRAM---------------------†
 200  §this is the main program†
 300  §initialize PS 300†
 400  INIT;
 500
 600  §manipulation of the object†
 700  sketch := BEGIN_S
 800                 set := SET LEVEL_OF_DETAIL TO 9;
 900                  i := increment level of detail;
1000                 WINDOW X=-10:10 Y=-10:10
1100                         FRONT=-5
1200                         BACK=40;        §specify the window†
1300                 rot_x := ROTATE IN X 0; §rotate in x†
1400                 rot_y := ROTATE IN Y 0; §rotate in y†
1500                 rot_z := ROTATE IN Z 0; §rotate in z†
1600                 scale := SCALE BY .1;   §scale by 1†
1700                 §all are initial rotations and scaling†
1800
1900                 INSTANCE OF box     §place for growing tree†
2000                 del := if level_of_detail = 10 then box;
2100                 CHARACTER SCALE .5;
2200                 CHARACTER  -0.1, -0.1,-5 '*';  §marker indicating the
2300                 ground and where the tree will originate†
2400           END_S;
68506  §defining the space/ground†
```

```
  100  §----------------------NETWORK MANIPULATE----------------------
  200  §this module is to do the rotation and scaling of the space†
  300  §x rotation is connected to dial 1†
  400  do_rot_x := F:DXROTATE;
  500  SEND 'x rotate' TO <1>DLABEL1;
  600  CONN DIALS<1>:<1>do_rot_x;
  700  CONN do_rot_x<1>:<1>sketch.rot_x;
  800  SEND 0 TO <2>do_rot_x;
  900  SEND 180 TO <3>do_rot_x;
 1000
 1100  §y rotation is connected to dial 2†
 1200  do_rot_y := F:DYROTATE;
 1300  SEND 'y rotate' TO <1>DLABEL2
 1400  CONN DIALS<2>:<1>do_rot_y;
 1500  CONN do_rot_y<1>:<1>sketch.rot_y;
 1600  SEND 0 TO <2>do_rot_y;
 1700  SEND 180 TO <3>do_rot_y;
 1800
```

```
1900   §z rotation is connected to dial 3†
2000   do_rot_z := F:DZROTATE;
2100   SEND 'z rotate' TO <1>DLABEL3
2200   CONN DIALS<3>:<1>do_rot_z;
2300   CONN do_rot_z<1>:<1>sketch.rot_z;
2400   SEND 0 TO <2>do_rot_z;
2500   SEND 180 TO <3>do_rot_z;
2600
2700   §scaling of object is connected to dial 4†
2800   do_scale := F:DSCALE;
2900   SEND 'scale' TO <1>DLABEL4;
3000   CONN DIALS<4>:<1>do_scale;
3100   CONN do_scale<1>;<1>sketch.scale;
3200   SEND 0 TO <2>do_scale;
3300   SEND 1 TO <3>do_scale;
3400   SEND 2 TO <4>do_scale;
3500   SEND .1 TO <5>do_scale;
3600
3700   §display sketch†
3800   DISPL sketch;
3900
4000   §-----------------------------------------------------------------†
4100
```

```
  50  (       PROGRAM          : APPLE.300                          )
 100  §       Programmer       : On Khong Lie                       †
 200  §       Date             : Winter, 1983                       †
 300  §       Place            : Utah State  University             †
 400
 500
 600
 700  §initialize PS 300†
 800  INIT;
 900
1000  sketch := BEGIN_S
1100                  WINDOW X=-10:10 Y=-10:10
1200                         FRONT=-5
1300                         BACK=40;          §specify the window†
1400                  rot_x := ROTATE IN X 0; §rotate in x†
1500                  rot_y := ROTATE IN Y 0; §rotate in y†
1600                  rot_z := ROTATE IN Z 0; §rotate in z†
1700                  scale := SCALE BY .1;   §scale by 1†
1800                  trans_x := TRANS BY 0, 0, 0;
1900                  trans_y := TRANS BY 0, 0, 0;
2000                  trans_z := TRANS BY 0, 0, 0;
2100                  §all are initial rotations  and scaling†
2200
2300                  INSTANCE OF box;
2400            END_S;
2500
2600  §Box is the global/general objects will be supplied from the †
2700  §FORTRAN program                                             †
```

```
 200  §Network Manipulate†
 300
 400  §this is the module to do rotations and scaling the object      †
 500
 600  §x rotation, connected to dial 1†
 700
 800  do_rot_x := F:DXROTATE;
 900
1000  SEND 'x rotate' TO <1>DLABEL1;
1100  CONN DIALS<1>:<1>do_rot_x;
1200  CONN do_rot_x<1>:<1>sketch.rot_x;
1300  SEND 0 TO <2>do_rot_x;
1400  SEND 180 TO <3>do_rot_x;
1500
1600  §y rotation, connected to dial 2†
1700
1800  do_rot_y := F:DYROTATE;
1900
```

```
2000    SEND 'y rotate' TO <1>DLABEL2;
2100    CONN DIALS<2>:<1>do_rot_Y;
2200    CONN do_rot_Y<1>:<1>sketch.rot_y;
2300    SEND 0 TO <2>do_rot_y;
2400    SEND 180 TO <3>do_rot_y;
2500
2600    §z rotation, connected to dial 3†
2700
2800    do_rot_z := F:DZROTATE;
2900
3000    SEND 'z rotate' TO <1>DLABEL3;
3100    CONN DIALS<3>:<1>do_rot_z;
3200    CONN do_rot_z<1>:<1>sketch.rot_z;
3300    SEND 0 TO <2>do_rot_z;
3400    SEND 180 TO <3>do_rot_z;
3500
3600    §scaling the object, connected to dial 4†
3700
3800    do_scale := F:DSCALE;
3900    SEND 'scale' TO <1>DLABEL4;
4000    CONN DIALS<4>:<1>do_scale;
4100    CONN do_scale<1>:<1>sketch.scale;
4200    SEND 0 TO <2>do_scale;
4300    SEND 1 TO <3>do_scale;
4400    SEND 2 TO <4>do_scale;
4500    SEND .1 TO <5>do_scale;
4600
4700    § X translation, connected to dial 5†
4800
4900    addx := F:ADDC;
5000    xtrans := F : XVECTOR;
5100
5200    SEND 'x trans' to <1>DLABEL5;
5300    CONN DIALS<5>:<1>addx;
5400    CONN addx<1>:<2>addx;
5500    CONN addx<1>:<1>xtrans;
5600    CONN xtrans<1>:<1>sketch.trans_x;
5700    SEND 0 TO <2>addx;
5800
5900    §y translation, connected to dial 6†
6000
```

```
6100    addy := F:ADDC;
6200    ytrans := F : YVECTOR;
6300
6400    SEND 'y trans' TO <1>DLABEL6;
6500    CONN DIALS<6>:<1>addy;
6600    CONN addy<1>:<2>addy;
```

```
6700   CONN addy<1>:<1>ytrans;
6800   CONN ytrans<1>:<1>sketch.trans_y;
6900   SEND 0 TO <2>addy;
7000
7100
7200
7300   addz := F:ADDC;
7400   ztrans := F : ZVECTOR;
7500
7600   SEND 'z trans' TO <1>DLABEL7;
7700   CONN DIALS<7>:<1>addz;
7800   CONN addz<1>:<2>addz;
7900   CONN addz<1>:<1>ztrans;
8000   CONN ztrans<1>:<1>sketch.trans_z;
8100   SEND 0 TO <2>addz;
8200
8300   §z translation, connected to dial 7†
8400   §display the sketch†
8500
8600   DISPL sketch;
```

Appendix C

Example Whisk-Tree Data File

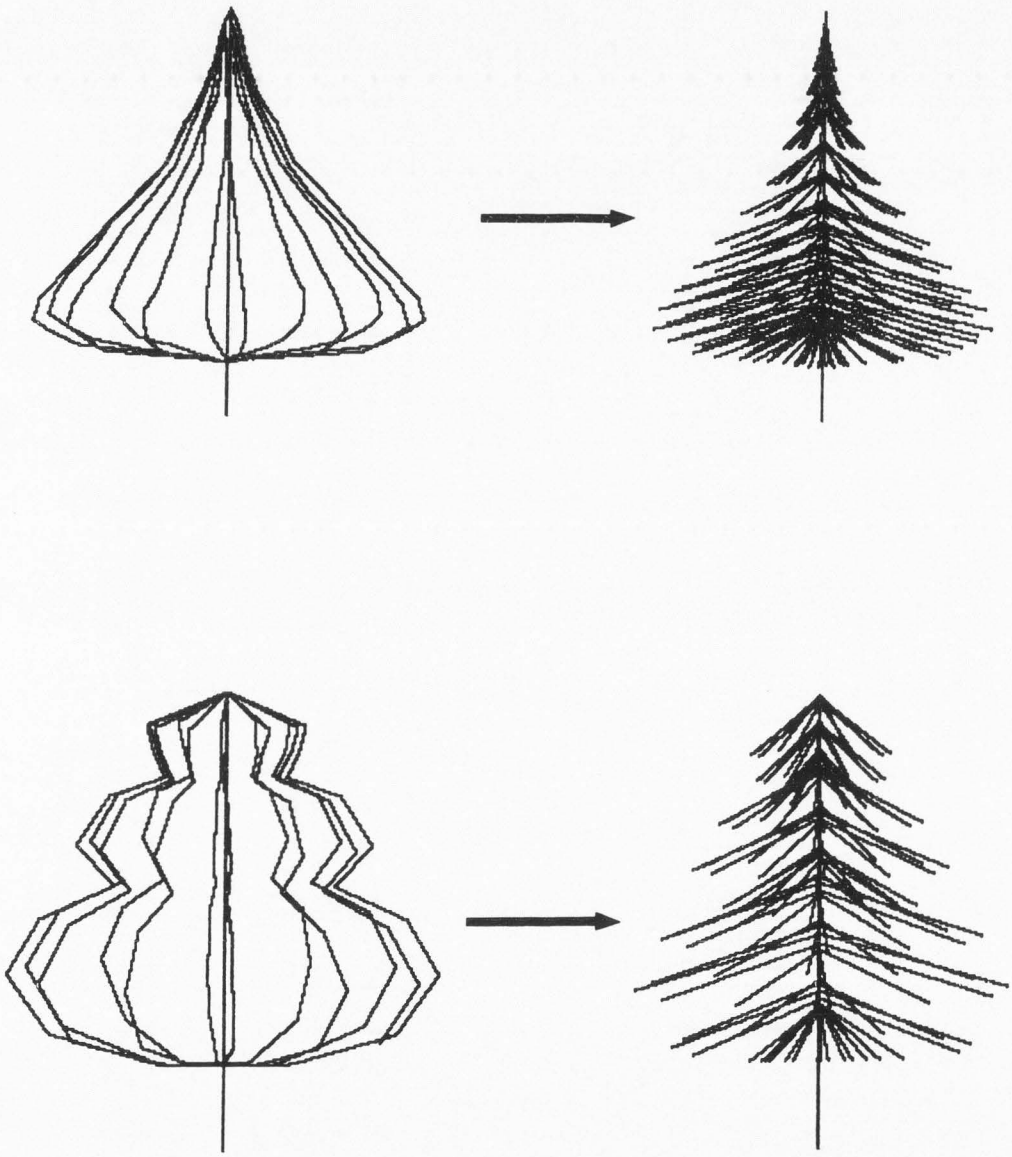
The following data list contains the coordinates that define the tree, ALTSPRUCE.DAT. The ALT prefix denotes that it has been altered from its Apple display format to PS-300 display format in which the raw coordinates generated by the Apple for display on its screen were manipulated with the program "Vector Scale" (Appendix A) to be compatible with the PS-300 command structure. Transfer of the file from the Apple to the VAX mainframe was facilitated by the commercial software "Visiterm" (Personal Software Inc.) using a telephone connection.

```
  50   BOX :=VECTOR_LIST ITEMIZED
 100   P 133,26,-30
 200   L 133,114,-30
 300   P 133,26,-30
 400   L 141,35,-30
 500   L 151,41,-30
 600   L 152,42,-30
 700   L 136,40,-30
 800   L 135,39,-30
 900   L 146,45,-30
1000   L 139,43,-30
1100   L 145,50,-30
1200   L 165,60,-30
1300   L 142,54,-30
1400   L 136,50,-30
1500   L 137,51,-30
1600   L 139,57,-30
1700   L 148,66,-30
1800   L 160,73,-30
1900   L 155,72,-30
2000   L 145,68,-30
2100   L 150,74,-30
2200   L 162,80,-30
2300   L 172,82,-30
2400   L 141,77,-30
2500   L 133,71,-30
2600   L 133,71,-30
2700   P 133,26,-30
2800   L 137,35,-36
2900   L 142,41,-45
3000   L 142,42,-46
3100   L 134,40,-32
3200   L 134,39,-31
3300   L 139,45,-41
3400   L 136,43,-35
3500   L 139,50,-40
3600   L 149,60,-57
3700   L 137,54,-37
3800   L 134,50,-32
3900   L 135,51,-33
4000   L 136,57,-35
4100   L 140,66,-42
4200   L 146,73,-53
4300   L 144,72,-49
4400   L 139,68,-40
4500   L 141,74,-44
4600   L 147,80,-55
4700   L 152,82,-63
4800   L 137,77,-36
4900   L 133,71,-30
5000   L 133,71,-30
5100   P 133,26,-30
5200   L 129,35,-36
5300   L 124,41,-45
5400   L 123,42,-46
5500   L 131,40,-32
5600   L 132,39,-31
5700   L 126,45,-41
5800   L 130,43,-35
5900   L 127,50,-40
6000   L 117,60,-57
6100   L 128,54,-37
6200   L 131,50,-32
6300   L 131,51,-33
6400   L 130,57,-35
6500   L 125,66,-42
6600   L 119,73,-53
6700   L 122,72,-49
6800   L 127,68,-40
6900   L 124,74,-44
7000   L 118,80,-55
7100   L 113,82,-63
7200   L 129,77,-36
7300   L 133,71,-30
7400   L 133,71,-30
7500   P 133,26,-30
7600   L 125,35,-30
7700   L 115,41,-30
```

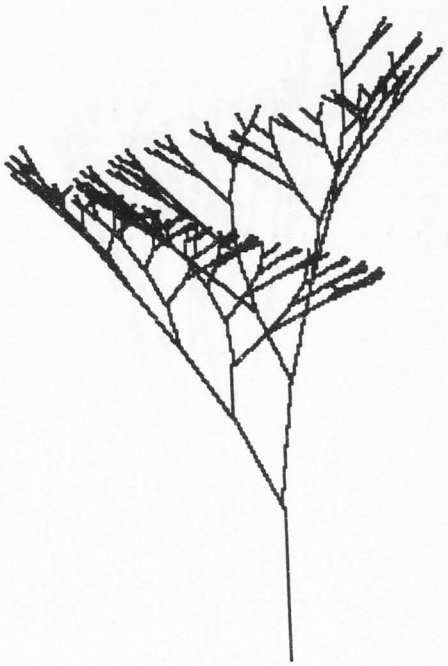
```
 7800   L 114,42,-30
 7900   L 130,40,-30
 8000   L 131,39,-30
 8100   L 120,45,-30
 8200   L 127,43,-30
 8300   L 121,50,-30
 8400   L 100,60,-30
 8500   L 124,54,-30
 8600   L 130,50,-30
 8700   L 129,51,-30
 8800   L 127,57,-30
 8900   L 118,66,-30
 9000   L 105,73,-30
 9100   L 110,72,-30
 9200   L 121,68,-30
 9300   L 115,74,-30
 9400   L 103,80,-30
 9500   L 93,82,-30
 9600   L 125,77,-30
 9700   L 133,71,-30
 9800   L 133,71,-30
 9900   P 133,26,-30
10000   L 128,35,-23
10100   L 123,41,-14
10200   L 123,42,-13
10300   L 131,40,-27
10400   L 131,39,-28
10500   L 126,45,-18
10600   L 129,43,-24
10700   L 126,50,-19
10800   L 116,60,-2
10900   L 128,54,-22
11000   L 131,50,-27
11100   L 130,51,-26
11200   L 129,57,-24
11300   L 125,66,-17
11400   L 119,73,-6
11500   L 121,72,-10
11600   L 126,68,-19
11700   L 124,74,-15
11800   L 118,80,-4
11900   L 113,82,4
12000   L 128,77,-23
12100   L 133,71,-30
12200   L 133,71,-30
12300   P 133,26,-30
12400   L 136,35,-23
12500   L 141,41,-14
12600   L 142,42,-13
12700   L 134,40,-27
12800   L 133,39,-28
12900   L 139,45,-18
13000   L 135,43,-24
13100   L 138,50,-19
13200   L 148,60,-2
13300   L 137,54,-22
13400   L 134,50,-27
13500   L 134,51,-26
13600   L 135,57,-24
13700   L 140,66,-17
13800   L 146,73,-6
13900   L 143,72,-10
14000   L 138,68,-19
14100   L 141,74,-15
14200   L 147,80,-4
14300   L 152,82,4
14400   L 136,77,-23
14500   ;
```

Appendix D

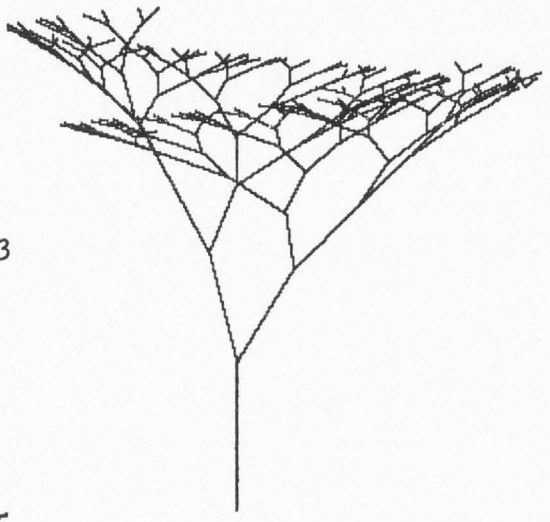
Additional Tree Examples



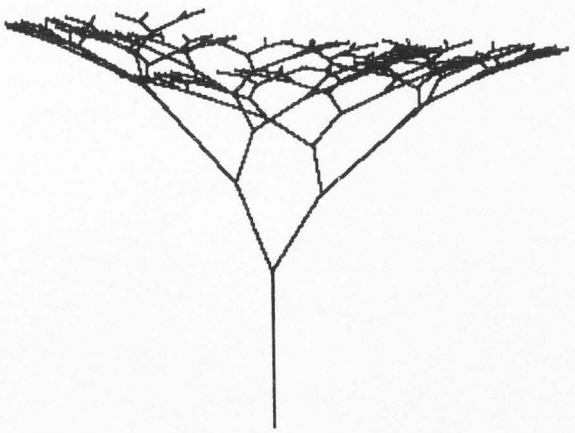
Alternative plotting procedure for Whisk Trees. Conifers implied by plotting from the vertical axis out to points along the canopy (right), rather than plotting points along canopy as continuous "ribs" (see also p.70).



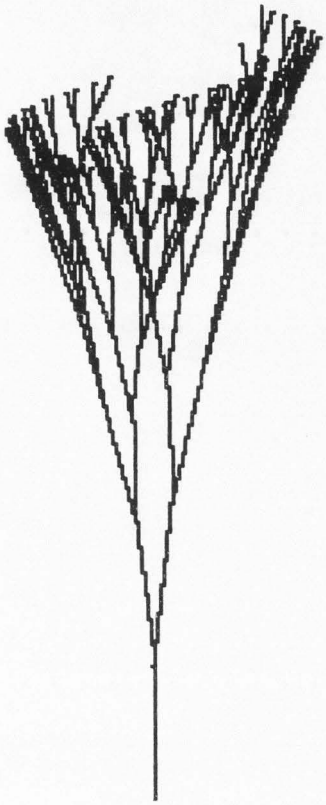
$\theta_1 = -35$, $\theta_2 = 10$, $R_1 = .73$, $R_2 = .83$

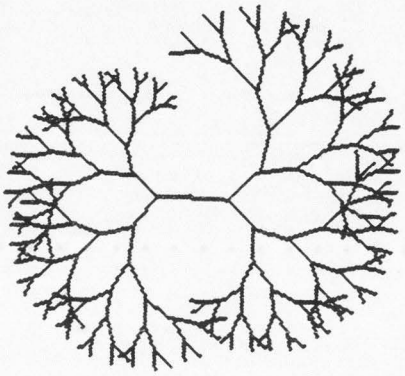


$\theta_1 = -40$, $\theta_2 = 20$,
$R_1 = .8$, $R_2 = .77$



$\theta_1 = -40$, $\theta_2 = 30$, $R_1 = .8$, $R_2 = .8$



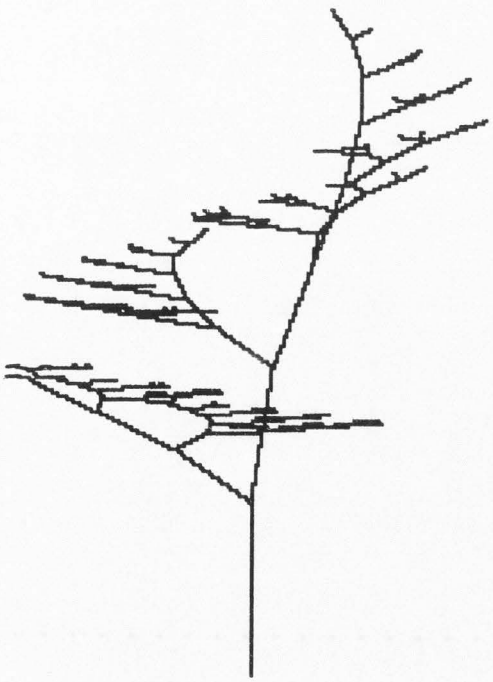


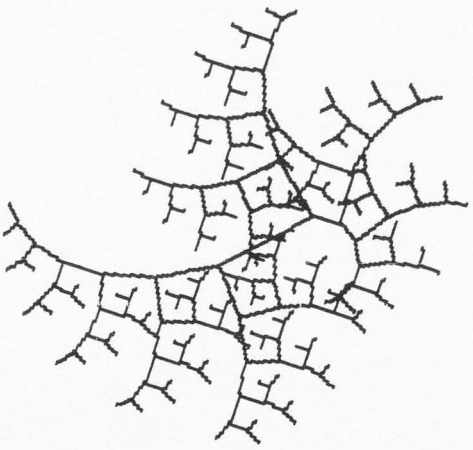
$\theta_1 = 10°$

$\theta_2 = -10°$

$R_1 = .73$

$R_2 = .83$



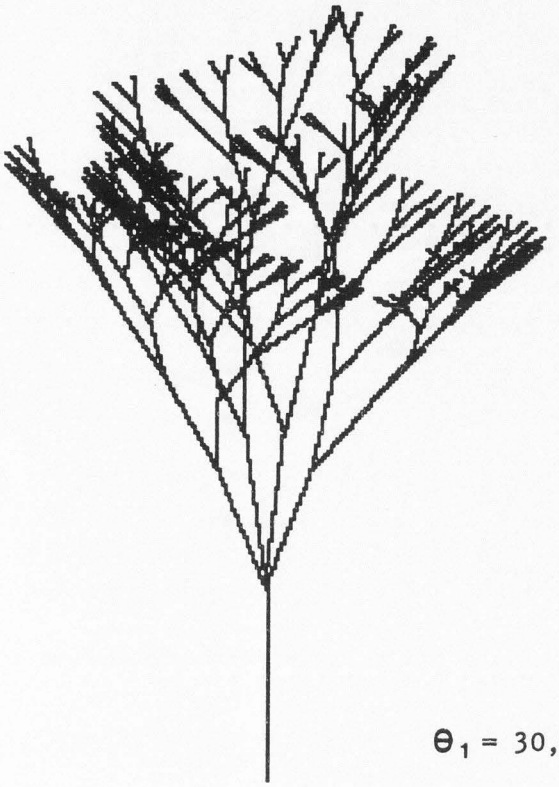


$\theta_1 = 15°$

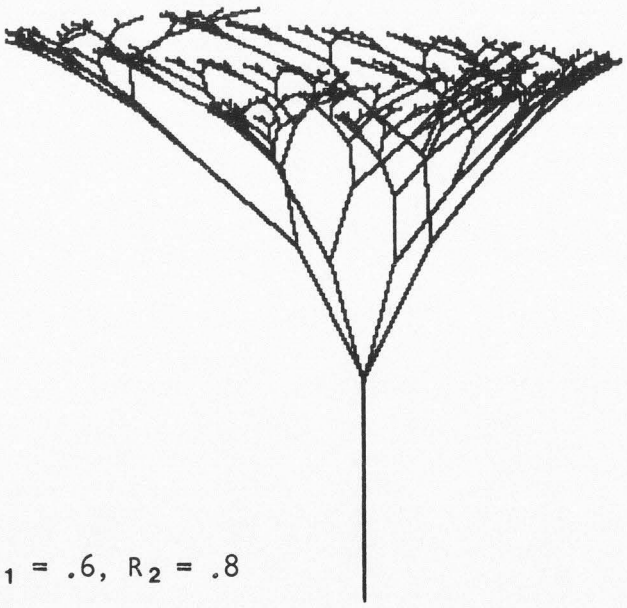
$\theta_2 = -65°$

$R_1 = .8$

$R_2 = .7$



$\theta_1 = 30,\ \theta_2 = -10,\ R_1 = .7,\ R_2 = .8$



$\theta_1 = -20,\ \theta_2 = 35,\ R_1 = .6,\ R_2 = .8$

Two examples of an alternate plotting procedure in which two views of the same tree are plotted at the same origin. One image is rotated along its vertical axis by 90° in relation to the other image.