T
D65-88
SAS

AN APPROACH TO GEOMETRIC MODELING OF SOLID OBJECTS

A THESIS

submitted in fulfilment of the
requirements for the award of the degree
of
DOCTOR OF PHILOSOPHY
in
ELECTRONICS AND COMPUTER ENGINEERING

By

S. SASIKUMARAN

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
UNIVERSITY OF ROORKEE
ROORKEE-247 667 (INDIA)

DECEMBER, 1988

DEDICATED

TO

MY PARENTS

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled 'An Approach to Geometric Modeling of Solid Objects' in fulfilment of the requirement for the award of the Degree of Doctor of Philosophy submitted in the Department of Electronics and Computer Engineering of the University is an authentic record of my own work carried out during a period from February 1986 to December 1988 under the supervision of Dr. R. C. Joshi and Dr. A. K. Sarje.

The matter embodied in this thesis has not been submitted by me for the award of any other Degree.

[illegible]

(S. SASIKUMARAN)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

[illegible]

(R. C. JOSHI)
Professor,
Electronics & Computer Engg.
Department

[illegible]

(A. K. SARJE)
Professor
Electronics & Computer Engg.
Department

26.12.1988

The candidate has passed the Viva-Voce examination held on____________at____________. The thesis is recommended for award of the Ph.D. Degree.

1. ____________________
(Signature of Guide)

1. ____________________
(Signature of External Examiner)

2. ____________________
(Signature of Guide)

ABSTRACT

Recent years have seen growing interest in the modeling (representation, manipulation and display) of solid objects with a computer. Three dimensional object modeling is essential for computer graphics, CAD/CAM systems, image understanding systems, and other applications.

The search for better modeling techniques have been and continue to be one of the major problems in solid modeling. This thesis presents a 'Hex-tree representational technique' for geometric modeling of solid objects. This is a tree oriented, recursive type new constructional technique for representation and display of any 3-D object. A single cubical cell is used as primitive in this approach. In the Hex-tree data structure, the full identity of a node (i.e. cubical cell) is represented in six Bits of a word, indicating the life entities in all directions which represent the six faces.

In order to get a realistic display of the Hex-tree based solid models, two Hidden-line algorithms are developed for simple and complex objects. These algorithms are suitable for Line-drawings display devices. A complete hidden lines removed perspective picture of solid models, viewed from various distances and/or positions in space could be perceived. Provisions are also incorporated to visualise the object models, without removing hidden lines. Same algorithm can be used for both planar and curve surfaced objects.

Necessary methods have been developed to calculate some of the integral properties of the Hex-tree based solid models. The volume, weight, center of gravity and moment of inertia of an unsymmetrical model have been computed and shown. The merits of providing 'Screen-layout' by creating multiple screen areas, for a better user-interface are also shown.

Using this approach, it is possible to define, modify and display any planar and curve surfaced three dimensional object. The implementation of the geometric modeling system has been done in the FORTRAN language on VAX-11/780 computer system using Tektronix-4027 graphics terminal.

ACKNOWLEDGEMENTS

The author wishes to express his sincere gratitude to Prof. R. C. Joshi and Prof. A. K. Sarje for their invaluable guidance and encouragement in planning and execution of this research work. Their painstaking efforts in correcting the manuscript and giving fruitful suggestions for its improvement are gratefully acknowledged.

The suggestions and cooperation extended by Prof. N. K. Nanda, former Head, and Prof. R. Mitra, present Head, Department of Electronics and Computer Engg., are gratefully acknowledged.

The author is thankful to Dr. K. N. Ramachandran Nair, Reader, Department of Computer Science, University of Poona, for the help he has rendered through discussions during the initial stage of this investigation.

Thanks are also due to Mrs. Deepa Chalisgaonkar for her sincere help during the execution of this work in the Computer Centre of National Institute of Hydrology, Roorkee, and Mr. Subash Kichlu of Remote Sensing Lab. of the same Institute for taking the photographs nicely.

The author is grateful to the authorities of Regional Engineering College, Calicut (Kerala) for sponsoring him to persue this research work under Quality Improvement Programme.

It is a distinct pleasure for the author to acknowledge and record his heartfelt appreciation to his wife Beena for her patience, understanding and in cheerfully shouldering the family responsibilities, and to his sons, Dawn & Dijo, for their understanding in bearing the burden imposed on them because of his involvement in this research programme.

No words can express the author's sincere gratitude to his Father-in-Law, Shri Raveendranathan, and other members of the family for the help, understanding and cooperation rendered during the tenure of this study.

The author is thankful to his fellow research scholars and other friends for their cooperation and help during the entire course of this research programme.

Finally, thanks are also due to the staff of Rashtriya Commercial Institute, CBRI Colony, Shantinagar, Roorkee for neat typing and Shri Y. D. Gupta for drawing the figures nicely.

Roorkee

December, 1988

[illegible]

(S. SASIKUMARAN)

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION AND STATEMENT OF THE PROBLEM

1.1 INTRODUCTION

Geometric modeling is the study of computer methods for generation, storage, analysis, manipulation and display of shape information. The system which provides facilities for entering, storing and modifying shape information is called a geometric modeling system (GMS). Geometric modeling systems are extensively used by a number of industries all over the world for creation and manipulation of 3-D object models in their computer-aided design and manufacturing activities. Plant design, architectural design, molecular modeling, etc. are few other application areas of geometric modeling system. Because of their 3-D object modeling capability, an object represented has the information content of a number of drawings, pages of specification and engineering data. In a typical geometric modeling system environment, the human-user sits at a graphic workstation and interacts with the system through various interactive devices (alphanumeric keyboard, digitizer, tablet, joystick, light pen, etc.) and display devices (raster-scan cathode-ray-tube, memory-tube displays, plotters, etc.) to get his job done. The objects created during one such interactive session can be stored in disc files for future examination or modification. Also, they can be passed on to any other program for analysis, such as, spatial interference checking, structural analysis, or for producing Numerical Control (NC) machine tool path data for their automatic manufacture. The possibility of integrating the design, analysis and manufacturing activities together, by providing a common data base is another important advantage of geometric modeling system.

Wireframe modeling system, surface modeling system and solid modeling system are the three types of 3-D geometric modeling systems available [57,35].In wireframe models, the edges of the object are shown as lines. For objects in which there are curved surfaces, contour lines can be added to indicate the surface. The image assumes the appearance of a frame constructed out of wire - hence, the name 'wireframe model'. These models are highly ambiguous description of object geometry, virtually not useful for complex manufacturing operations. In wireframe models, all the lines defining the edges (and contoured surfaces) of the model are shown in the image. This can cause the image to be somewhat confusing to the viewer, and in some cases, the image might be interpretable in several different ways. Though by removing the hidden lines in the image this interpretation problem can be alleviated to some extent, it requires substantial manual intervention. The true volume of the object also cannot be known through the model, as there is no knowledge of the surfaces between the lines.

The surface models describe an object in terms of points, edges and faces between these edges. These models are also ambiguous due to the fact that there is no knowlege of what is 'solid' material or what is the 'inside' of an object or the 'outside'. Surface modelers are useful to find the intersection of complex surfaces, to produce shaded colour pictures, to generate Numerical Control tool path data for complex surface machining etc. However, surface modelers suffer from the following drawbacks.

i) They cannot reliably remove 'hidden lines' from any view

ii) The creation of internal sectioned views is a tedious task

iii) The accuracy and reliability of mass properties (volume, moment of inertia, etc.) extracted from surface models have limitations.

An improvement over Wireframe Model and Surface Model, both in terms of realism to the user and definition to the computer, is the Solid Model. The solid modeler (also referred to as 'Geometric' or 'Volumetric' modeler) holds a complete description of an object, in terms of space, which it occupies. This description is unambiguous, in the sense that the system always knows whether any point is 'inside' or 'outside' an object, or if it lies on the surface of the object. There is very little chance of misinterpretation as the models are displayed as solid objects to the viewer. When colour is added to the image, the resulting picture becomes strikingly realistic. Other advantages of solid modelers are :

i) They can produce accurate mass properties data which truly reflects the reality of the part.

ii) Interference between any parts can be detected and displayed

iii) Views can be produced from any point of view, with automatic removal or dashing of hidden lines.

Perspective can be generated for any view. Because of this tremendous potential, geometric modeling system, with solid modeling capability, will find wide range of applications outside the realm of computer-aided design and manufacturing, such as, training simulators, robotics, animation in movies, etc. [57,55, 35,34].

Various representation methods have been proposed for geometric modeling of solid objects [55]. Among these, Constructive Solid Geometry (CSG) and Boundary Representation (B-Rep) are the two widely accepted schemes for commercial geometric modeling systems. However, these methods have several limitations.

The representation capabilities of these schemes are not sufficiently robust to easily handle the object complexities required in a realistic environment, as the objects are constructed from a limited number of mathematically well defined surfaces or solid primitives [45]. Adding new powerful primitives to a system based on these methods or generalising the use of existing one not only requires extensive development of mathematical tools and significant software modification, but also needs substantial additional computation and memory space.

In any of these representations, object manipulations, such as boolean operations (union, difference and intersection) or display operations place an unacceptable burden on computational resources. In these operations, it is necessary to calculate the edge of intersection in three dimensions of two objects (represented by several thousand primitive shapes or surface patches), interference detection and hidden surface removal, which require a very large amount of computation. The searching and sorting tasks for the comparison of primitives also increases the processing time.

Problem also arises when there is a need to store data in a database. For a database, it is an essential requirement to secure consistency of the stored data, like physical 3-D interference of mechanical parts. In order to check the consistency of stored data in these representations, a lot of computation is required [20].

The computation of integral properties, such as, volume, surface area, moment of inertia, etc. of objects represented by Constructive Solid Geometry scheme is a difficult operation. If an object is defined as the union of two primitives which penetrate each other, then the volume of the object can be computed only if we know what portion is common to these two primitives. The CSG representation of the object does not explicitly give this information. Hence, computation of these properties require more computational efforts.

Another disadvantage of Boundary Representation scheme is its verbosity [55]. Even for a simple object like a cube, the amount of information (contains details of its six faces, twelve edges, eight vertices and their adjacency information) to be stored is very large.

Even though Constructive Solid Geometry and Boundary Representation schemes are the well known methods for solid modeling, another technique known as Oct-tree (a tree based hierarchically oriented approach) is becoming popular in recent years [30,45]. This Oct-tree data structure is basically an approximation of three dimensional object, by a set of cubes in various sizes. This method also suffers from several limitations.

As the Oct-tree is an approximation of an object with smooth free surfaces by cubes, it is inevitable that the encoded object entails some notched surfaces [20]. In order to avoid this jagged surface, it is necessary to represent an object by a very deep level Oct-tree, which requires enormous data storage and a high speed processor. This characteristic considerably affects the important advantages of Oct-tree, such as, processing speed, memory economy, etc. Also, because of this approximation, only approximate integral properties of object can be calculated.

Geometric transformations are also problematic in this scheme. As geometric transformation is a sort of combinatorial problem in the Oct-tree representation, a manipulation sequence affects the calculation time considerably. Another major limitation of Oct-tree data structure is the difficulty of incorporating it into existing graphic software systems [13]. An object created on a system built around either Boundary Representation scheme or primitive solids, and transformed into an Oct-tree, to take avantage of boolean operations, can no longer be reconstructed back to a Boundary Representation scheme. This irreversibility of the transformation is a drawback of using Oct-tree in conjunction with existing Computer-Aided Design/ Computer-Aided Manufacturing (CAD/CAM) systems and databases. For display operations also, it is not a desirable data structure [79].

1.2 MOTIVATION FOR THE PRESENT WORK

In the light of the discussion about the computational and other problems associated with the popular solid modeling schemes, it is observed that these methods have limitations in some form or the other. This outlines clearly a need for developing new techniques or improvements in the existing techniques. Some of the desirable 'design requirements' in a computer-aided design environment are discussed in this context.

The acceptance of any computer-aided modeling system heavily depends upon the flexibility of the modeling technique. The approach should be simple, reliable and unified one, so that a wide-range of 3-D objects from polyhedral to solid objects with curve surfaced could be modeled. As far as possible, the method should be sound enough to model the desired shapes directly by avoiding frequent boolean operations.

Designing is an active process. As the design proceeds, various ideas strike for improvement. It should be possible to feed these information with computer assistance. This could be attained only if the system is provided with a good User-interface. Through better User-interface, not only the shape of the object in the designer's mind can be made, but also he gets the feel of designing with his own hands. This also allows him to correct erroneous instructions at any stage of the design. There can also be situations for modification of models by combining different object models. Such modifications can be done through boolean operations on solid models. The system should be supported with these facilities, so that the designer-user will be able to model the shapes of his interest. Incorporation of suitable algorithms for removing the boundary of the model, that are not visible from a given vantage point, is considered as another essential requirement. This will ensure the user a realistic display of the modeled object. Inclusion of software for the computation of volume, moment of inertia and similar properties of solid models is also a desirable requirement in a computer-aided design set up.

In view of the above discussion, it is clear that there is a need for developing new methods or improvements in the available schemes for the representation, manipulation and/or display of solid objects and also to incorporate design requirements, like, friendly user-interfaces, modification facilities, hidden parts removal algorithms, engineering analysis software, etc. These observations provided motivation for this thesis work.

1.3 STATEMENT OF THE PROBLEM

To state the specific tasks, which the present work is intended to cover, a list of major objectives is given below:

1. To explore the possibilities for the development of a new representational technique for geometric modeling of solid objects.
2. To develop Hidden-line elimination algorithms for efficient display of solid models, based on the proposed technique.
3. To provide suitable user-interface to the geometric modeling system based on the suggested method.
4. To develop necessary methods for the computation of integral properties like, volume, moment of inertia, etc. of the proposed technique based solid models.

1.4 ORGANIZATION OF THE THESIS

A chapter-wise summary of the thesis is as follows :

In Chapter 2, an overview of geometric modeling systems and general considerations are discussed.

This Chapter begins with the historical developments of geometric modeling systems. General properties of various representational schemes of solid objects, human-computer interface and application areas of geometric modeling systems (GMS) are also presented in this Chapter.

The concept of new representational technique and the data structure along with suggested definitions are explained in Chapter 3. The usefulness of this scheme to perform boolean operations and the development of curved boundaries are also discussed, with supporting photographs.

In Chapter 4, two hidden-line removal algorithms for the display of the Hex-tree represented solid models are explained. These algorithms are suitable for line-drawings display devices.

The implementation details of the Hex-tree based geometric modeling system with photographs of solid models produced by the system are given in Chapter 5. An overall idea about the various facilities incorporated in this system for a friendly user-machine interface are also explained in this Chapter.

The computation of volume, center of gravity and moment of inertia of Hex-tree based solid models are discussed in Chapter 6. Computed results of a typical model are also shown.

Finally, the summary of the research work and areas for further investigations are presented in Chapter 7.

—

CHAPTER 2

OVERVIEW OF GEOMETRIC MODELING SYSTEMS

2.1 HISTORICAL DEVELOPMENTS

The use of graphic displays as output devices for computers has been experimented from the very early days of computers; in the year 1950, for example, a cathode ray tube was attached to MIT's Whirlwind I computer to generate simple pictures. However, it was only a decade later the potential of interactive computer graphics was realised when, in 1962, Ivan Sutherland developed the 'SKETCHPAD' as a part of his Ph.D work at M.I.T [65]. As its name indicates, one can feed into the computer graphical information like points, lines and curves by drawing them on the CRT screen with a pen like tool. This experiment demonstrated the possibility for human beings to communicate with a computer through pictures, and it launched the new subject of interactive computer graphics. In the late sixtees several projects in interactive computer graphics were taken up not only in universities but also in research laboratories and industries like Bell Telephone labs, General Motors Company and Lockheed Aircraft Corporation [48].

From the very beginning, industrial applications of computer graphics were widely recognized. Layout design of printed circuit boards and generation of photographic masks for the production of integrated circuits were done using computer graphics techniques. In these applications, the objects handled are purely two dimensional and their geometry is simple. In the area of mechanical engineering design and manufacture, we are mainly dealing with solid objects and their shape is an important information for their designers and manufacturers. Traditionally, drawings are used to communicate shape information among various persons involved in the design and manufacture of a product. The first efforts in computer aided design, naturally, were to computerise drafting operations. Graphic input devices like digitizers could be used to convert existing drawings to their computer representations; computerised drafting packages provided help in interactively creating new drawings, and modifying existing ones. The usefulness of these drafting packages were very limited since they were merely trying to imitiate manual drafting operations. The first step in increasing the power of these systems was to incorporate in them three dimensional capabilities. With a three dimensional representation of an object, many views of the object can be produced from a single representation, unlike in the case of drafting packages where each view needs a different representation. For a 3-D representation, a list of edges of the object, known as wireframe representation, is usually stored in the system. Wire frame representations are ambiguous in the sense that a single wireframe representation may correspond to more than one solid object [57]. This serious drawback of the wireframe representation prevented the automation of many useful operations on solid objects, such as, mass property calculations, sectioning, hidden line elimination and so on. Geometric modeling systems were introduced to remove these handicaps.

Geometric modeling systems were first introduced in the early seventies. Prominent among the early systems were the BUILD system of the Cambridge University, U.K. [9], and the Part and Assembly Description language (PADL) system of the University of Rochester, U.S.A. [50]. The Build system used the Boundary Representation Scheme in which solid objects are defined by specifying their bounding surfaces. PADL used the Constructive Solid Geometry Scheme (CSG) for representing solids, a technique in which an object is built up from certain simple primitives like cubes and cylinders by means of geometric transformations and boolean operations. A number of other systems based on these two solid representation schemes were also developed at many universities in U.S., Europe and Japan [57,3]. Once the possibility of representing solid objects unambiguously and of performing operations on them with the aid of computers was demonstrated by these experimental systems, some industrial organizations and software firms began to develop commercial geometric modeling systems. GM solid [8,7], based on PADL, developed at the General Motors, and ROMULUS [71], based on the BUILD Modeler, developed by the Shape Data Ltd., are examples of such systems. These systems had very good user interfaces and they could be incorporated into computer aided engineering systems which are capable of computing several useful properties of solid objects and also giving assistance in their manufacture.

Along with the development of experimental systems, attention was also paid to build up a theoretical basis for solid modeling, especially by researchers of the Prouction Automation Project of the University of Rochester [55]. Among their contributions include the development of a rigorous mathematical model of solid objects in terms of compact regular semianalytic sets and regularised boolean operations, study of formal properties of representation schemes, and development of efficient algorithms for the boundary evaluation of solids defined by the CSG scheme. Also around this period, many interesting applications of solid modelers like finite element mesh generation, interference analysis, kinematic simulation, Numerical control program generation and verification, process and assembly planning were attempted [35]. Another important development in the late seventies is the introduction of a new solid representation scheme known as Oct-tree [29,30]. Many algorithms for representing solids and performing operations on them using the Oct-trees representation are now being developed [30, 16, 45, 79, 80]. The development of another data stucture known as 'Polytree' which is a generalization of Oct-tree data structure was also reported [13].

Attempts are also being made to extend the geometric coverage of geometric modeling systems to include solids bounded by sculptured surfaces. These are surfaces like those of car bodies, ship hulls and aircraft fuse lages which cannot be defined by a single mathematical equation. Even though the area of sculptured surfaces is well developed, the use of these surfaces in defining solid objects is of recent origin, and many problems need to be solved before they can be successfully incorporated in commercial GMS. A work in the geometric modeling of solids bounded by sculptured surface was reported in [51].

2.2 GENERAL PROPERTIES OF REPRESENTATION SCHEMES

A solid representation is a symbol structure over a finite alphabet specified according to some syntatic rules. A representation scheme is then a mapping from the mathematical model space to a set of such symbol structures [55].

Let M be the mathematical modeling space for solid objects, i.e., M is the set of all subsets of the three dimensional Eucledean space E_3 which are bounded, closed, regular and semi-analytic. Members of M are called r-sets. Let R be the set of all syntatically correct representations produced by a grammer. R is called a representation space. Then a mapping s from M to R is called a **representation scheme.** The properties of representation schemes can be categorised as follows :

2.2.1 Formal Properties

In a precise mathematical fashion, the various properties are discussed here. **Domain** of the representation scheme is the domain of s. The domain of a representation scheme characterizes the descriptive power of the scheme. Members of R included in the range of s are called **valid** representations, i.e., they correspond to elements of M and are not nonsense objects. Validity is equivalent to semantic correctness. One of the issues in geometric modeling system design is to decide that who will be responsible for checking validity of objects in the system. Even though it is good to have the system automatically check validity of its objects, in some representation schemes, the validity check is a costly operation in terms of computer time and hence is left to the user. In such a system, if the user fails to make sure that the objects he has created are valid objects, then later on, it may lead to serious troubles and probably program termination when operations are attmpted to be performed on invalid objects. Validity checking by user is not always possible in a system which tries to automate certain stages of design activity because in such systems, objects are sometimes created by programs within the system.

A representation r in the range V of the scheme s is said to be **complete** or **unambigous** if it corresponds to a single object, i.e. if $s^{-1}(r)$ contains only one element. The representation scheme itself is **complete** or **unambigous** if all its valid representations are unambigous. A complete representation of an object has enough information to distinguish that entity from all other entities in the domain D of the scheme. So, if we exclude properties defined by functions that are not computable, then theoretically a complete representation provides all necessary information to answer any question asked about the geometry of the object. Completeness of the representation scheme is hence an essential property of general purpose geometric modeling systems which are built to tackle a range of applications not known in advance.

An object is said to have **unique** representation if there is only one symbol structure that can be used to represent it. A representation scheme s is unique if all the objects in its domain are uniquely represented, i.e, the mapping s is a single valued function. Uniqueness of the representation scheme helps in easily determining equality of two objects in it. In the absence of uniqueness, two objects A and B can be tested for equality only by (i) computing 'S' using the expression S = (A-*B)U*(B-*A), and (ii) checking whether S is null object or not.

2.2.2 Informal Properties

There are a number of desirable properties of representation schemes which are informal in nature. Some of the informal properties are now discussed. **Conciseness** is one of the properties which refers to the 'size' of representations in a scheme. Concise representations are convenient to store and to transmit over data links, and contain relatively few redundant data. In practical systems, it is common to add redundant information to object representations to help the speeding up of certain frequently performed computations. Another desirable property of representations is **ease of creation**. The ease with which (valid) representations may be created by users of modeling systems is of importance, especially if the users are human. Concise representations generally are easier to create than verbose ones. Some systems with verbose representation scheme provide input subsystems to help the users to conveniently create objects. Yet another desirable property of a solid representation scheme is **efficacy in the context of applications**. Even though a complete representation contains enough informations to do most of the application processing, it may happen that given a representation, there is no convenient or efficient way of implementing some algorithms.

Since no single representation scheme available which is good for all applications, practical geometric modeling systems do sometimes contain multiple representations of the same object. There are a number of problems associated with multiple reprsentations. First of all, the system should ensure consistency, i.e. different representations of the same object should agree with each other. Secondly, algorithms should be provided to convert from one representation to another. Sometimes it is not possible to convert one representation of an object to another representation of the same object since the domain of coverage of the two representations may not be same. In such cases approximate conversion algorithms are employed, which compute an object in the other representations that closely approximates the original object.

2.3 REPRESENTATION SCHEMES FOR SOLID MODELS

There are a number of different schemes for representing solid models in a computer.

2.3.1 Constructive Solid Geometry (CSG)

This is probably the most important scheme for representing non-sculptured or functional objects. In this scheme, a solid object is created by combining several primitive solids of simple shapes. The primitives commonly used are cubes, cylinders, spheres, cones, etc. Geometric transformations are provided to change their size and to position them in space. They are then combined together by boolean operations (Union, difference and intersection). So, in this scheme an object is represented by a binary tree whose leaf nodes represent either primitives or parameters of geometric transformations and whose internal nodes represent geometric transformations or boolean operations. A formal definition of such a CSG tree is given below:

```
<CSG tree> :: = <Primitive leaf>|
                <CSG tree><Boolean operator node><CSG tree>|
                <CSG tree> <transformation node> <transformation
                                                   parameters>
```

There are a number of variations possible from the scheme mentioned above. One of them is to use half-spaces as primitives instead of bound solids. A half space is defined as a set of points in the three-dimensional Eucledean space E_3 given by $ki\{P:F(P)<0\}$, where F is a real valued analytic function in E_3 and k and i are respectively the closure and interior operators in the usual E_3 topology [53].

One of the main advantages of the CSG scheme is the possibility of ensuring the validity of objects represented by it. Primitive shapes like cubes, cylinders, spheres, cones, etc. represent valid physical objects. Since the operations performed on these primitives are regularised boolean operations, the resulting objects are also guaranteed to be valid [54]. However, this advantage is lost if half spaces are used as primitives. In this case there is a need to ensure that the objects in the system are bounded and it cannot be done without spending a good amount of computational effort.

Given a CSG representation, there exists only one solid that corresponds to it and hence the CSG scheme is unambigous. On the other hand, given an object there may be more than one way of decomposing it into primitive solids and so this scheme does not possess the uniqueness property. Consequently, special algorithms are necessary to determine whether two representations in the scheme represent the same object or not. One important advantage of CSG scheme is its consciseness. If the object to be represented can be split up into the primitive solids available in the scheme, then the user will be able to specify that object with the help of a few primitive instances and boolean operations on them.

It is quite easy to perform boolean operations on two objects represented in a CSG scheme. All one has to do is to create a new CSG tree with its root node containing the required boolean operation and with the two sons of this root being the two operands of the boolean operation. There is no other solid representation scheme in which the boolean operations are so easy to perform. The simplicity of performing boolean operations is due to the implicit nature of the CSG scheme of representing solids. This very same implicitness is a draw back of the scheme for most other operations we would like to perform with solid objects.

One example of such a difficult operation is the display of objects. In order to display an object we should know what portion of the solid is visible. Clearly it is only a subset of boundary of the object that will be visible and all the interior portions are invisible, assuming that the object is opaque. The boundary of the solid object is a subset of the boundaries of the primitives that constitute the object. It may be a proper subset since some of the primitives may penetrate each other. There are algorithms available to determine the boundary of an object defined by CSG tree [8]. Once the boundary of an object is evaluated, it can be displayed after removing the hidden portions of the boundary, if necessary. Another approach to displaying objects represented by a CSG tree, known as ray casting [61] is to determine the visible portion of the object directly from the boundary of the primitives that constitute the object, instead of going through the intermediate step of evaluating the boundary of the object. In this method rays (semi-infinite straight lines) along all directions and originating from the point of observation are first generated. Then the points of intesection of these rays with the boundaries of the various (transformed) primitive instances are computed. These points of intersection are then sorted and the point on each ray which is nearest to the point of observation is then displayed. Special care must be taken to avoid repetitive calculations and make this algorithm efficient [2,57]. One of the disadvantages of this method is the necessity to recompute the intersection of rays and primitive surfaces whenever the viewing parameters change. In the case of conventional approach, boundary evaluation can be performed once for all and only the hidden portion removal is to be repeated when the viewing parameters change. The basic operations in ray casting can be done in parallel for various rays so, if a number of special purpose hardware is available for doing this, then display by ray casting will become faster.

Because of the ease of creation, consciseness and guarantee of validity, the CSG scheme is used as the main representation scheme in many existing geometric modeling systems. Most important among these are the PADL systems developed by the production Automation Project of the University of Rochester and the GMsolid, based on PADL-2, developed by the General Motors. Other systems based on the CSG scheme are GDP of IBM, TIPS of the Hokkaido University, Synthavision of Magi Inc., Unisolids of McAuto, Catsoft of Catronix, etc. [3,57].

2.3.2 Pure Primitive Instancing

In this scheme there are a fixed number of predefined object types. An object of a particular type is completely specified by a few parameters. There is no provision to combine objects to make complex solids, and hence the domain of the scheme is very limited. When an object type is included in the system, the condition for the validity of an instance of this can be easily expressed in terms of its parameters and the corresponding checks can be incorporated in the system. This type of representation is concise and easy to use, but algorithms to compute properties of solids represented in this scheme will have to treat each type of object separately.

2.3.3 Boundary Representation Scheme (B-Rep)

One of the most widespread representation techniques used in geometric modeling systems is the Boundary Representation. In this scheme, the solid is represented by its bounding surface. This surface is usually specified by means of a list of faces and each face is represented by its equation and bounding edges. The edges in turn are represented by their equation and endpoints. The geometric coverage of the scheme depends on the type of surfaces available to represent faces of solids in the scheme. Planar and quadratic surfaces are sufficient to cover most of the nonsculptured solids. Sculptured solids require bicubic parametric patches with facilities to modify their shape interactively by the user. Among the available representation schemes, Boundary Representation provides maximum geometric coverage.

It is easy to display objects from their Boundary Representation. For fast display generation during interactive sessions the entire surface can be drawn without eliminating hidden portions. For a more realistic display hidden-line or hidden-surface algorithms can be used. In the case of line drawings it is necessary to have techniques to detect silhouette edges of curved surfaces. In the case of raster scan devices shading algorithms are needed to compute the intensity of each pixel to be displayed. Graphic interaction is easier in the case of an object represented by its boundary. Boundary information is also useful in many other problems like collision detection of an object which moves in the presence of other objects.

Implementation of boolean operations in this scheme is somewhat difficult. If A and B are two solid objects and C is the result of performing a boolean operation on A and B, then it is required to compute the boundary of C from the boundaries of A and B. Clearly, the boundary of C is a subset of the union of boundaries of A and B. In order to determine which portion of the boundaries of A and B constitute the boundary of C we need a classification algorithm. The objective of this classification algorithm is the following. Given a surface and a solid object, determine which portion of the surface lies within, on, and outside the solid object. The basic idea behind the classification algorithm is to split the surface into a number of patches such that each patch is completely inside, on, or outside the solid. In order to determine this splitting, we first compute the curves of intersection of the surface with the boundary of the solid. These curves of intersection together with the original edge of the surface effects the desired splitting of the surface since the classification of the surface will change only when it crosses the boundary of the solid. Using this classification algorithm we can classify the boundaries of each of the solid A and B with respect to the other solid and then pick up and connect together the correct surface patches to form the boundary of C. One thing that is clear from this brief description of the algorithm to compute boolean operation is that, it involves heavy computation. Another problem is in finding the curve of intersection of two surfaces when a number of surface types are present in the system. In the case of bicubic surface patches, the curve of intersection between two such surfaces cannot be exactly represented by a single equation of low degree, but can only be approximated by straight line segments. Further it is also inconvenient to represent a portion of such a surface bounded by these approximate curves so that it can take part in subsequent boolean operations. These type of problems have compelled the designers of some of the existing geometric modeling systems with sculptured surface facility to disallow boolean operations to be performed when two sculptured surfaces intersect.

Inspite of some of its drawbacks, this scheme is very popular and is used in many geometric modeling systems. The BUILD Modeler developed at the Cambridge University, U.K., is one of the earliest examples of a GMS using Boundary Representation. Two commercial systems ROMULUS and DESIGN based on the BUILD Modeler also use the Boundary Representation scheme. GEOMED, UNISCAD, CADD, COMPAC, EUCLID, GLIDE, MEDUSA and PROREN-2 are some of the other systems that use the Boundary Representation [3,57]. The main advantage of Boundary Representation is that it is best suitable for display of objects. Display of surfaces have been studied for a very long time in computer graphics and many algorithms like hidden portion removal and shading are well developed. Current research activity in the area of geometric modeling may increase the importance of the types of representations that are peculiar to solid modeling like the CSG and Octtree representation, but as long as display of object is necessary Boundary Representation will remain as an important solid representation scheme.

2.3.4 Cell Decompositions

In this approach, a solid object is represented by dividing or decomposing its volume into smaller volumes or cells which are mutually contiguous and do not interpenetrate. The cell shape is not necessarily cuboid, nor are the cells all necessarily identical in shape. For example, a rectilinear polyhedron can be triangulated by decomposing it into a number of tetrahedra which are either disjoint or are touching along a common face, edge or vertex. Curved polyhedra can be similarly be decomposed into curved tetrahedra. The domain of the scheme depends on the type of cells that are available. Validity of the representation is not easy to check. The representation is unambiguous.

In general, cell decomposition produces an approximate representation of an object, since, some cells will straddle the object boundary, so that they are partly in and partly out of the object. Representing the object by all cells (those entirely in, those entirely out, and those partially in the object volume) will therefore include some 'empty space' in the description. Such an approximate representation is often troublesome, because it can change the 'topology' of the object. Thus, a small hole or void in the object may not be included in the cellular model, if it lies entirely within one of the cells - it will be replaced by the cell enclosing it. We can obviate many of the problems associated with these approximation by allowing the cell shape and size to vary, so that it conforms to the object boundary.

Cell decompositions of solid objects are useful for performing finite element analysis. Spatial occupancy enumeration and Oct-tree representations are two particular types of cell decomposition scheme [35].

2.3.4.1 Spatial Occupancy Enumeration

The portion of the three dimensional space that is of interest to the user is divided uniformly into a number of small cubes, called Voxels, of same size and faces parallel to the coordinate planes. An object can then be specified by enumerating the voxels that constitutes it. The scheme is unambiguous and unique but the domain is restrictd to a subset of the set of rectilinear polyhedra. The scheme is very verbose, but some improvements are possible by choosing voxels that suit the particular application area at hand. Another variation of the scheme is to choose voxels that are of varying size.

2.3.4.2 Oct-tree Representation Scheme

Another solid modeling scheme which is becoming popular in recent years is Octtree. This scheme is based on a hierarchy of different cell sizes. The first level contains the largest cells, and at the second level these cells may be sub-divided into smaller cells. At each subsequent level further subdivisions may be performed, giving rise to smaller and smaller cell sizes and hence to higher and higher spatial resolution. Usually the linear resolution doubles at each successive level, and always the set of smaller cells derived from a given cell on the previous level 'justfills' the volume of this 'parent' cell. The cells are classified as empty, full or partial depending on whether it is entirely outside, entirely inside or partially inside the object to be represented. Cells which are designated as empty or full are not sub-divided further. Cells which are designated as partially full are sub-divided and hence taken to the next level. The whole sub-division process can be viewed as forming a (rooted) tree structure with the root node representing a single enclosing cell, called the universe and the first level branch nodes representing a sub-division into the largest cells. Some (but usually not all) first level nodes then have branches to second level nodes, and so on for each subsequent level in the tree. The cells are cubes and are subdivided into eight sub-cubes of half the linear dimensions, so this hierarchical scheme is known as Oct-tree. These eight sub-cubes are also known as eight octants. This recursive sub-division process continues until a level is reached that the octants become partially filled or empty. The accuracy of the object to be represented depends on going to very deep level Oct-tree. Oct-tree scheme can be used to represent any complex object shapes.

This scheme is a generalisation of the quadtree scheme used in image processing to represent two dimensional areas.

Quadtrees in turn are a generalisation of binary trees used extensively in computer science. Since the cubes are of varying size, less number of cubes are sufficient to approximate an object and hence Oct-tree is more space efficient than spatial occupancy enumeration. The computation of integral properties of an object represented by an Octtree is easy because the cubes are quasidis-joint (i.e., they are disjoint or just touch along a face).

Because of the enormous number of cubes required to approximate objects, it is difficult for a human user to directly specify objects in terms of their Oct-tree representation. So, in geometric modeling systems based on Oct-tree representation, some convenient representation for specifying objects and a conversion algorithm to convert from the input representation to Oct-tree is to be provided. Validity of an Oct-tree representation is easy to check. Since Oct-trees are usually created by programs, validity is guaranteed. The Oct-tree representation is unambigous. Any complex object shape can be approximated by Oct-trees by going to very deep levels. One of the special advantages of this scheme is the possibility to work with variable precision. In interactive sessions when quick response is important we can work with a few high levels of the Oct-trees giving a coarse approximation of the object. One of the most important factors to be taken care of in Oct-tree method is that of an efficient storage scheme. It is not advisable to store the explicit tree structure as in that case pointers will occupy a lot more space than useful data. Tree codes and leaf codes schemes are two main approaches for compact storage and both are based on similar techniques used for quadtrees [78].

Algorithms for performing boolean operations are very simple in Oct-tree representation. The two objects on which the operation is to be performed are first stored in the same universal cube, if they are not already so, and then trees corresponding to these objects are simultaneously traversed. At any of the leaf nodes of the trees the boolean operations are tivial to perform and the result of such operations are used to create a new tree. The nodes of this tree are merged together whenever possible, and the resulting tree represents the object which results from the operation.

Even though quadtrees were being used in image processing for quite some time, their generalisation to Oct-trees and its application to modeling of solid objects is relatively of recent origin. It was first suggested by Hunter in 1978 [29]. Oct-trees were also studied independently by several authors around this time [30, 45]. Many algorithms for geometric transformations, boolean operations and display were developed and some experimental systems based on Oct-tree representation were built [16,29,79]. But commercial systems based on this scheme are not available. Huge amount of storage required to represent objects and slowness of some of the algorithms for object manipulations are the probable reasons for the lack of availability of such systems. In many algorithms that operate on Oct-trees, there are some operations that can be done in parallel on all eight octants of a cube. This factor has encouraged researchers to build special purpose hardware to perform operations on Oct-trees in parallel. When this type of hardware becomes widely available, Oct-tree representation will be more popular and probably have an edge over other forms of object representations.

2.3.5 Sweep Representations

When an object moves in space, it sweeps out a volume and this resulting solid can be represented by the object which moves, and its trajectory. Two particular instances of this general scheme are well known, viz., translational and rotational sweeping. In translational sweeping, we take a subset of the two dimensional space and moves it in a direction perpendicular to the plane of this object. The resulting object can easily be represented by the boundary of the planar set being translated and the distance through which it is moved. Rotational sweeping is similar to translational sweeping, but instead of translatory motion, the planar object is rotated about a fixed axis to produce a solid object. The domains of translational and rotational sweepings are limited respectively to objects having translational and rotational symmetry. The object generated by the sweep operation is valid and unambigous, if the object being swept are respectively valid and unambiguous. The representation is not always unique.

Instead of planar objects, solids also can be used in the sweeping operation. The ability to compute the volume swept by a solid object is of great use in several practical applications. One example is dynamic interference checking where it is required to find out whether a solid object can move along a given path without colliding with its surrounding objects. It can also be used to check the correctness of cutter path movements in a part program to produce a desired object from a given workpiece. The general sweeping technique is not mathematically well understood and algorithms for computing properties of solids represented by this method are not available.

2.4 HUMAN - COMPUTER INTERFACES

In the previous section, we have seen various schemes for representing solid objects. Whatever be the representation chosen, facilities should be given to the user to conveniently create objects inside the system and to examine the shape of objects that are already created. Some times it may be necessary to have extra representation forms whose sole purpose is to serve as convenient user interfaces. Such representations are called volatile representations. The input created by the user in a volatile representation may not be stored in that representation, but will be converted to the main representation of the system which is also called working representation. Similarly, when the working representation is not the boundary representation, the system may perform a boundary evaluation for the purpose of display generation and the resulting boundary representation may not be stored permanently. In this section, we are discussing some of the commonly used input and output techniques in geometric modeling systems.

Input techniques

The simplest way to provide the input is to give a sequence of characters. This sequence of characters may either be typed in from a terminal, or be specified by a sequence of menu selections. This character sequence may contain only commands to create geometric objects, or it may be imbedded in a programming language. In the former case the user need not be a trained programmer, whereas the later case a user with the knowledge of the language in which the commands are imbedded will be able to use the full power of that language to create complex objects.

Editing facilities must be provided when the user wants to make minor modifications on an existing object, otherwise it will be very inconvenient to respecify the modified object from the scratch. Also the user should be able to save on the backup store the objects created in one interactive session. He should later on be able to read this object definition and use it directly or edit it to make any desired changes. It is also good if facilities are available to create a library of frequently used objects. Another important facility that can be provided is the ability to convert the existing manual representations of solids to machine representations. Drawings which are used as the storage medium of geometric information in manual systems can easily be read into the system by means of digitizers. But such drawings are usually ambiguous and algorithms to convert them to solid representations need advice from human uses to create the desired object. A closely related problem is that of converting a wireframe representation of an object to a solid representation of the same object. In a wireframe representation an object is represented by the set of its edges and hence can be ambiguous [56]. Many existing drafting packages use this type of representation and hence it is profitable to have an algorithm to convert from wireframe to solid representation. Algorithms are available for solving the two conversion problems mentioned above [43,75].

For users who are familiar with manual techniques of specifying solid objects, facilities similar to that can be provided. One example of such a facility is the creation of the so-called two-and-a-half dimensional objects, i.e., objects having translational symmetry, by specifying their cross section with drafting like graphic input techniques and then using a translatory sweep to create the solid. The use of graphic input devices for specifying geometric information with reference to objects already displayed on the screen should be provided rather than having to type in coordinate values as sequence of digits. Here all the available techniques of interactive computer graphics can be made use of [48].

Output techniques

Since GMS do have complete representations of solid objects, it is possible to output models of objects automatically so that the use can examine their shape [38]. The easiest method of conveying shape in formations to human users is through display of objects, and all GMS provide this facility. The medium on which pictures are drawn - CRT screen or paper - is two dimensional and therefore solid objects are to be mapped from the three dimensional Eucledean space to the two dimensional Eucledean plane by a geometric projection. Specification of viewing parameters and type of projection can be done by methods available in computer graphics [19]

There are two types of displays possible - line drawings and shaded pictures. In line drawings, only certain curves that lie on the boundary of the object are drawn. In the case of a rectilinear polyhedron these curves are the straight line edges of the polyhedron. For solids covered by curved surfaces, such as spheres, cylinders, cones, etc., there are two types of curves to be drawn - boundary edges and silhouette edges. The position of boundary edges on the object are fixed whereas the position of silhouette edges vary with the point of observation. There are techniques available to detect silhouette edges of quadratic surfaces [74].

The other type of display, namely, shaded pictures, is suitable for raster scan terminals whose screen is made up of a matrix of pixels. For generating a shaded display, we have to compute the set of pixels that constitute the boundary of the object being displayed and the intensity with which these pixels are to be displayed. The first of these two operations is done by scan conversion algorithms, the second by shading algorithms, and these algorithms are discussed in standard books on Interactive Computer Graphics [24,19,48].

2.5 APPLICATION AREAS

Computer models of solid objects have many useful applications. One of the common applications is designing an object having a desired shape. If the required shape can be characterised mathematically, then probably some conventional surface fitting methods may be used to automatically create the shape. But, there are a number of practical applications where the desired properties are based on aesthetical cnsiderations and hence cannot be directly mathematically formulated. In such situations the designer creates a shape, examines it by displaying on the CRT, makes suitable modifications and repeats the process until he is satisfied with the output. So, the display of an object from its computer model itself is a useful application of the model.

In mechanical engineering design, complex machineries are designed, first by designing the components separately and then assembling these components together. One of the design problems to be solved in the assembly process is to check whether the components interfere each other. This is known as static interference checking and can easily be perfomed in a geometric modeling system in which boolean operations on solid objects can be performed. For this we need only computer intersection of pairs of components and check whether the resulting objects are null or not. Algorithms for checking whether an object is null or not when it is represented by a CSG scheme are available [68]. In the case of other schemes null object detection is a trivial computation. Another problem that is closely related to that of static interference checking is the dynamic interference checking. Here, we have an object, a path along which it is to move, a set of objects in the surrounding and the problem is to decide whether the proposed motion of the object is collision free or not. If we are able to compute the volume swept by the moving object, then the dynamic interference chekcing is reduced to static interference checking.

The facilities described above - graphics and interference checking - are available in almost all geometric modeling systems. There are a number of advanced features that are desirable for a GMS to possess like automatic manufacturing, finite element mesh generation, assembly planning and so on. Currently available commercial geometric modeling systems may support one of these advanced features, but it will require considerable research effort before they can be implemented efficiently. Some of the research activities going on in these areas are reported in a survey article on GMS [58].

In the area of Numerical Control (NC) machines, GMS can provide program verification support by taking as input, an NC program, and then displaying the object that will be generated if this program were used on an actual NC machine. In addition to such visual aids, fully automated NC program verification systems were also studied. In automatic systems, there are mainly two types of checkings are done. First of all, for each tool motion, invoked by an NC command, the system verifies whether it is feasible. Other than certain technical aspects such as directional admissibility, spatial accessibility and so on, dynamic interference checking is perfomed to make sure that the tool does not collide with the surrounding objects. The volume swept by the moving tool is then subtrated from the workpiece to get the object that results from the machining operations. This object is then compared with the target object by a same object detection algorithm to check the success of the entire process of machining. Automatic generation of NC program - is also being attempted in the research laboratories.

Finite element method for structural analysis, requires the object under study to be decomposed into a number of small components of simple shapes like cubes or tetrahedra similar to cell decomposition scheme used in GMS. Algorithms for automatically generating such triangulations from their Boundary representations are available. Similar techniques for Oct-tree and CSG representations are being studied.

Because of the availability of unambigous solid representation, GMS are finding applications in a number of diverse areas. In robotics, geometric modelers are being used to solve such problems as that of finding collision free path for the movement of robot arms. Even though integrated circuits (IC) are essentially two dimensional structures, some properties of circuits like capacitance can be accurately determined only from a three dimensional model and at IBM, solid modeling techniques are being used in IC design [36].

Inspite of their inherent limitation that solid objects cannot be unambigously represented, existing commercial wireframe systems provide support to some important applications like numerical control machining. In such systems solid modelers can act as very good frontends. In this scheme, the user can make use of all the powerful facilities of solid modeling systems to create and modify objects. The system then converts them to wireframe representation and uses the existing algorithms for wireframe objects to do the application processing. Such systems will be used in practice until application algorithms for solid modelers are fully developed.

2.6 GEOMETRIC MODELING SYSTEMS

In this section, we describe some of the important geometric modeling systems developed so far. Our aim is not to give an exhaustive list of all systems developed so far, but we are interested in describing only those systems which introduced major innovations and which are influential in the development of many other systems.

We first describe the BUILD group of Geometric Modelers whose main characteristic is the use of Boundary Representation in defining solid objects. This group of systems originated with BUILD-1, a small system developed by Braid I.C. as a part of his Ph.D thesis work at the University of Cambridge, U.K. in the early seventies [9]. BUILD-1 was followed by the development of bigger systems namely BUILD-2, which was a group effort led by Braid, but still conducted in the academic environment [11]. This was an experimental system for investigating the techniques and algorithms for geometric modeling. Even though it was based on Boundary Representation, there were six standard shapes that user could make use of-cube, wedge, tetrahedron, cylinder, cylindrical segment and fillet. For building up complex shapes, not only boolean operations but operators for local modifications like tweaking, fillets, chamfers and draft angles were provided. The topological structure of the boundary of solid objects were stored in Winged-edge polyhedron [4] structure introduced by Baumgart. In this scheme the set of edges of the object are stored along with their adjacency information - pointers to faces and vertices adjacent to the edge. BUILD-2 also used B-spline surface patches for representing curved surfaces and B-spline curves for approximating curves of intersection of these patches with plane surfaces. Boolean operation in the system could not handle objects with two intersecting B-spline surfaces. Being an experimental system its user interface provided only limited capabilities. One of the important facility provided in BUILD-2 was automatic dimensioning and tolerening analysis [27,28]. Two commercial geometric modeling systems were developed based on BUILD system. They are the Design developed by Manufacturing Data Systems Inc. (MDSI) and the ROMULUS developed by Shape Data Ltd. In addition to the facilities available in the BUILD system, these two systems provided very good user interfaces also. They help users in visualising shape of solid objects by providing continuous rotation of wireframe display of objects. This is implemented by the hardware of the display processor and is fast enough for user interaction unlike the software implemented hidden-line and hidden-surface elimination procedures. Both of them were incorporated in existing integrated computer aided engineering systems. More details of these system can be had from [26].

A great deal of research and development work in geometric modeling was conducted in the Production Automation Project at the University of Rochester, U.S.A. One of the important contributions made by the Rochester group was the building up of a firm mathematical foundation of solid modeling. They developed in a rigorous fashion an approach to mathematical modeling of solid objects [52]. The introduction of the concept of set membership classification for the boundary evaluation of objects defined by a CSG scheme is also important [69]. Based on these mathematical foundations, a small but robust modeler, PADL-1, was designed and implemented [50]. PADL-1 used CSG representation where primitives allowed were only rectangular blocks and cylinders whose axes were parallel to coordinate axes. One of the facilities provided in PADL-1 which is of practical importance is the specification of tolerencing and dimension information. This system was developed during 1975-77 by a team of researchers in the University. As a part of making available this new technique widely to industrial users, a bigger project, PADL-2, was started in 1978 [11,57]. This was sponsored by several industrial organisations and a U.S. government agency. The aim of this project was to build a set of procedures that will form the core of several geometric modelers to be built from it by interested users and software firms. The geometric coverage was increased by allowing primitives like block, cylinder, sphere and cone and also permiting any rigid body transformations to be performed on them. Specification of dimension and tolerencing information was not implemented in this system. To demonstrate the utility of the core routines, a sample modeler, P2/mm, was developed based on these core procedures [12]. This system used Boundary Representation as an additional representation. Boundary evaluation algorithms were developed to convert from CSG representation to Boundary representation. Quick wireframe display of objects were then possible. For hidden-line eliminated and shaded displays, ray casting was used. Mass property computations were also performed by Ray casting [61]. General motors, one of the industrial sponsors of the PADL-2 project, built a solid modeler called GMSOLID based on PADL-2. General motors were using computer graphics techniques for a very long period in their design activities. They already had a Corporate Graphic System (CGS) which is a powerful wireframe system supporting many applications. GMSOLID was incorporated into this system so that it can make use of the existing application procedures. More details about GMSOLID are available in [7,8].

The systems described were mainly designed to support mechanical engineering design and manufacturing activities. There are a few geometric modeling systems that were developed for other applications as well. The BDS and GLIDE [17,57] developed at Carnegie-Mellon University were mainly intended to be used for architectural design. BDS or Building Design System was an earlier prototype of GLIDE. These systems are based on Boundary Representation. They use Euler operations to define the topology of the object. Boolean operations on solids are also available. Another important feature of these systems is the incorporation of special data base constructs that are suitable to handle geometric objects.

IBM also has shown interest in Solid Modeling. Initially they marketed the CADCAM system developed by Lockheed Corporation. This system had only 2D drafting facilities. Later on they were supplying the CATIA system from the French Aerospace Company Dassault. CATIA had solid modeling facilities and it was interfaced to CADCAM. Then IBM developed their own system called Geometric Design Processor (GDP) for finding collision free path for robot arms [76,77]. This modeler had only blocks and polyhedral approximation to cylinder, as primitives. Later on a user interface for this modeler was also developed called GRIN (GRaphic INput Subsystem) [18]. Another important problem in the area of solid modeling that was tackled by IBM researchers is how to convert existing geometric database to solid models. Two types of conversion algorithms were developed - one for converting wireframe representations and another for converting projections [43,75]. Since both of these representations are ambigous, conversion algorithms take advice from the human users to construct solid models from these representations.

Attempts have been made to use Oct-trees as the main representation scheme for solid modelers. It was first proposed by Jackins and Tanimoto [30] and several others. Many of the geometric algorithms for processing Oct-tree encoded objects were presented in [45]. A lot of attention is being paid [16,79,80] to develop algorithms for processing Oct-trees.

In addition to the important developments described above, there are a number of systems that were developed in various countries all over the world. In Japan, at the University of Tokyo, a modeler 'HOSAKA' was developed around 1975. This was followed by another system called 'Geomap'. From early seventies, work was being done in Solid Modeling at the Hokkaido University and the modeler TIPS was developed [49]. The system has a representation scheme based on half spaces which are intersected to generate solids, which in turn can be combined by set union to generate more complex objects. The type of half spaces provided are cylindrical, spherical and bicubic. The geometric modeler is a part of a large system for automatic design and manufacture. Work has also been reported from University of Tokyo on sculptured Solids [14] and Oct-tree encoded objects [79,80]. In the continental Europe also, a number of systems were developed. Examples of such systems are the 'COMPAC' of the Technical University of Berlin, 'PROREN' of the University of Rhur, 'EUCID' developed in France, 'EUKLID' developed in

Switzerland, GWB developed in Finland and so on [57].

Geometric modeling systems were first developed in the early seventies. Within a short period of ten to fifteen years, a number of significant contributions have been made by people who were responsible for developing these systems. Very good survey articles are available giving more information on the subject [3,56,57,58]. The views of various industrial users as well as software firms confirm that there is a bright future for solid modeling [46,70]. Within few years we will have enough experience of using this technology in the production environment. The experience so gained will be a valuable feedback to designers of the system that will help them to produce still better systems. Meanwhile, research work is going on in areas such as the development of new representational schemes, in corporation of sculptured surface capabilities, development of more efficient application processing algorithms, linking of computer aided design with computer-aided manufacture, incorporation of efficient methods for integal properties calculations, development of better user interfaces, introduction of Artificial intelligence in solid modeling, and so on. By the end of this decade or early nineties, solid modeling will, probably be well established and become a standard tool in industrial design and manufacturing.

—

CHAPTER 3

THE HEX-TREE APPROACH

3.1 INTRODUCTION

Out of the several schemes evolved for computer-aided geometric modeling of solid objects, Constructive Solid Geometry, Boundary Representation and Oct-tree are the 3 major representational techniques widely recognized. Constructive Solid Geometry scheme uses a building-block approach whereby simple primitive shapes are combined using boolean operators to make complex shapes. In Boundary Representation, a solid object is represented in terms of its boundary (enclosing surface). Oct-tree representation approximates the solid object with the variably sized spatial cubes obtained by dividing a cubic 'object space' large enough to accommodate the object to be modeled. However, these methods have various limitations as discussed earlier. Hence, search for new methodological basis for solid modeling has become essential.

As it is known, in the raster CRT graphics display devices, the shapes are displayed by connecting discrete cells called picture elements (Pixels) that are arranged in a two dimensional matrix form. This shaping process can be viewed as a 2-D constructional procedure, originating from a pixel and growing in the X and Y coordinate directions. If this growing process is extended in the Z coordinate direction also, it can be culminated into a 3-D shaping procedure. This idea inspired for the development of a new geometric modeling approach for solid objects, by applying the pixel concept of 2-D with a cubical cell (for 3-D). Thus, 3-D object shapes can be built by constructing cubical cells on the faces (representing the coordinate axes) of an initial cubical cell as shown in Fig. 3.1

This construction procedure can be considered as forming a tree structure, with the root node representing the initial cubical cell and the first level branch nodes representing 6 cubical cells on the faces of the rooted initial cubical cell. Some (need not be all) first level nodes then have branches to the second level nodes, and so on for each subsequent level in the tree. Hence, this hierarchical structure is named as HEX-TREE. In this constructional approach, solid object is modeled by the collection of equally sized cubical cells. This new representational method [33] is explained in this chapter.

3.2 HEX-TREE DATA STRUCTURE AND REPRESENTATIONAL DETAILS

Three-dimensional geometric modeling systems maintain internal data structures that allow users to manipulate and display arbitrary objects. These data structures permit implicit or explicit definitions of the boundaries of a solid region within a region of space. A more general representation depicts a solid body as a three dimensional array. The array contains information indicating the material found at every point in space. If a large region of space contains a single material, then many adjacent elements of the space array contained within that region will have the same value.

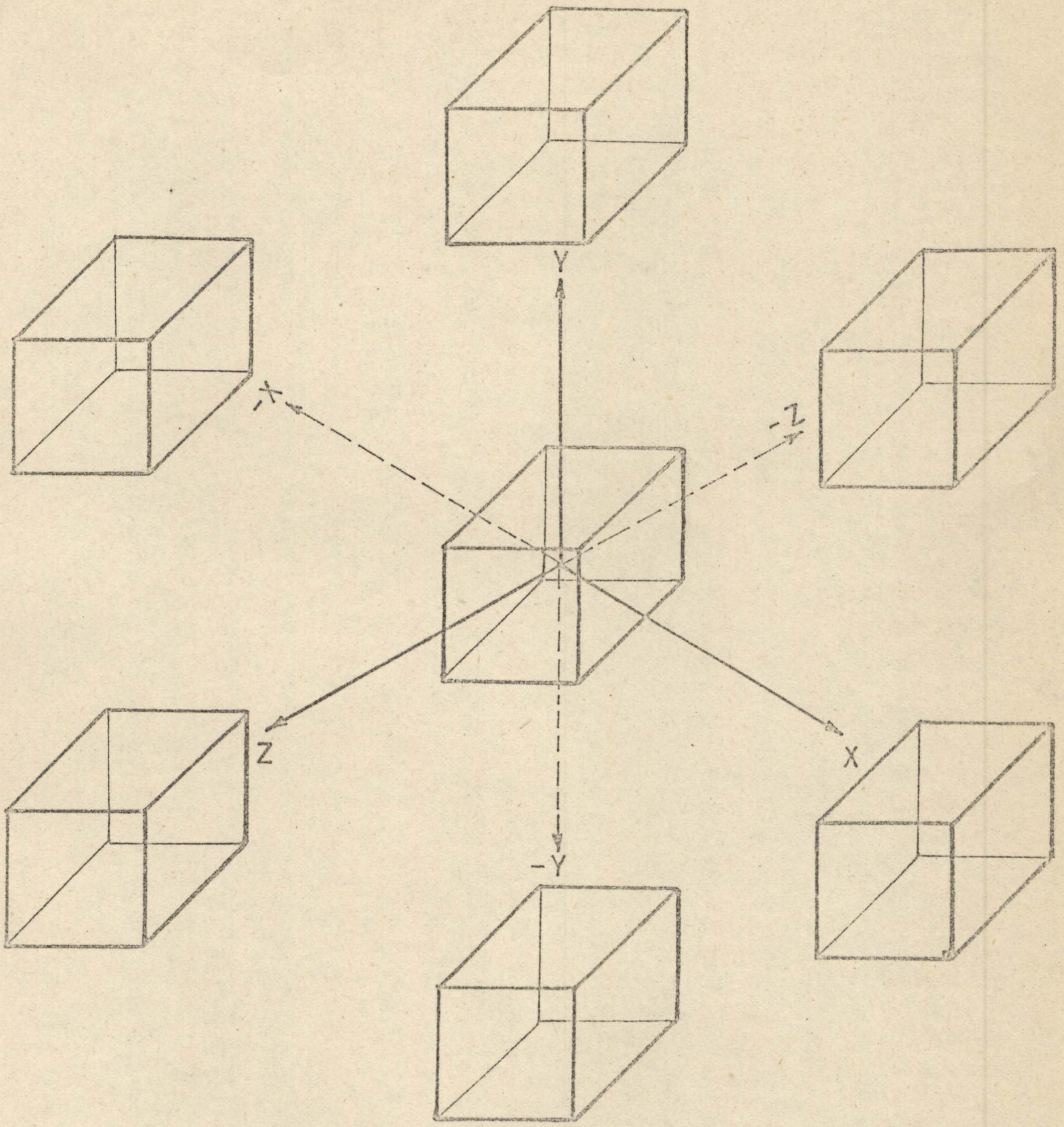


Fig. 3.1 Strategy for the Modeling

In the Hex-tree representational approach, any 3-D object can be made using a single cubical cell as primitive. The number of nodes at the nth level is equal to 6^n, where n varies from zero to any finite number in this hierarchically oriented, recursive type data structure. The full identity of a node (i.e. cubical cell) is represented in 6 bits of a word length, indicating the life entities in all six directions (i.e. the Bits 0 and 1 represent the presence of a cube in positive and negative X-directions, Bits two and three represent positive and negative Y-directions and Bits four and five represent positive and negative Z-directions). Fig. 3.2 illustrates the details of the Hex-tree representation. In this figure, the root node corresponds to the initial cubical cell. Fig. 3.3 shows the Bit presentation of a word.

3.2.1 Cell description

As it is mentioned, the primitive for the proposed representational technique is a cubical cell. Fig. 3.4 shows the cubical cell with various details. A left handed co-ordinate system is followed in this geometric modeling system based on Hex-tree.

3.2.2 Definitions

The different terminology suggested in this method are explained below :

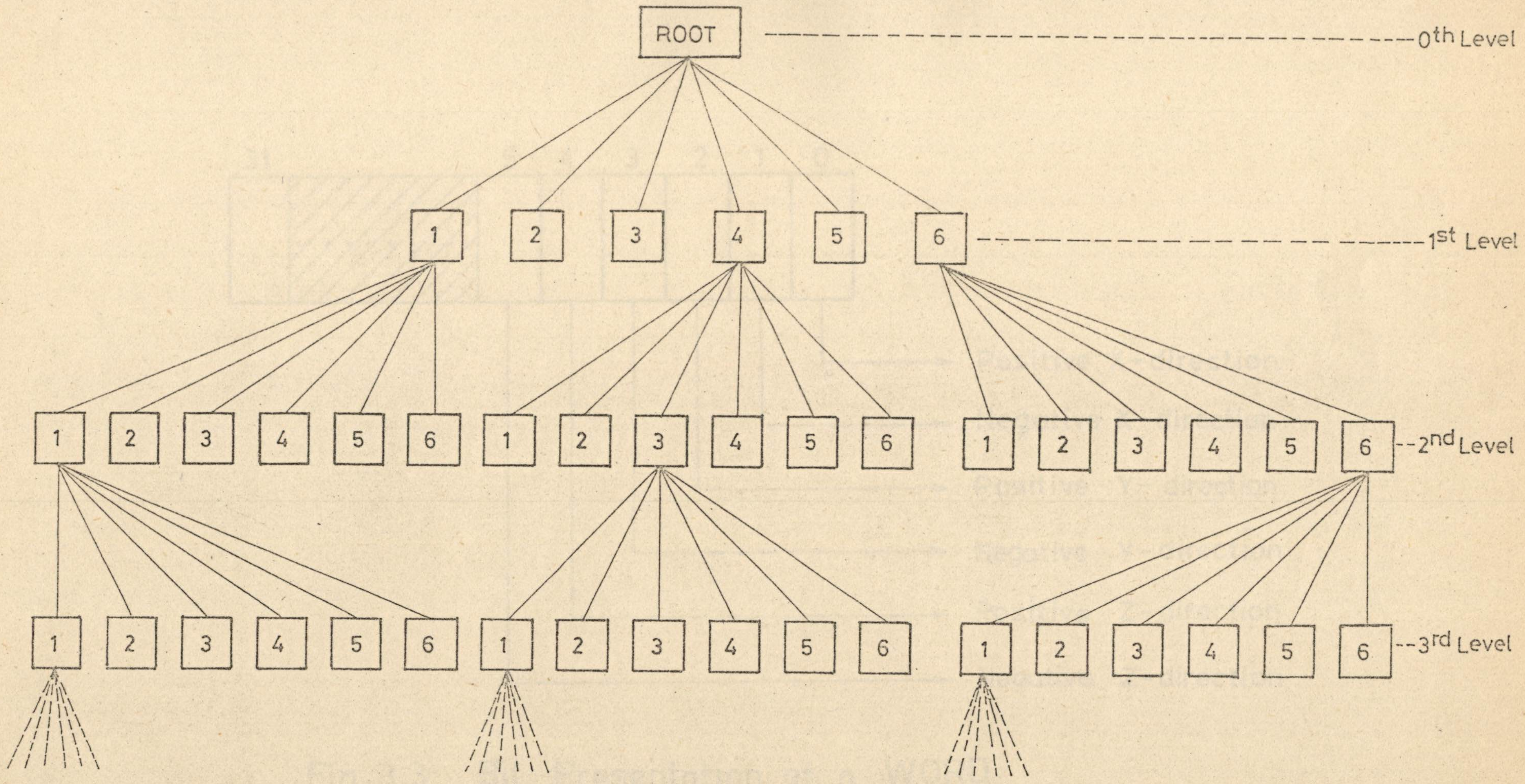


Fig. 3.2 Hex - tree Representation

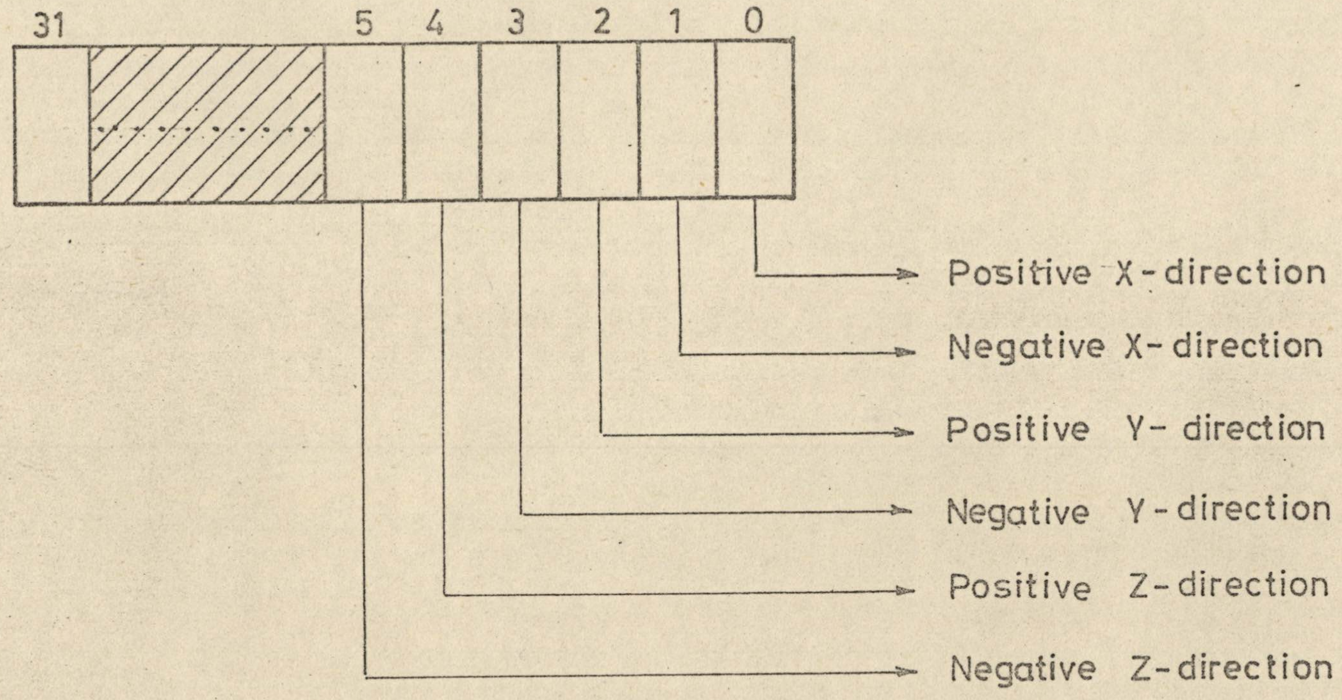
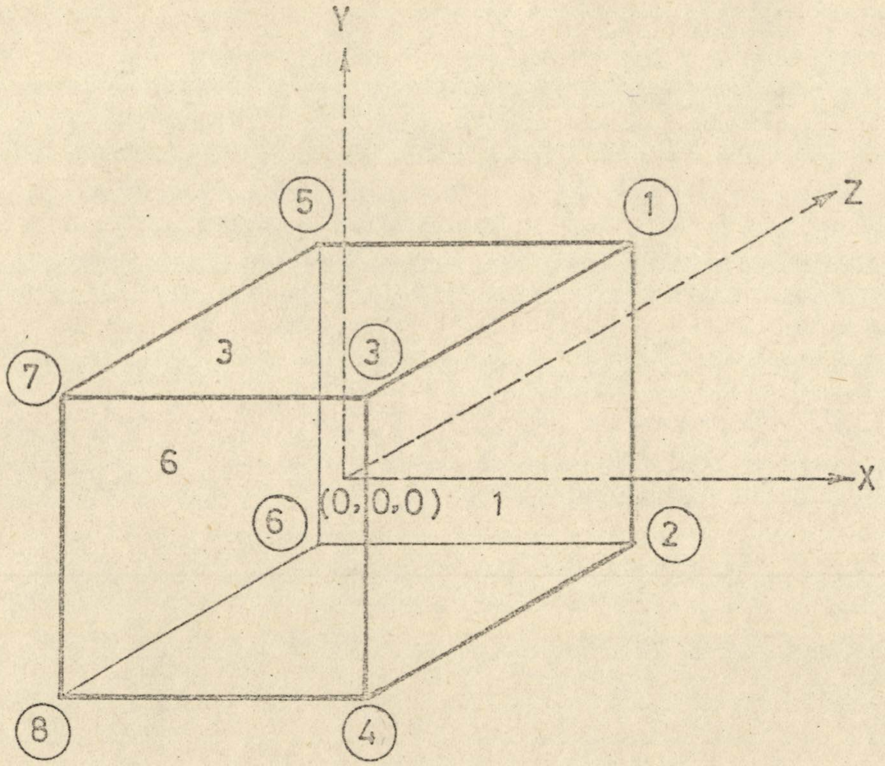


Fig. 3.3 Bit Presentation of a WORD



FACES

1	1	2	4	3
2	5	6	8	7
3	1	3	7	5
4	2	4	8	6
5	1	2	6	5
6	3	4	8	7

Fig. 3.4 Primitive cell with Origin and System Coordinates

Cell

Cell is a cubical unit entity having six faces, the numbering of the faces are shown in Fig. 3.4.

Life

A cell is said to have life in a particular direction if and only if another cell is attached to it in that direction.

Life Sought Face

A life seeking face becomes a life sought face as soon as one of the faces of another cell is attached to it.

Boundary Cell

A cell at the boundary of the Hex-tree is termed as the boundary cell.

Boundarial Face

All the faces of the boundary cell except the life sought faces are called boundarial faces.

Sleepy Face

All the faces common to two trees during the period of boolean operations are called sleepy faces. 'S' indicates sleepy faces in Figs. (3.6 - 3.8).

Fig. 3.5 shows an example of a Hex-tree. For the boundary cell 9, Faces 1, 3, 4, 5, 6 are boundarial faces and face 2 is a life sought face.

3.3 BOOLEAN OPERATIONS ON HEX-TREE

Almost all geometric modeling systems support boolean operations (Union, difference and intersection) on solid objects.

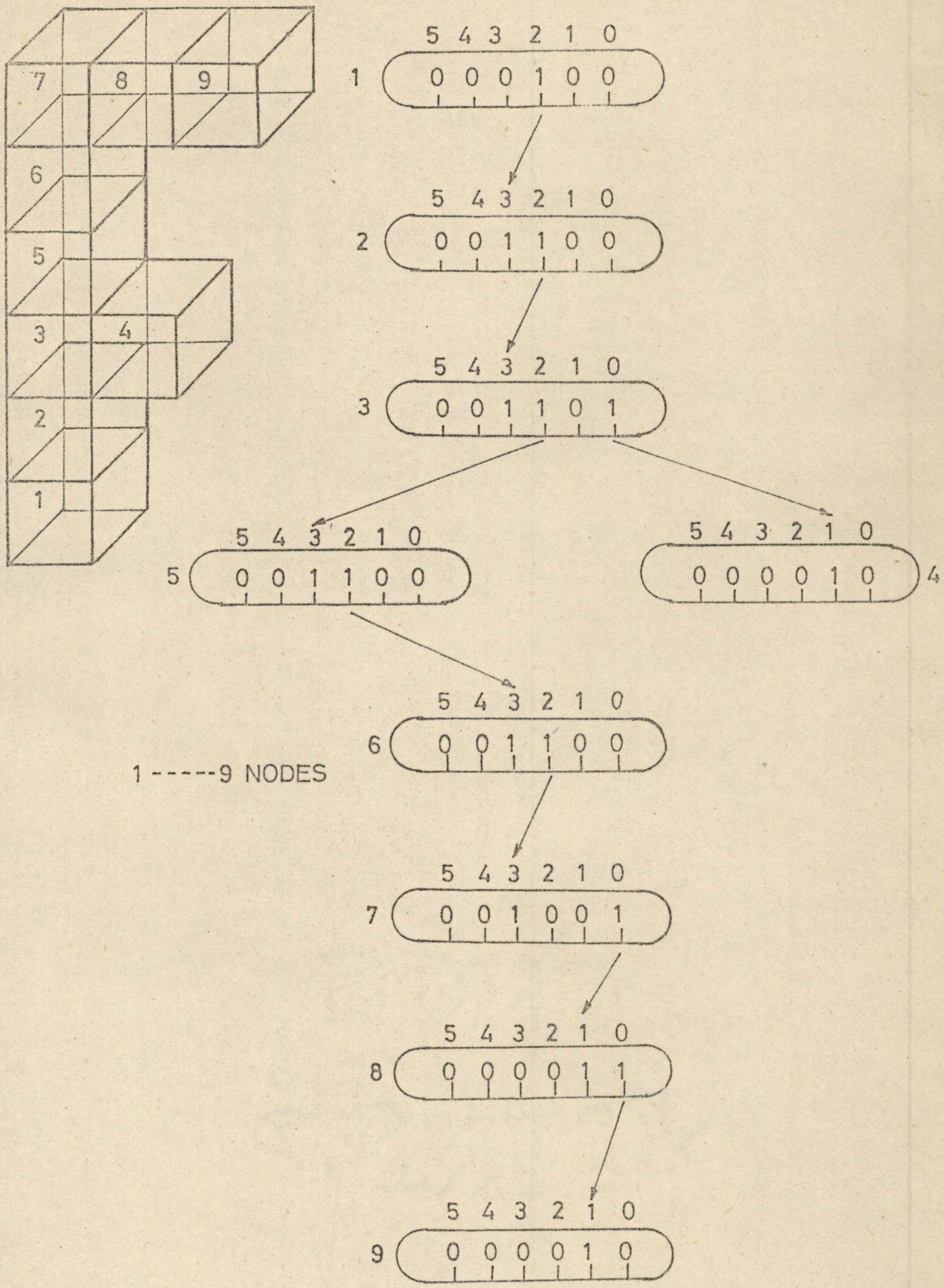


Fig. 3.5 An example of a Hex-tree

Boundary representations (face/edge/vertex structures) for solids defined through boolean operations are generated in these modelers by using so-called boundary evaluation and boundary merging procedures. These are the most complex and delicate software modules in a solid modeler [59]. The systems developed so far seem to have various problems [81].

i) The algorithms for boolean shape operations are very complicated and the systems are very large. This could cause reliability and maintenance problems when they are brought into practical use.

ii) Processing speed is not fast enough.

The boolean operations algorithms for the Hex-tree method does not require boundary evaluation. These operations are simple, fast and unified one, regardless of the surface types (i.e. planar or curved).

In order to perform boolean operations on Hex-tree approach consider two trees Hex-tree 'X' and Hex-tree 'Y' (representing object 'X' and object 'Y'). Let Hex-tree 'Z' (representing the object 'Z') be the resultant tree obtained after merging these two trees. While scanning the two trees for boolean operations, during the initial scan the sleepy faces are identified, and depending on the operations to be performed, the face identity is modified to 1 or 0. UNION, DIFFERENCE and INTERSECTION processes are shown in Figs. (3.6 - 3.8). Various steps

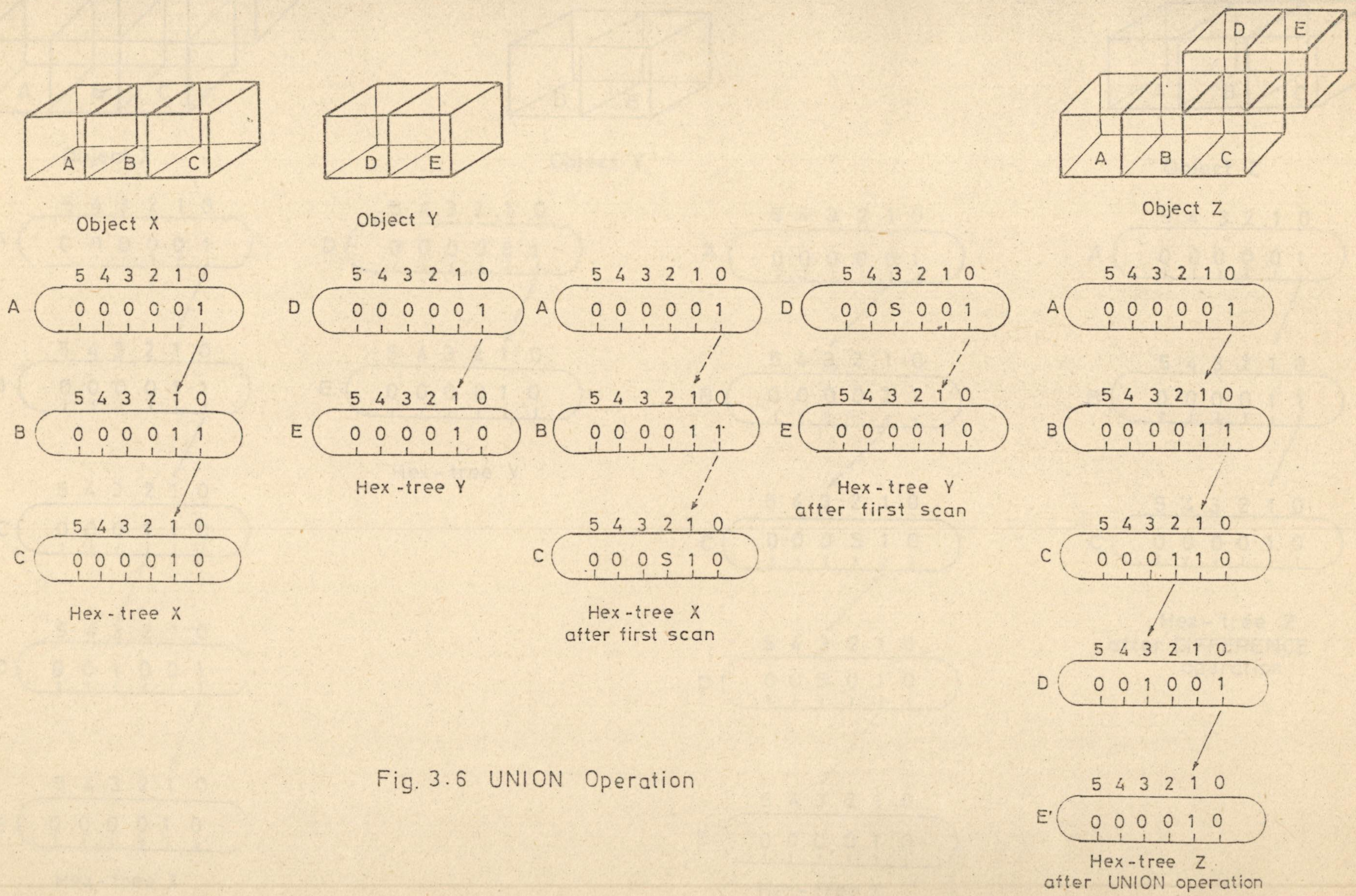


Fig. 3.6 UNION Operation

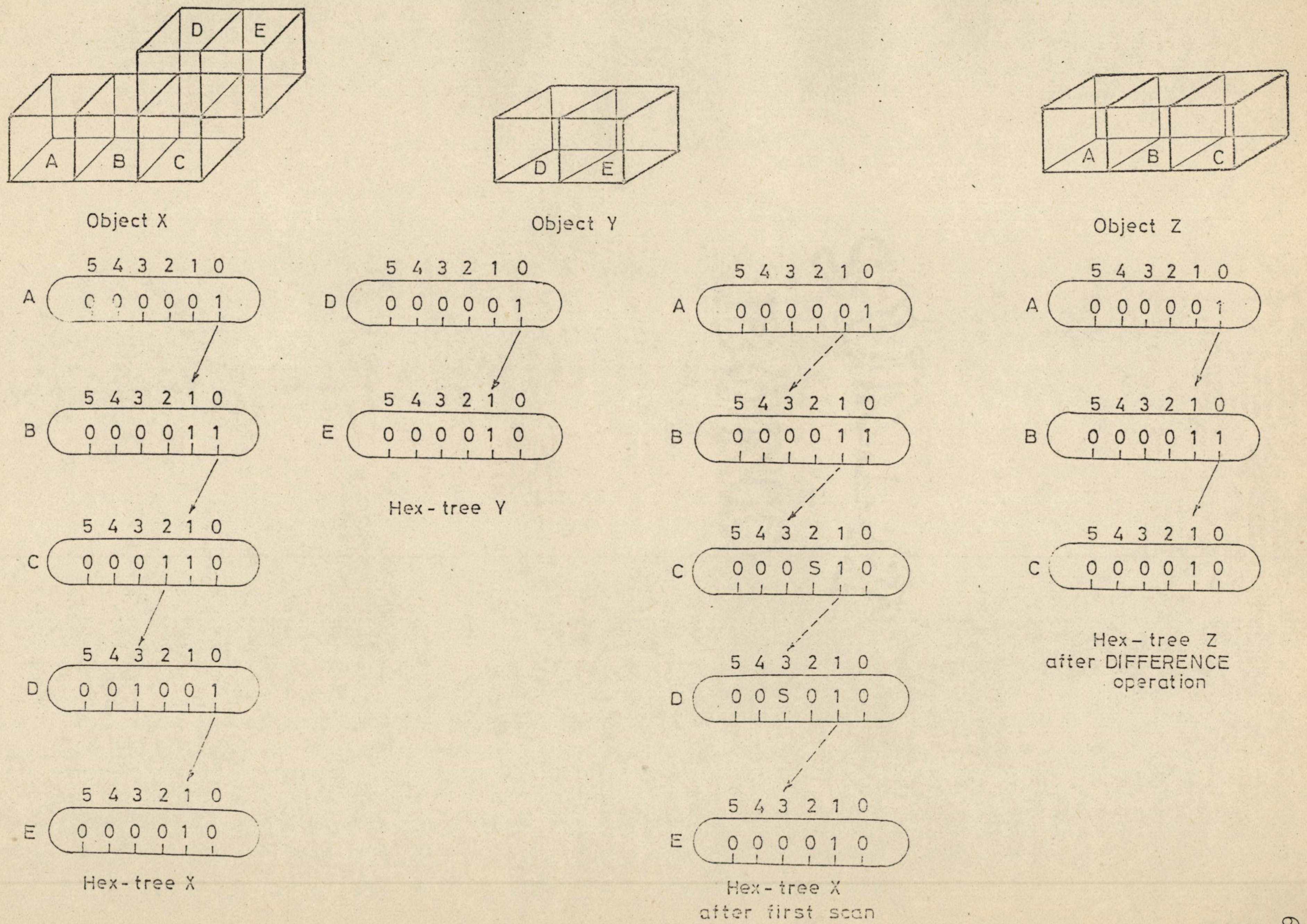


Fig. 3.7 DIFFERENCE Operation

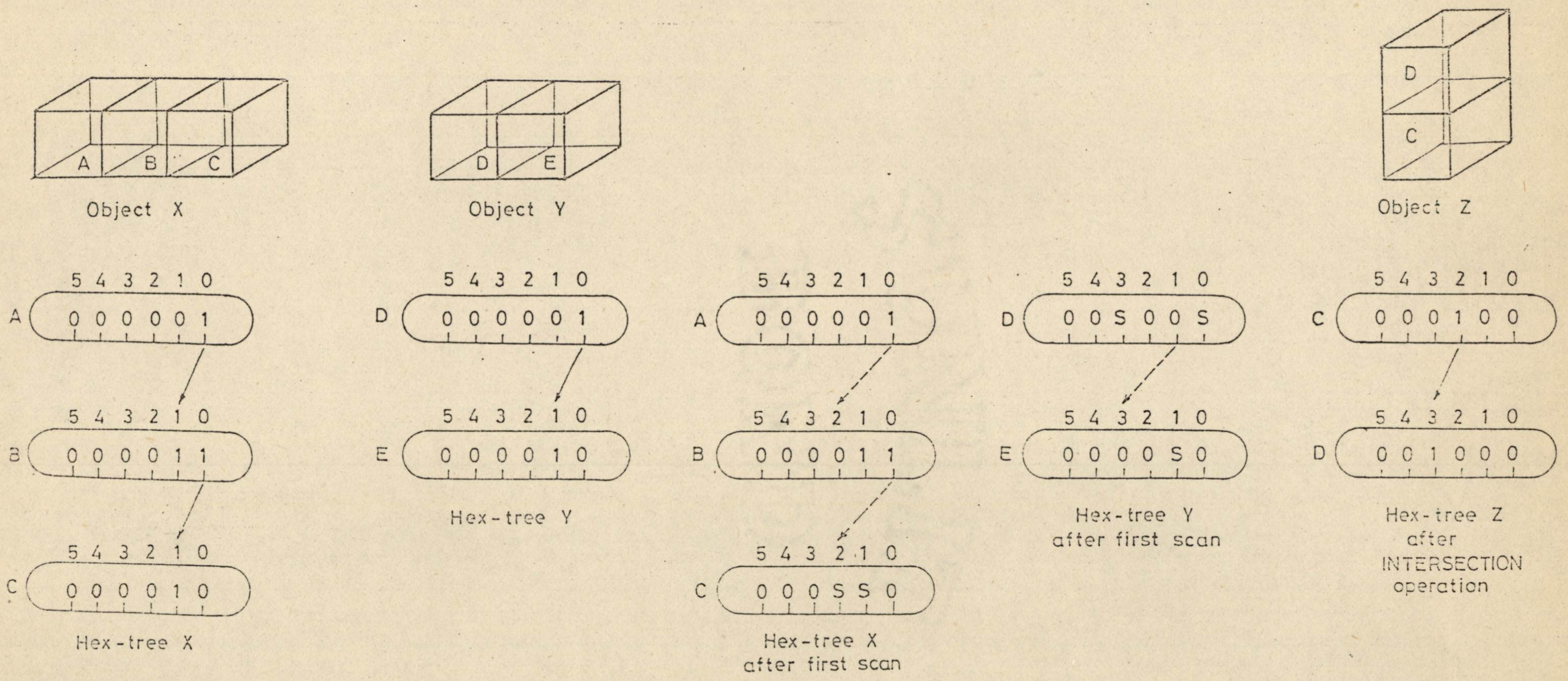


Fig. 3.8 INTERSECTION Operation

for the boolean operations are given here through a typical example.

1. Form the Hex-tree 'X' and Hex-tree 'Y' representing the two objects 'X' and 'Y'.

2. Select the nodes of Hex-tree 'X' involved in the boolean operations.

3. Change the identity of these nodes as explained in Figs. (3.6,3.7 & 3.8) depending on the boolean operations to be performed.

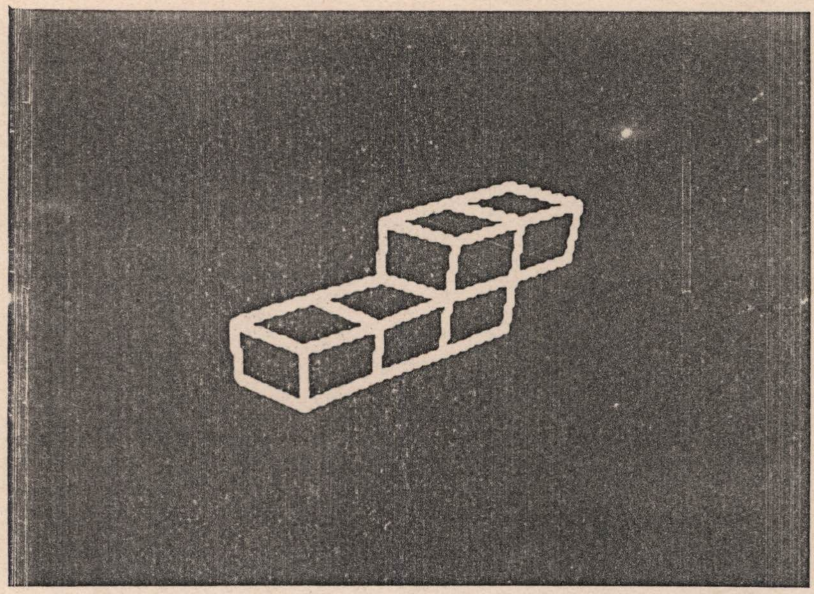
4. Repeat steps 2 and 3 for the Hex-tree 'Y'.

5. Form the Hex-tree 'Z' representing the resultant object 'Z' as shown in Figs. (3.6, 3.7 & 3.8).

Results of boolean operations of two Hex-tree encoded objects (object 'X' and object 'Y') are given in Figs. (3.9, 3.10 & 3.11).

3.4 DEVELOPMENT OF CURVED SURFACES

In shape designing process, it is required to represent wide range of objects with planar surface to curved surface. Most of the computer aided geometric modeling systems currently available consider these shapes independently due to their underlying theory and practice. In a computer aided design environment, this is definitely not a desirable feature, because



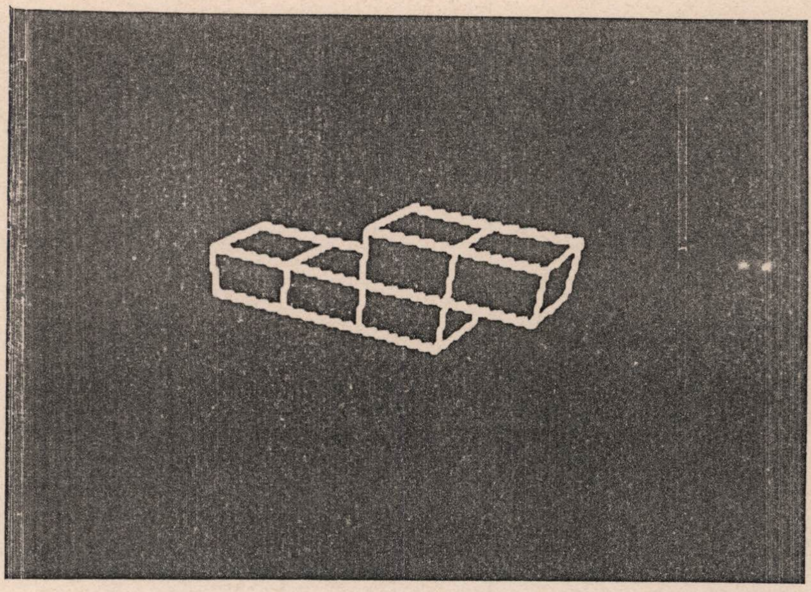


Fig. 3.9 Two views of UNION operation on two Hex-tree encoded objects

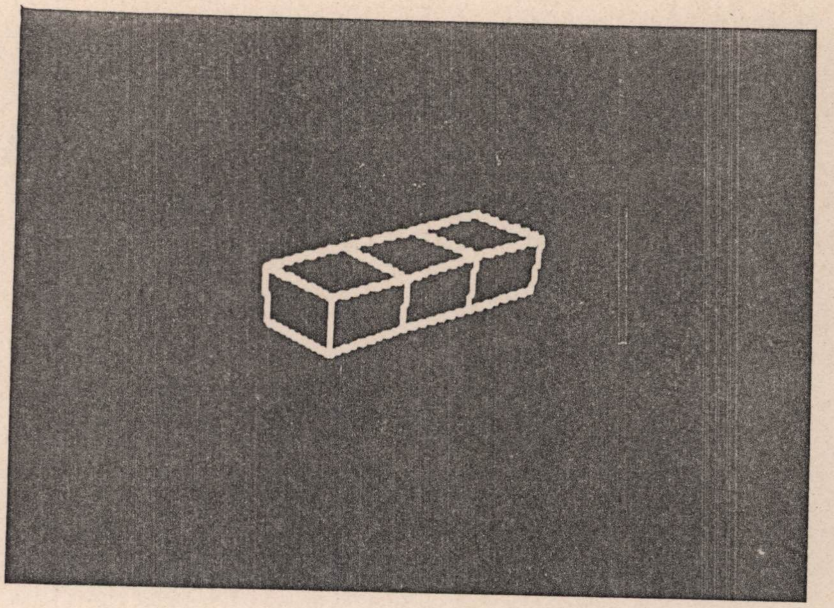


Fig. 3.10 DIFFERENCE operation on two Hex-tree encoded objects

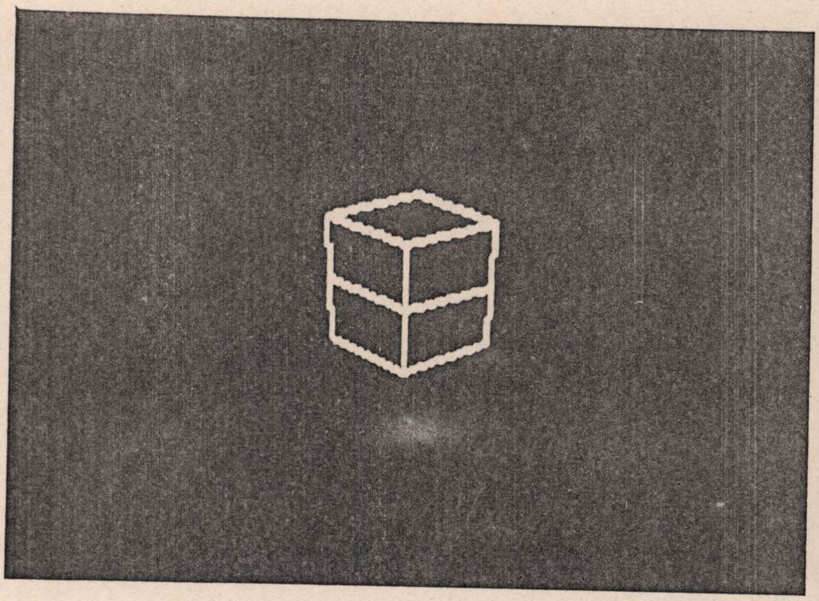


Fig. 3.11 INTERSECTION operation on two Hex-tree encoded objects

practical engineering objects have wide range of shapes. Also the important matters in modeling shapes are, not only producing fine and precise models quickly but also flexible representations of the shape for various modifications and useful techniques of changing the shapes. Attempts are already initiated for developing unified shape modeling techniques to design objects having polyhedral shape to curved surfaces [14,44,31]. The Hex-tree representational technique is a unified approach for modeling objects having planesurface to curved surface.

The curve-surfaced objects are modeled by growing the primitive along the six faces. If the approximate object shape is not obtained, repeat the modeling by changing the size of the primitive. The process of changing the primitive size and checking whether the target object shape is reached or not continue, until a satisfactory result is arrived. The smoothness of the curved surface depends on the size of the primitive cubical cell. An advantage is that it requires only a single set of manipulation and analysis algorithm for all types of curved shapes. Figs. (3.12 - 3.17) show views of different curved objects.

(Note: Figs. (3.12, 3.14 & 3.16) show the results on large cubical cell basis. The true representation of curved surfaces can be achieved by taking smaller cells).

3.5 DISPLAY TECHNIQUE FOR HEX-TREE REPRESENTED OBJECTS

One may attempt to develop a number of representational

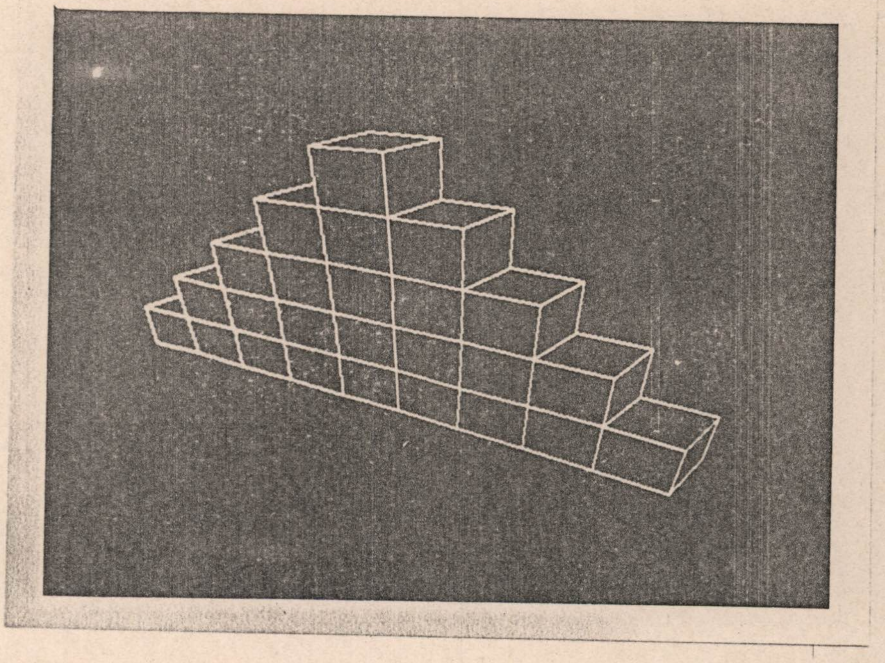


Fig. 3.12 A single layer curved object with 25 cells

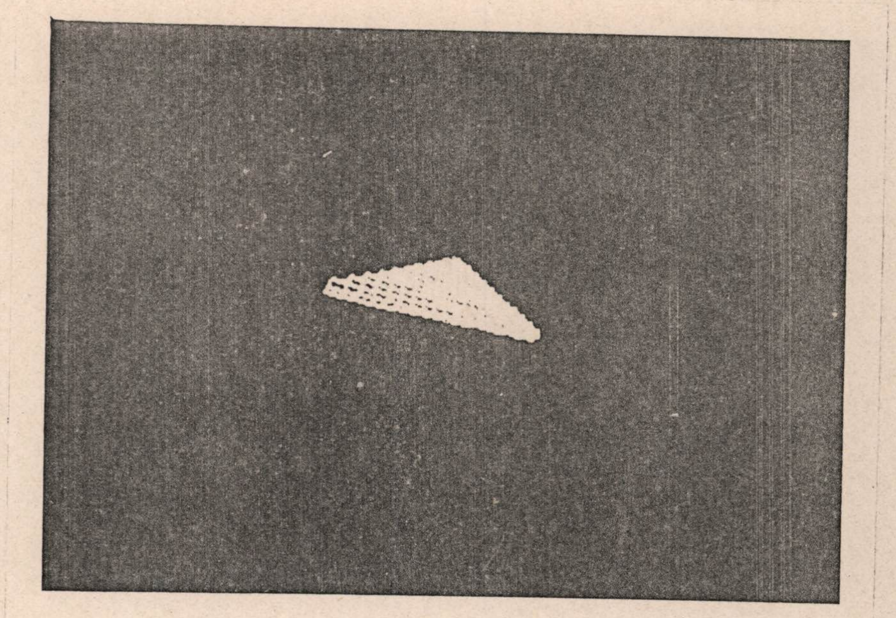


Fig. 3.13 A single layer curved object with 121 cells

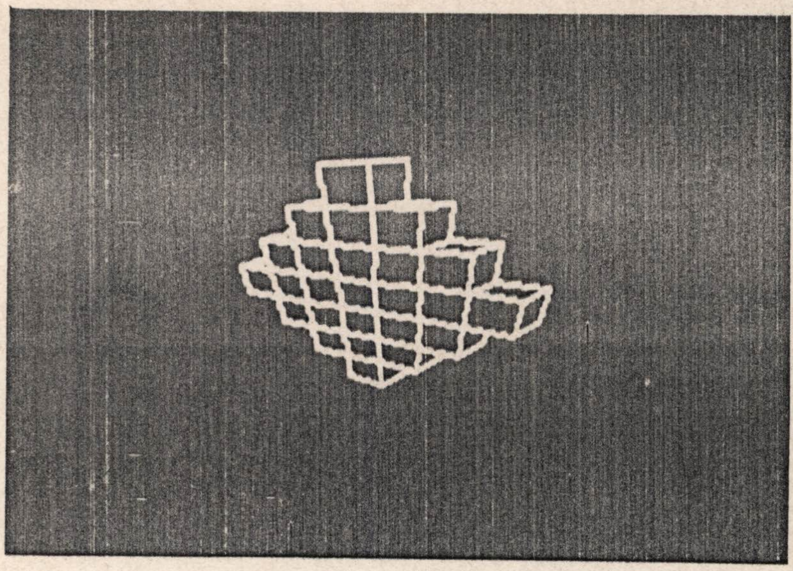


Fig. 3.14 A single layer curved object with 25 cells

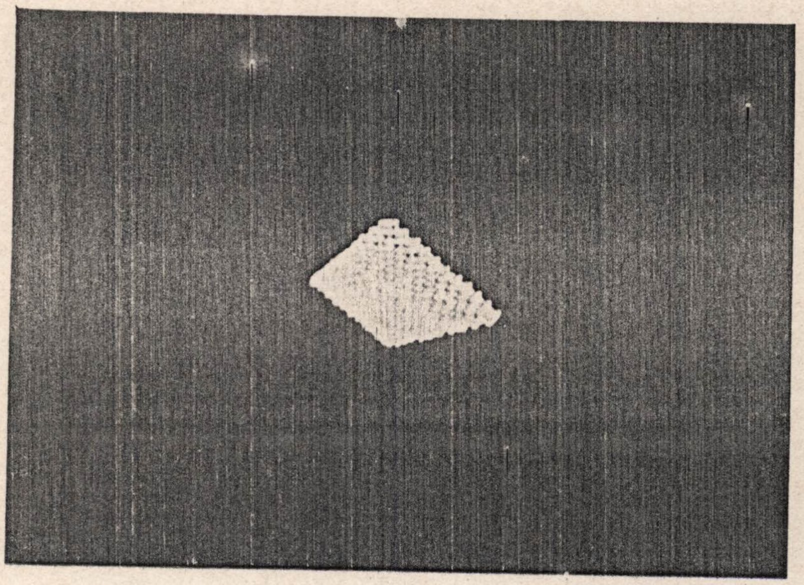


Fig. 3.15 A single layer curved object with 221 cells

techniques for an object, but the measure of undestanding increases considerably with a pictorial display. All geometric modeling systems necessarily provide display facility as this is the easiest method of conveying shape information to the user. Hence, through display, not only the designer gets an opportunity to examine the shape of the modeled object, but also he can modify it if necessary, before finalising the design. Since display operation is done frequently, a quick display function is desirable.

Popular solid modeling techniques are not efficient in display operation. In CSG and B-Rep schemes, it is necessary to determine the boundary of the object in a situation when one primitive penetrate on another. So, much computation is required for finding out whether a specific section is to be included in the final object or not, before displaying it. In order to get a realistic line-drawings display of an Oct-tree encoded object, it is required to evaluate the boundary of the Oct-tree as a set of polygons [51] and then apply hidden line elimination algorithms.

In the Hex-tree method, since the whole chain of representation and display is on face basis, the question of penetration of primitives does not arise. Hence, boundary evaluation as required in Constructive Solid Geometry and Boundary Representation Schemes are not needed. Since the method is suitable for line-drawings display, the display operation problems of Oct-tree are also not there. Display process of this approach

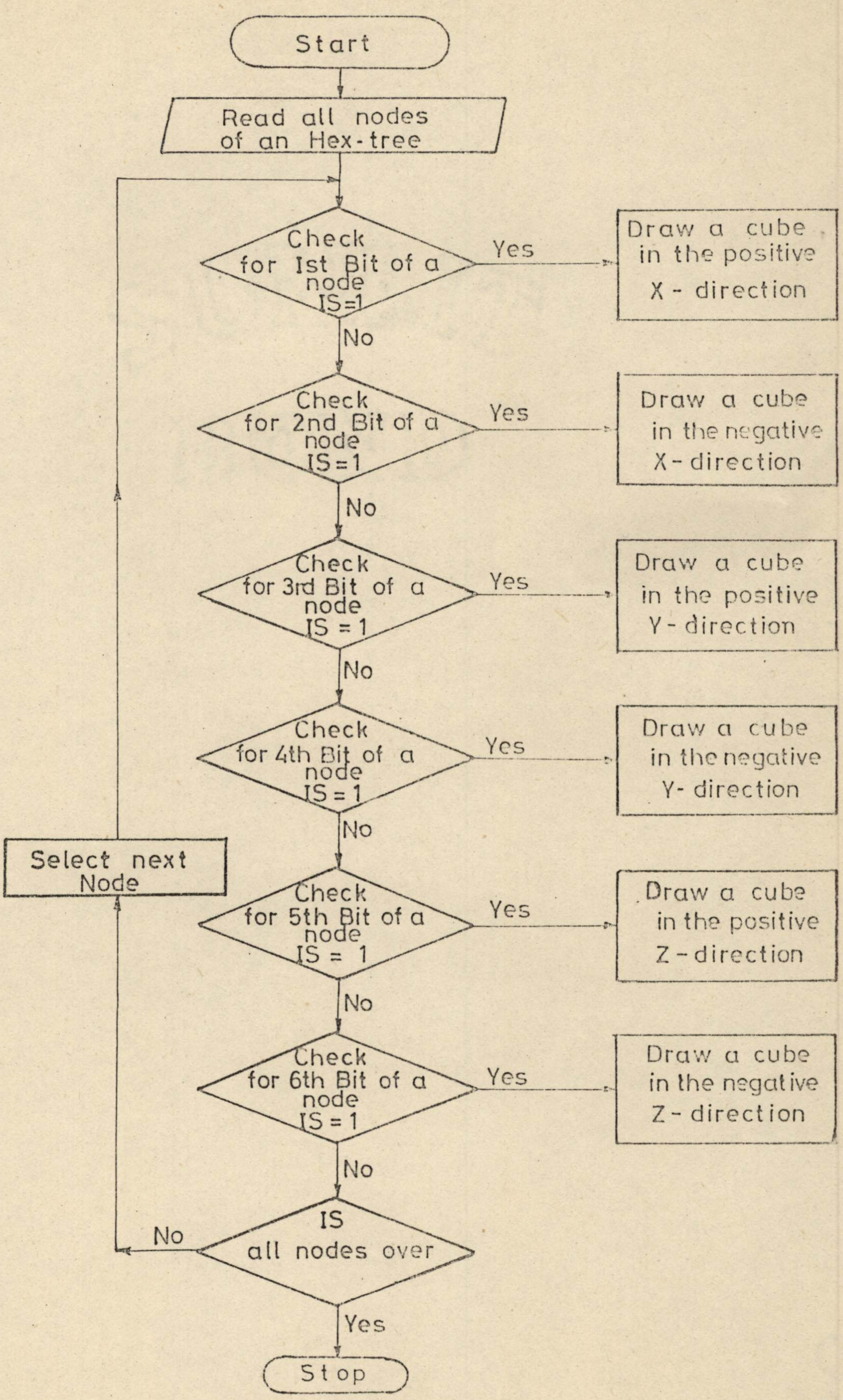

Start
Read all nodes of an Hex-tree
Check for Ist Bit of a node IS=1
Yes
Draw a cube in the positive X - direction
No
Check for 2nd Bit of a node IS = 1
Yes
Draw a cube in the negative X - direction
No
Check for 3rd Bit of a node IS = 1
Yes
Draw a cube in the positive Y - direction
No
Check for 4th Bit of a node IS = 1
Yes
Draw a cube in the negative Y- direction
No
Select next Node
Check for 5th Bit of a node IS = 1
Yes
Draw a cube in the positive Z - direction
No
Check for 6th Bit of a node IS = 1
Yes
Draw a cube in the negative Z - direction
No
IS all nodes over
No
Yes
Stop

Fig. 3.18 Flow diagram for display operation

is given in Fig. 3.18. Searching is done by logic operations. Because of these reasons, Hex-tree displaying is faster and more efficient.

3.6 CONCLUSIONS

A new constructional solid modeling scheme known as Hex-tree has been proposed for the representation and display of any type of 3-D objects, using a single cubical cell as primitive. It provides several advantages over existing popular representational techniques. Using a common hierarchical data structure, any 3-D objects of arbitrary complexity can be represented to a specified precision of the primitive size. Efficient algorithms have been developed for boolean and display operations. Due to Bit level approach, the memory requirements are less. Because of the face basis approach and efficient searching algorithms for tree traversal, the suggested technique is computationally faster and efficient.

—

CHAPTER 4

DEVELOPMENT OF HIDDEN-LINE ELIMINATION METHODS

4.1 INTRODUCTION

Hidden-line removal is one of the challenging problems in Computer Graphics. This is the problem of determining which edges of a solid object are visible (and which invisible) from a given vantage point. The hidden-line problem is not as easy as it might at first seem. A reason for this is the variety of possible complex three-dimensional solid shapes that it has to cope with.

Robert [60] developed the first program capable of removing hidden lines in 1963. The algorithm required an inordinate amount of computer time even for relatively simple objects. The same has been found characteristic of a number of other algorithms in computer graphics. After Robert's work, the race to develop a fast hidden-line (removal) algorithm was on. In 1966, Ruth Weiss [74] developed the program called BEVISION which consists of a Fortran Subroutine package for generating orthographic projections of both planar faced and quadric-surfaced objects. Although capable of producing impressive pictures, the program was based essentially on a point by point testing approach and required long computation times. Weiss's work was followed by a series of papers describing hidden-line algorithms limited to planar-faced bodies. Appel introduced the concept of quantitative invisibility, which permitted the information about the invisibility of a vertex or a line segment to be transferred to adjacent vertices and line segments [1]. This approach was also taken by Loutrel [42], Galimberty and Montanari [22]. Loutrel paper was based on the concept that once the invisibility (or visibility) of one vertex was determined, maximum use of this information should be made in determining the invisibility (or visibility) of neighbouring vertices and edges. In effect, the problem, would thus be solved by a process that propagated from vertex to vertex until all vertices and edges had their visibility established. During these periods algorithms capable of solving hidden-line problem of surfaces bounded by quadric surfaces like spherical, ellipsoidal, cylindrical, paraboloidal etc. were also developed.

Simultaneously with the foregoing developments, work on hidden line elimination was also going forward following an approach that concentrated more on the 2-D projections than on the 3-D object from which it was derived. Notable among the latter was the algorithms of Warnock[72], Watkins [73] and Newell, et al. [47]. Unlike Appel-Loutrel- algorithms which are vector (line-drawing) oriented, these algorithms are raster oriented. Bouknight and Kelley [6] presented an algorithm producing shaded pictures with shadows and movable light sources Newell's algorithm is well suited for generating half-tone (and colour) images on a raster type display. Gouraud [23] had shown how one could smooth the surfaces represented by multiple planar sections. An important algorithm closely related to the hidden--line problem is that of clipping. Sproull and Sutherland [63], Sutherland and Hodgman [66] brought clipping algorithms applicable to both 2-D and 3-D displays. These methods provide a good basis for subsequent hidden-line computation.

Kubert, Szabo and Giulieri in their paper describe a perspective transformation by which a surface S = F(X,Y) is projected on to a plane from an arbitrary observation point and then plotted with a digital plotter. He adopted the technique of eliminating hidden lines in perspective projections of 3-D surfaces described as functions (single or multivalued) of two variables.

It seems that most of the work on hidden line elimination took place during the seventies and early eighties. To the author's knowledge, Stuart Sechrest and Donald P. Greenberg [64]. Yoshio Ohno [82], John R. Rankin [32] are a few researchers whose papers are seen in this area in the middle of eighties and afterwords.

A survey paper [67] gives a comprehensive details regarding various approches on hidden line and hidden surface elimination. Hidden surface problem is very similar in nature as Hidden line elimination problem, except the fact that one must include or omit entire or partial surface areas rather than just the lines or part of the lines representing edges. Some methods are heavily dependent on memory and involves less computation. Other methods involve more processing time and have less memory demands. Most algorithms however apply only to special types of three dimensional objects and they apply to specific kinds of graphics equipment [32]. Hidden line and hidden surface algorithms have been classified as object space

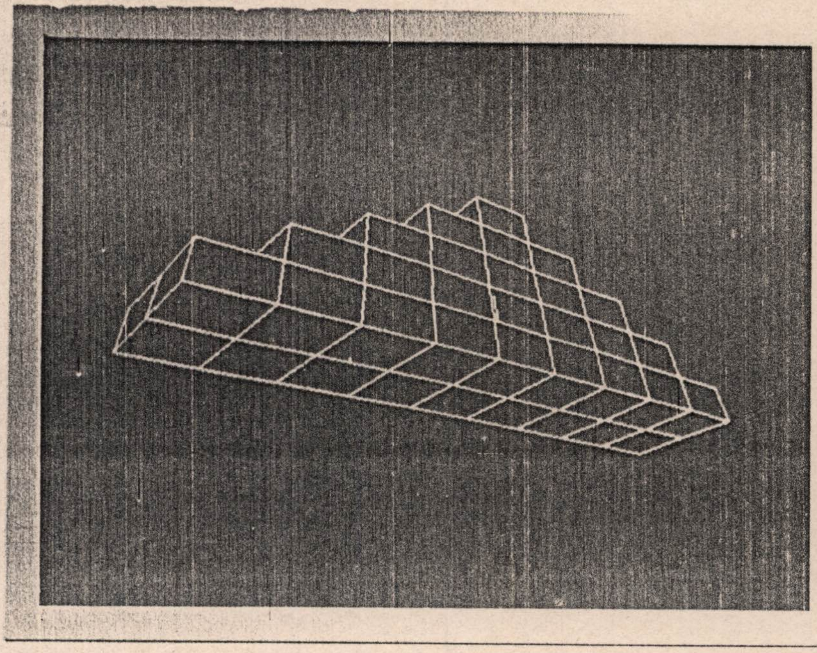


Fig. 3.16 A double layer curved object having 50 cells

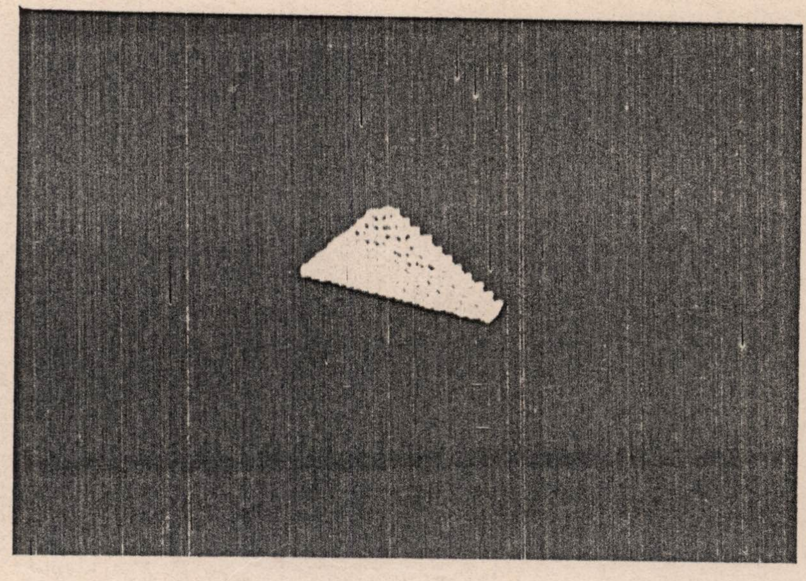


Fig. 3.17 A double layer curved object with 242 cells

methods or image space methods or a combination method. Generally hidden-surface algorithms use raster image-space methods while most hidden line algorithms use object space methods.

Object space algorithms are implemented in the physical co-ordinate system in which the objects are described. Very precise, generally to the precision of the machine can be obtained. These results can be satisfactorily enlarged many times. These algorithms are particularly useful in precise engineering applications. Image space algorithms are implemented in the screen co-ordinate system in which the objects are viewed. Calculations are performed only to the precision of the screen representation.

Some of the principles [67] based on which hidden line elimination methods can be developed, depends on the modeling techniques, equipment availability, application requirement etc., are :

1. Use of plane equations
2. Minimax tests,
3. Surrounder test,
4. Edge intersection tests,
5. Segment comparisons.

These principles can be applied individually or in combination depends on the modeling environment. Sorting principles also plays an important role in the development of the algorithms.

In short, it appears that even though there are many different hidden-line algorithms available, none of these is the best. Some are application oriented and others need improvements and modifications before applying them for a particular application. In this chapter, two hidden-line algorithms, developed for the Hex-tree based solid models are explained [62].

4.2 DEVELOPMENT OF HIDDEN-LINE ALGORITHM FOR SIMPLE SHAPES

In this algorithm, when a line of the object is tested for visibility, it is assumed that the line is completely visible or invisible. Visibility checking is made under the assumption that if a face of an object is found visible, then all the edges which form this face are considered as visible. Due to this reason, the method is found satisfactory in modeling of simple object shapes only. Provisions are also incorporated to get an overall shape of the object by showing the invisible edges as 'dashed lines', if desired.

Since the primitive in the Hex-tree technique is a cube, the functional representation of the surface in a simple linear polynomial in x,y and z is $f(x,y,z) = Ax + By + Cz - D$. This also enables us to divide space into two parts. For all points (x,y,z) such that $f(x,y,z) = 0$ lie on the surface, those with $f(x,y,z) > 0$ lie in one part and those with $f(x,y,z) < 0$ lie in the other. If it is required to find out whether two points lie on the same side of a surface or not, all that is to be done is to check whether the sign of the above equation in these two points are same. If these have opposite signs, then the two points lie in opposite sides of the surface. In such a case the line joining these two points intersect the surface.

The perspective projection of any point can be calculated as follows. Assume that the origin is the eye and the axis of cone of vision is the positive z axis. Let the perspective plane is kept perpendicular to the cone of vision at a distance d (known as PD) from the eye as shown in Fig. 4.1. To find out the coordinates of $P' = (x', y', d)$, the projection point of $P = (x,y,z)$ in the $z = d$ plane, the principle of similar triangles can be used. Considering, the y-coordinates, we can write

$$y'/d = y/z$$

$$\text{i.e. } y' = y(d/z) \qquad \ldots \qquad (4.2.1)$$

Similarly,

$$x' = x\,(d/z) \qquad \ldots \qquad (4.2.2)$$

hence, $P' = (x(d/z), y(d/z), d)$.

Equations 4.2.1 and 4.2.2 can be used to find out the perspective projections of points.

These principles are used in this algorithm to check whether a face of the object is visible or not when an observer at some general point is looking towards the origin of the object

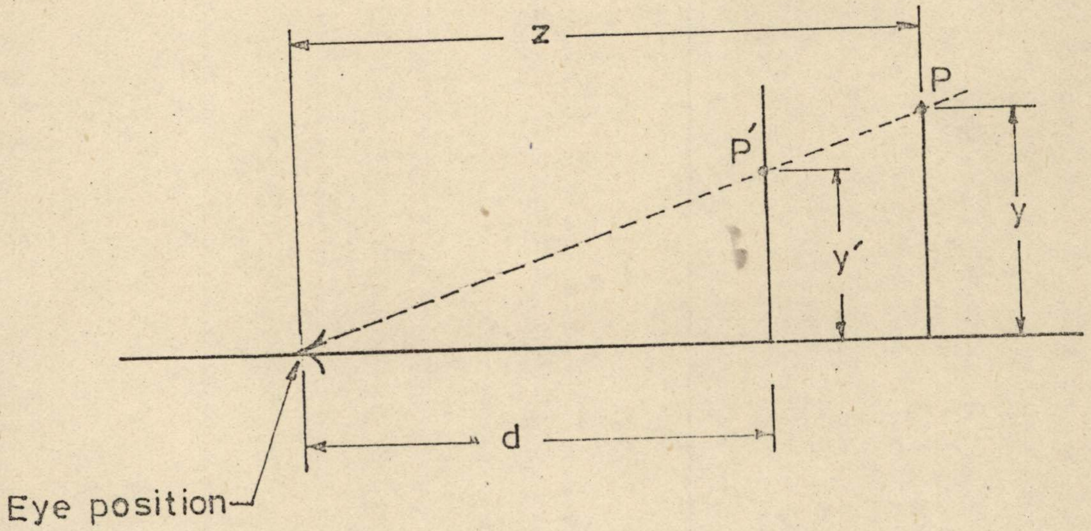


Fig. 4.1 The principle for the calculation of perceptive projection of any point

and also to calculate the perspective projection. If the particular face under visibility test is visible, then the functional equations in (x, y, z) coordinates of the origin and the observer point give opposite signs. A stepwise description of this Hidden line elimination method is given below :

1. Select a face of the object modeled using Hex-tree technique.
2. Take two adjacent sides of this face which give three non-collinear points, as one point is common.
3. Get the (x,y,z) co-ordinate of these points
4. Calculate A, B, C, D values of the plane equation $Ax + By + Cz - D = 0$ using the (x,y,z) co-ordinates of these three points.
5. With the A, B, C, D values known, check whether the origin of the object and any observer point line in the same side of the plane by putting the (x,y,z) co-ordinates values of the origin and observer point in the functional form of the equation $f(x,y,z) = Ax + By + Cz - D$.
6. If the signs are opposite, then the face is visible and hence all the sides of this face are considered as visible.
7. Put these sides of the face in a list
8. Repeat the process 1 to 7 for all the faces of the object.
9. Arrange a list of the visible edges of the object by deleting duplications, if any
10. Draw the complete perspective picture of the object, based on the above list
11. Form another list of invisible edges of the object from the total list of the edges of the object

12. Draw the invisible edges in dashed line if the full shape of the object is required.

4.3 DEVELOPMENT OF GENERAL HIDDEN-LINE ALGORITHM

In this general algorithm, in order to produce hidden line picture of the Hex-tree based modeled object, each line on the model is tested for visibility with every face. In such a situation it is possible that part of the line may be visible and parts invisible (behind a face). This algorithm is developed to meet such a requirement. The method is suitable for removing hidden lines of any complex object.

Let (x_1', y_1', z_1') and (x_2', y_2', z_2') are two end points of a typical line L_3 in 3-D space. A general point on this line is $(1-\emptyset)(x_1', y_1', z_1') + \emptyset(x_2', y_2', z_2')$ where $\emptyset$ is similar to α in equation A.1. Suppose that these two points are perspectively projected on to two points (x_1, y_1) and (x_2, y_2) on the perspective plane. Thus L_3 is projected to the line L_2 in this plane with the general point $(1 - \lambda)(x_1,y_1) + \lambda(x_2, y_2)$ where $\emptyset$ is not necessarily equal to λ. Let a typical face F_3 be projected to an area F_2 on the perspective plane, and assume that the vertices on this projected face are

$$\delta = \left\{(\overline{x_i}, \overline{y_i}) \;\middle|\; i = 1, \ldots\ldots\ldots, N)\right\}$$

where N is the number of vertices

Thus the ith edge in F_2 has a general point

$$(1-u)(\overline{x_i}, \overline{y_i}) + u\,(\overline{x_{i+1}}, \overline{y_{i+1}}) \quad \text{where } 0 \leq u \leq 1$$

where u, λ are semilar to α in equation A.1. In an ordinary perspective picture every line L_2 would be drawn on the screen. If a face F_3 lies between the eye and L_3 then part, and perhaps all, of L_2 will be hidden. If L_3 lies in the face F_3, then L_3 is on the surface of the face and any view of this line cannot be obscured by that face. If this line is not an edge of the face, a detailed examination is necesary. If F_2 is not intersected by L_2 then F_3 can have no effect on the line. In order to find out whether L_2 intersect F_2 assume that L_2 cuts the extended ith edge of F_2 at the point,

$$(1-u_i)(\bar{x}_i, \bar{y}_i) + u_i(\bar{x}_{i+1}, \bar{y}_{i+1})$$

If $u_i < 0$ or $u_i > 1$, then L_2 intersects the ith edge at a point outside the area F_2; if $0 \leq u_i \leq 1$ then L_2 crosses the area F_2 at a point on its ith edge. Since the perspective projection of a convex face is a convex area on the perspective plane, then the number of crossing points will be either zero (and hence there is no intersection) or two. In this latter case we find the two crossing points on the line L_2 given by the values u_{min} and u_{max}, $u_{min} < u_{max}$, that is the points are $(1 - u_{min})(x_1, y_1) + u_{min}(x_2, y_2)$ and $(1 - u_{max})(x_1, y_1) + u_{max}(x_2, y_2)$.

It is now necessary to discover whether the subsegment of L_2 between these two points is visible or not. This is checked by finding the midpoint of the segment

$$(x_{mid}, y_{mid}) = (1 - u_{mid})(x_1, y_1) + u_{mid}(x_2, y_2), \qquad \ldots 4.1$$

where $u_{mid} = (u_{max} + u_{min})/2$

Let $(\hat{x}, \hat{y}, \hat{z})$ be a point on L_3 that has (x_{mid}, y_{mid}) as its perspective projection. The line subsegment is hidden if and only if $(\hat{x}, \hat{y}, \hat{z})$ and any obsrver point lie on opposite sides of the infinite plane containing F_3. $(\hat{x}, \hat{y}, \hat{z})$ can be found out as follows:

Since x_{mid} and y_{mid} are the perspective projection of $(\hat{x}, \hat{y}, \hat{z})$, we can write, based on Fig. 4.1,

$$x_{mid} = \frac{\hat{x} \text{ X PD}}{(\hat{z} + DT)} \qquad \ldots\ 4.2$$

$$y_{mid} = \frac{\hat{y} \text{ X PD}}{(\hat{z} + DT)} \qquad \ldots\ 4.3$$

Where PD is the distance between the observer point (eye position) and the perspective plane, and $DT = \sqrt{EX^2 + EY^2 + EZ^2}$. EX, EY, EZ are the coordinates of the observer point.
As $(\hat{x}, \hat{y}, \hat{z})$ lies on L_3, for some value of $\emptyset$

$$\begin{aligned} \hat{x} &= (1 - \emptyset)\, x_1' + \emptyset x_2' \\ \hat{y} &= (1-\emptyset) y_1' + \emptyset y_2' \\ \hat{z} &= (1-\emptyset) z_1' + \emptyset z_2' \end{aligned} \qquad \ldots\ 4.4$$

Substituting equation 4.4 in equation 4.2 and 4.3 we get

$$x_{mid} = \frac{(1-\emptyset)x'_1 + \emptyset x'_2 \text{ x PD}}{(1-\emptyset)z'_1 + \emptyset z'_2 + DT}$$

$$= \frac{[x'_1 + \emptyset\,(x'_2 - x'_1)] \times PD}{[z'_1 + \emptyset\,(z'_2 - z'_1)] + DT} \qquad \ldots \quad (4.5)$$

Similarly,

$$y_{mid} = \frac{[y'_1 + \emptyset\,(y'_2 - y'_1)] \times PD}{[z'_1 + \emptyset\,(z'_2 - z'_1)] + DT} \qquad \ldots \quad (4.6)$$

Using equations 4.5 and 4.6, Ø can be written as

$$\left.\begin{array}{l} \emptyset = \dfrac{x_{mid}\,(z'_1 + DT) - x'_1 \times PD}{(x'_2 - x'_1) \times PD - x_{mid}(z'_2 - z'_1)} \\ \text{or} \\ \emptyset = \dfrac{y_{mid}(z'_1 + DT) - y'_1 \times PD}{(y'_2 - y'_1) \times PD - y_{mid}\,(z'_2 - z'_1)} \end{array}\right] \qquad \ldots \quad (4.7)$$

This value of Ø is utilized to find out the point $(\hat{x}, \hat{y}, \hat{z})$. Now we have the point (0, 0, - DT) of the observer point in the negative Z-axis, and the point $(\hat{x}, \hat{y}, \hat{z})$ in the line subsegment. Using these two points and the equation $f(x, y, z) = Ax + By + Cz - D$, the visibility of this line subsegment is tested. Various steps involved in this algorithm are given below

1. Select an edge of the model
2. Select a face of the model
3. Check whether the edge considered in step 1 is one of the edges of the face considered in step 2. If yes go to step

2, otherwise go to step 4. This checking is done at 3-D level.

4. Find the intersection, if any, between the edge and the face at the perspective 2-D level. Check whether the gradient value u is >1 or <0. If yes discard the face and go to step 2. Otherwise go to step 5.

5. If u value is between 0 and 1, consider the next edge of the face at 2-D level and do the intersection test. Let the u values be u_{min} and u_{max}.

6. Find out x_{mid} and y_{mid} using equation 4.1
7. With x_{mid} and y_{mid}, find the 3-D point $(\hat{x}, \hat{y}, \hat{z})$ using equations 4.7 and 4.4

8. Check the subsegment visibility using (0, 0, - DT) of observer point and $(\hat{x}, \hat{y}, \hat{z})$ by putting in the functional equation for plane $f(x, y, z) = A_x + B_y + C_z - D$. If both are having opposite signs, then the subsegment is visible. If subsegment is visible go to step 2. If not go to step 9.

9. Adjust the visible segment arrays and the corresponding u values. If all the faces are checked, go to 10 otherwise go to step 2.

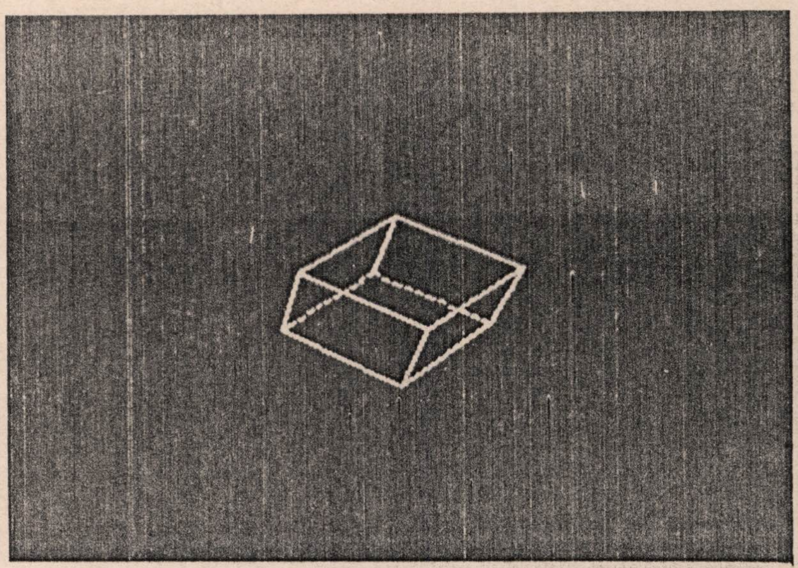
10. Visible edge segments are drawn at perspective 2-D level for the edge under consideration

11. Repeat the steps 2 to 10 for other edges of the model.

Few results obtained using the above mentioned algorithms are given in Figs. (4.2 - 4.7).

4.4 CONCLUSIONS

Two hidden-line removal algorithms, suitable for line-drawings display devices, developed for the Hex-tree representational technique based solid models, are presented. A face basis visibility testing was adopted in the first algorithm whereas a part of line segment level visibility testing was followed in the second algorithm. The complete hidden line perspective picture of an object viewed from various distances and/or positions are facilitated. In case an overall shape of the object model is desired, provisions are incorporated in the first algorithm to perceive the invisible parts with dashed lines. Few photographs obtained using these algorithms are also given.



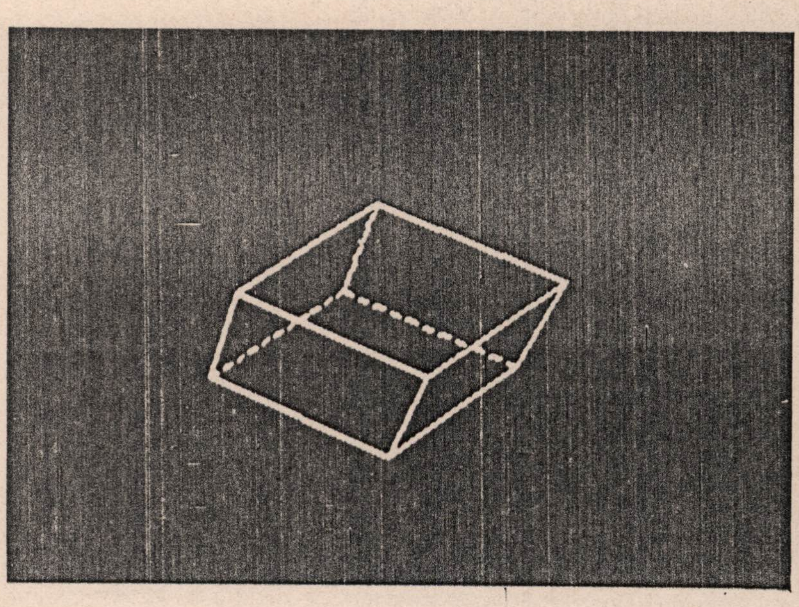


Fig. 4. 2 Hidden lines 'dashed' single cell with different perspective plane distances

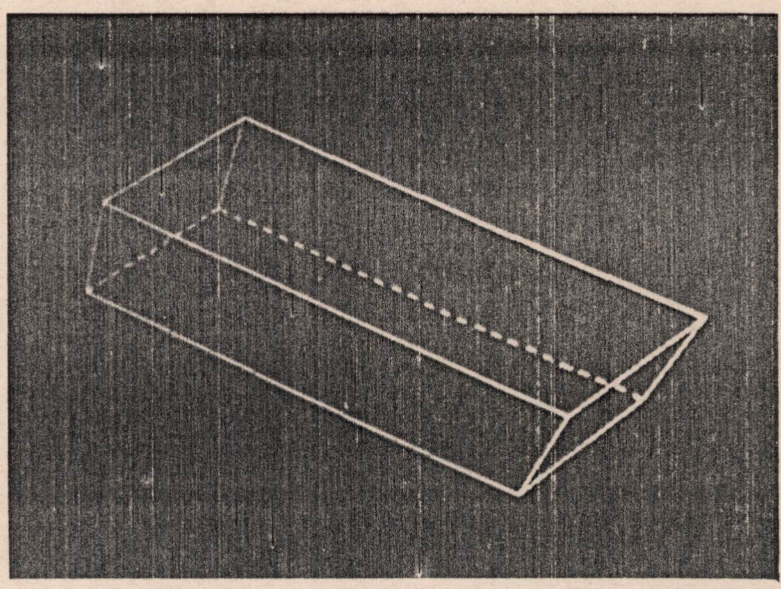


Fig. 4.3 A Rectangular parallelepiped with hidden lines 'dashed'

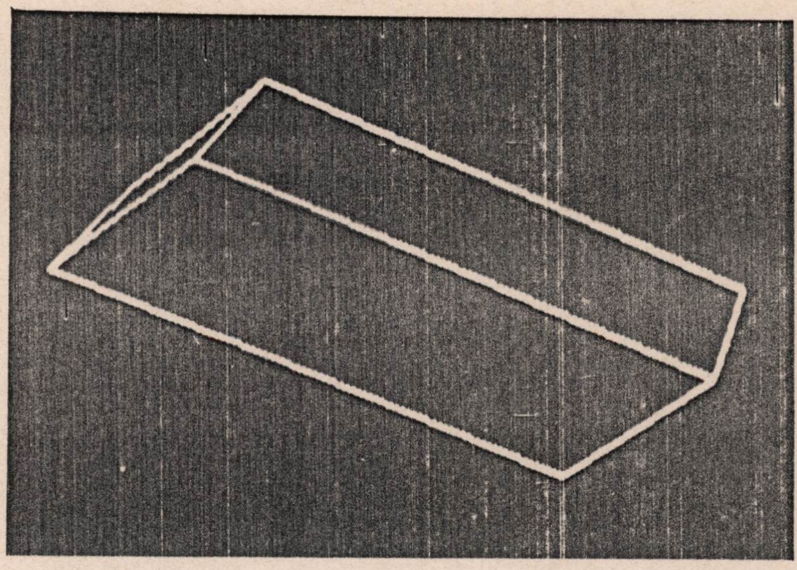


Fig. 4.4 A Rectangular parallelepiped with hidden lines removed

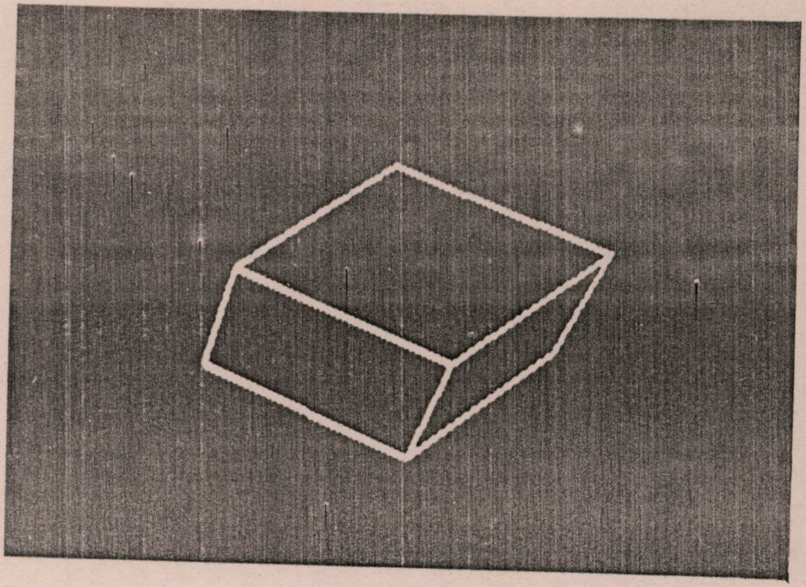


Fig. 4.5 Single cell with hidden lines removed

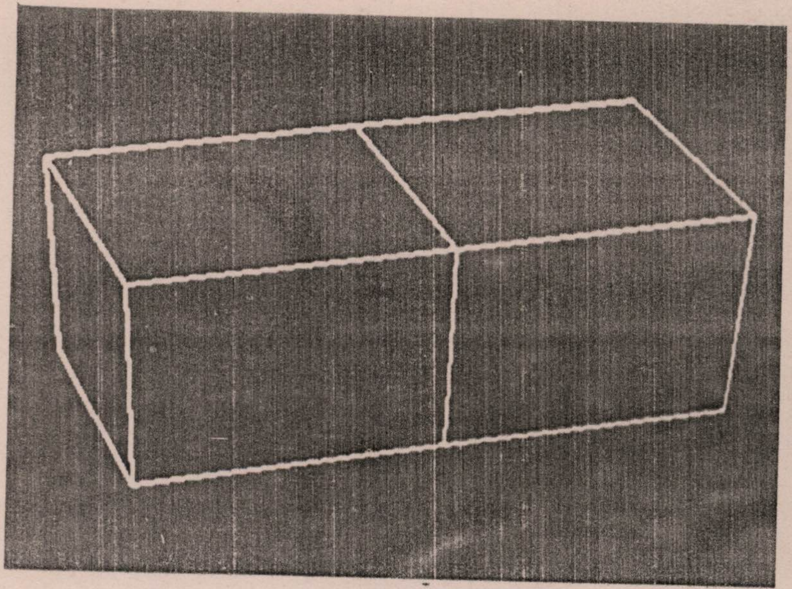
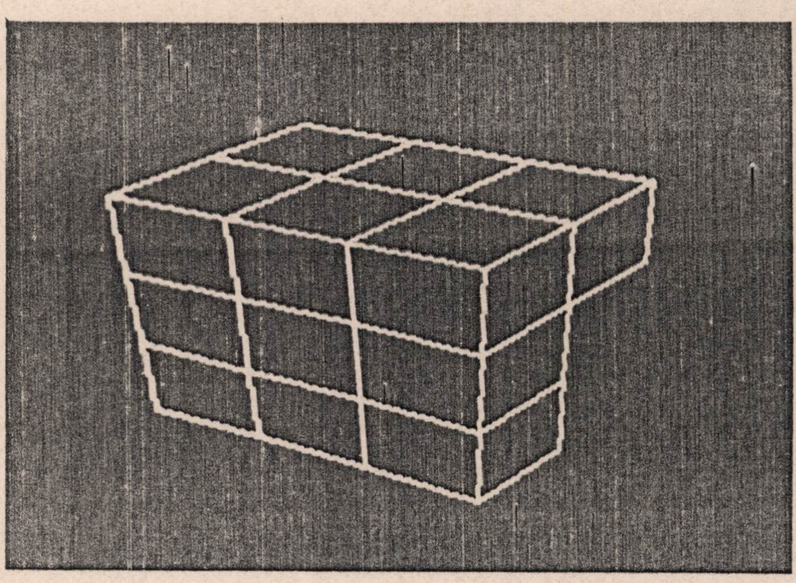


Fig. 4.6 A double cell object with hidden lines removed



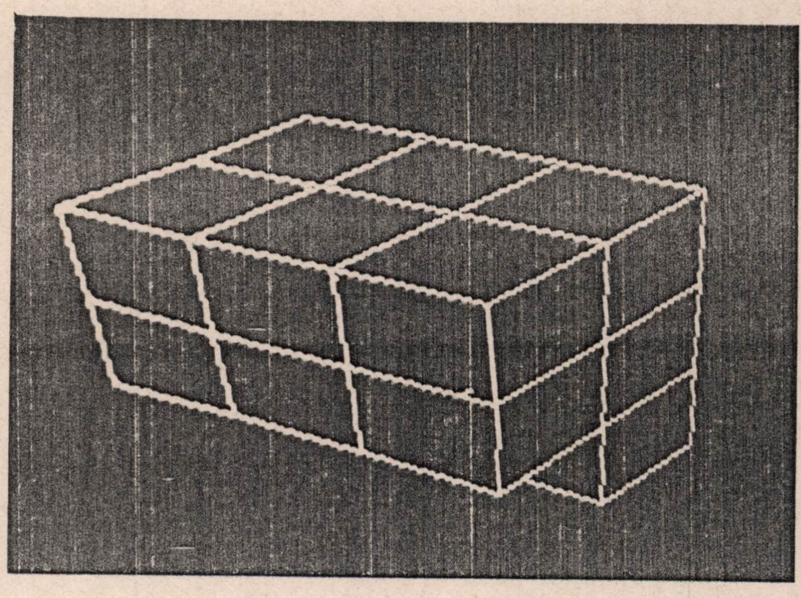


Fig. 4.7 Two objects with hidden lines removed

CHAPTER 5

THREE DIMENSIONAL STRUCTURE MODELING-IMPLEMENTATION

5.1 INTRODUCTION

Engineering components have a wide range of geometric shapes and there are different ways to represent the geometry of these object shapes in a computer. Even though it is necessary to represent solid objects having shapes from polyhedral to curve-surfaced shapes, present day geometric modeling systems based on popular solid modeling schemes are not been able to achieve these requirements satisfactorily due to their various limitations. So the search for developing new solid modeling techniques capable for handling such complex shapes continues. Once the representation details of the shape has been stored, it can be used in various applications.

The basic problems behind 3-D structure modeling is the development of a suitable data structure and display of the represented object shape. The data structure must be versatile enough, so that a large variety of shapes can be represented. The display must be clear so that a realistic image of the model can be visualised. The user should have the flexibility to view the modeled object at any observer point in space. If the user is interested to view the model without going for a realistic image (by removing the hidden portion from a given vantage point) the system should be able to present such pictures. In case the designer wants his model to be seen by putting the hidden lines as 'dashed lines' in order to get an idea about the overall shape of the model, this should also be facilitated.

The ability to create and manipulate shapes is made a great deal easier, if the basic boolean operations of union, difference and intersection are available in the geometric modeling system. However powerful a modeling system may be, it should be supported with friendly user-machine interfaces for its acceptance among user communities. One of the ways of achieving this is by creating multiple screen areas.

In the foregoing chapters, the description about the Hex-tree data structure and the hidden-line algorithms were given. The development of screen-layout and the implementaion details of the geometric modeling system are discussed here.

5.2 SCREEN LAYOUT - USER INTERFACE

The acceptance of a computer aided geometric modeling system among a user group depends very much on its User-Interface as this is the window through which the designer can look into the system. CAD/CAM has reached a stage that it is widely used in development and production in industry. As the users of the systems are designers with little or no background in computer usage, it is required that the system should be capable of guiding the unexperienced user the modeling process and if the user is an experienced one, he should have the possibilities of making shortcuts to speed up the process. The user controls the system's processes by using an input language which is familiar to him. If graphic input is used, the input language may contain elements like 'light pen selection', 'locate operations', etc. Likewise the system formulates its output to the user in an output language which may consist of elements like 'a segment of a line on the screen', 'a prompt message', 'a menu', etc. In an interactive designing process, a session is a dialogue between the user and the system where input language constructs are being interpreted by the system, and followed by output language constructs to be interpreted by the user. The interface is good if the following criteria are fulfilled.

Concise

The user shall be able to control much of the system's tasks in 'few words', and get much information in 'few words'.

Consequent

Similar tasks shall be controlled by similar inputs language constructs, and output information of similar character shall be given with similar output language sentences.

Familiar

Input and output language constructs shall reflect the way the user thinks, and not the way the system works.

Logical

Output - and especially - input language constructs shall reflect as close as possible the user's model of the world he is modeling and/or simulating. They shall reflect the real world's entities, their relations and the processes affecting them.

The constructs must reflect as close as possible the user's knowledge of the underlying system. This knowledge may vary from user to user, therefore, the User-Interface should be able to provide the user with further information if needed, e.g., by an inline help function.

The best User-Interfac require extensive use of computer graphics displays - as the whole human-computer information exchange takes place through displays. Hence, an efficient and optimum screen layout is an important aspect of the User-Interface. A typical screen layout can have the following information areas as shown in Fig. 5.1.

<u>System status area</u>

This area informs the user about current system status, e.g., module name, date, etc.

<u>Master menu area</u>

This information area gives a display of the various command.

<u>Submenu area</u>

Display the description of the menu item selected from the master menu area.

<u>Message area</u>

This category is a display of the interactive alpha numeric dialogue between user and the system, i.e., prompts, message, user input etc.

<u>Output area</u>

Used to display the created models, listing of database contents, etc.

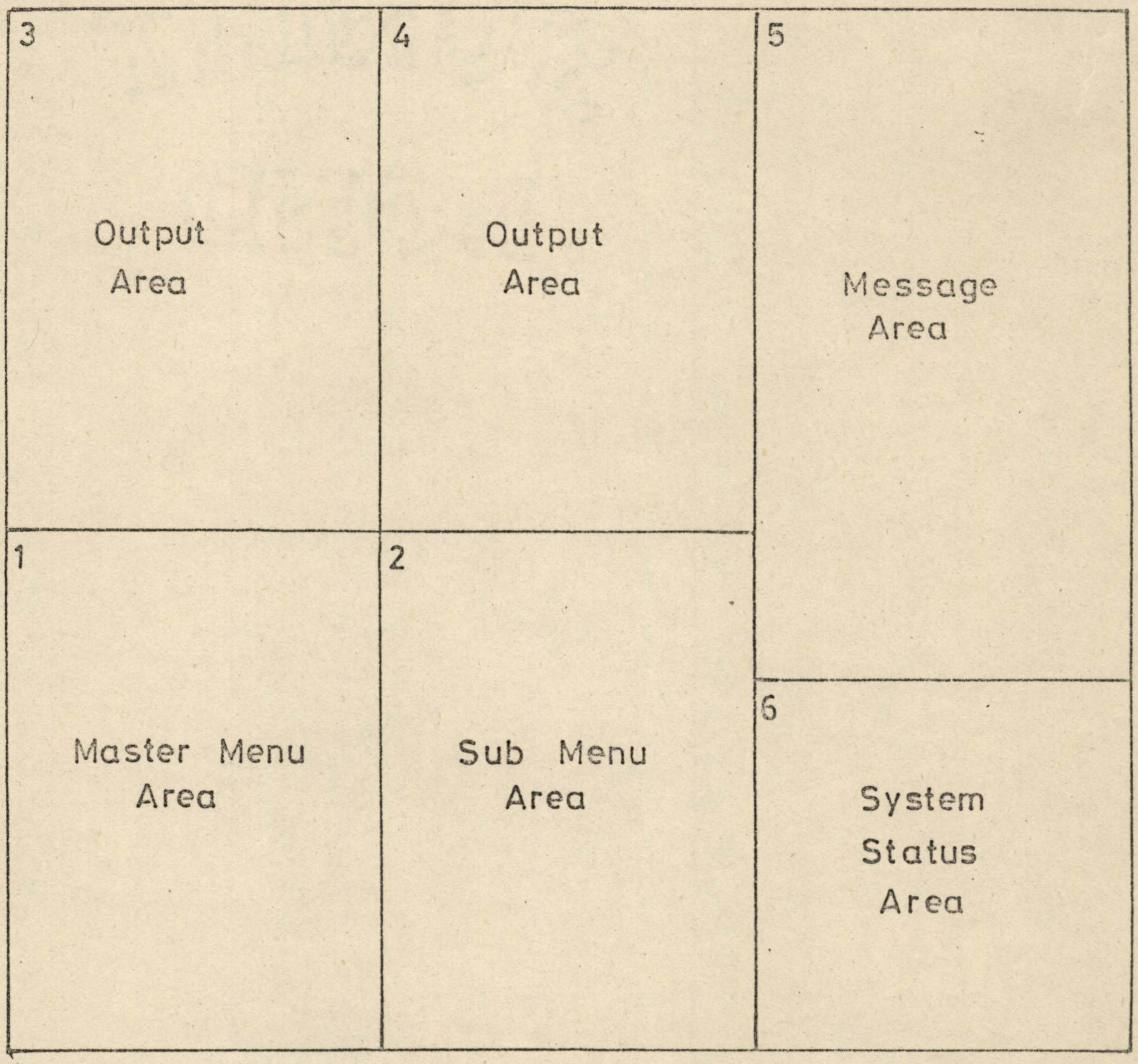


Fig. 5.1 A typical screen-layout

The screen layouts incorporated with the Hex-tree modeling system are shown in Figs. (5.2 & 5.3). Fig. 5.3 gives a screen-layout as shown in Fig. 5.1. The flow chart for creating multiple screen areas is shown in Fig. 5.4. Using various alphanumeric keys, the required screen areass can be obtained.

5.3 IMPLEMENTATION DETAILS

The various features of the geometric modeling system based on the Hex-tree solid modeling technique are explained here by adopting a building block procedure. The schematic diagram of the geometric modeling system is shown in Fig. 5.5

In the **interactive mode**, the designer can not only make the shapes of the object in his mind, but also he can get the feel of designing by his own hands. He is able to define the life constraints at every step, and correct erroneous instructions at every stage. He could move at any node irrespective of any format. There is also provision in the system to respond with a message or life presence if life is defined on an already live face. Using Hex-tree method, it is possible to model many shapes directly without performing boolean operations. Fig. 5.6 and Fig. 5.7 show two typical photographs of such shapes. Through **direct mode**, the modeling of objects whose shapes and details are known can be performed. Boolean operations are essential in any designing set up as modification of shape is a common phenomenon in designing process. In the **boolean mode**, boolean operations on pairs of solid objects represented by

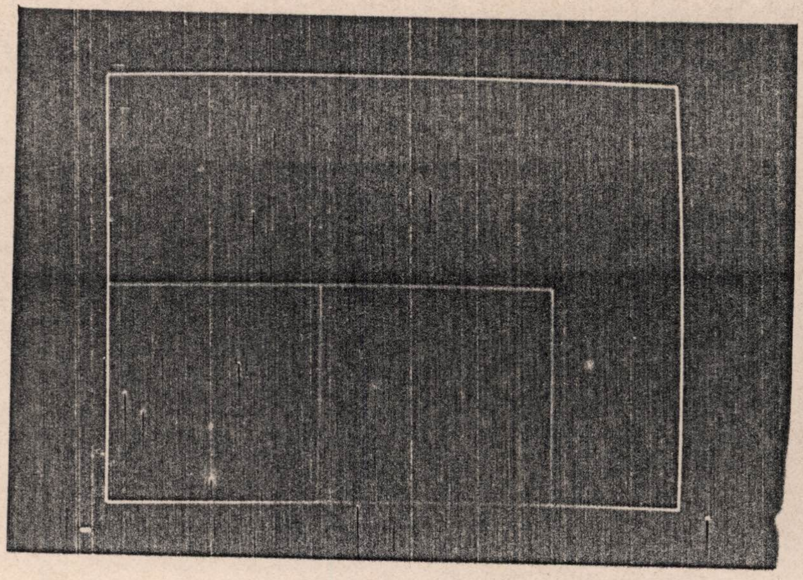


Fig.5.2 A screen-layout with three screen areas

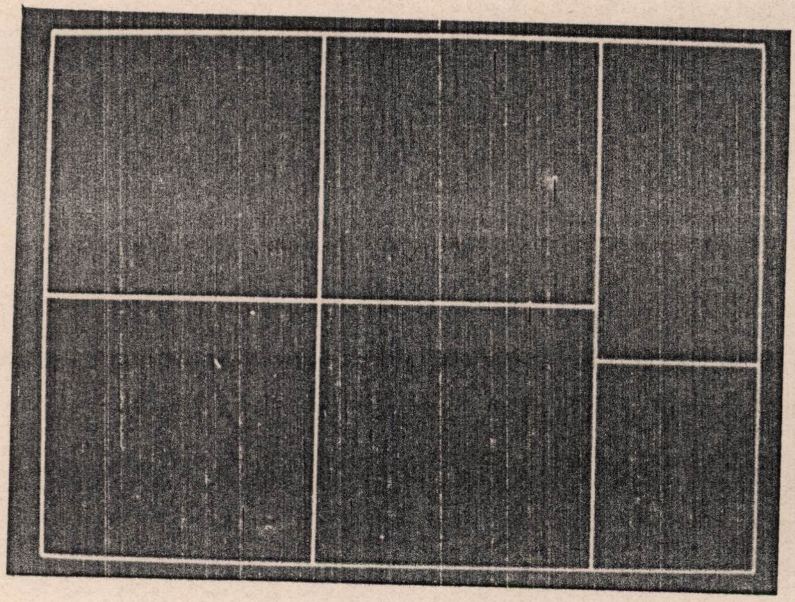


Fig. 5.3 A screen-layout with six screen areas

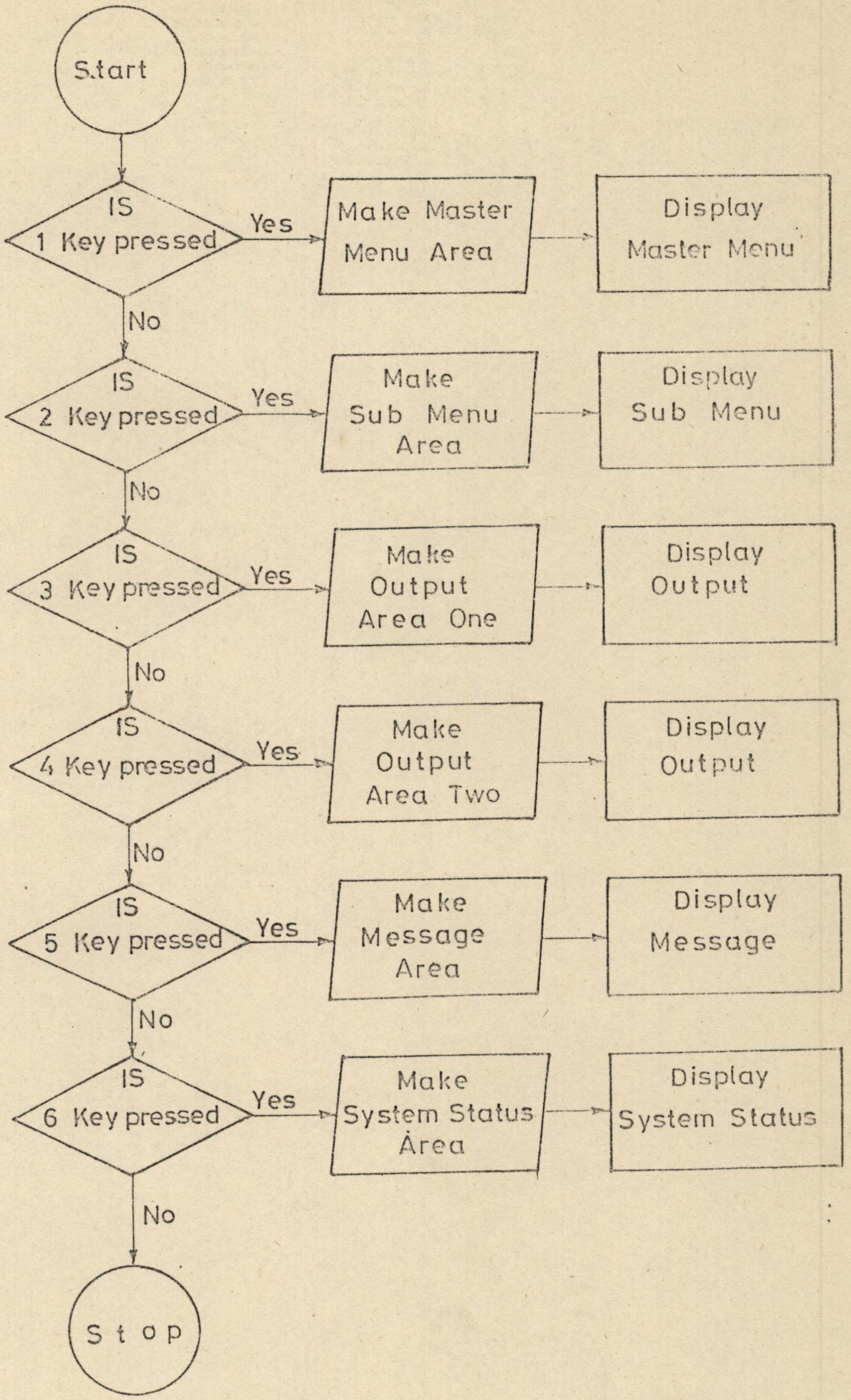


Fig. 5.4 Flow Diagram of the Screen Layout

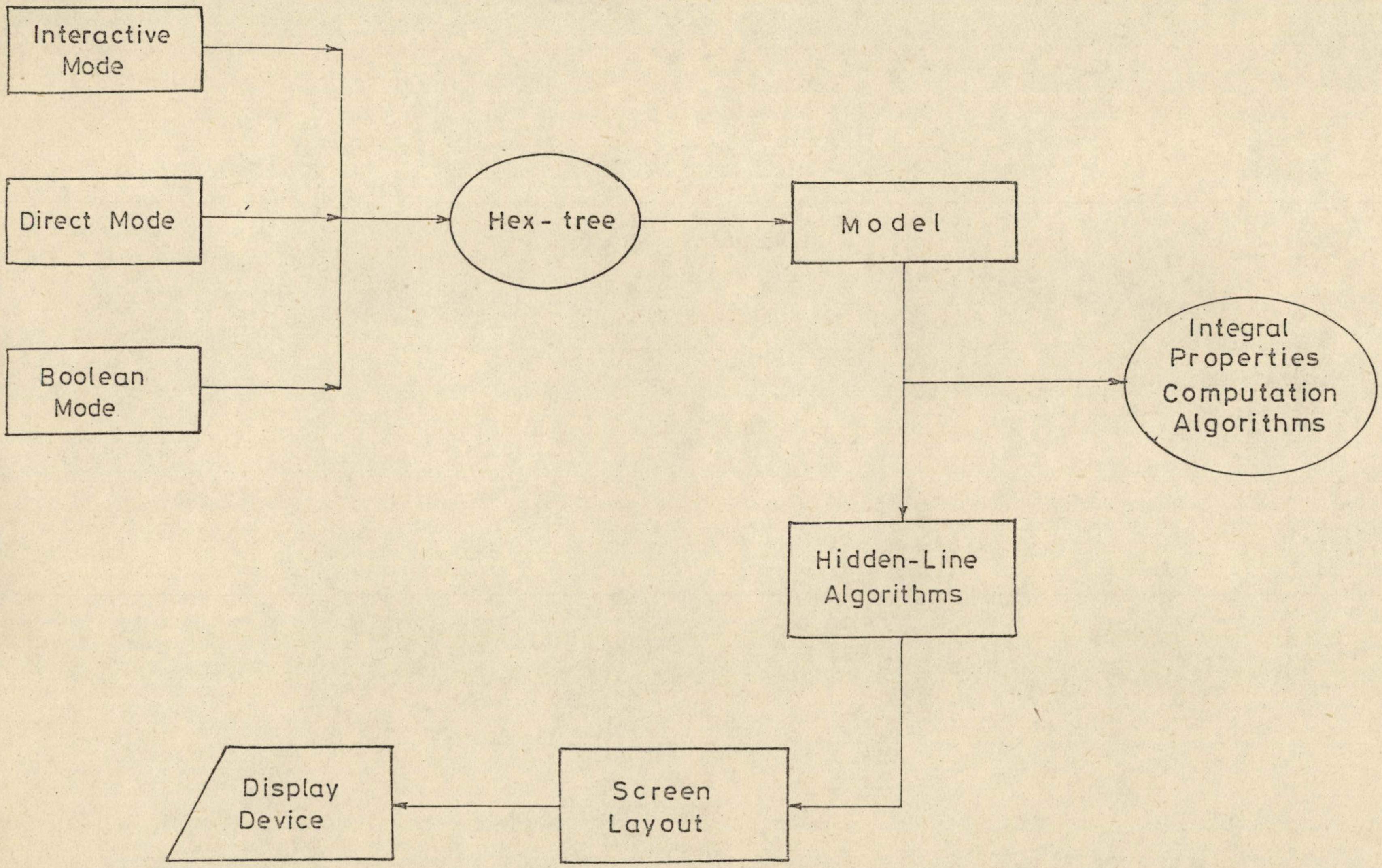


Fig. 5.5 Schematic Representation of the Geometric Modeling System

Hex-tree can be performed. Typical photograhs obtained after performing these operations are shown in Figs. (3.9-3.11).

Once the Bit indices of the nodal words are allocated based on Hex-tree data structure, the next step is to get the display of the model. The potential faces of each cell which control the building up of the model in all the positive and negative (X, Y, Z) directions are obtained, by logically searching the nodal Bits. In this searching, in order to find out the status of the first Bit position of a particular node (i.e. zero or one), a 32 bit word in which the first Bit position is one and the remaining Bit positions carry zero was utilized. Similarly for second Bit position checking, another word with second Bit position is one and the remaining Bit positions carry zero was used and so on. Fig. 5.8 dpicts this process. On a similar way this searching process was extended to check the status of all Bit positions of the nodal words. Nodal checking, and the application of geometric transformation on the primitive cell, continues until the whole Hex-tree nodes are visited. Once this process is completed, the whole topological and geometric information of the model are available. By applying **hidden-line algorithms** explained in chapter 4, the hidden lines removed picture could be seen through the **display device.** The **screen-layout** discussed in Section 5.2 added the functioning of the system more interactive. With the available geometric information of a particular model, the **integral properties** can be calculated using various algorithms which are discussed in Chapter 6. The system is supported by the following facilities

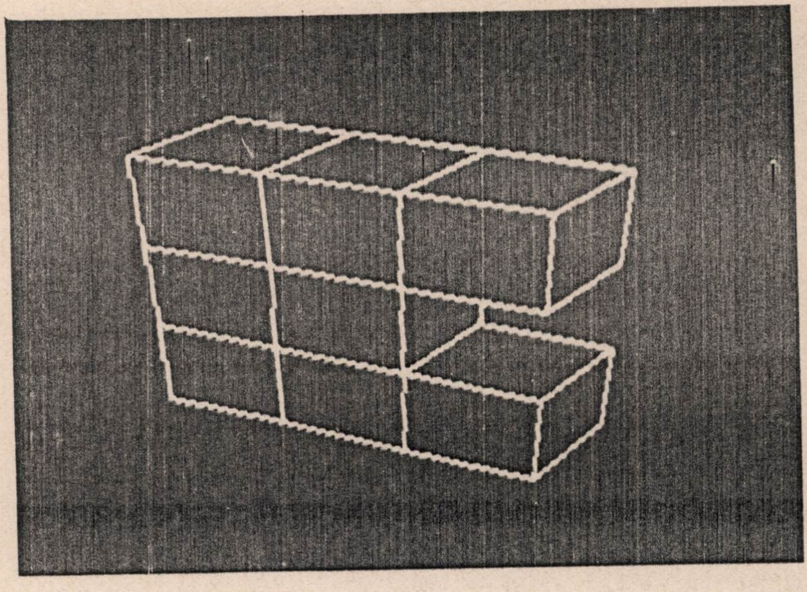


Fig. 5.6 An object with a hole

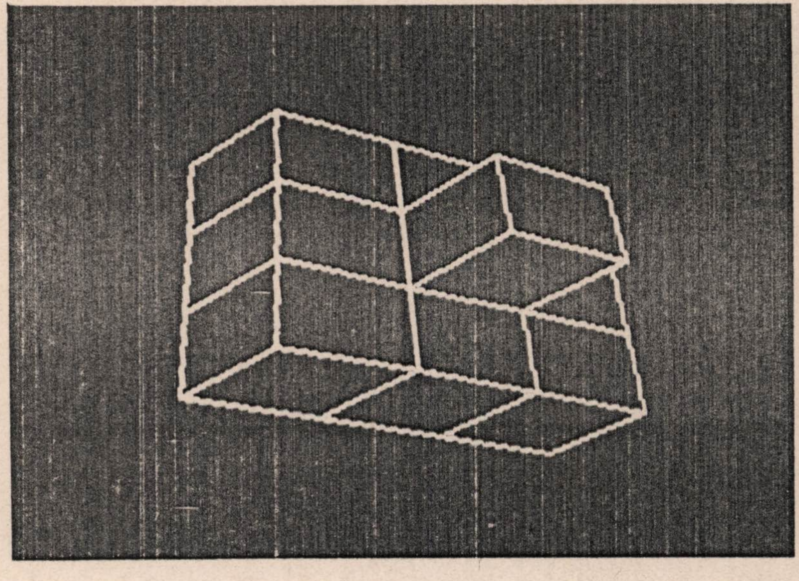


Fig.5.7 An object with a central growth

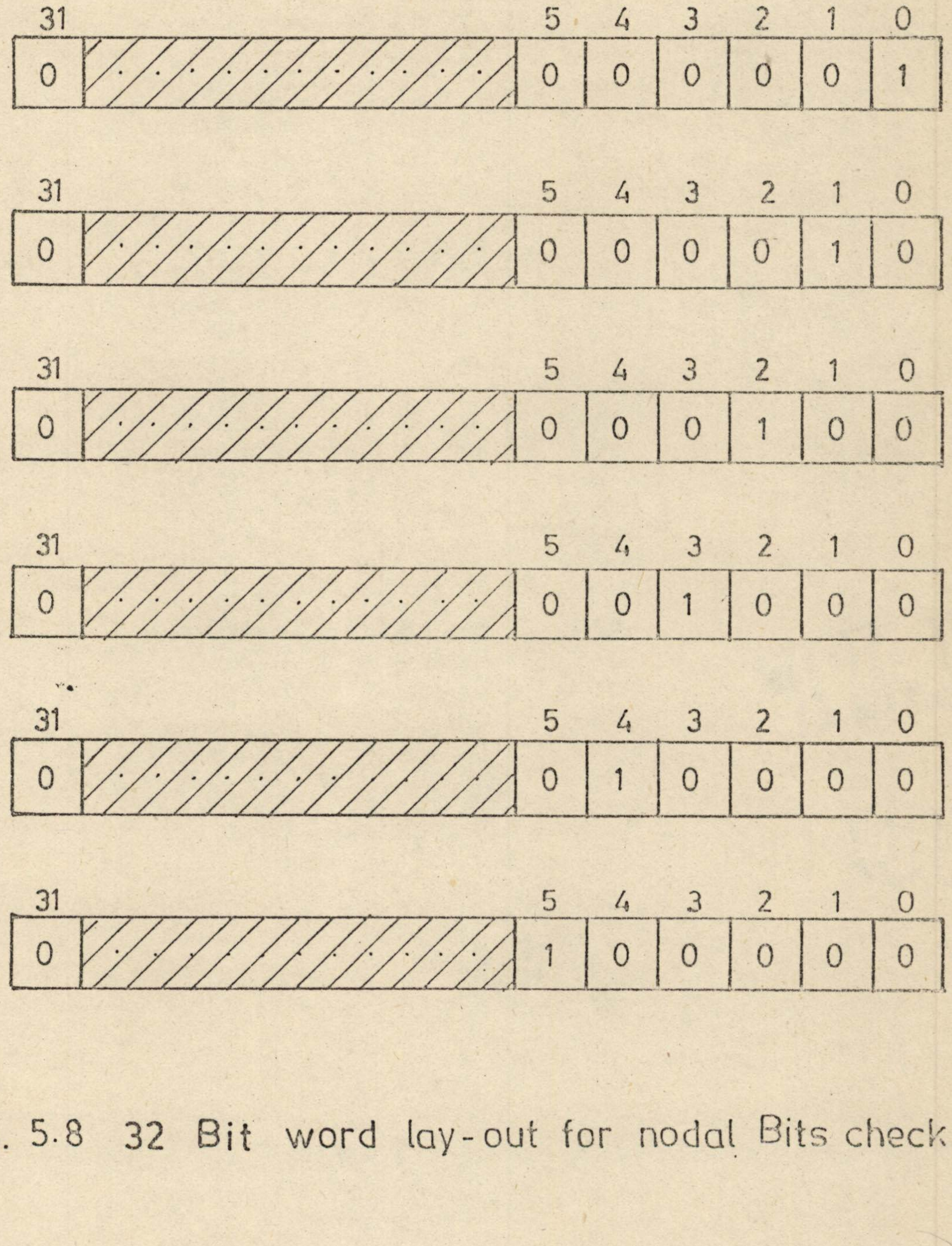


Fig. 5.8 32 Bit word lay-out for nodal Bits checking

for visualising the models from any distance and/or position in space:

i) Perspective views without the hidden lines removed

ii) Perspective views with the hidden lines removed

iii) Perspective views with the hidden lines dashed

The geometric modeling system was implemented in FORTRAN Language on VAX-11/780 computer system using Tektronix-4027 graphics terminal. Graphics support was provided by the Interactive Graphics Library (IGL) package. Figs. (5.9 - 5.18) show 3-D views of various modeled objects.

5.4 CONCLUSIONS

In this chapter, an overall description about the developmental and implementation details of the geometric modeling system based on Hex-tree technique have been given. A detailed discussion regarding the importance and the basic requirements for good user-interface has also been done. A schematic representation of the Hex-tree based geometric modeling System is given and its various features are discussed. Finally, 3-D views of the modeled objects are presented through various figures (Fig. 5.9 to Fig. 5.18).

—

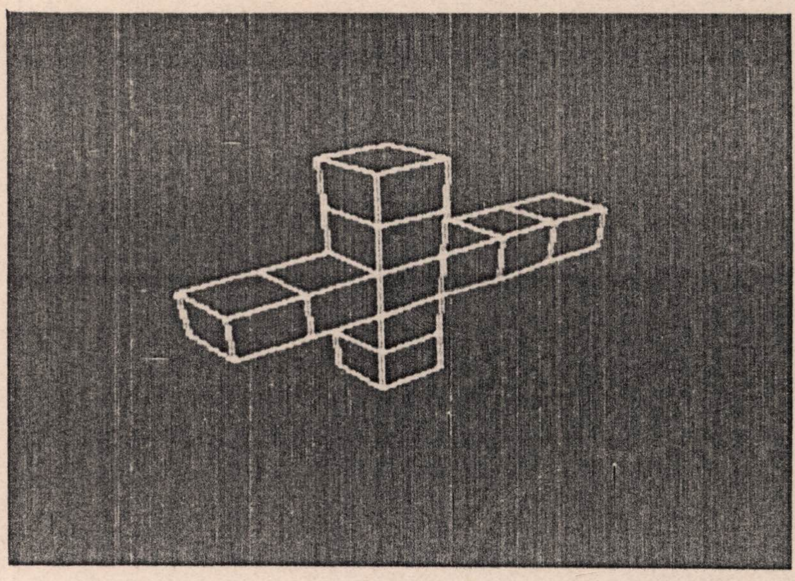


Fig. 5.9 An object showing cell growth in $\pm$ X and $\pm$ Y directions

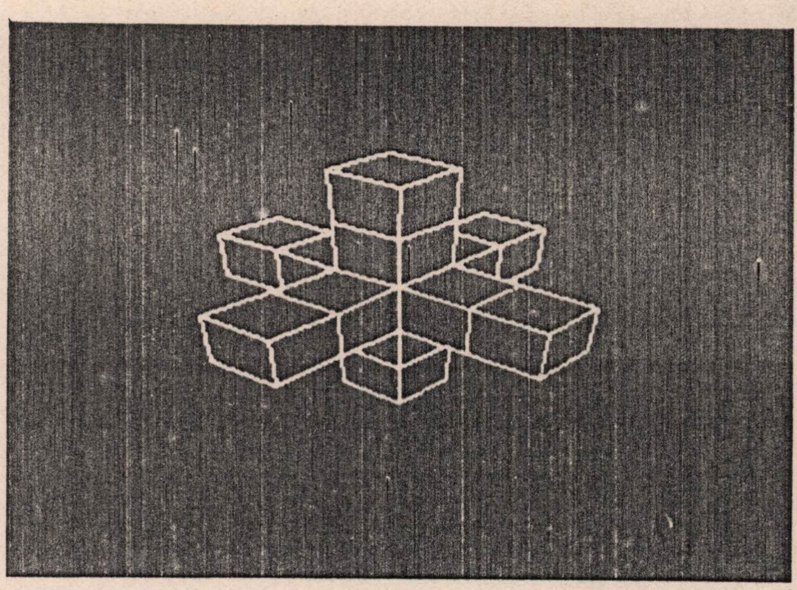


Fig. 5.10 An object showing cell growth in $\pm$ X, $\pm$ Y and $\pm$ Z directions

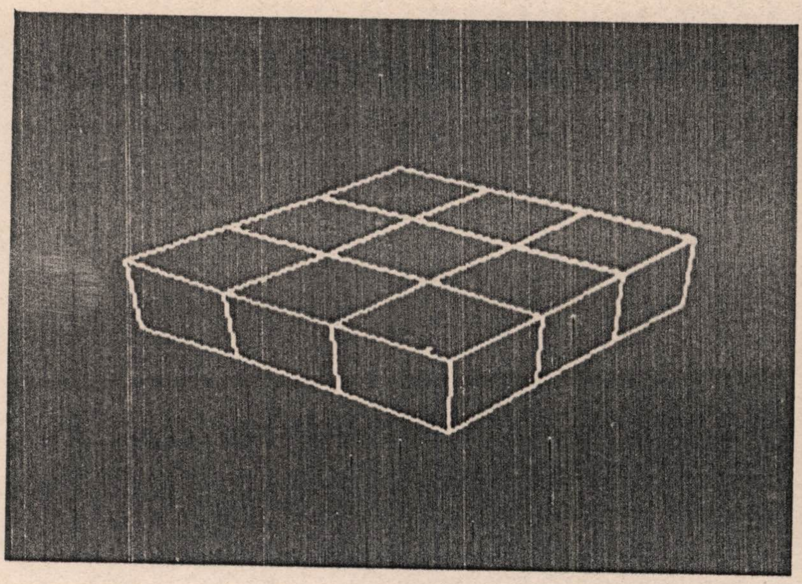


Fig. 5.11 An object grown in X-Z plane

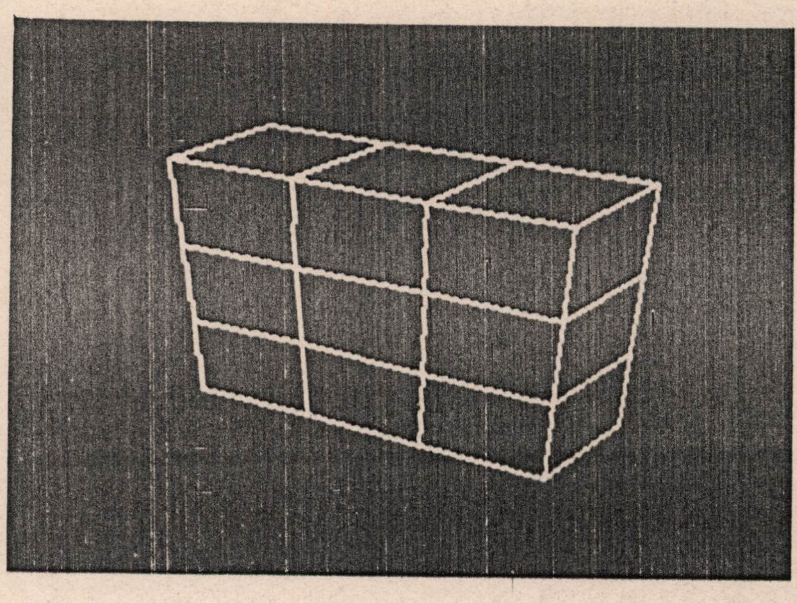


Fig. 5.12 An object grown in X - Y plane

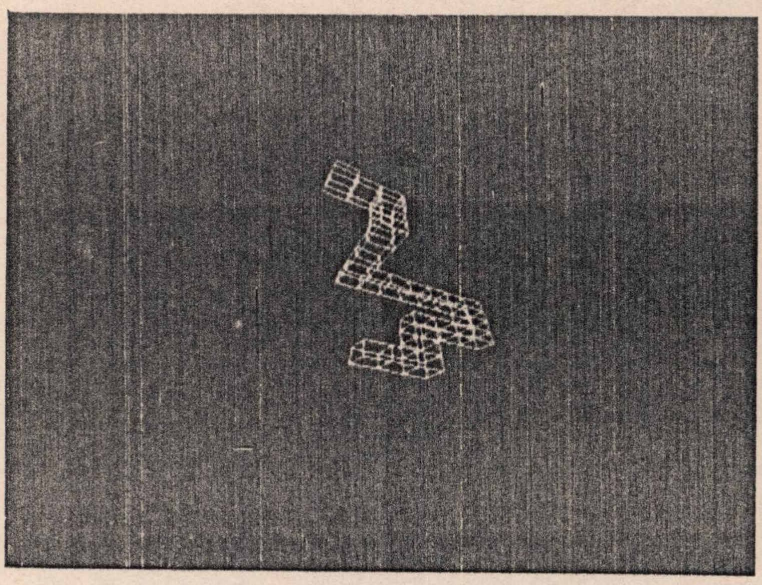


Fig. 5.13 An object with hidden lines

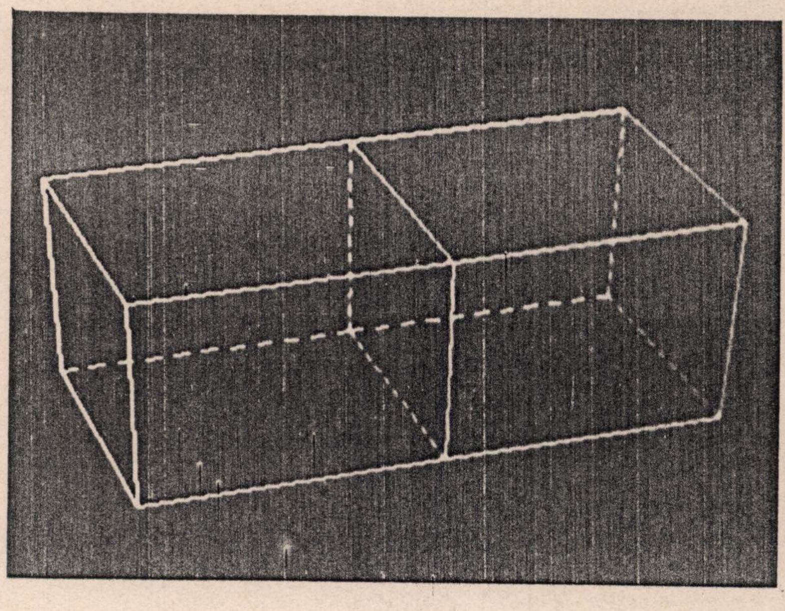
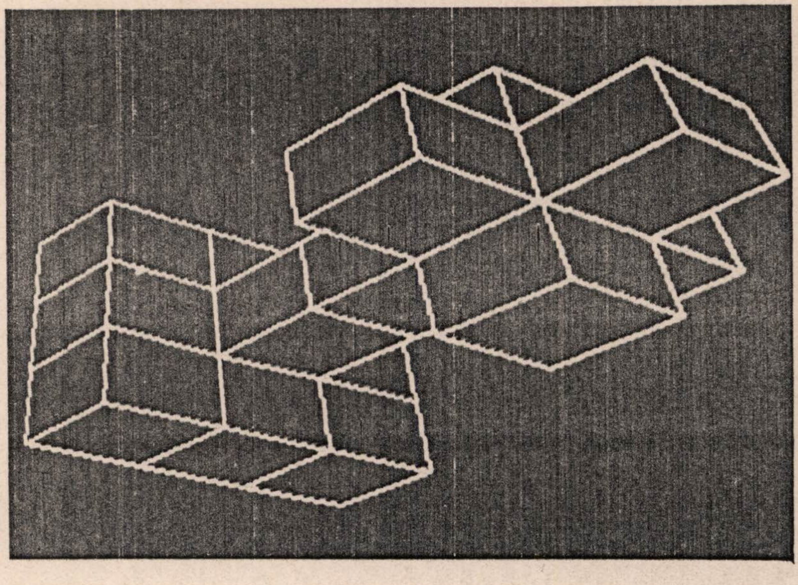


Fig. 5.14 A hidden lines 'dashed' object



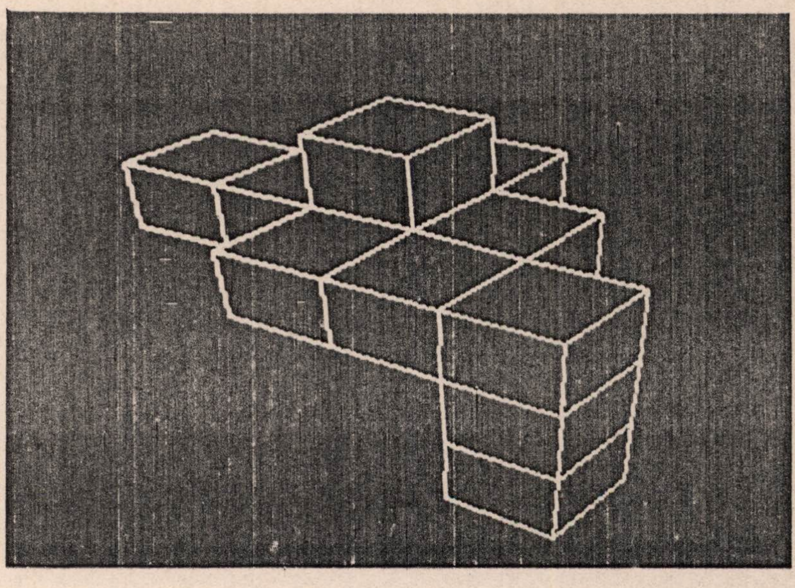
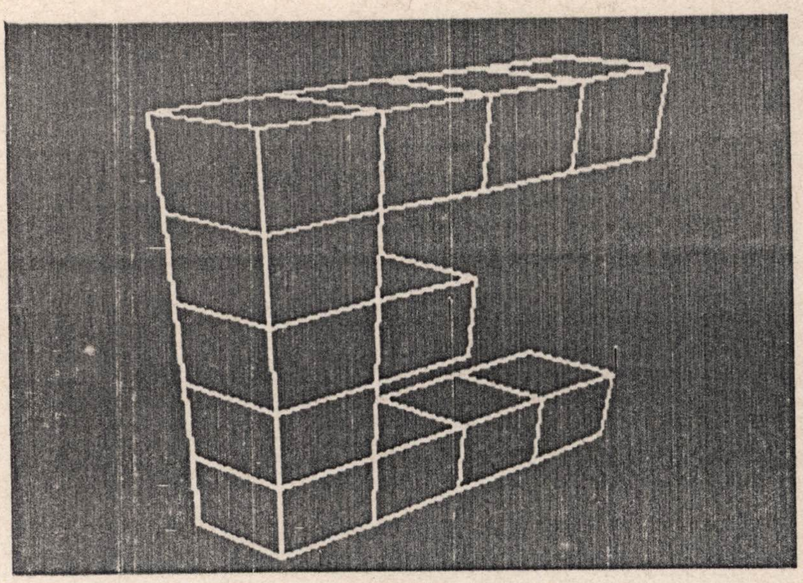


Fig. 5.15 Two hidden lines removed objects



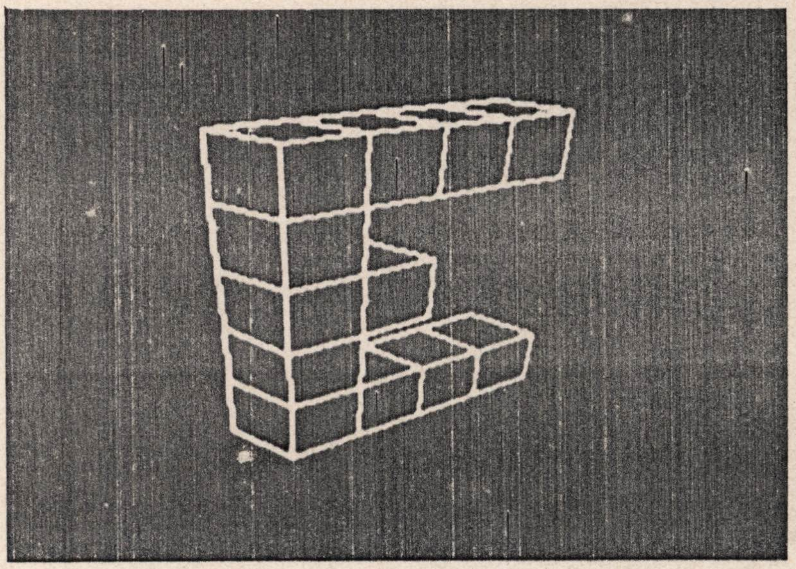


Fig. 5.16 An object viewed from same observer position with different perspective plane distances

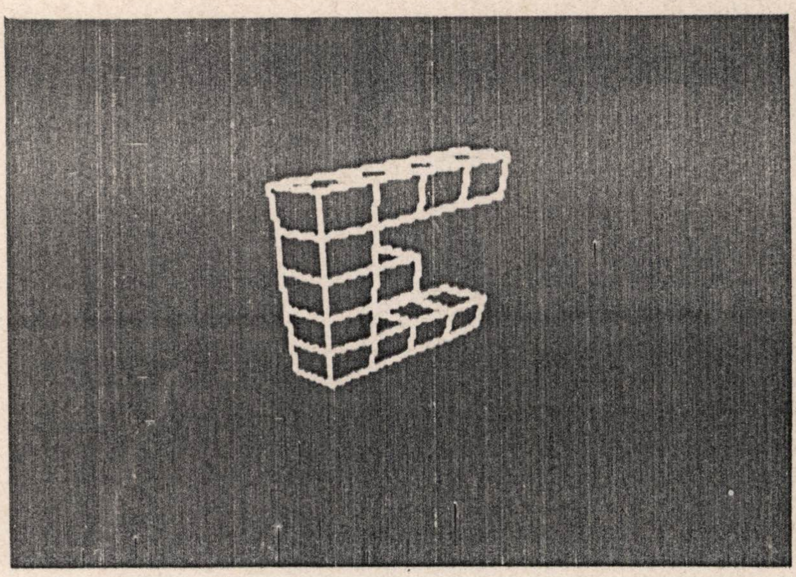


Fig. 5.17 The object shown in Fig. 5.16 viewed from same observer position with another perspective plane distance

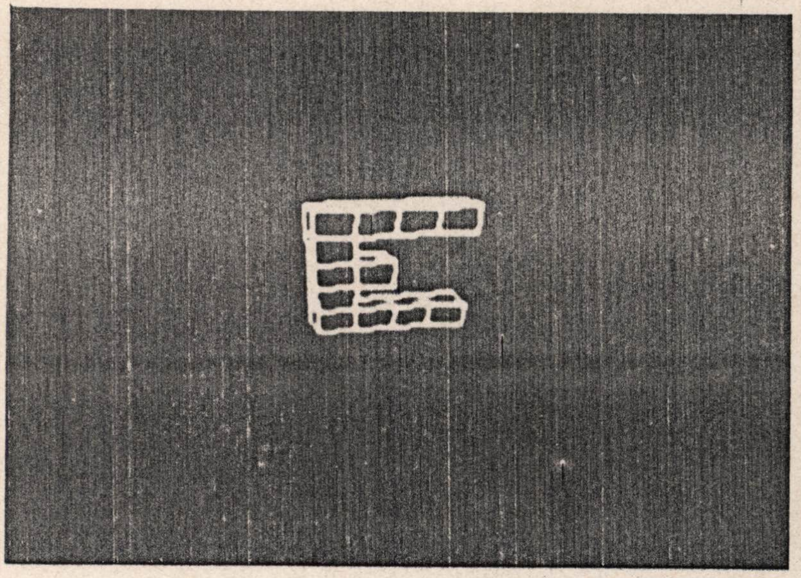


Fig. 5.18 A rotated 'E' object

CHAPTER 6

COMPUTATION OF INTEGRAL PROPERTIES

6.1 INTRODUCTION

When any shape has been modeled, it will often require engineering analysis to determine its integral properties (volume, center of gravity, moment of inertia and similar properties). The computation of these properties is an important problem in computer aided design, robotics and other related fields. These properties of a solid 'S' can be defined by triple (volumetric) integrals over subsects of three dimensional Eucledean space.

i.e. $I = \int_S \rho f(p)\, dV$ 6.1.1

where, $P = (x,y,z)$ is a point of Eucledean 3-space (E^3), dV is the volume differential, f is a polynomial and ρ is the density of the material. Depending upon the value of f(p), the equation defines various integral properties of a homogeneous solid, S. For example, when $f(P) = \frac{1}{\rho}/1$, the equation defines the volume/mass of the solid respectively and when $f(P) = x^2 + y^2$, it defines the moment of inertia of the solid about z axis.

Most of the known methods for calculating these properties of solid models are associated with representation methods [40]. Integral properties of solid represented by Boundary representation scheme are evaluated by surface integration using direct integration. In this method, the integral of a function f(x,y,z) over a polyhedral solid (e.g. prism) is evaluated by adding appropriately signed contributions of the prisms defined by the faces and their xy projections. This method is attractive for polyhedral objects represented by their boundaries. In the case of curved objects, a polyhedral approximation method is used. The curved objects faces are approximated with triangular facets in this approach.

One of the techniques for computing integral properties of solid objects represented by Constructive Solid Geometry scheme is by converting CSG representation into approximate cellular decompositions based on equally or variably sized blocks [41]. In such methods, the algorithms generate a collection of quasi-disjoint cells whose union approximate the solid and compute the integral properties of this solid by adding the contributions of the individual cells.

A quasidisjoint decomposition of a solid is a segmentation of the solid into smaller solids which have no 'holes' and have disjoint interiors. Any quasidisjoint decompositions of a solid 'S' takes the form

$$S = U_i C_i \quad \ldots \quad \ldots \; 6.1.2$$

where, C_i represent the smaller solids (cells).

Since cells C_i have disjoint interiors, any integral over S can be decomposed into a sum of integrals form

$$\int_S \rho f dV = \sum_i \int_{C_i} \rho f \, dV \quad \ldots \; 6.1.3$$

Evaluation of each C_i integral is easy when the cells are of simple shape like cubical cell. This approach is used in cellular decomposition based schemes.

6.2 CALCULATION OF INTEGRAL PROPERTIES OF HEX-TREE BASED SOLID MODELS

In the Hex-tree scheme, a solid object is represented as the collection of cubical cells of equal size. Hence, the integral properties can be calculated in a similar way as it was discussed in equation 6.1.3. Some of the integral properties can be derived as follows.

Let

1,b,h	=	length, breadth, height (depth) of the cubical cell shown in Fig. 6.1a.
N	=	the number of cubical cells in a model
ρ	=	density of the material
g	=	acceleration due to gravity
m	=	mass of a cubical cell

Volume

Volume of a cubical cell $v = l x b x h$

Volume of a model $V = N \times v$ ... 6.2.1

Weight

Weight of a cubical cell $w = m \times g$

$= v \times \rho \times g$

Weight of a model $W = N \times w$

$= N \times v \times \rho \times g$ ... 6.2.2

Let w_1, w_2 .... w_N be the weight of each cubical cell, (x_1, y_1, z_1), (x_2, y_2, z_2) .... (z_N, y_N, z_N) be the center of gravity of each cell, and $(\bar{x}, \bar{y}, \bar{z})$ be the center of gravity of the model shown in Fig. 6.1b. The coordinates of the center

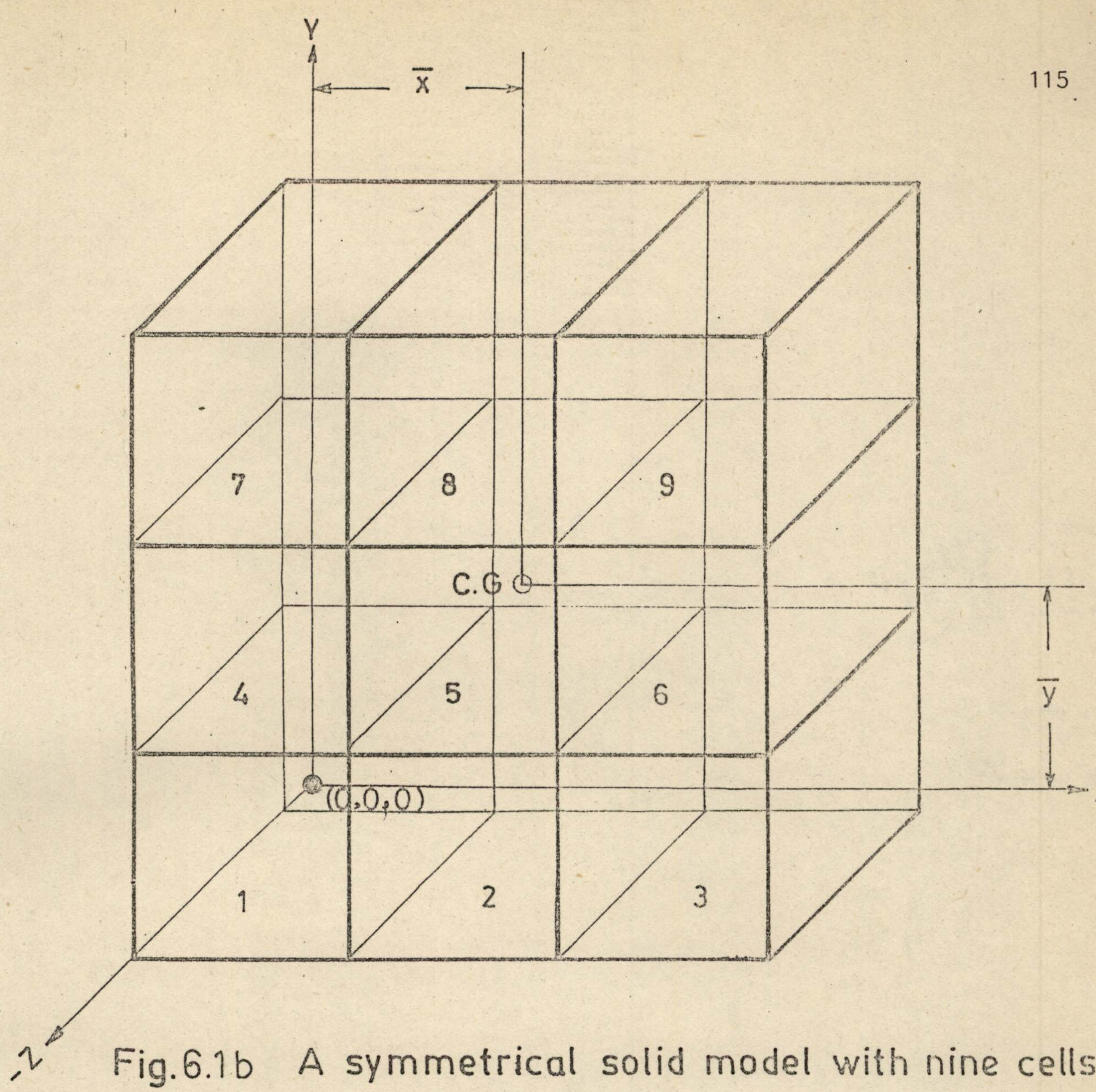


Fig. 6.1b A symmetrical solid model with nine cells

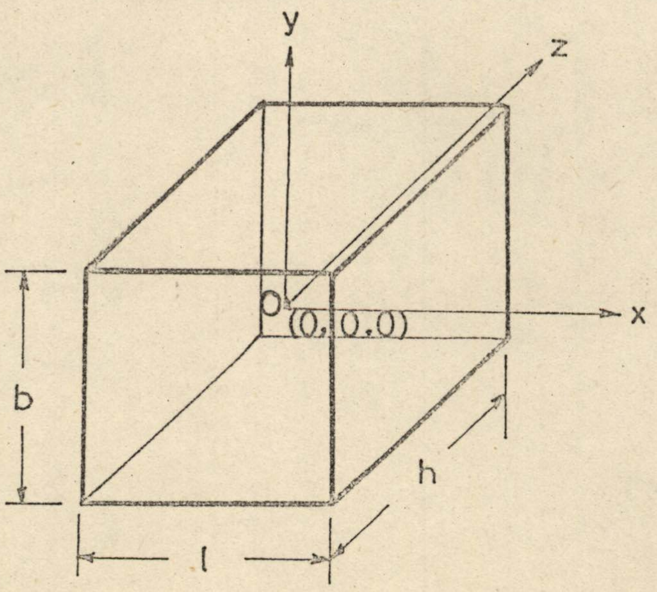


Fig. 6.1a A cell details

of gravity of the model can be found out using the following relations :

$$\left.\begin{aligned}
\bar{x} &= \frac{\sum_{i=1}^{N} (w_i x_i)}{\sum_{i=1}^{N} (w_i)} \\
\bar{y} &= \frac{\sum_{i=1}^{N} (w_i y_i)}{\sum_{i=1}^{N} (w_i)} \\
\bar{z} &= \frac{\sum_{i=1}^{N} (w_i z_i)}{\sum_{i=1}^{N} (w_i)}
\end{aligned}\right] \qquad \ldots \quad 6.2.3$$

Moment of Inertia

Consider a cubical cell having mass m, shown in Fig. 6.1a. The moment of inertia of this cell about the axes ox, oy, oz passing through its center of gravity are :

$$\left.\begin{aligned}
I_{ox} &= 1/12\ m[b^2 + h^2] \\
I_{oy} &= 1/12\ m[l^2 + h^2] \\
I_{oz} &= 1/12\ m[l^2 + b^2]
\end{aligned}\right] \qquad \ldots \quad 6.2.4$$

In order to transfer the moment of inertia of this element from axes passing through its center of gravity (G) to parallel axes passing through some other point, the following relations are used. In this regard, G has coordinates x_G, y_G and z_G defined from the new coordinates axes ox', oy' and oz'.

$$\left.\begin{aligned} I_{ox}' &= I_{ox} + m(y_G^2 + z_G^2) \\ I_{oy}' &= I_{oy} + m(x_G^2 + z_G^2) \\ I_{oz}' &= I_{oz} + m(x_G^2 + y_G^2) \end{aligned}\right] \quad \ldots \quad 6.2.5$$

(a) Moment of Inertia of Symmetrical Models

Consider the symmetrical Hex-tree model shown in Fig. 6.1b Let n_x, n_y and n_z be the number of cubical cells in the X,Y,Z directions. The moment of inertia I_X, I_Y, and I_Z about the axes passing through the center of gravity of the model can be calculated using the equations given below :

$$\left.\begin{aligned} I_X &= 1/12\ m[(n_y b)^2 + (n_z h)^2] \\ I_Y &= 1/12\ m[(n_x l)^2 + (n_z h)^2] \\ I_Z &= 1/12\ m[(n_x l)^2 + (n_y b)^2] \end{aligned}\right] \quad \ldots \quad 6.2.6$$

(b) Moment of Inertia of Unsymmetrical Models

The moment of inertia equations in this case can be derived in a similar way as equations 6.2.5. Consider an unsymmetrical model shown in Fig. 6.2. Let (x_1, y_1, z_1), (x_2, y_2, z_2) .... (x_N, y_N, z_N) are the centers of gravity of N individual cubical cells of the model defined with respect to the reference axes OX, OY, OZ and $\bar{x}$, $\bar{y}$, $\bar{z}$ are the coordinates of the center of gravity of the model.

Moment of inertia $(I_1, I_2 \ldots . I_N)$ of the cells (1,2...N) about the x-axis passing through the center of gravity of the model can be written as :

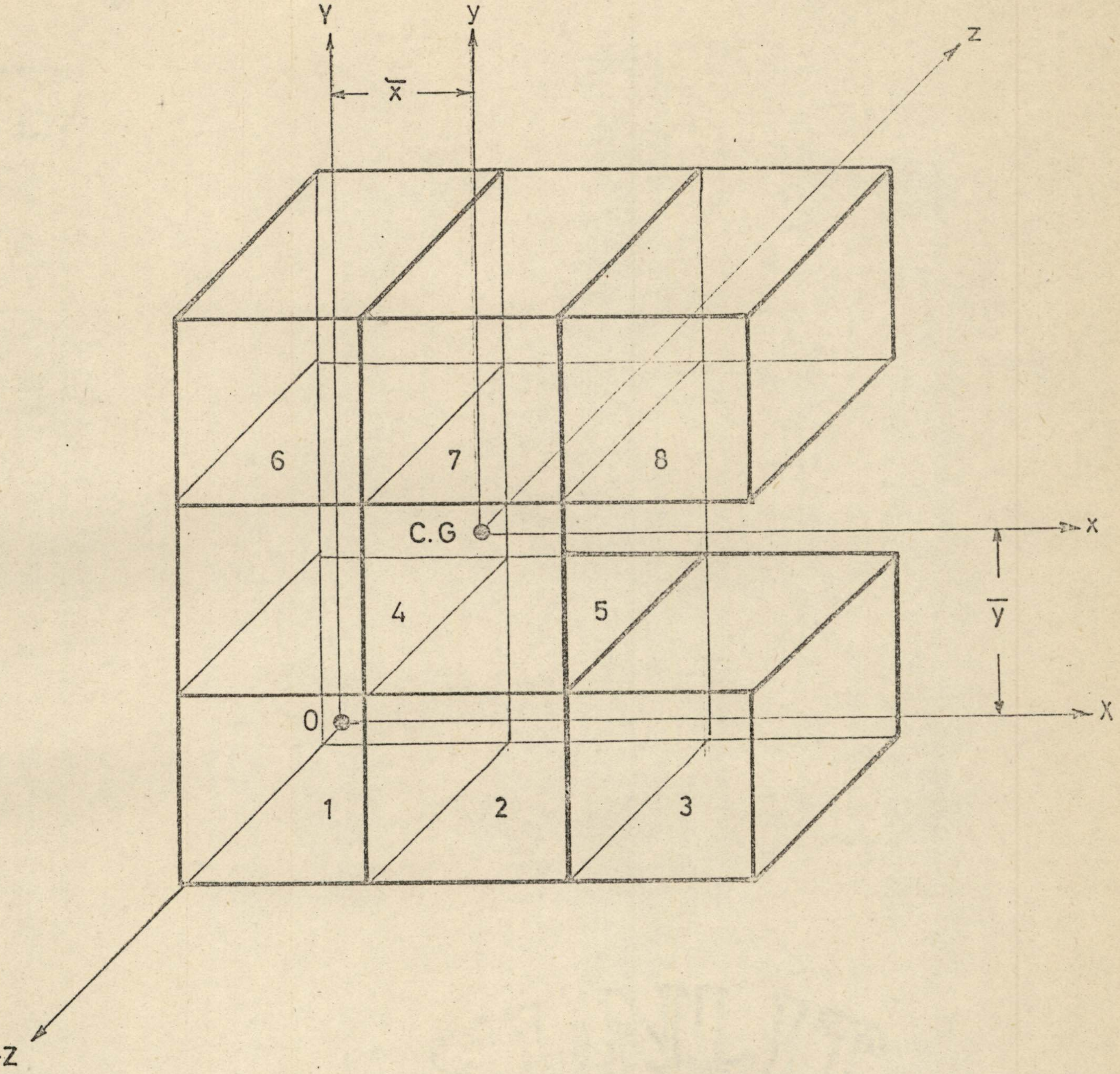


Fig. 6. 2 An unsymmetrical solid model with eight cells

$$I_1 = 1/12\ m_1(b^2 + h^2) + m_1[(\bar{y} - y_1)^2 + (\bar{z} - z_1)^2]$$

$$I_2 = 1/12\ m_2(b^2 + h^2) + m_2[(\bar{y} - y_2)^2 + (\bar{z} - z_2)^2]$$

$$I_3 = 1/12\ m_3(b^2 + h^2) + m_3[(\bar{y} - y_3)^2 + (\bar{z} - z_3)^2]$$

$$\vdots$$

$$I_N = 1/12\ m_N(b^2 + h^2) + m_N[(\bar{y} - y_N)^2 + (\bar{z} - z_N)^2]$$

Hence, the moment of inertia of the model about the x-axis passing through its center of gravity can be formed as

$$I_x = \sum_{i=1}^{N} m_i \left[\frac{1}{12} (b^2+h^2) + [(\bar{y}-y_i)^2 + (\bar{z}-z_i)^2] \right] \quad \ldots\ 6.2.7$$

Similarly, the moment of inertia of the model about the y and z axes can be derived as

$$I_y = \sum_{i=1}^{N} m_i \left[\frac{1}{12} (l^2+h^2) + [(\bar{x}-x_i)^2 + (\bar{z}-z_i)^2] \right] \quad 6.2.8$$

$$I_z = \sum_{i=1}^{N} m_i \left[\frac{1}{12} (l^2+b^2) + [(\bar{x}-x_i)^2 + (\bar{y}-y_i)^2] \right] \quad \ldots\ 6.2.9$$

As the cells are cubes of same size l = b = h, and is equal to 'a' (say). Hence the equations (6.2.7-6.2.9) can be rewritten as

$$I_x = \sum_{i=1}^{N} m_i \left[\frac{1}{12} (a^4) + [(\bar{y}-y_i)^2 + (\bar{z}-z_i)^2] \right] \quad \ldots\ 6.2.10$$

$$I_y = \sum_{i=1}^{N} m_i \left[\frac{1}{12} (a^4) + [(\bar{x}-x_i) + (\bar{z}-z_i)^2] \right] \quad \ldots \quad 6.2.11$$

$$I_z = \sum_{i=1}^{N} m_i \left[\frac{1}{12} (a^4) + [(x-x_i)^2 + (y-y_i)^2] \right] \quad \ldots \quad 6.2.12$$

The following steps are involved in the calculation of integral properties of a solid model represented by the proposed technique.

1. Obtain the values of the parameters ρ, g
2. Get the Hex-tree details and the cell dimensions
3. Find out the appropriate cell positions by searchng the Hex-tree nodes and then obtain (x_i, y_i, z_i) for $i = 1,2,\ldots N$
4. Compute the volume and weight of the model using equations 6.2.1 and 6.2.2 respectively.
5. Calculate the center of gravity of the model using equation 6.2.3. Moments of inertia are calculated using equation 6.2.6 (For symmetrical models) and equations 6.2.10, 6.2.11 and 6.2.12 (For unsymmetrical models).

An unsymmetrical Hex-tree model consists of 8 cells as shown in Fig. 6.2 (a view of this model is shown in Fig. 5.6) has been considered for computing the integral properties. It is assumed that the model is made of Aluminium material. The properties of this model are computed and given below a = 2.00 cm $\qquad$ d = 2700 kg/m^3

Volume = 64E - 06 m^3

Weight = 1.693 kg.wt

Center of gravity = C(1.75, 2.00, 0.00)

Moment of inertia

$$I_x = 0.74822E - 04 \text{ kg.m}^2$$

$$I_y = 0.65063E - 04 \text{ kg.m}^2$$

$$I_z = 1.16899E - 04 \text{ kg.m}^2$$

6.3 CONCLUSIONS

The calculation of integral properties of Hex-tree based solid models have been discussed. A brief description about the methods of computing these properties for constructive solid geometry, Boundary representation and Cellular decomposition schemes are also given. The volume, weight, center of gravity and moment of inertia of a typical Hex-tree solid model have been computed and the results are shown. These calculations are easier and fast as the Hex-tree approach is based on equally sized cubical cells.

—

CHAPTER 7

CONCLUSIONS AND SUGGESTIONS FOR FURTHER INVESTIGATIONS

7.1 SUMMARY AND CONCLUSIONS

Modeling the geometry of rigid solid objects using computer based systems is becoming increasingly important in CAD/CAM, computer vision, robotics and other fields that deal with spatial phenomena. The success of a geometric/solid modeling system greatly depends on the unambiguous representation of 3-D objects.

The major objective of this thesis work has been to explore the possibility of developing a representational technique for geometric modeling of solid objects. Towards this end, the thesis has culminated in the design and implementation of a new geometric modeling approach based on HEX-TREE representational technique using a cubical cell as primitive. The system so developed can create, modify and display solid objects of any shape.

The modeling of 3-D objects has been done, by constructing cubes over the six faces of this primitive and continuing this process till the desired shape is obtained. This constructing procedure can be viewed as forming a tree structure with root node as the primitive cell and the first level branch nodes as six cubical cells on the faces of the rooted cubical cell, and in a similar way the subsequent level's branch nodes. The full identity of a node (i.e. cubical cell) is represented in six Bits of a word length indicating the cell presence or absence on the six faces.

The development of plane-surfaced object models was straight forward. However, curve-surfaced objects have been modeled as follows :

(i) Selection of a primitive

(ii) Grow the primitive along the six faces

(iii) Check whether the approximate shape of the target object is obtained or not

(iv) If the desired shape is not achieved, change the primitive size till a satisfactory result is reached.

Figs (3.12 - 3.17) show few models having curved boundaries.

As the modeling of plane and curve-surfaced objects are based on single cubical cell primitive, and due to the hierarchical nature of the tree oriented data structure, any type of object could be modeled easily as compared to other popular schemes: For example, multi-directional joint based models (Figs. 5.9 - 5.10), multi-layer and multi-directional objects (Fig. 5.15), object with hole (Fig. 5.6), object with central growth (Fig. 5.7) etc. This peculiar characteristics strongly justify the development of this scheme for curved object modeling also.

To get the realistic view of the models, hidden line elimination algorithms have been developed for plane as well as curve surfaced objects. Out of the two algorithms developed, the first one is suitable for simple objects, in which visibility checking is done on the face-basis of the object. The second algorithm is meant for complex shapes, in which the visibility testing is performed on part of a line-segment level. The complete hidden lines removed perspective picture of an object model, viewed from various distances and/or positions could be perceived. Provisions are incorporated in the first algorithm to get the hidden lines 'dashed', if so desired by the viewer, in order to get an overall shape of the model. It is also possible to get the screen image, without applying hidden line algorithms. Various shapes generated using these algorithms are shown in Figs. (4.2 - 4.7) and Figs. (5.11 - 5.18).

The feasibility of the approach has been shown for performing boolean operations (Union, difference and intersection). The details have been discussed and typical photographs obtained after performing union, difference and intersection operations on two objects, shown in Figs. (3.9 - 3.11). The methods can be extended in a similar way to complex objects also.

Necessary methods have been developed to calculate some of the integral properties of the Hex-tree based solid models. The volume, weight, center of gravity and moment of inertia of an unsymmetrical model have been computed and shown in Chapter 6.

Screen layout is one of the ways of providing a friendly User-Interface. Through the screen layout-interface provided in the system, the designer is able to create multiple screen areas depending on his requirements (Figs. 5.2 & 5.3).

The reported approach has several advantages over existing popular schemes. In addition to the advantages of a tree based, recursive type data-structure, the following are the major achievements of the Hex-tree approach.

(a) It was experienced that any three dimensional object could be modeled using this technique. The same data structure is sufficient for both plane-faced and curve-surfaced objects. The curved object's precision is limited to the size of the cubical cell.

(b) Same hidden-line algorithm is applicable for the display of both plane-faced and curve-surfaced shapes. It is possible to view the model from various distances and/or positions in space. Because of the avoidance of interference checking, the display of the hidden lines removed model are fast and efficient.

(c) Boolean operations are simple and fast in this approach, as the whole chain of representation and display are performed on the face basis.

(d) Computation of integral properties, such as, volume, center of gravity and moment of inertia of a Hex-tree based model is easier and faster. This is because, in this method, the primitive is a cubical cell and all the cells are of equal size. Moreover, as the approach is based on equal-sized cubical cells, the additional computational requirement for representation conversion or polyhedral approximation of curved objects required in popular schemes are avoided.

(e) An overall improvement in computational efficiency and memory saving is also achieved. This is due to the Bit level representation, and logic based search algorithms for tree traversal.

The object represented by the Hex-tree method is **valid** as the primitive cubical cell represent a valid physical object. Given a representation, there exists only one solid that corresponds to it and hence this scheme is **unambiguous**. Given an object, there is only one way of decomposing it into primitive solid and hence the scheme possesses the **uniqueness** property also. Another property of this scheme is its **consciseness** as Hex-tree representations are convenient to store and transmit over data links and contain relatively few redundant data.

7.2 SUGGESTIONS FOR FURTHER INVESTIGATIONS

Following are the suggestions for further investigating the work carried out in this thesis:

In the reported approach the primitive is a cubical cell. It is desirable to explore the possibilities of combining common faces of adjacent cells. An attempt in this line will result in a good amount of memory saving. This will also result in an enhancement in saving the computing speed for hidden line removal process.

The smoothness of the periphery of a curved object depends on the primitive size. This problem can be alleviated if surface developing methods such as B-spline or Bezier techniques are incorporated with the Hex-tree technique. This sort of hybrid approach can be thought of as another investigation.

In this work we have not developed solid models of sufficiently complex objects as our objective was mainly for the development of a feasible method. Also, we have not tried to make use of those developed models for any specific application. It is a good idea to take up an application area where simple or complex models are involved and build up an entire system based on this modeling scheme.

The success of interfacing technique of a well designed software package is closely related to the interface between the package and the user. A well adapted command language is necessary to create an interactive environment for the system to control which processes should be activated and which data should be passed. The command language must be simple and so adaptive that it allows the user the option either of utilizing his knowledge and ideas without being disturbed by a dialogue with the system, or if desired, of answering the dialogued queries of the system. This kind of facilities and other interfacing techniques can also be incorporated to the system to make the user interface more attractive.

In the formation of any engineering design project, some type of analysis is required. The analysis may involve stress-strain calculations,heat-transfer computations, integral or mass property calculations, etc. As we have developed methods only for computing some integral properties, the development of appropriate methods for other engineering analyses can also be investigated.

—

APPENDIX

CALCULATION OF THE POINT OF INTERSECTION OF TWO LINES

A general point $\bar{P} \equiv (x, y)$ on the line joining two points $\bar{P}_1 = (x_1, y_1)$ and $\bar{P}_2 = (x_2, y_2)$ can be written as

$$(1 - \alpha)(\bar{P}_1) + \alpha(\bar{P}_2) \qquad \ldots (A.1)$$

where $0 \leq \alpha \leq 1$

α = (distance of $\bar{P}$ from $\bar{P}_1$)/(distance of $\bar{P}_2$ from $\bar{P}_1$)

$\equiv$ means "equivalent to"

The point of intersection $\bar{P} \equiv (x, y)$ of two lines joining

i) $\bar{P}_1 \equiv (x_1, y_1)$ to $\bar{P}_2 \equiv (x_2, y_2)$; and

ii) $\bar{P}_3 \equiv (x_3, y_3)$ to $\bar{P}_4 = (x_4, y_4)$

can be found out as follows :

Let a general point on line (i) is $(1 - \alpha)\bar{P}_1 + \alpha \bar{P}_2$ for some α , and a general point on line (ii) is $(1 - \beta)\bar{P}_3 + \beta \bar{P}_4$ for some β, where β is similar to α . Thus the point of intersection $\bar{P}$ lies on both lines and all that is necessary to calculate $\bar{P}$ is to find the unique values of α and β such that

$$\bar{P} \equiv (1-\alpha)\bar{P}_1 + \alpha\bar{P}_2 = (1 - \beta)\bar{P}_3 + \beta\bar{P}_4 \qquad \ldots \text{ A.2}$$

that is

$$(1 - \alpha)x_1 + \alpha x_2 = (1 - \beta)x_3 + \beta x_4$$

and

$$(1 - \alpha)y_1 + \alpha y_2 = (1 - \beta)y_3 + \beta y_4$$

Hence

$$\alpha(x_2 - x_1) + \beta(x_3 - x_4) = (x_3 - x_1)$$

and

$$\alpha(y_2 - y_1) + \beta(y_3 - y_4) = (y_3 - y_1)$$

Thus we have two equations in two unknowns

Let $\triangle = (x_2 - x_1)(y_3 - y_4) - (x_3-x_4)(y_2-y_1) \neq 0$ ... A.3

then

$\alpha = [(x_3-x_1)(y_3-y_4) - (x_3-x_4)(y_3-y_1)]/\triangle$ ... A.4

Using equation A.4, we can get the point P by substituting in equation A.2.

—

8. Boyse, J. W., 'Preliminary Design for a Geometric Modeler', Res. Pub. GMR-2768, Comp. Science Dept., General Motors Research Laboratories, Warren, Mich, 1978.

9. Braid, I. C. and Lang, C. A., 'Computer Aided Design of Mechanical Components with Volume Building Blocks', pp. 173-184 in [25].

10. Braid, I.C., 'Designing with Volumes : Computations of Weight, Centre of Gravity, Moments of Inertia and Principal Axes', C.A.D. Group Doc. 81, University of Cambridge, U.K., 1973.

11. Braid, I. C., 'Geometric Modelling-ten Year On', CAD Group Doc. 103, Computer Laboratry, Cambridge University, Cambridge, UK, 1979.

12. Brown, C. M., 'PADL-2: A Technical Summary', IEEE Computer Graphics and Applications, Vol. 2, pp. 69-84, March 1982.

13. Ingrid Carlbon, Indranil Chakravarty and David Vandersehel', 'A Hierarchical Data Structure for Representing the Spatial Decomposition of 3-D Objects', IEEE Computer Graphics and Applications, pp. 24-31, 1985.

14. Chiyokura, H. and F. Kimura, 'Design of Solids with Free Form Surfaces', Computer Graphics (Proceedings of SIGGARAPH'83) Vol. 17, pp. 289-298, 1983.

15. Cook, P. N., et.al., 'Volume and Surface Area Estimates Using Tomographic Data', IEEE Trans. on PAMI, Vol. 2, pp. 478-479, Sept. 1980.

16. Doctor, L. J. and J. G. Torborg, 'Display Techniques for Octree Encoded Objects', IEEE Computer Graphics and Applications, Vol. 1, pp. 29-38, 1981.

17. Eastman, C. M., et al., 'GLIDE, : a system for implementing design data bases', Res. Rep. No. 76, Inst. of Phys. Planning, Carnegie - Millon University, 1978.

18. Fitzgerald, W., et al., 'GRIN : Interactive Graphics for Modeling Solids', IBM Journal of R&D, Vol. 24, pp. 281-294, 1981.

19. Foley, J. D. and A. Van Dam, 'Fundamentals of Interactive Computer Graphics', Addison Wesley, Reading, Mass, 1982.

20. Fujimura, K., Toriya, H., Yamaguchi, K. and Kunni, T. L., 'An Enhanced Oct-tree data structure and operation for solid modeling', Tech. Report 83-01, Dept. of Information Science, Faculty of Science, University of Tokyo, Japan, 1983.

21. Fujimura, K., Toriya, H. Yamaguchi, K. and Kunni, T. L., 'Oct-tree algorithms for solids Modelling', Tech. Report 83-04, Dept. of Information Science, Faculty of Science, University of Tokyo, Japan, 1983.

22. Galimberti, R., and Montanari, U., 'An algorithm for Hidden-line Elimination', Comm. ACM, Vol. 12, No. 4, pp. 206-211, April 1969.

23. Gouraud, H., 'Continuous Shading of Curved Surfaces', IEEE Trans. on Computers, Vol. C-20, No. 6, pp. 623-629. June 1971.

24. Giloi, W. K., 'Interactive Computer Graphics', Prentiçe Hall, Englewood. Cliffs, NJ, 1978

25. Hatvany, J. (Ed), 'Computer Languages for Numerical Control', North-Holland Publishing Company, Amsterdam,. 1973.

26. Hillyard, R. C., 'The Build Group of Solid Modelers', IEEE Computer Graphics and Applications, Vol. 2, pp. 43-52, 1982.

27. Hillyard, R. C. and Braid, I. C., 'Characterising Non-ideal Shapes in terms of Dimensions and Tolerances' Computer Graphics (Proceedings of SIGGRAPH '78) Vol. 12, pp. 234-238. 1978.

28. Hillyard, R. C. and Braid, I. C.,'The Analysis of Dimensions and Tolerances in Computer Aided Mechanical Design', Computer Aided Design, Vol. 10, pp. 161-166, 1978.

29. Hunter, G. M., 'Efficient Computation and Data Structures for Graphics', Ph.D. Dissertation, EE and CS Dept., Princeton University, 1978.

30. Jackins, C. L. and Tanimoto, S. L.,'Octrees and their use in Representing Three Dimensional Objects', Computer Graphics and Image Processing, Vol. 14, pp. 249-270, 1980.

31. Jared, G.E.M. and Varady, T., 'Synthesis of Volume Modelling and Sculptured Surfaces in Build.', Proce. 6th International Conf. and Exhibition on Comp. in Design and Engineering, 3-5 April 1984, pp. 481-495.

32. John R. Rankin, 'A Geometric, Hidden-Line Processing Algorithm', Comp. and Graphics, Vol. 11, No. 1, pp. 11-19, 1987.

33. Joshi, R. C., Durbari, H., Goel. S. and Sasikumaran. S., 'A Hierarchical Hex-tree Representaional Technique for Solid Modelling', Comput. and Graphics, Vol. 12, No. 2, pp. 235-238, 1988.

34. Joshi, R. C., Khushinder Nath and Sasikumaran. S., 'Volumetric Representation: A Technic for Representing and Manipulating 3-D Objects', National Workshop on Robotics, Organised by Dept. of Science and Technology, Govt. of India, PUNE, 29-30th April, 1987.

35. Joe, R. and Philip. S. (Ed.), 'Principles of Computer-Aided Design', Pitman Publishing Company, London, 1987.

36. Koppelman, G. M. and Wesley, M. A., 'OYSTER : A study of Integrated Circuits as Three Dimensional Structures', IBM Journal of R&D, Vol. 27, pp. 149-163, 1983.

37. Kubert, B., Szabo, J., and Giulieri, S., 'The Perspective Representation of Functions of Two Variables', Journal of ACM, Vol. 15, No. 2, pp. 193-205, April 1968.

38. Lang, C. A., 'A Three Dimensional Model Making Machine', pp. 109-118 in [25].

39. Lee, Y. T. and Requicha, A.A.G., 'Algorithms for Computing the Volume and Other Integral Properties of Solid objects', TM-35, Production Automation Project, University of Rochester, 1981.

40. Lee, Y. T. and Requicha, A.A.G., 'Algorithms for Computing the Volume and Other Integral Properties of Solids. I. Known Methods and Open Issues, Comm. of the ACM, Vol. 25, No. 9, pp. 635-641, 1982.

41. Lee, Y. T. and Requicha, A.A.G., 'Algorithms for Computing the Volume and Other Integral Properties of Solids II. A family of Algorithms Based on Representation Conversion and Cellular Approximation', Comm. ACM, Vol. 25, No. 9, pp 642-650, 1982.

42. Loutrel, P. P., 'A Solution to the Hidden-line Problem for Computer-drawn Polyhdera', IEEE Transactions on Computers, Vol. EC-19, pp. 205-211, 1970.

43. Markowsky, G. and Wesley, M.A., 'Fleshing Out Wireframes', IBM Journal of R&D, Vol. 24, pp. 582-597, 1980.

44. Marshall, P., 'Computer-Aided Process Planning and Estimating as part of an Integrated CAD/CAM System', Computer Aided Engineering Journal, pp. 187-192, October 1983.

45. Meagher, D., 'Geometric Modeling Using Octree Encoding', Computer Graphics and Image Processing, Vol. 19, pp. 129-147, 1982.

46. Myers, W., 'An Industrial Perspective on Solid Modeling', IEEE Computer Graphics and Applications Vol. 2, pp. 86-97, 1982.

47. Newell, M. E., Newell, R. C. and Sancha, T. L., 'A Solution to Hidden Surface Problem', Proce. ACM National Conf., 1972.

48. Newman, W. M. and Sproull, R. F., 'Principles of Interactive Computer Graphics', 2nd Ed., McGraw-Hill, New York, 1979.

49. Okino, N., et al., 'TIPS-1: Technical Information Processing System for Computer Aided Design, Drawing and Manufacturing', pp. 141-150 in [25].

50. Production Automation Project, 'An Introduction to PADL: Characteristics, Status, and Rationale', Tech Memo No. 22, Production Automation Project, University of Rochester, 1974.

51. Ramachandran Nair, K. N. and Sankar, R., 'An approach to Geometric Modeling of Solids Bounded by Sculptured Surfaces', Comput. and Graphics, Vol. 11, No. 2, pp. 113-120, 1987.

52. Requicha, A.A.G., 'Mathematical Models of Rigid Objects', Tech. Memo No. 28, Production Automation Project, University of Rochester, 1977.

53. Requicha, A.A.G. and Voelcker, H. B., 'Constructive Solid Geometry', Tech. Memo No. 25, Production Automation Project, University of Rochester, 1977.

54. Requicha A.A.G. and Tilove, R. B., 'Mathematical Foundations of Constructive Solid Geometry: General Topology of Closed Regular Sets', Tech. Memo. 27, Production Automation Project, University of Rochester, 1978.

55. Requicha, A.A.G., 'Representations of Rigid Solids - Theory, Methods and Systems', Computing Surveys, Vol. 12, pp. 437-464, 1980.

56. Requicha, A.A.G. and Voelcker, H.B., 'An Introduction to Geometric Modeling and its Applications in Mechanical Design and Production', In 'Advances in Information System Sciences' (Ed.) T. Tou, Vol. 8, pp. 293-238, 1981.

57. Requicha, A.A.G. and Voelcker, H.B., 'Solid Modeling: A Historical Summary and Contemporary Assessment', IEEE Computer Graphics and Applications, Vol. 2, pp. 9-26, 1982.

58. Requicha, A.A.G. and Voelcker, H. B., 'Solid Modeling: Curent Status and Research Directions', IEEE Computer Graphics and Applications, Vol. 3, pp. 25-37, 1983.

59. Requicha, A.A.G. and Voelcker, H.B., 'Boolean Operations in Solid Modeling: Boundary Evaluation and Merging Algorithms', Proce. of IEEE, Vol.73, No. 1, pp. 30-44, January 1985.

60. Roberts, L. G., 'Machine Perception of Three Dimensional Solids', Tech. Report No. 315, MIT Lincoln Laboratory, May 1963.

61. Roth, S. D., 'Ray Casting as a Method for Solid Modelling', Research Publication GMR-3466, Computer Science Dept., General Motors Research Laboratories, Waren, Mich., Oct. 1980.

62. Sasikumaran, S., Joshi, R. C., Sarje A. K. and Darbari, H., 'A Novel Constructional Approach for Computer Aided Design', International Conference on Computer Graphics, Singapore, 15-16th September, 1988.

63. Sproull, R. H., and Sutherland, I.E., 'A Clipping Divider', Proc. 1968, AFIPSFJCC, Vol. 33, AFIPS Press, Montvale, N. J., pp. 765-775.

64. Stuart Sechrest and Donald P. Greenberg, 'A Visible Polygon Reconstruction Algorithm', Computer Graphics (Proceedings of SIGGARAPH'81) Vol. 15, No. 3, August 1981.

65. Sutherland, I.E., 'SKETCHPAD: a man-machine graphic communication system', MIT Lincoln Lab. Tech. Rep. 296, May 1965.

66. Sutherland, I.E., and Hodgman, G. W., 'Reentrant Polygon Clipping', Comm. ACM, Vol. 17, No. 1, pp. 32-42, January 1974.

67. Sutherland, I.E., Sproull, R.F., and Schumaker, R.A., 'A Characterization of Ten Hidden-Surface Algorithms', Computing Surveys, Vol. 6, pp. 1-55, 1974.

68. Tilove, R. B., 'Exploting spatial and Structural Locality in Geometric Modeling', Tech. Memo No. 38, Production Automation Project, University of Rochester, 1981.

69. Tilove, R. B., 'Set-membership classification: 'A unified approach to Geometric Intersection Problems', IEEE Trans. on Computers, Vol. C-29, pp. 874-883, 1981.

70. Tony Bishop, David Howard and Norman Schofield, 'CAE-Key issues for future Success', Computer-Aided Engineering Journal, pp. 161-175, October 1985.

71. Veenman, P., 'ROMULUS - The Design of a Geometric Modeller', in Geometric Modelling Seminar, W.A. Carter, (Ed.), P-80-GM-01, CAM-I, Inc., Bournemouth, U.K., pp. 127-152, Nov. 1979.

72. Warnock, J. E., 'A hidden Surface Algorithm for Computer Generated Halftone Pictures', Tech. Report No. 4-15, University of Utah, Salt Lake City, Utah, June 1969.

73. Watkins, G. S., A real-time Visible Surface Algorithm', Tech. Report No. UTECH-CSC-70-101, University of Utah, Salt Lake City, Utah, June 1970.

74. Weiss, R. A.,'BE VISION, A package of 7090 FORTRAN Programme to Draw Orthographic Views of Combinations of Plane and Quadratic Surface', JACM, Vol. 13, pp. 194-204, 1966.

75. Wesley, M. A. and Markowsky, G., 'Fleshing Out Projections', IBM Journal of R&D, Vol. 25, pp. 934-954, 1981.

76. Wesley, M. A., et al., A Geometric Modeling System for Automated Mechanical Assembly', IBM Journal of R&D, Vol. 24, pp. 64-74, 1980.

77. Wesley, M. A., 'Construction and use of Geometric Models', in 'Computer Aided Design', Lecture Note in Computer Science, No. 89, Ch. 2, J. Encarnacao (Ed), Springer Verlay, New York. 1980.

78. Woodwark, J. R. 'Compressed Quad Trees', The Computer Journal, Vol. 27, No. 3, pp. 225-229, 1984.

79. Yamaguchi, K., et al., 'Octree related data structures and algorithms', IEEE Computer Graphics and Applications, Vol. 4, pp. 53-59, 1984.

80. Yamaguchi, K., et al., 'Computer-Integrated Manufacturing of Surfaces using Octree Encoding', IEEE Computer Graphics and Applications, Vol. 4, pp. 60-65, 1984.

81. Yamaguchi, F. and Tokieda, T., 'A Unified Algorithm for Boolean Shape Operations, IEEE Comp. Graphics and Aplications, pp. 24-37, June 1984.

82. Yoshio Ohno, 'A Hidden Line Elimination Method for Curved Surfaces', Computer Aided Design, Vol. 15, no. 4, pp. 209-216, July 1983.
