

**CONCEPTS FOR COMPUTER-AIDED PRELIMINARY
DESIGN AND MODELING OF ASSEMBLIES
WITH MOVING PARTS**

By

XIANG HONG

Bachelor of Science in Engineering

Shanghai Jiao Tong University

Shanghai, P. R. China


1986

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 1991

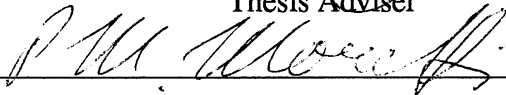
Shesis
1991
H772c


CONCEPTS FOR COMPUTER-AIDED PRELIMINARY
DESIGN AND MODELING OF ASSEMBLIES
WITH MOVING PARTS

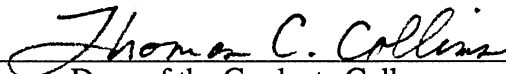
Thesis Approved:



Thesis Adviser







Dean of the Graduate College

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my appreciation to my thesis adviser, Dr. R. D. Delahoussaye for his invaluable assistance, and guidance in the development of this thesis, and encouragement throughout my master studies.

I would also thank my committee members, Dr. P. M. Moretti and Dr. J. D. Spitler, for their suggestions and directions to my thesis.

My heartfelt gratitude is extended to my parents, Mrs. and Mr. Xingxia Hong, for their encouragement during my studies.

Appreciation is extended to Mr. and Mrs. Dalton for their help during my study.

TABLE OF CONTENTS

Chapter	Page
I. THE RESEARCH PROBLEM	1
Introduction	1
Motivation	2
Concepts for Computer-Aided Preliminary Design	2
Objectives	4
II. LITERATURE REVIEW	6
Computer-Aided Preliminary Design	6
Geometric Modeling System	9
Solid Modeling System	13
Decomposition Model	13
Exhaustive Enumeration	13
Octree Representation	20
Cell Decomposition Representation	24
Constructive Solid Geometry Model	28
Boundary Representation Model	35
Summary	45
III. COMPUTER-AIDED PRELIMINARY DESIGN WITH MOVING PARTS	48
Introduction	48
Overview of System Capabilities	49
Interactive Motion Simulation	53
Component Data Structure	59
IV. DEVELOPING A SOLID MODELING SYSTEM	61
Introduction	61
Solid Representation in Solid Modeling System	61
CSG Primitives	62
Boundary Representation Data Structure	62
Using Data Structure to Express CSG Primitives	65
Boundary Evaluation Procedure	65
Geometric Calculation	71
V. CONCLUSIONS AND RECOMMENDATIONS	77
Results	77
Conclusions	79

Chapter	Page
Recommendations	81
Computer-Aided Preliminary Design	81
Solid Modeling System	81
REFERENCES	83

LIST OF TABLES

Table	Page
I. Values of Cube Origin	17
II. Values of Sub-Cube Origin	23
III. Face Table in Polygon Based Data Structure.	36
IV. Face Table in Vertex Based Data Structure	38
V. Vertex Table in Vertex Based Data Structure.	39
VI. Face Table in Edge Based Data Structure	41
VII. Vertex Table in Edge Based Data Structure.	42
VIII. Edge Table in Edge Based Data Structure.	43
IX. Face Table in Winged-Edge Based Data Structure	46
X. Vertex Table in Winged-Edge Based Data Structure.	46
XI. Relation Table in Winged-Edge Based Data Structure.	47
XII. Changing Face Rules	70

LIST OF FIGURES

Figure	Page
1. Camera Shutter System	3
2. Constraint Change	8
3. Wireframe Model Ambiguity	11
4. Wireframe Model without Silhouettes	12
5. Exhaustive Enumeration Representation	14
6. One Object	16
7. Using Exhaustive Enumeration to Model an Object	18
8. Do Boolean Set Operations Based on Exhaustive Enumeration Model	19
9. Model an Object by Using Octree Representation	21
10. A Cube is Divided by Using Octree Representation	22
11. Using Octree Representation to Do Union Operation	25
12. Using Octree Representation to Do Difference Operation	26
13. Using Octree Representation to Do Intersection Operation.	27
14. Define Primitives in CSG Model	29
15. Built a Half Sphere by Doing Difference Operation	30
16. General Intersection Boolean Set Operation	32
17. Regularized Intersection Boolean Set Operation	33
18. Binary Tree for CSG	34
19. Modeling Block Using Polygon Based Data Structure	36
20. Modeling Block Using Vertex Based Data Structure	38
21. Modeling Block Using Edge Based Data Structure	41

Figure	Page
22. Camera Shutter and Actuator	50
23. Computer-Aided Preliminary Design System Configuration	51
24. Slide	52
25. Objects Contact.	54
26. Maximum Collision Distance	55
27. Movement by Using Large Time Step and Small Time Step	57
28. Movement with Large Collision and Small Collision	58
29. Pin, Slot and Outloop.	60
30. Solid Modeling System Configuration	63
31. Using Boundary Representation to Model Block Primitive	66
32. Create New Edges and Vertexes	67
33. Two Blocks Do Union Boolean Set Operation	67
34. Changed Faces by Doing Regularized Union Boolean Set Operation.	69
35. Relation Between Face and Object	70
36. Two Planes	72
37. Calculate Common Intersection Segment	73
38. Two Polygons Do Not Intersect	75
39. Design 1 Initial Position	78
40. Design 1 in Second Stable Position	78
41. Design 2 Initial Position	80
42. Design 2 With Lens Opening During Shutter Actuator Goes Back	80

CHAPTER I

THE RESEARCH PROBLEM

Introduction

In today's world, an engineer designs a wide range of tools and devices to suit the people's purposes. In modern society, these design activities and manufacturing activities are separated. The manufacturing process usually can not start until the design process is complete. This is why design activities become so important.

In order to speed up the design process, computer technologies are applied to design. Today, there are many Computer-Aided Design software packages, like CADAM, CAEDS, etc. These CAD softwares aid engineers in design activities. For example, software supplies an easy and efficient way to design mechanical parts, and do analysis of the mechanical parts. The design activities are so complex, however, that no current CAD software can perform the entire design process, supplanting the need for people.

Even though current CAD software can support the individual mechanical part design, few CAD applications can handle the whole design process, such as the design of mechanical assemblies. In particular, there is no CAD software which can support preliminary design, which is an important part in the design process. This is the reason why we are developing a framework for CAD software for preliminary design.

Because in the real world, design activities involve "real world objects" (solids), solid modeling has become very popular today. Actually, solid modeling plays an important role in computer graphics, computer vision, computer-aided mechanical and civil engineering. It is the heart of Computer-Aided Design and Computer-Aided Manufacturing. The

purpose of the solid modeling is to use a mathematical model, which is widely used in all scientific fields and engineering, to represent engineering activities such as representing individual mechanical part design, mechanical part manufacturing, mechanical assembly and inspection. In this way, the time for the whole production cycle can be rapidly reduced.

In this thesis, we will develop concepts for preliminary design modeling and test some of these concepts by implementing them in software.

Motivation

Most machines with what appear to be complex 3-D motion are actually a collection of planar sub-systems, located in planes with various 3-D orientations. The planar sub-systems are connected by relatively simple 3-D interfaces. For example, in Figure 1 the movement of the camera shutter is in the x-y plane, the movement of the shutter actuator is in the x-z plane. The two movements are synchronized in three dimensions. Design for planar motion of components in an assembly is common, non-trivial and very often difficult, and therefore important. This is why we have chosen to focus on the design of assemblies of planar moving parts.

Concepts for Computer-Aided Preliminary Design

In 1985, French suggested that a typical design process can be divided into three stages, preliminary design stage, embodiment design stage and detail design stage. In preliminary design stage, we are doing an initial plan and approximate dynamic performance analysis and motion analysis. In embodiment design stage, we are choosing components and make them to an assembly. In detail design stage, we are doing engineering computations, accurate dynamic analysis and various drawings. The goal of computer-aided preliminary design is to provide a platform for designers to do better and faster evaluation during the preliminary design stage. From the discussion of our

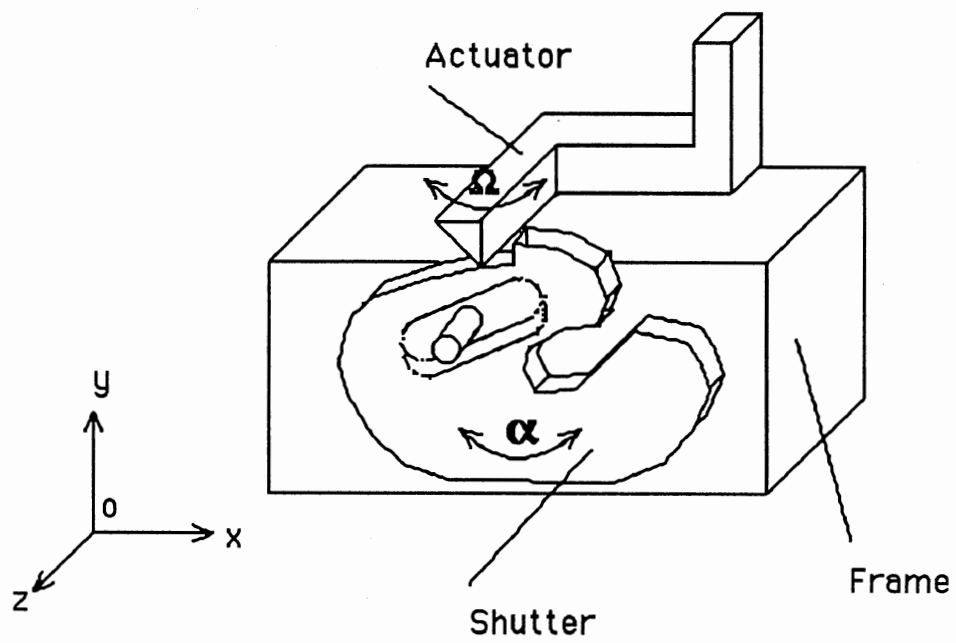


Figure 1. Camera Shutter System

motivation, we will restrict our attention to the motion design of planar assemblies. Our concepts of software for computer-aided preliminary design include the following:

- 1) It should have a user interface that:
 - a) Allows specification of geometry with the same ease as pencil and paper.
 - b) Allows the constraint relationships between components to be defined, such as a pin in a slot.
 - c) Allows rapid editing of the geometry.
 - d) Allows the designers to interact with the design by "pushing" on parts to make them move.
 - e) Allows easy adjustment of model parameters to give fast, but reasonable motion.
- 2) Because many designs rely on large deformations of component geometry, it should handle the motion of compliant parts.
- 3) It should provide some "understanding" of the motion of the assembly using only shape information. Constraint changes should be detected and handled without user intervention.
- 4) It should be able to assemble multiple planar sub-systems into their proper 3-D orientations to model the complex motion generated.
- 5) It should provide a connection to a solid modeler for better visualization of the design and for possible enhanced analysis.

Objectives

The primary objective of this research is to implement in software some of the concepts described in the previous section, and to investigate the validity of these concepts by using the software to examine example designs.

The following will be implemented:

- 1) Develop a minimal user interface which can create components, and connect them

in an assembly.

- 2) Create a simple dynamic model by which we can get the fast, realistic and interactive motion simulation.
- 3) Develop a method for automated detection of constraint changes.
- 4) Develop a "force block" which the designer can use to apply forces to the model interactively.
- 5) Develop a means of quickly changing the value of the model parameters.
- 6) Develop a prototype solid modeler in order to provide better visual understanding of the design and to allow potential refined analysis.

The development of a solid modeler listed in item 6 is actually a second major objective for two reasons. Although commercial solid modelers are available, their source code is not available. In order to do further research in design modeling, a solid modeler with source code is needed. Also, the geometric algorithms developed as part of a solid modeler are useful in other applications. In particular, the polygon intersection algorithms can be used directly in the preliminary design modeler for detection of constraint changes.

CHAPTER II

LITERATURE REVIEW

Computer-Aided Preliminary Design

From ancient times, people understood how to design the mechanical system to function properly. But at that time, it took a long time to get the final system since they did not have simulation methods or experimental methods. Today, computers are used widely not only for scientific computation but also for aiding the various engineering system designs. Computer-aided design can save significant amount of time and money while reducing the complete product design cycle.

In order to apply the CAD technology to the engineering design process, we must know what the engineering design process is. Usually, the engineering design process involves three stages: preliminary design, embodiment design, and detail design. In the preliminary design stage, the goal is to analyze the design problem and propose possible solutions based on engineering, practical, and manufacturing knowledge. Approximate dynamic performance analysis and motion analysis should be done during this stage. In the embodiment design stage, the focus is on choosing subassemblies and components and connecting them together, depending on the preliminary design. Some feedback information should be sent to preliminary design. In the detail design stage, there are various engineering computations, more accurate dynamic analysis, and strength analysis. Individual components and subassemblies are better modified in order to get the greater performance of the total system. Various drawings, not only for the design process, but also for the manufacturing process and the assembly process, are created.

Computer-aided design technology has been applied to the engineering design area for a long time. Today, various CAD softwares have been developed to help the design process. Engineers can use these packages to establish the model, then do analysis and simulation. For example, in 1988, Thatch and Myklebust listed that IMP, DRAM, ADAMS and Dymac, are software packages for mechanism design analysis. Kinsyn, Mecsyn, LINCAGEA, and Recsyn are used for mechanism synthesis.

There are two shortcomings for these current computer packages. The first one is that all these packages require the user to input a large amount of data for problem formulation and other technical information such as linkage information for mechanism design. This impedes usage of these packages.

The second shortcoming involves the packages lack of the ability to handle kinematic constraint changes. It is difficult to simulate the mechanical system because in general, the kinematic constraints of the mechanical system are always changing. This will change the motion equations. For example, in Figure 2 (a), object A has no constraints, it is a free body object. In Figure 2 (b), object A has one contacting point with the wedge. In Figure 2 (c), object A has two contacting points with the wedge. In Figure 2 (d), object A not only contacts two points with wedge, but also has one contacting point with wall C. So we can see that, during the whole process, the constraints of object A have been changed three times. If we analyze the motion of the object A, we should change the motion equations when the constraint conditions have been changed. Now the problem is how to detect the constraint changes and change the motion equations.

Mechanical systems are made of subsystems which connect together through constraints. The changing of constraints will change the whole system. An elastic mechanical joint (linkage with pin clearance) is a typical constraint changes example. In 1971, Dubowsky and Freudenstein represented this constraint changes by using the Impact Pair model in which spring and damping are used between the two objects which have a relative motion. The reason they can use spring and damping to model the constraint

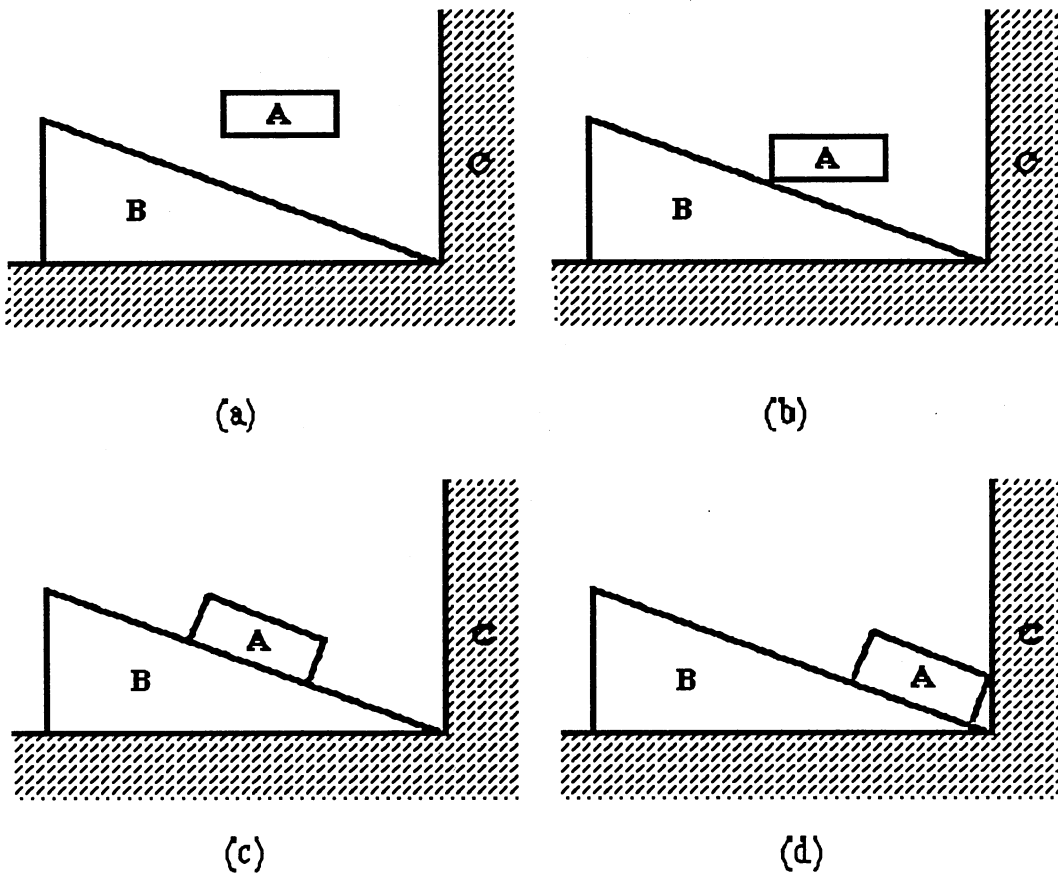


Figure 2. Constraint Change

changes is that Hertzian nonlinearities affect little in the dynamic response of the model.

In 1973, Winfery, Anderson and Gnilka represented the general approach to handle constraint changes. They apply finite element technology to the elastic machinery system with general constraint changes. Even though this method can handle the different constraints, it still needs the user to indicate when and where the constraint change occurs and to choose the appropriate equations.

In 1991, Gilmore and Cipra presented a method which can automatically predict and detect the kinematic constraint change and reformulate the equations of motion. They use the concepts of point to line contact kinematic constraints, force closure, and ray firing together with the rigid body boundary information, with state variables and with reaction forces to characterize the kinematic constraint changes.

Geometric Modeling System

Geometry, a branch of mathematics concerned with the shape and spatial relations of objects, is one of the fundamental ways in which engineering objects are described. In current production industries, the geometric specification, which is two dimension(2-D) projection drawing, is widely used from design to manufacturing, from assembly to inspection. It becomes an "essential language" for today's engineer.

Taking advantage of computers in the 1960's, many computer companies developed computer systems which can replace the basic engineer routine -- drawing. This computer revolution improves productivity and efficiency.

In 1979, Baer, Eastman and Henrion made a survey of geometric modeling system which is described as following. Between 1955 and 1964, interactive computer graphics became available. Almost all CAD systems were based on the 2-D wireframe geometric model. The internal representation of 2-D wireframe is lists of lines and arcs, which can replace the engineer drawing and produce the point-to-point NC (Numerical Control) codes for drilling and punching operations. In 1970, three dimensions (3-D) wireframe systems

appeared, which could represent the segments of 3-D space curves instead of 2-D lines and arcs. Using the computer graphics principles, we can also set orthographic, isometric, and perspective views. However, there are two flaws.

One flaw is that more than one object can be imagined based on the wireframe model. This is called wireframe ambiguity. This implies that wireframe model representation of the object sometimes is not unique, so this representation is incomplete. An example is given in Figure 3.

Another flaw is that 3-D wireframe model can not represent the curved object properly due to missing the "profile line" or "silhouettes" with the object is displayed by using different view point. See Figure 4.

Consequently, the wireframe method can not completely represent the 3-D object. The second method to represent the 3-D object is called polygonal schemes, which were developed in order to make visual effects and support the realism computer graphics. It is used in real-time 3-D animation systems, such as flight training simulation. This method uses two technologies. One is that an object is represented by many planar polygons. Another uses a spatial clipping operation to trim the part or whole polygons which are hidden in order to get a realistic image.

The third method is the sculptured surface method which uses mathematics to define curves and surfaces, developed by Coons, Bezier and Gordon from 1967 to 1974., in order to support the model dealing with a curved surface object (such as a car body, ship hull, aircraft, and so on). Although the sculptured surface theory and technology has developed rapidly and has helped design engineers solve more problems concerning how to establish the sculptured curves and surfaces models in the computer, the theory has been applied very little to solid model development. Until 1980, the B-Splines had been applied in solid modeling to represent "general" halfspace. Another advantage of using the sculpture surface model is that we can get NC codes from it directly.

Currently, the more important and popular method to represent the 3-D objects is

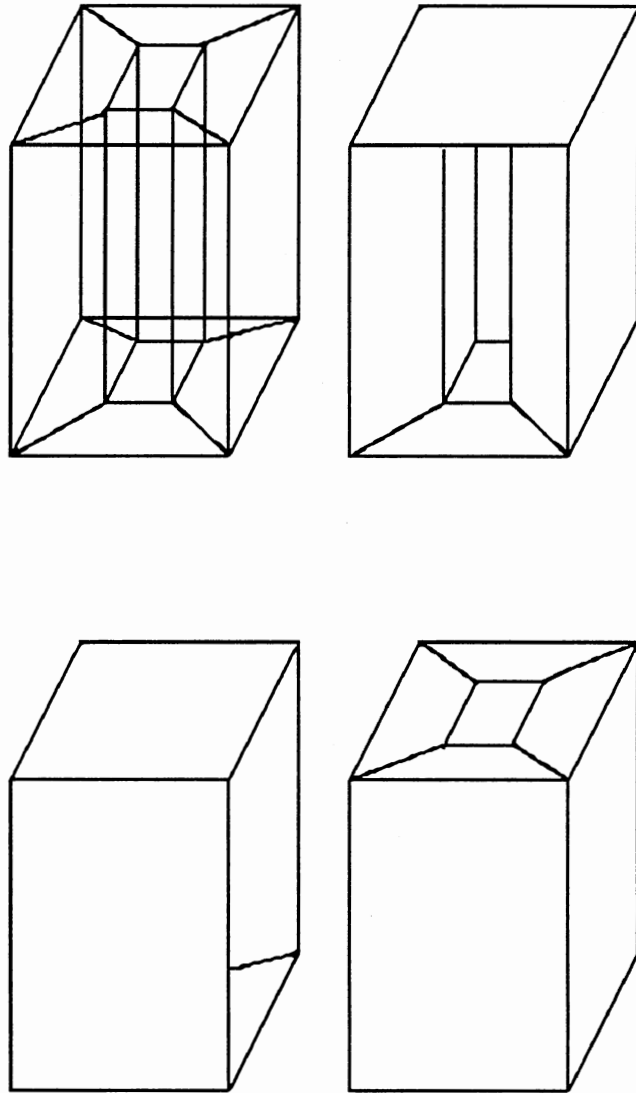


Figure 3. Wireframe Model Ambiguity

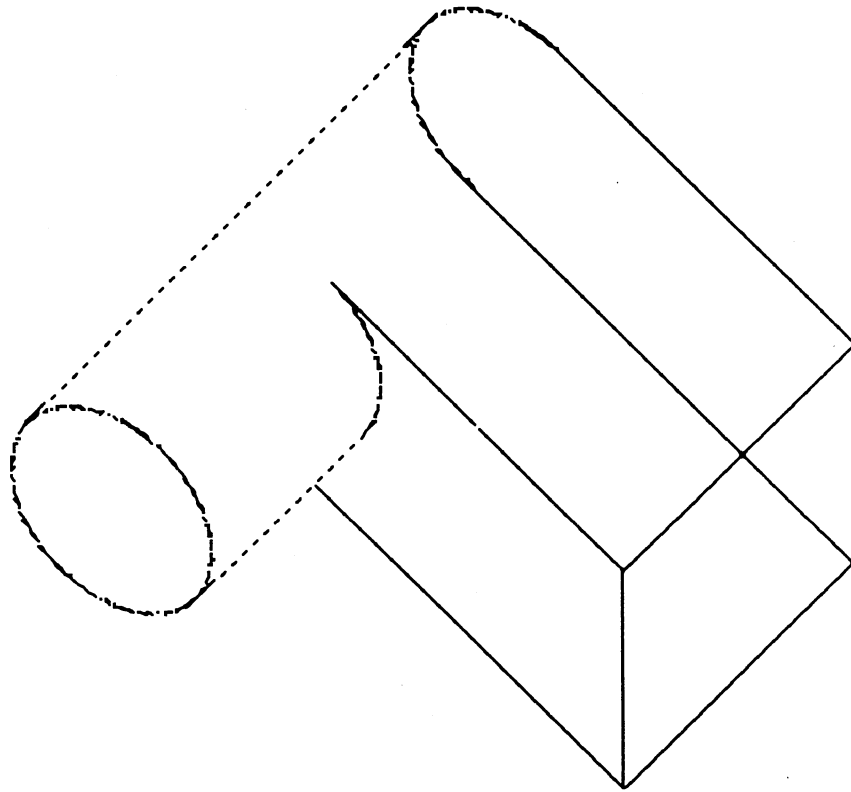


Figure 4. Wireframe Model without Silhouettes

called solid modeling. Using this method, we can create the unambiguous, complete, and unique model in a computer to describe the real world object. There are several solid model schemes. The first one is called Decomposition Model, the second is called Constructive Solid Geometry Model, and the third is called Boundary Representation Model.

Solid Modeling System

"Solid Modeling" means an "informationally complete" representation of the physical object of which some properties, like volume or surface area, should be calculated automatically without human help. In 1970, the solid modeling system became more popular since it could not only offer many new utilities but also link CAD/CAM together. Generally, there are three types of models called Decomposition Model, Constructive Solid Model, and Boundary Model. They are explained in more detail below.

Decomposition Model

Decomposition Model uses the "Finite Element" principle. The physical object is cut into a hundred or a thousand small simple element volumes. By using different rules, these element volumes are combined together to represent a new object. So there is one important technology -- how to divide the physical object into a thousand small element volumes. There are several methods to divide a physical object into many small element volumes.

Exhaustive Enumeration. Since a physical object is made of continuous material, which implies that it is composed of infinite points in three-dimensional space, we can not divide a physical object into points. However, as in converting the continuous system into the discrete system, in 1980 Christessen suggested that we could divide the physical object into a set of smaller element volumes (usually cubes), according to our accuracy. In this way, an object is represented by a list of tiny cubes which are completely or partially contained in the solid. This representation is called exhaustive enumeration (see Figure 5).

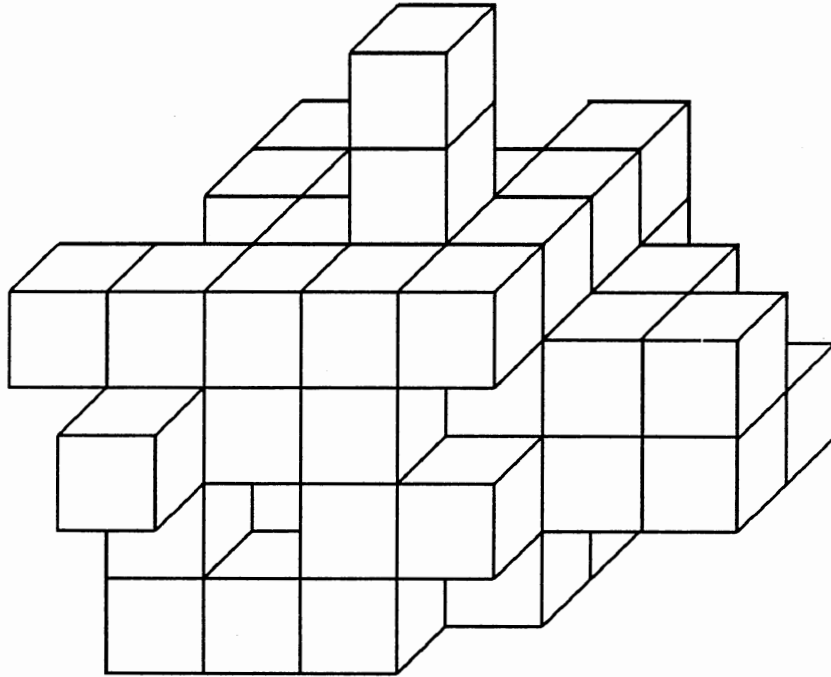


Figure 5. Exhaustive Enumeration Representation

Since the physical object is now represented by a list of small cubes which have no overlapping, uniform dimension and origin, we can define a data structure to describe the object. The data structure is defined as follows:

```
struct EXHAUSTIVE_ENUMERATION
{
    int    id;

    float  x;          /* x value of origin of the cube */
    float  y;          /* y value of origin of the cube */
    float  z;          /* z value of origin of the cube */

    struct EXHAUSTIVE_ENUMERATION *next;
};
```

Because we already know how many element cubes there are and each element cube's origin (x', y', z') in global coordinate, we can use the link data structure to connect all these element cubes together. There is an example in Figure 6, Figure 7 and Table I.

Through this example, we can see that exhaustive enumeration is a good way to represent the physical object. However, how to get these element cubes? We can not get these element cubes by solid description language directly or input these element cubes manually. We should use the other models first, such as Constructive Solid Geometry Model, which has a good user interface to create a model, then we use the conversion algorithm to convert the CSG Model to Exhaustive Enumerations Model.

By using Exhaustive Enumerations representation, we can do Boolean Set Operations to create a new object. There are examples in Figure 8.

Since the exhaustive enumeration representing the physical object is always a valid solid, the Boolean Set Operation should also make the object valid. For example, one element cube disconnecting with other element cubes is not allowed. Every element cube should have at least one face to connect with other element cubes.

One shortcoming of Exhaustive Enumeration Model is that it takes a lot of memory

space based on increased resolution. Another is that it is an approximate representation method.

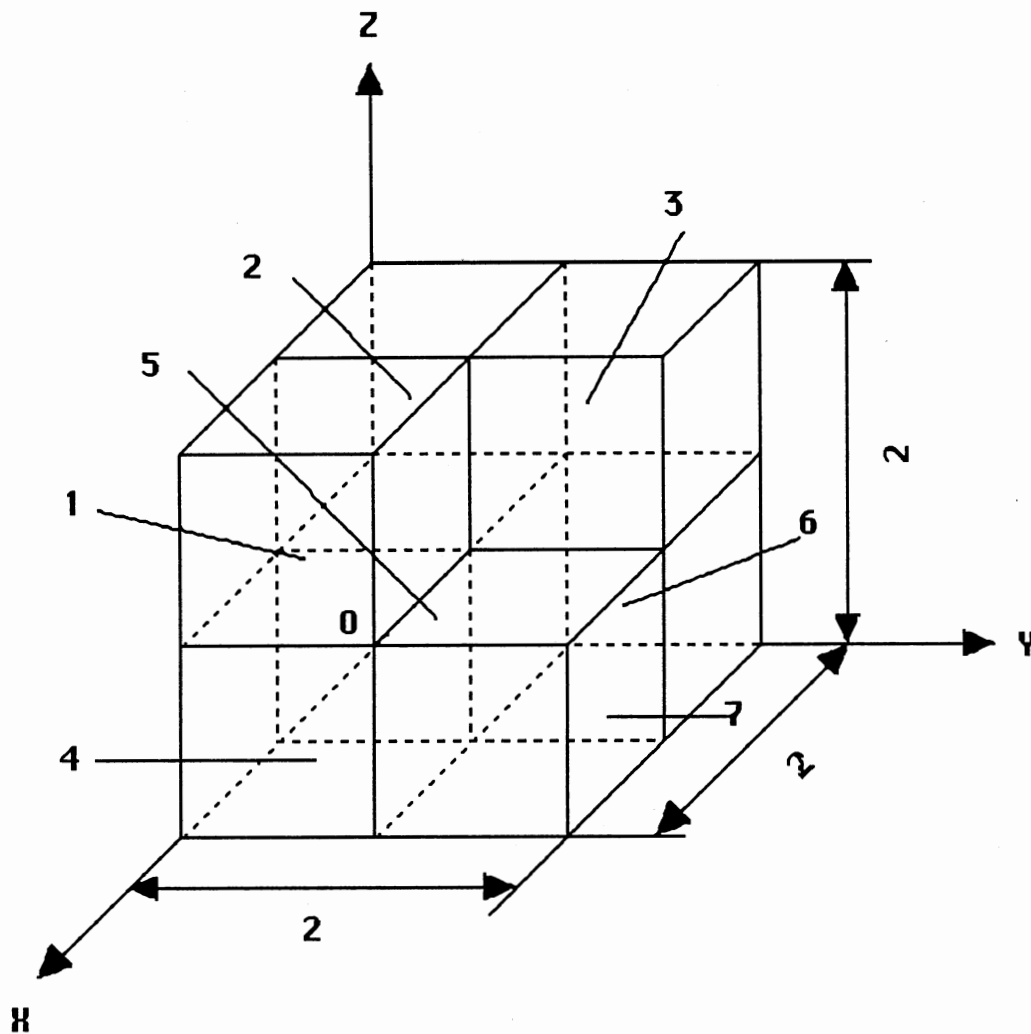


Figure 6. One Object

TABLE I
VALUES OF CUBE ORIGIN

Element Cube Id	ox'	oy'	oz'
1	1.0	0.0	1.0
2	0.0	0.0	1.0
3	0.0	1.0	1.0
4	1.0	0.0	0.0
5	0.0	0.0	0.0
6	0.0	1.0	0.0
7	1.0	1.0	0.0

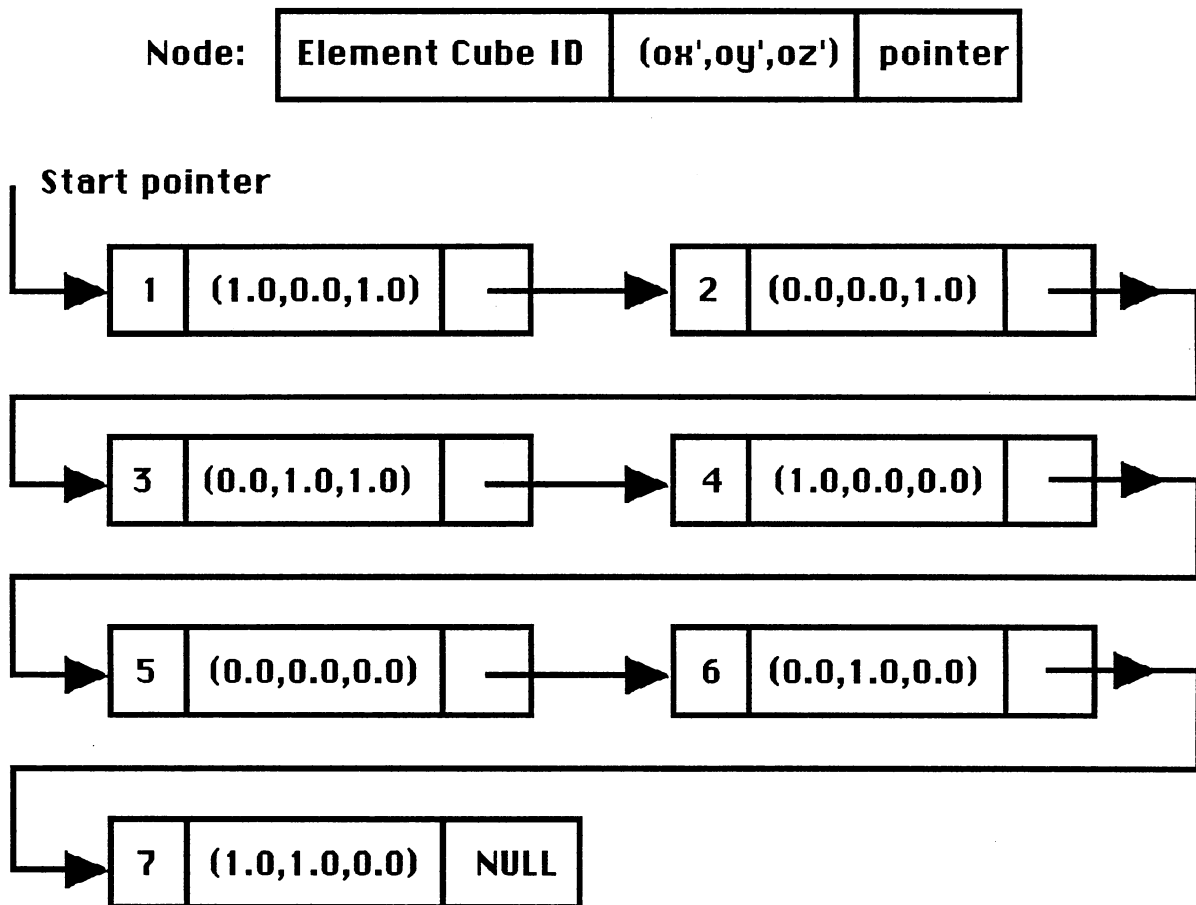


Figure 7. Using Exhaustive Enumeration to Model an Object

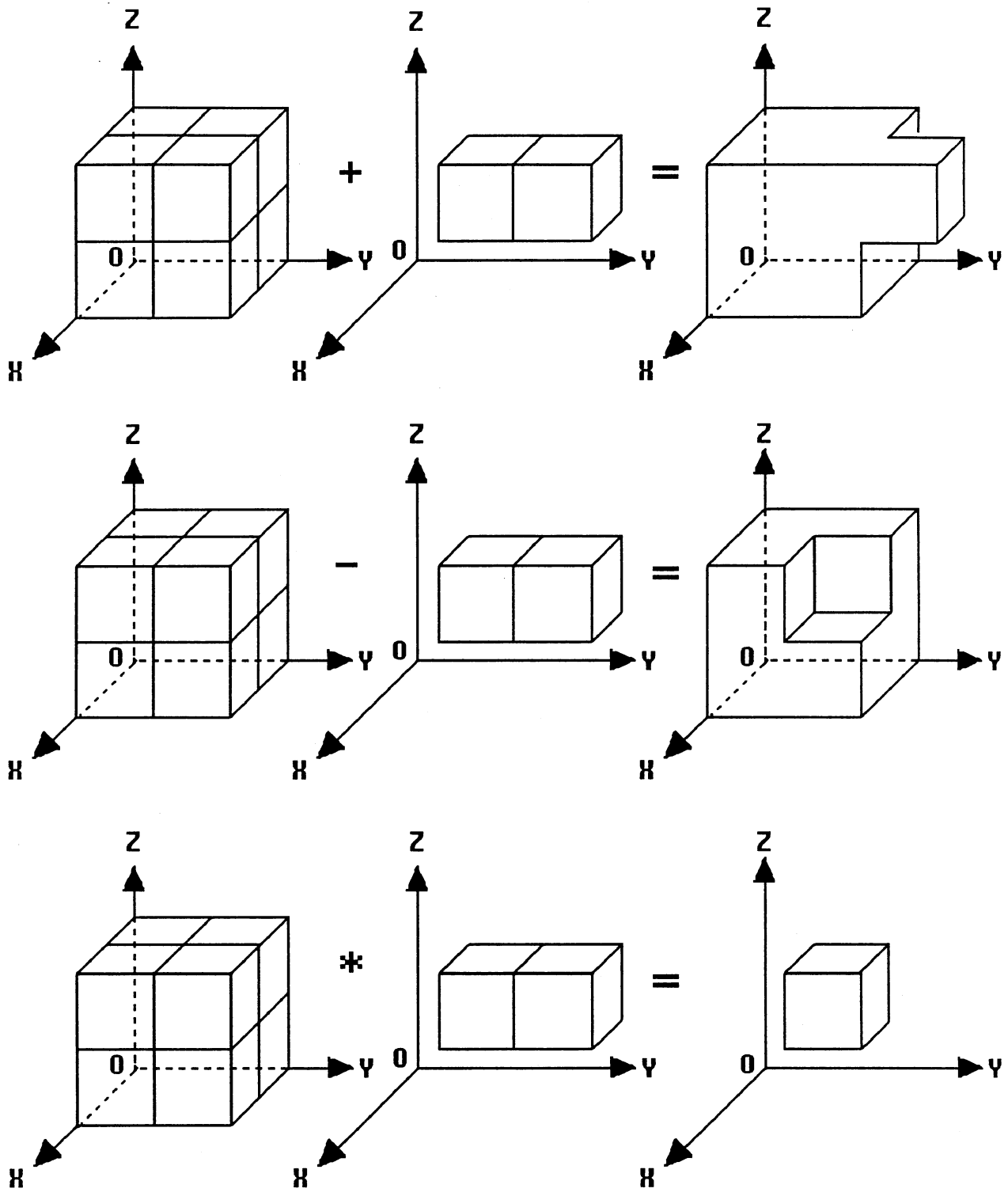


Figure 8. Do Boolean Set Operations Based on Exhaustive Enumeration Model

Octree Representation. Since Exhaustive Enumeration Model takes a lot of memory space, other space subdivision schemes have been developed to overcome this shortcoming. Instead of using a regular element cube as a space subdivision, a more efficient and adaptive space subdivision has been chosen for representing the physical object. In 1980, Jackins and Tanimoto developed an adaptive space subdivision scheme called Octree representation, which is analogous to quadtree representation developed by Samet in 1984 for two dimension objects. The object is represented by the Octree scheme using recursive subdivision of the space which contains a complete or partial object. The object in Figure 9 is represented by Octree representation.

Although the element cubes have different size in Octree Representation, they do not overlap and have their own origins. So we can define a tree data structure to describe the physical object.

The data structure is defined as follows:

```
struct OCTREE
{
    int    id;
    int    flag;        /*flag */
    float  x;          /* x value of origin of the cube */
    float  y;          /* y value of origin of the cube */
    float  z;          /* z value of origin of the cube */
    float  size;       /* size of cube */
    struct OCTREE *octree[8];
};
```

By using this data structure, we can represent the physical object if we know the origin and dimension of each element cube. See example in Figure 10 and Table II.

Although the Octree model is a better scheme for representation, how do we get it? The procedure is as follows.

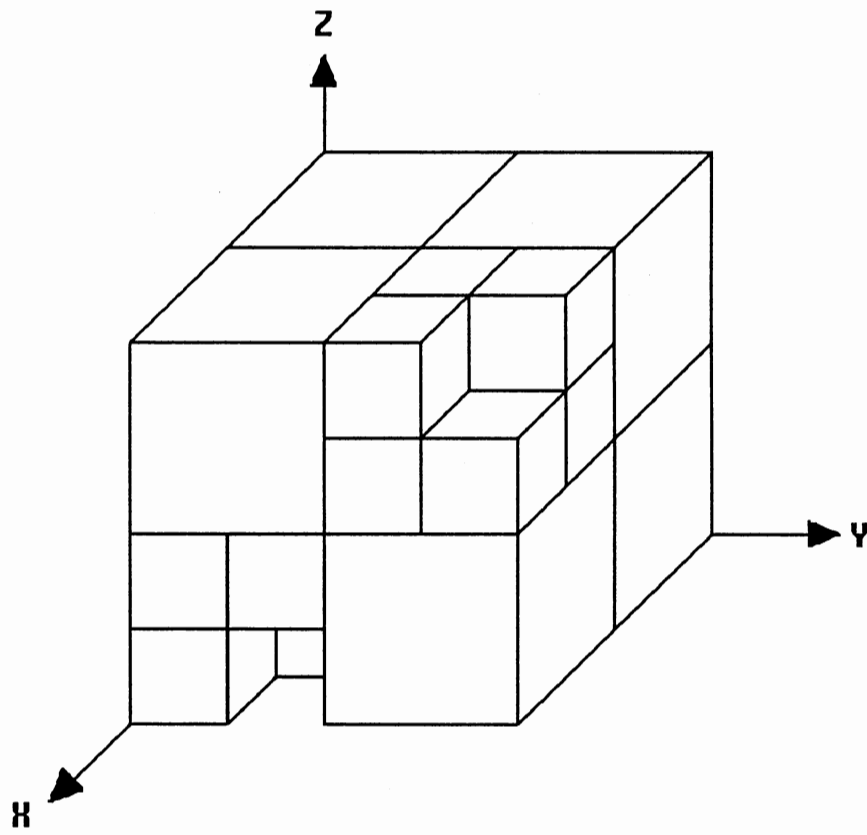


Figure 9. Model an Object by Using Octree Representation

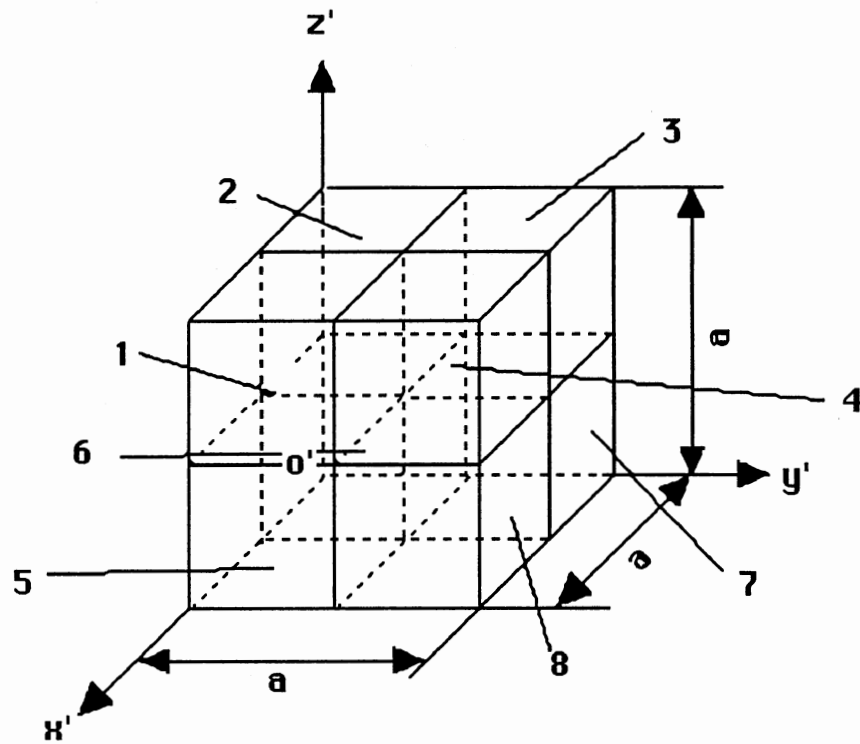


Figure 10. A Cube is Divided by Using Octree Representation

TABLE II
VALUES OF SUB-CUBE ORIGIN

Sub-Cube Id	ox'	oy'	oz'
1	$a/2.0$	0.0	$a/2.0$
2	0.0	0.0	$a/2.0$
3	0.0	$a/2.0$	$a/2.0$
4	$a/2.0$	$a/2.0$	$a/2.0$
5	$a/2.0$	0.0	0.0
6	0.0	0.0	0.0
7	0.0	$a/2.0$	0.0
8	$a/2.0$	$a/2.0$	0.0

First, we should find the maximum cube to contain the object, then we divide this cube into eight octants. In every octant, we test whether the object partially fills it in, completely fills it in or does not fill it in at all. If the object partially fills in an octant, we can use this octant as the original one, and do the same above procedure. We can recursively do this procedure until we achieve the accuracy we require.

The advantage of using Octree representation is that we can calculate some geometric properties, like volume, center of mass and moments of inertia easily. Also using this representation, we can do Boolean Set Operation easily. There are some examples in Figure 11, Figure 12, and Figure 13.

Compared to Exhaustive Enumeration Representation, the Octree representation saves more memory space. However, compared to other representations, such as CSG, Boundary Representation, Octree Representation still occupies more memory space. Another shortcoming is that the Octree Representation is an approximate representation.

Cell Decomposition Representation. Just as there are two basic elements, triangular and rectangular, in two dimension finite element model, in 1978, Elliott suggested that we could define some basic volume elements, such as cube, block, prism, polyhedron etc. in three dimensions and use these elements to represent the physical objects by doing the glue operation which combine the two objects together along their bounding surface.

The advantage of this method is that it can save more memory space, compared with Exhaustive Enumeration and Octree Representation. Sometimes it is an accuracy model to represent the solid. Also we can make use of this representation to do finite element analysis.

However, by using different volume elements, it is difficult to get this model by converting from other models.

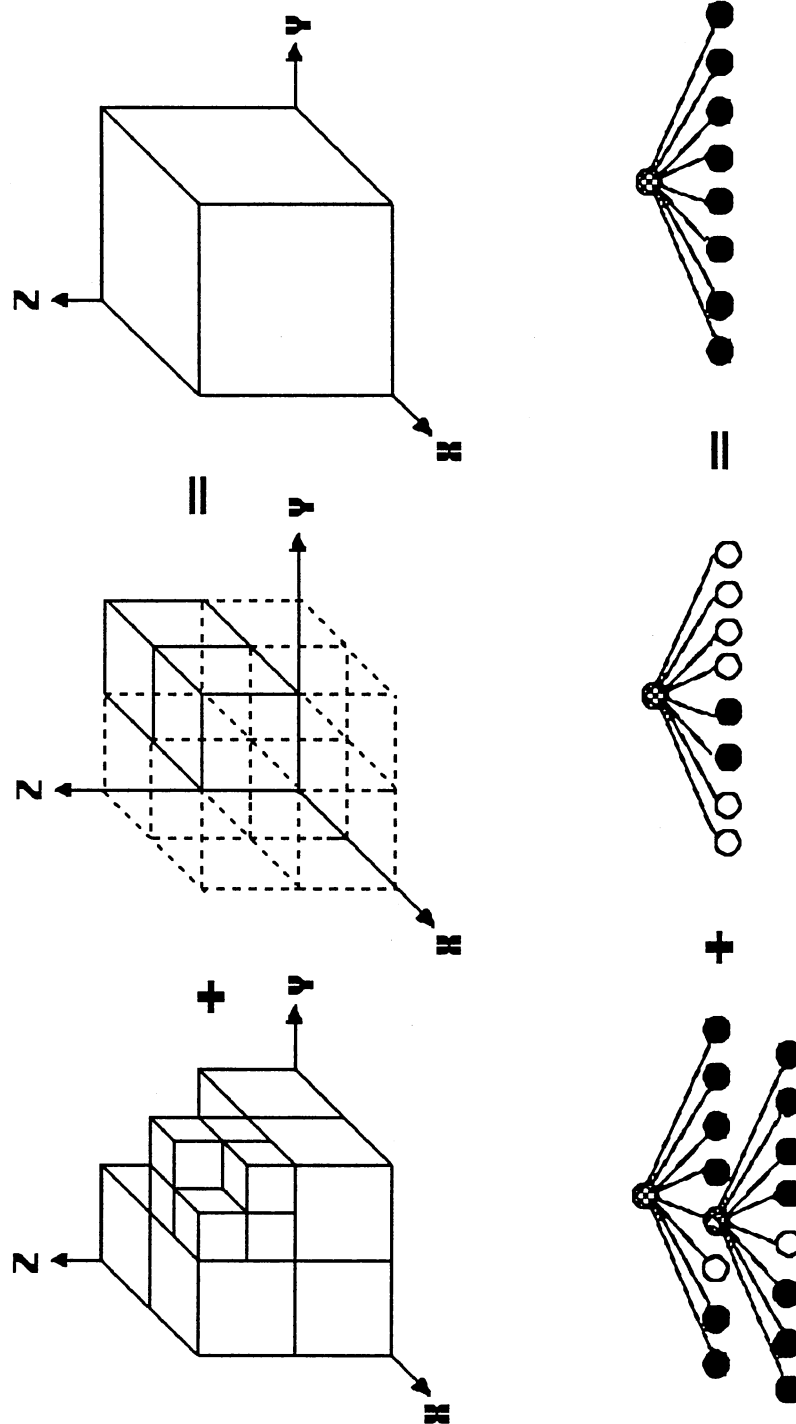


Figure 11. Using Octree Representation to Do Union Operation

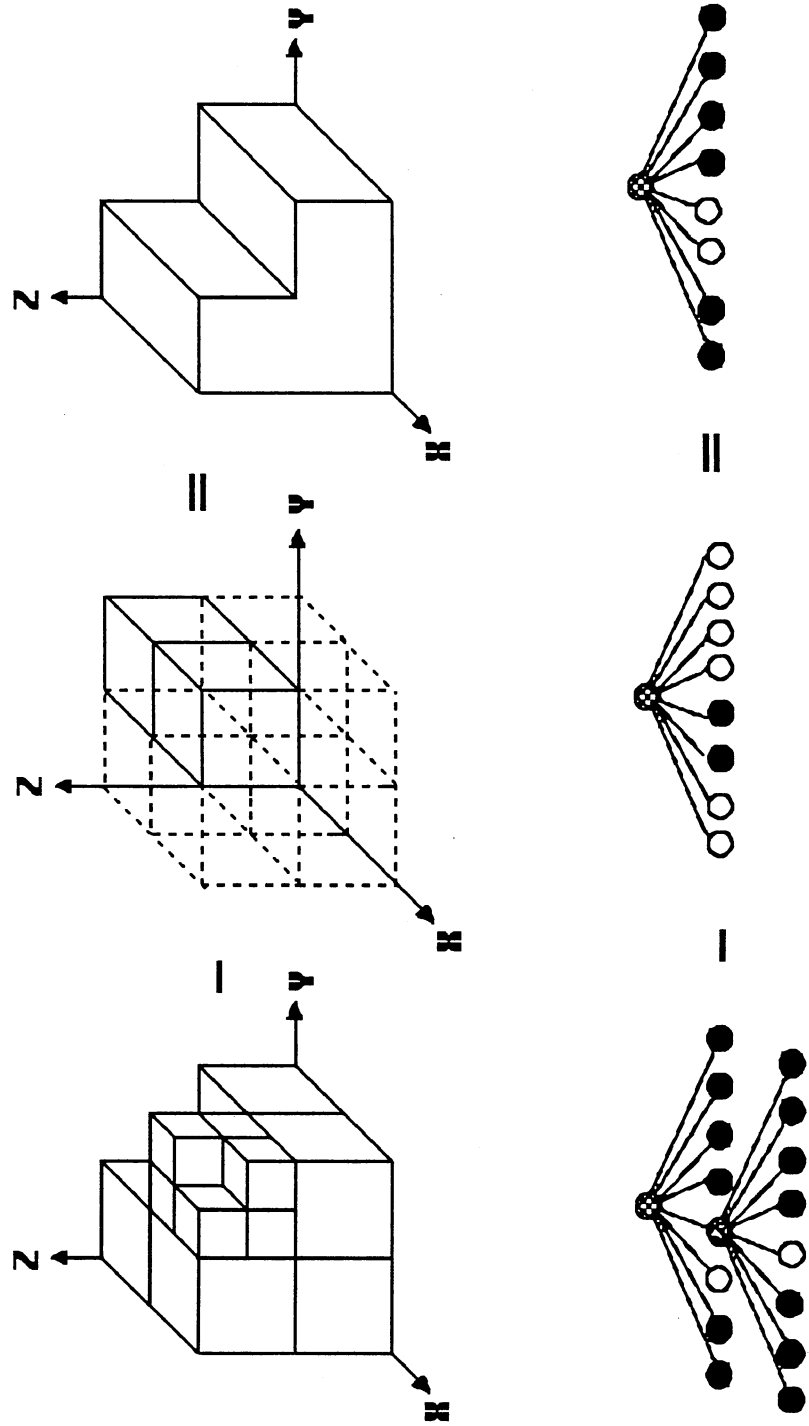


Figure 12. Using Octrees Representation to Do Difference Operation

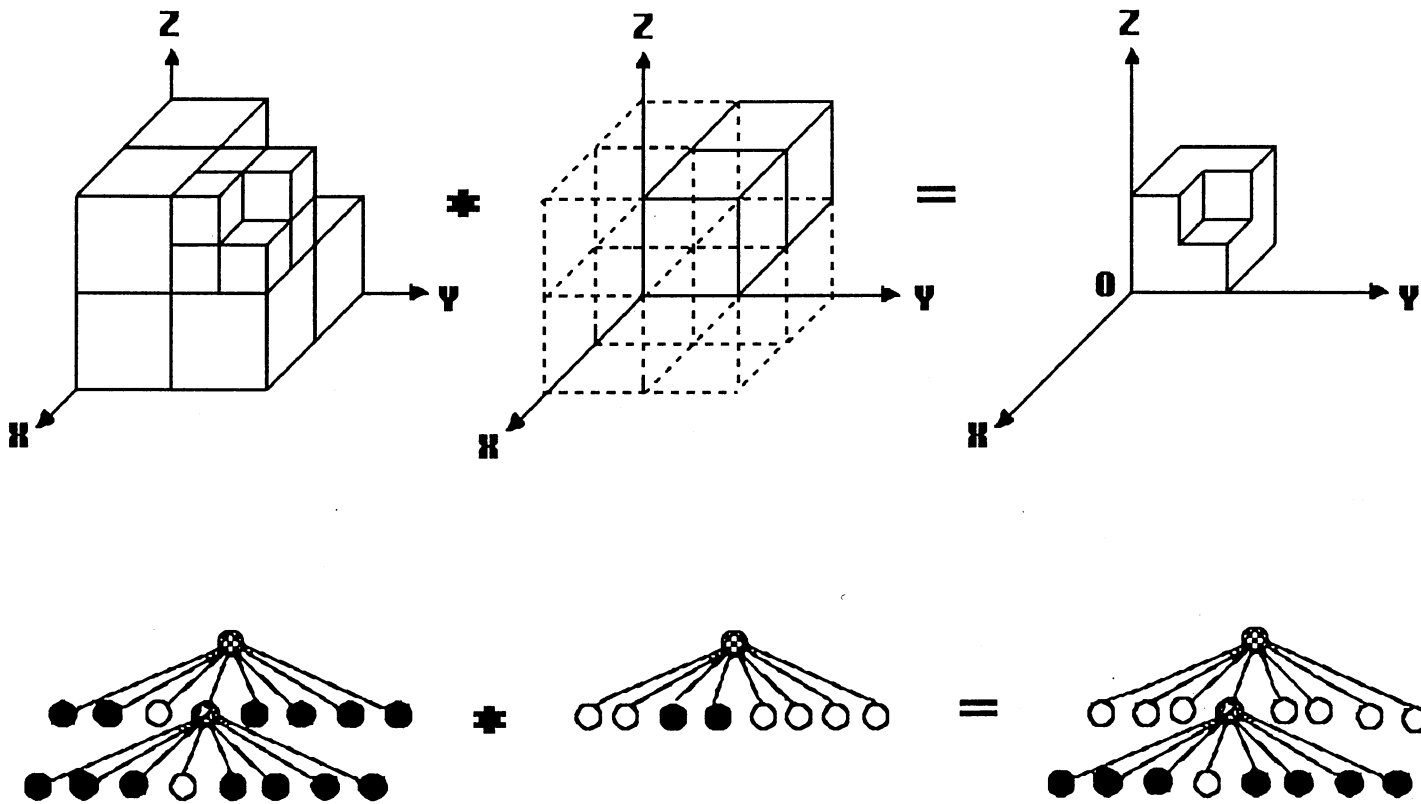


Figure 13. Using Octree Representation to Do Intersection Operation

Constructive Solid Geometry Model

In Decomposition Model, all cells do "glue" operations only to represent the physical object. In 1977, Requicha and Tilove developed Constructive Solid Geometry Model, in which there are more primitives, like block, cylinder, cone, and sphere, etc. Also they defined Regularized Boolean Set Operation to combine these primitives together. The defined primitives are all parameterized instances. For example, we can use three variables, length, width, and height, to define a block. Here coordinate is the block local coordinate. By choosing the different value of parameters, we can get the different size block. Just as we define block, we can also define cylinder, cone, and sphere. (see Figure 14).

The above primitives are called default primitives. Also, we can define primitives which are built by doing a set of Boolean Set Operations on default primitives. For example, a half sphere can be built by a whole sphere minus a block. See Figure 15. So in this way, the user can create primitives by themselves and use these primitives to do Boolean Set Operation.

The most important in CSG is that we can use three basic Boolean Set Operations, Union, Intersection, and Difference. However, though the Boolean Set Operation is used in boolean algebra, it should be different from constructive geometric solid Boolean Set Operation since the set domain is totally different! What we are concerned with in CSG model is the real world object which should be the bounded and closed subset of whole three-dimensional space. Every real world solid should be filled by material homogeneously. This implies that every solid representation should have volume mathematically. Since "dangling" edge and face do not have volume, a complete solid model should not contain them.

In the primitives, we see that all default primitives have the real world object property: that is, the objects are bounded and closed solids. However, using general Boolean Set

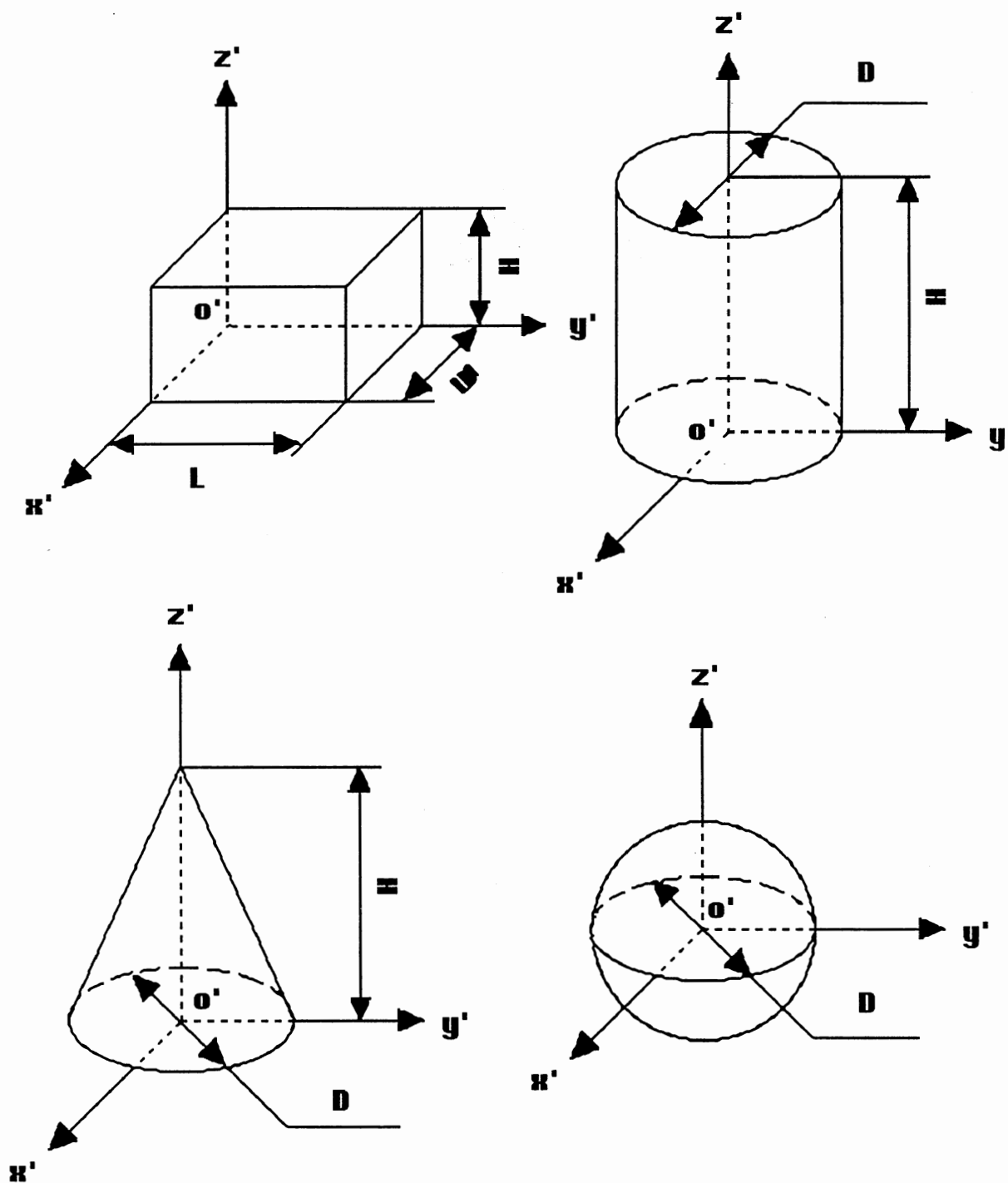


Figure 14. Define Primitives in CSG Model

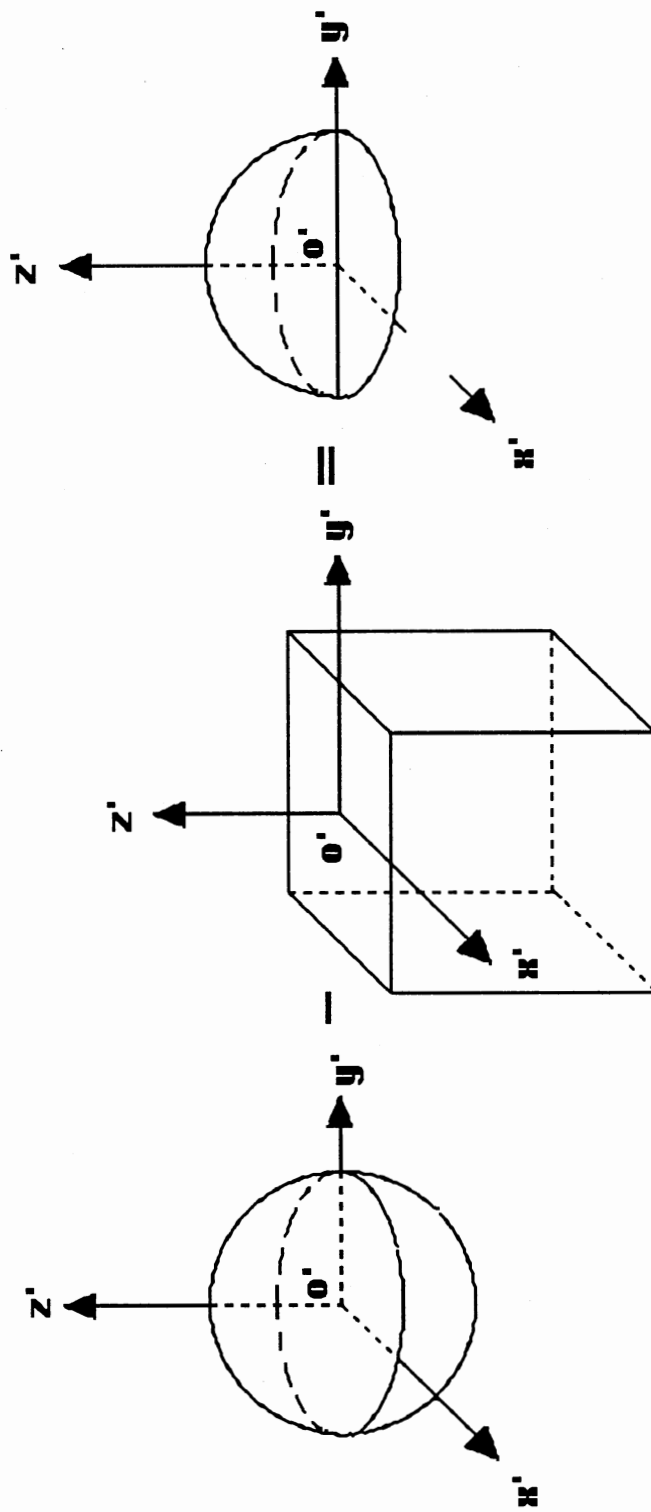


Figure 15. Built a Half Sphere by Doing Difference Operation

Operation, which is the same used in boolean algebra, we can not guarantee that the final solid is a "real" solid since it may contain some dangle edges or faces. See the example in Figure 16 below.

In order to keep the solid, which is built by a series of Boolean Set Operations, to be a "real" solid, in 1977, Requicha and Tilove added more constraints on general Boolean Set Operation and call them "Regularized Boolean Set Operation ". Using Regularized Boolean Set Operation will guarantee that the final object is a real solid. We can define the Regularized Boolean Set Operation, Union, Intersection, and Difference like Union*, Intersection*, and Difference*. So object A and object B doing Regularized Boolean Set Operation can be defined as follows.

$$A \text{ Union* } B = r(A \text{ Union } B)$$

$$A \text{ Intersection* } B = r(A \text{ Intersection } B)$$

$$A \text{ Difference* } B = r(A \text{ Difference } B)$$

By using the same primitives on Figure 16 and doing Regularized Boolean Set Operation, we can get the correct result. See Figure 17.

In Constructive Geometric Solid Model, the user uses basic primitives to build the desired solid by doing a series of Regularized Boolean Set Operations. We can use binary tree data structure to represent this model. The root of the binary tree is the final object. Every node is an intermediate object. The leaf is a primitive. See Figure 18. When we have established a binary tree, we can traverse the whole tree and do the proper Regularized Boolean Set Operations. Finally, we will get a solid.

One important property in CSG Model is that we can use different Regularized Boolean Set Operations to build the same object. So it is difficult to use CSG model directly to determine whether the two CSG models express the same objects.

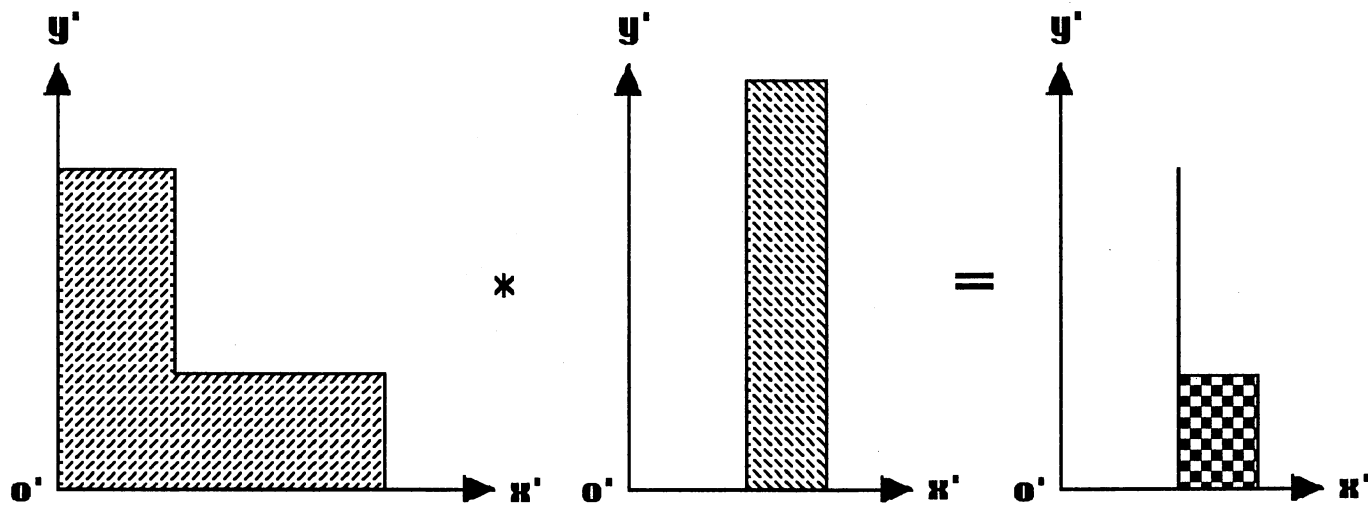


Figure 16. General Intersection Boolean Set Operation

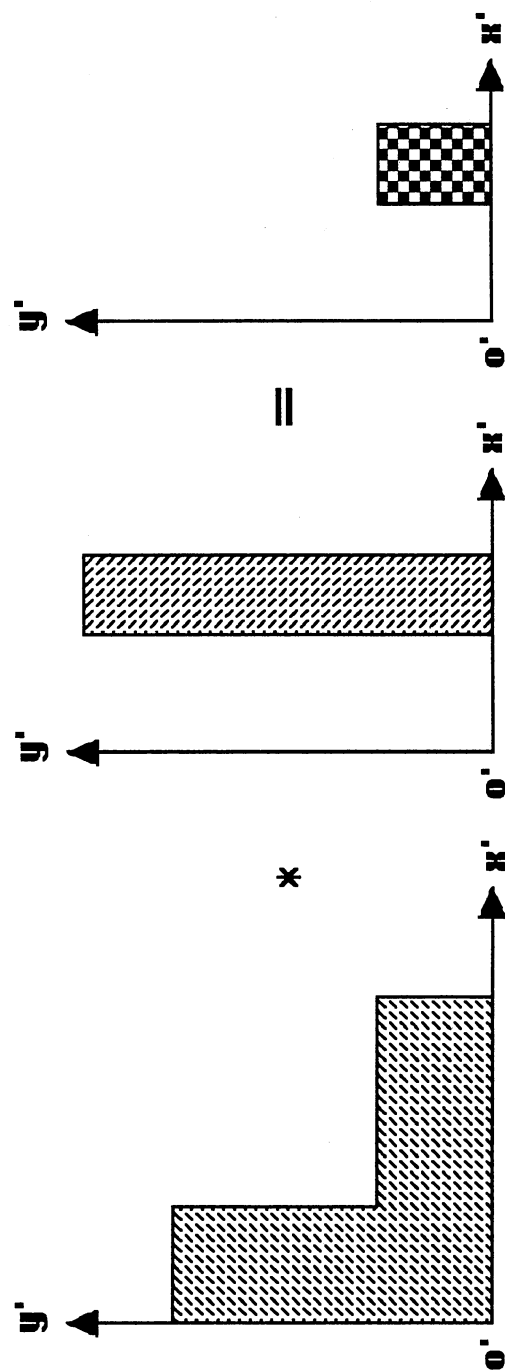


Figure 17. Regularized Intersection Boolean Set Operation

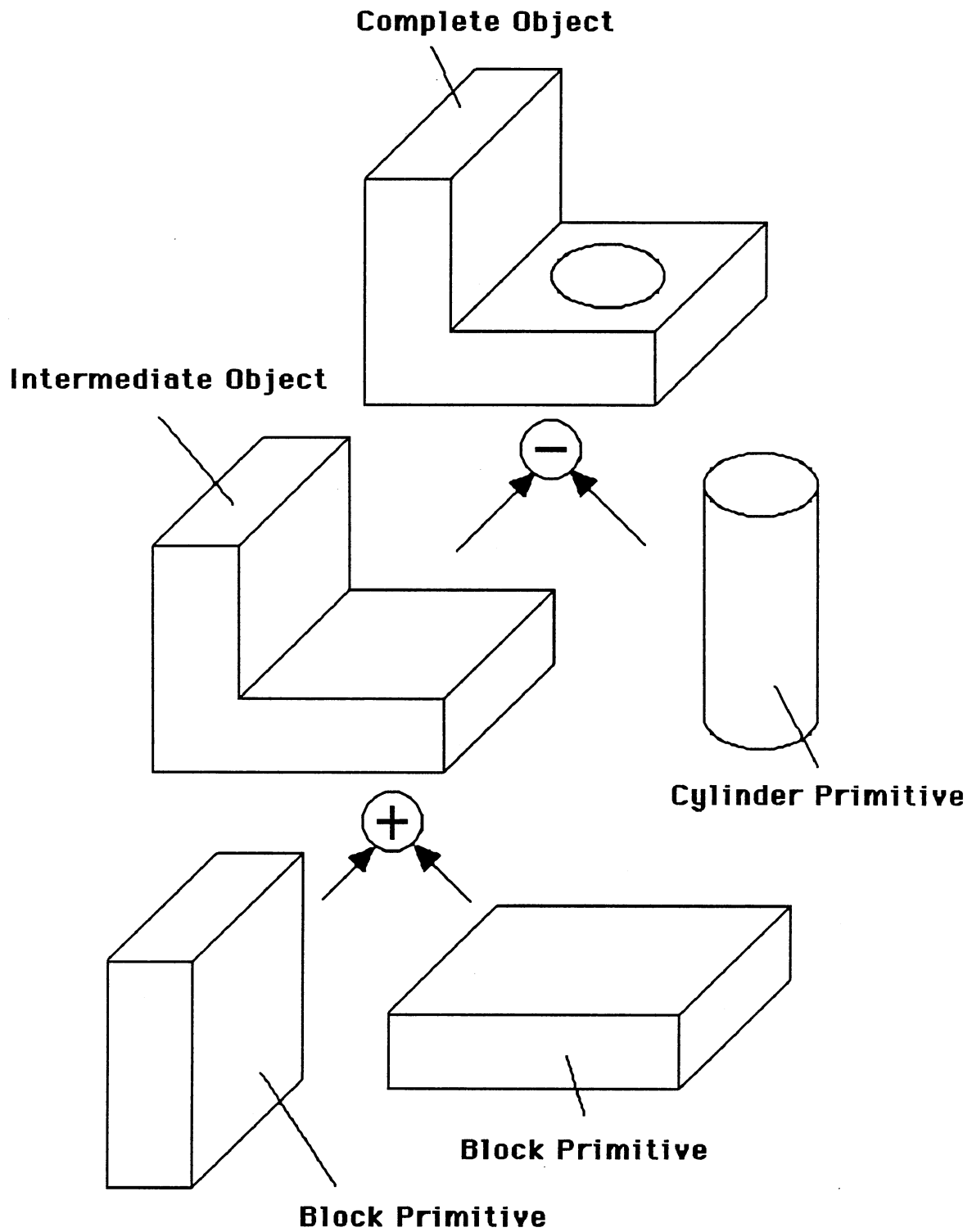


Figure 18. Binary Tree for CSG

Boundary Representation Model

In 1979, Baer, Eastman, and Henrion discussed Boundary Representation.

Boundary Representation Model views the physical objects via their bounding surfaces. It divides the surface into many "single faces", which can be easily described by the mathematics formula, and connects these faces together based on "topology". Since the Boundary Representation Model uses faces to represent the solid, face representation is the heart of Boundary Representation Model. There are several data structures to represent the face as follows.

1) Polygon Based Data Structure

Generally, the faces in Boundary Representation Model are planar faces and polyhedrons which we can use polygon model to represent. The direct data structure to model a polygon is defined as follows:

```
struct POLYGON
{
    int    polygon_id;
    int    vertex_number;
    float  x[N], y[N], z[N];
    struct POLYGON    *next;
};
```

where x, y, z, are pointer coordinates. N is the largest pointer number.

By Using this data structure, we can use Table III to represent the block in Figure 19.

2) Vertex Based Data Structure

In Polygon Based Data Structure, there is a redundancy caused by using the same vertex coordinates to represent the faces. We can introduce the modified data structure which uses the vertex ID instead of using vertex coordinates. This method saves a lot of space since vertex ID is one dimension. The data structure is defined as follows:

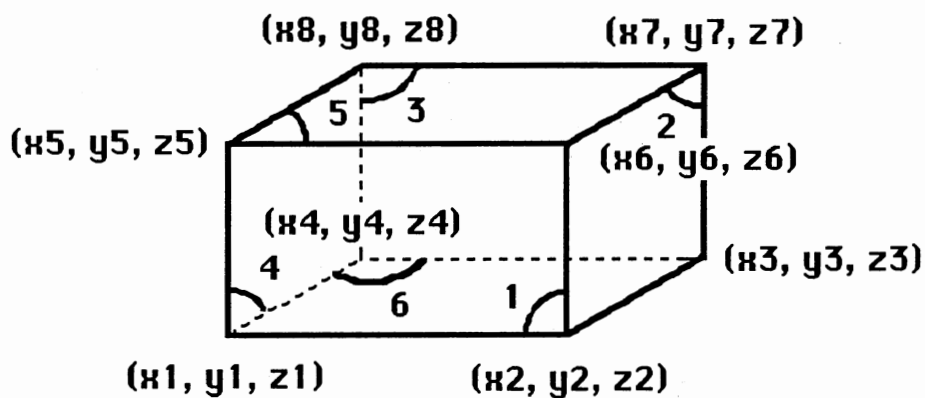


Figure 19. Modeling Block Using Polygon Based Data Structure

TABLE III
FACE TABLE IN POLYGON BASED DATA STRUCTURE

Face Id	Vertex Link
1	$(x1, y1, z1), (x2, y2, z2), (x6, y6, z6), (x5, y5, z5)$
2	$(x2, y2, z2), (x3, y3, z3), (x7, y7, z7), (x6, y6, z6)$
3	$(x7, y7, z7), (x3, y3, z3), (x4, y4, z4), (x8, y8, z8)$
4	$(x5, y5, z5), (x8, y8, z8), (x4, y4, z4), (x1, y1, z1)$
5	$(x5, y5, z5), (x6, y6, z6), (x7, y7, z7), (x8, y8, z8)$
6	$(x1, y1, z1), (x4, y4, z4), (x3, y3, z3), (x2, y2, z2)$

```

struct VERTEX
{
    int          vertex_id;
    float        x, y, z;
    struct VERTEX *next;
};

```

```

struct POLYGON
{
    int    polygon_id;
    int    vertex_number;
    int    vertex_id[N];
    struct POLYGON *next;
};

```

By Using this data structure, we can use Table IV and Table V to represent the block in Figure 20.

3) Edge Based Data Structure

Since the polygon is bounded by many connecting straight line segments, we can also use these segments to represent the polygon. The data structure is defined as follows.

```

struct VERTEX
{
    int    vertex_id;
    float  x, y, z;
    struct VERTEX *next;
};

```

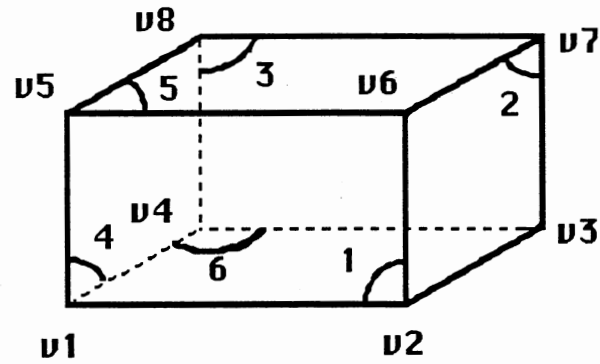


Figure 20. Modeling Block Using Vertex Based Data Structure

TABLE IV
FACE TABLE IN VERTEX BASED
DATA STRUCTURE

Face Id	Vertex Link
1	V1, V5, V6, V2
2	V2, V3, V7, V6
3	V3, V4, V8, V7
4	V5, V8, V4, V1
5	V5, V6, V7, V8
6	V1, V4, V3, V2

TABLE V
VERTEX TABLE IN VERTEX BASED
DATA STRUCTURE

Vertex Id	Coordinate
1	(x1, y1, z1)
2	(x2, y2, z2)
3	(x3, y3, z3)
4	(x4, y4, z4)
5	(x5, y5, z5)
6	(x6, y6, z6)
7	(x7, y7, z7)
8	(x8, y8, z8)


```

struct EDGE
{
    int    edge_id;
    int    start_vertex_id, end_vertex_id;
    struct EDGE    *next;
};

struct POLYGON
{
    int    polygon_id;
    int    edge_number;
    int    edge_id[M];
    struct POLYGON    *next;
};

```

where M is the largest number of line segments.

The Block in Figure 21 can be represented by Table VI, Table VII and Table VIII.

4) Winged-Edge Data Structure

In order to combine two or more objects to create the new object, we should do a lot of calculation to get the correct geometry and topology informations of the object. In this process, we always need to search the connect faces which share one edge, or search the faces which share the same vertex. So we should establish a data structure which can support these calculations rapidly.

In 1974, Baumgart developed Winged-Edge Data Structure and applied it to the computer vision system. Now Winged-Edge Data Structure has been widely used in Solid Model System. By this data structure, we can search the vertexes, edges, and faces more efficiently. Winged-Edge Data Structure is defined as follows.

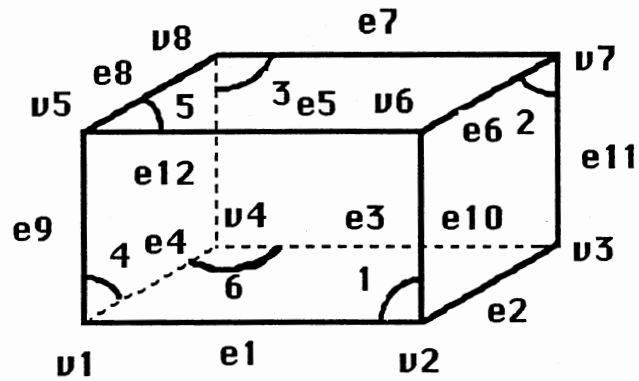


Figure 21 Modeling Block Using Edge Based Data Structure

TABLE VI
FACE TABLE IN EDGE BASED
DATA STRUCTURE

Face Id	Coordinate
1	e1, e9, e5, e10
2	e2, e10, e6, e11
3	e3, e12, e7, e11
4	e4, e9, e8, e12
5	e5, e6, e7, e8
6	e1, e4, e3, e2

TABLE VII
VERTEX TABLE IN EDGE BASED
DATA STRUCTURE

Vertex Id	Coordinate
1	(x1, y1, z1)
2	(x2, y2, z2)
3	(x3, y3, z3)
4	(x4, y4, z4)
5	(x5, y5, z5)
6	(x6, y6, z6)
7	(x7, y7, z7)
8	(x8, y8, z8)

TABLE VIII
EDGE TABLE IN EDGE BASED DATA STRUCTURE

Edge Id	Start Vertex	End Vertex
1	v1	v2
2	v2	v3
3	v3	v4
4	v4	v1
5	v5	v6
6	v6	v7
7	v7	v8
8	v8	v5
9	v1	v5
10	v2	v6
11	v3	v7
112	v4	v8

```
struct VERTEX
{
    int    vertex_id;
    float  x, y, z;
    struct VERTEX    *next;
};

struct EDGE_RELATION
{
    int    face_id;
    int    previous_edge_id, next_edge_id;
    struct EDGE_RELATION    *next;
};

struct RELATION
{
    int    edge_id;
    int    start_vertex_id, end_vertex_id;
    struct EDGE_RELATION    fcw, fccw;
    struct RELATION    *next;
};

struct POLYGON
{
    int    polygon_id;
    int    first_edge_id;
    struct POLYGON    *next;
};
```

Using this data structure, we can quickly find two faces which share the same edge, and the vertex which is met by three edges. Also we can get the edge of face quickly by

traversing a RELATION table.

We can use Table IX, Table X, and Table XI to represent the block in Figure 21.

Boundary Representation Model is often used as the internal representation of the Solid Modeling System. However, it is very difficult to get it directly from input. Usually, we can get Boundary Representation Model by converting other modeling system like CSG.

Summary

Above we have discussed computer-aided preliminary design. There are several methods to handle the constraint change. Some methods can also predict the constraint change automatically.

Also we have discussed three methods to represent the physical object. Each method has its own advantages and disadvantages. Sometimes, we need more than one kind of representation in the system in order to do more analysis. This requires us to convert one kind of representation to another quickly. Actually, Solid Modeling System usually uses multi-representation to get good user interface and keep more information for applications.

TABLE IX
FACE TABLE IN WINGED-EDGE BASED
DATA STRUCTURE

Face Id	Start Edge
1	e1
2	e2
3	e3
4	e4
5	e5
6	e1

TABLE X
VERTEX TABLE IN WINGED-EDGE BASED
DATA STRUCTURE

Vertex Id	Coordinate
1	(x1, y1, z1)
2	(x2, y2, z2)
3	(x3, y3, z3)
4	(x4, y4, z4)
5	(x5, y5, z5)
6	(x6, y6, z6)
7	(x7, y7, z7)
8	(x8, y8, z8)

TABLE XI
RELATION TABLE IN WINGED-EDGE BASED DATA STRUCTURE

Edge	Vstart	Vend	Fcw	Ncw	Pcw	Fccw	Neew	Pccw
e1	v1	v2	f6	e2	e4	f1	e9	e10
e2	v2	v3	f6	e3	e1	f2	e10	e11
e3	v3	v4	f6	e4	e2	f3	e11	e12
e4	v4	v1	f6	e1	e3	f4	e12	e9
e5	v5	v6	f1	e10	e9	f5	e8	e6
e6	v6	v7	f12	e11	e10	f5	e5	e7
e7	v7	v8	f3	e12	e11	f5	e6	e8
e8	v8	v5	f4	e9	e12	f5	e7	e5
e9	v1	v5	f1	e5	e1	f4	e4	e8
e10	v2	v6	f2	e6	e2	f1	e1	e5
e11	v3	v7	f3	e7	e3	f2	e2	e6
e12	v4	v8	f4	e8	e4	f3	e3	e7

CHAPTER III
COMPUTER-AIDED PRELIMINARY
DESIGN WITH MOVING PARTS

Introduction

Applying computer technology to Mechanical Design is no longer a new approach. However, there are only specific computer programs for aiding specific mechanical design problems. Also there is no computer-aided preliminary design package. Developing a general approach for aiding mechanical design is still a developing area.

Since we want to use computers for aiding mechanical design, we should analyze the design process first. The typical mechanical design process involves three stages described as preliminary design, Embodiment design and detail design.

- 1) Preliminary design is doing an initial plan, performance estimation, rough structural analysis, and principal dimension.
- 2) Embodiment design is doing structural analysis, approximate motion analysis, and simulation.
- 3) Detail design involves various engineering computations, dynamic analysis, and various drawings.

Of the above definitions, we can deduce that the preliminary design is the more important stage since it is the start of the design process. If we can do better analysis and evaluation in the preliminary stage, time can be saved since we must redesign the system if we find that we can not meet some requirements in the detail stage. This is the reason we pay close attention to every aspect and do more simulation during this stage.

Another characteristic for mechanical design is that a large portion of complex mechanical design involves assemblies of planar moving parts. We usually design the planar sub-assemblies, then assemble these planer sub-assemblies together. So it is important to develop the general approach to design the planer sub-assemblies. Figure 22 is an example of designing the camera shutter and actuator. The shutter rotates in the x-y plane. The actuator rotates on the x-z plane. When the actuator rotates on the x-z plane, it pushes the shutter to move on the x-y plane. Therefore, the complex 3-D motion can be synthesized by combining planar sub-systems in various 3-D orientations and providing simple connection between sub-systems.

Overview of System Capabilities

According to our objectives, we have developed a preliminary design model with following capabilities. We have developed the simple dynamic model which detects the constraint changes automatically. We have provided a minimal user interface and the capability to replay the model motion in 3-D. The computer-aided preliminary design system configuration is shown in Figure 23.

In order to do motion simulation, we have implemented a simple dynamic model which can handle the planar motions. Newton's second law has been used to generate the equations of motions.

When the object moves, the constraints will change. We can automatically detect the constraint changes by using the polygon collision technology. Using this technology, the user will not be interrupted during the simulation. Also, the reaction forces have been calculated in order to maintain the constraints.

A minimal user interface has been implemented. We can use this interface to define the components with the geometric features, such as the slot and the pin, and combine these components together to make an assembly. In order to interact with the model, we developed the "force block" which is used by the user to apply an interactive force on the

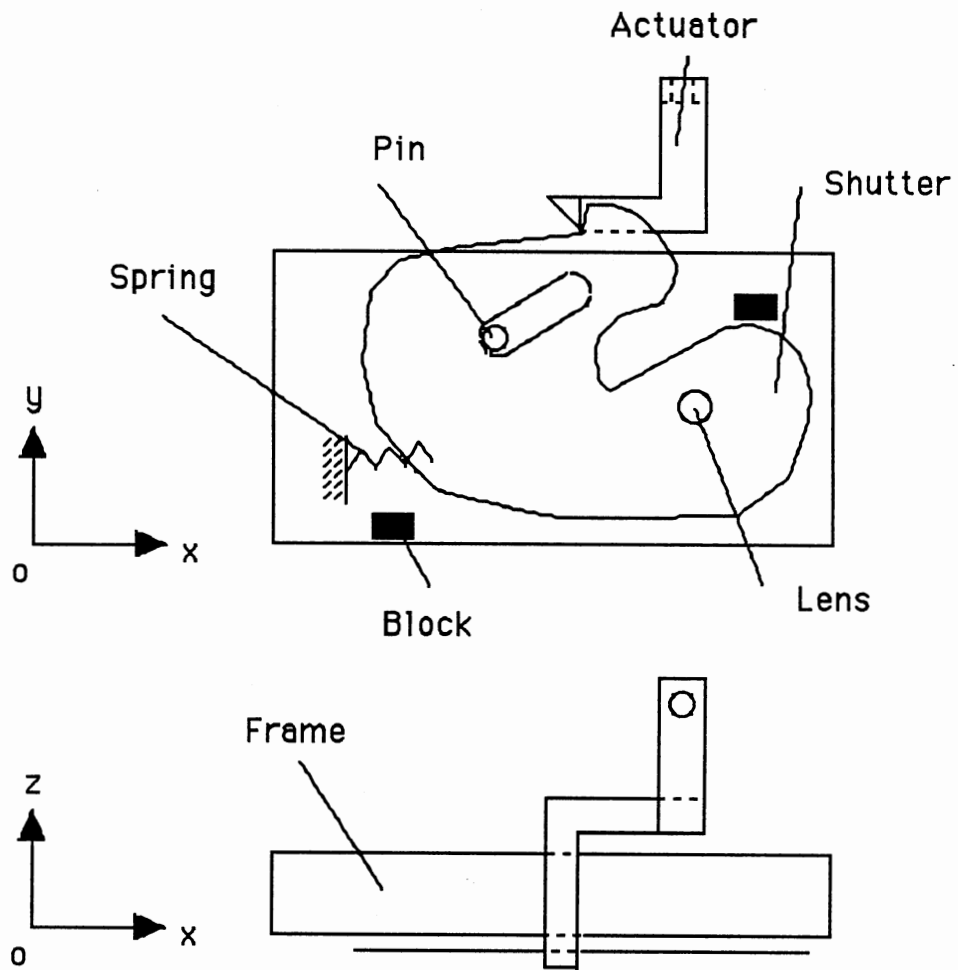


Figure 22. Camera Shutter and Actuator

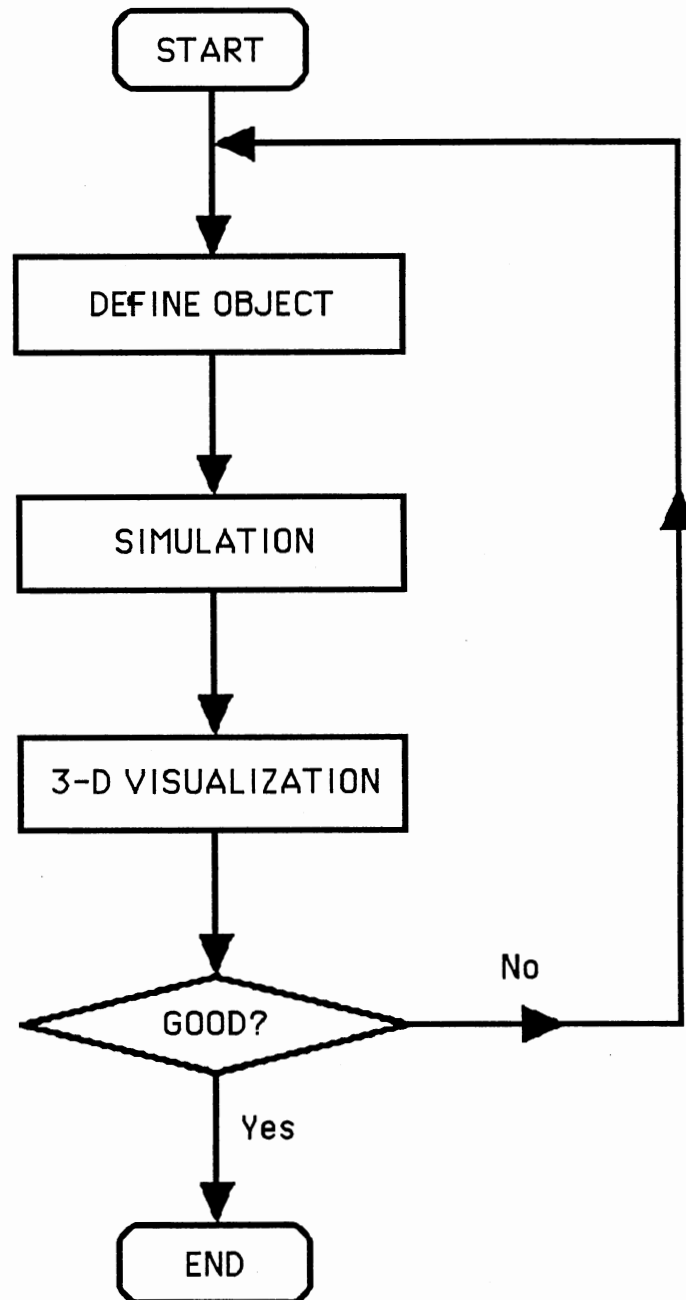
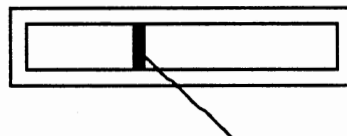


Figure 23. Computer-Aided Preliminary Design System Configuration

moving component. Also we developed the "slide" (see Figure 24) to control the parameters of the model, such as spring stiffness, time step, etc. in order to speed up the motion.

We have implemented the 3-D replay by using the motion data in 2-D to get better visual understanding of the design and keep geometric data of the model to further analysis.

Variable Name
Variable Value



Slide to Change the Variable Value

Figure 24. Slide

Interactive Motion Simulation

The important part of the computer-aided preliminary design is how to detect the constraint change, how to change the motion equations and how to make the mechanical part move reasonably. Since we are only concerned with two dimensional object now, we can apply Newton's second law to a free body object.

$$\begin{cases} \Sigma F = m * a \\ \Sigma M = J * \varepsilon \end{cases} \quad (3.1)$$

When the body is free, it does not have any constraints. The total force is equal to the external force. When the object contacts another object, the total force of the object should be equal to the reaction force plus external force. See Figure 25.

Now we can use the boundary information of the objects to detect the two polygons of the objects. If two polygons collide, then the reaction force should be created. We first calculate the maximum collision distance. See Figure 26.

Since

$$\begin{cases} F_1 = k_1 * x_1 * w \\ F_2 = k_2 * x_2 * w \end{cases} \quad (3.2)$$

Where k_1 and k_2 are stiffness of object1 and object2. x_1 and x_2 are deformation of object 1 and object 2. w is the width of the collision.

From (3.2), we get the following,

$$\begin{cases} x_1 = \frac{F_1}{k_1 * w} \\ x_2 = \frac{F_2}{k_2 * w} \end{cases} \quad (3.3)$$

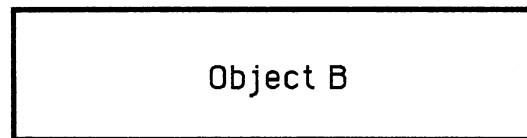
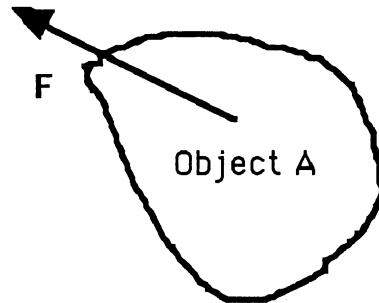
So:

$$x = x_1 + x_2 = \frac{F_1}{k_1 * w} + \frac{F_2}{k_2 * w}$$

Since

$$F_1 = F_2$$

External Force



External Force

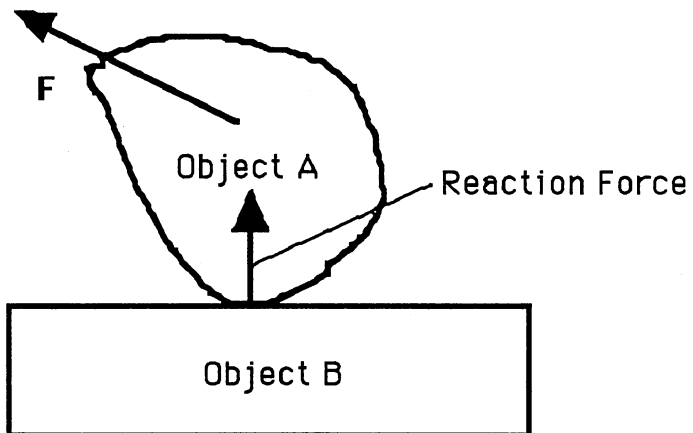
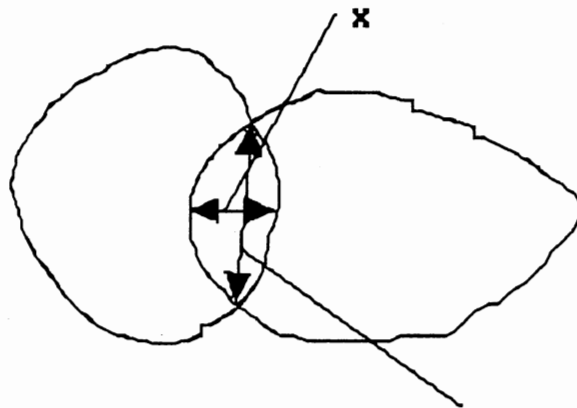


Figure 25. Objects Contact

Maximum Collision Distance of Two Objects



Width of Collision of Two Objects

Figure 26. Maximum Collision Distance

then

$$x = \frac{F_1}{k_1 * w} + \frac{F_2}{k_2 * w} = \frac{k_1 + k_2}{k_1 * k_2 * w} * F_1$$

$$F_1 = F_2 = \frac{k_1 * k_2}{k_1 + k_2} * w * x \quad (3.4)$$

So by using (3.4), we can calculate the reaction forces.

By knowing all the forces at any time, we can calculate the acceleration and the angular acceleration at any time. Using the continuous time formulas below:

$$\begin{cases} v = v_0 + a * t \\ s = s_0 + v_0 * t + \frac{1}{2} * a * t^2 \end{cases}$$

$$\begin{cases} \omega = \omega_0 + \epsilon * t \\ \alpha = \alpha_0 + \omega_0 * t + \frac{1}{2} * \epsilon * t^2 \end{cases}$$

We can derive the discrete time formulas as follows:

$$\begin{cases} v_{t+h} = v_t + a_t * h \\ s_{t+h} = s_t + v_t * h + \frac{1}{2} * a_t * h^2 \end{cases} \quad (3.5)$$

$$\begin{cases} \omega_{t+h} = \omega_t + \epsilon_t * h \\ \alpha_{t+h} = \alpha_t + \omega_t * h + \frac{1}{2} * \epsilon_t * h^2 \end{cases} \quad (3.6)$$

By using (3.5) and (3.6), we see that if we know the previous step a_t , v_t , s_t , ϵ_t , ω_t and α_t , we can calculate the next time step v_{t+h} , s_{t+h} , ω_{t+h} and α_{t+h} .

Since we use discrete time formulas to calculate the step v_t , s_t , ω_t and α_t , we should reduce the time step h in some situation in order to get reasonable motion. For example, in Figure 27, we can see if the time step h is large, the moving object will go out of slot A. This is not reasonable. So we reduce the time step h until the pin does not totally go out of the slot. In Figure 28, there is another situation in which we should reduce the time step h . We see in Figure 28 (a), the collision is too large, so we reduce the time step h to get the smaller collision. Actually, when we calculate the s_t , we make a backup step if necessary. This means that we first try a large time step h , if the collision is too large, abort this s_t . Reduce the time step h , calculate the s_t again until we get the smaller collision.

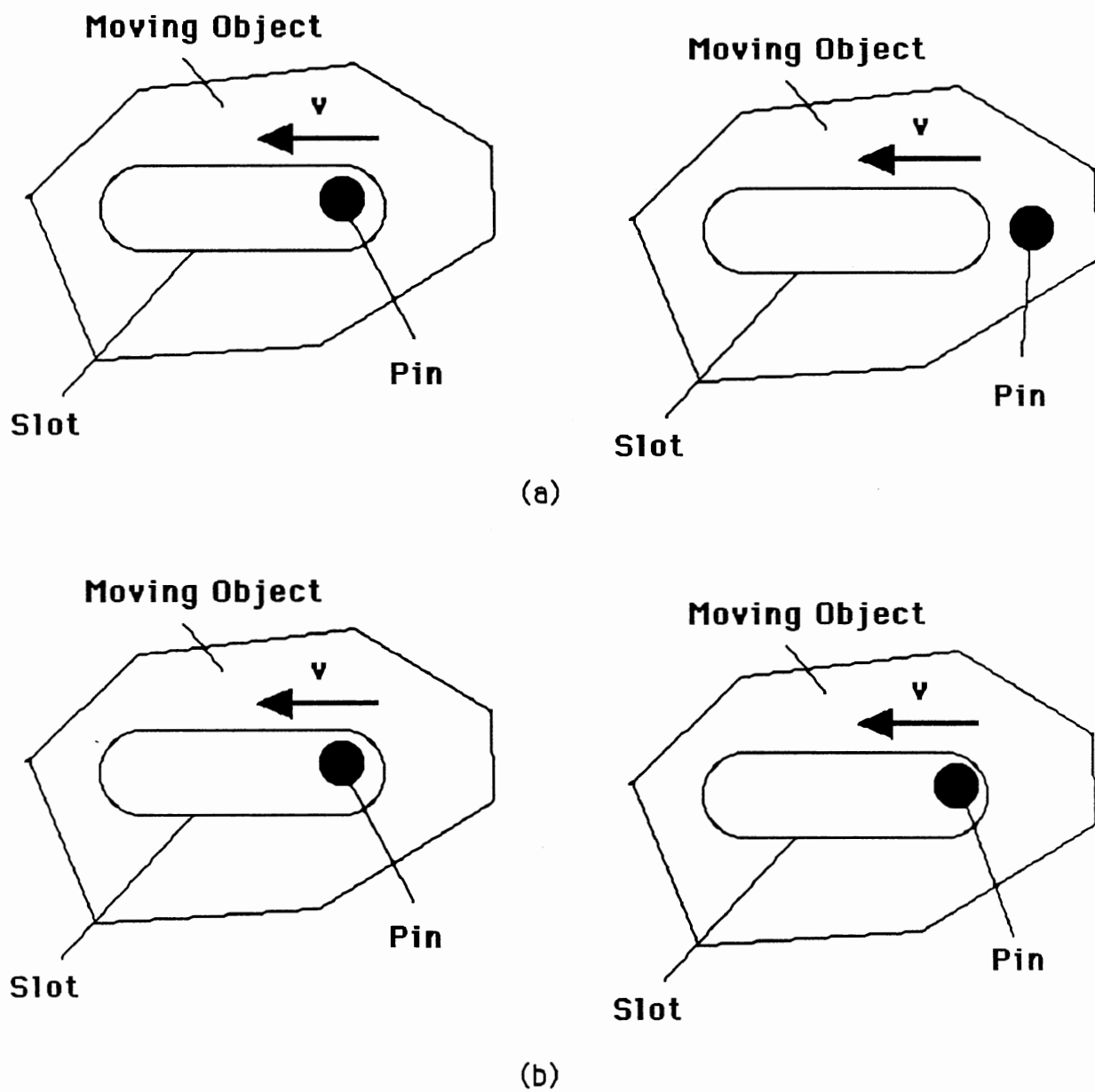


Figure 27. Movement by Using Large Time Step and Small Time Step

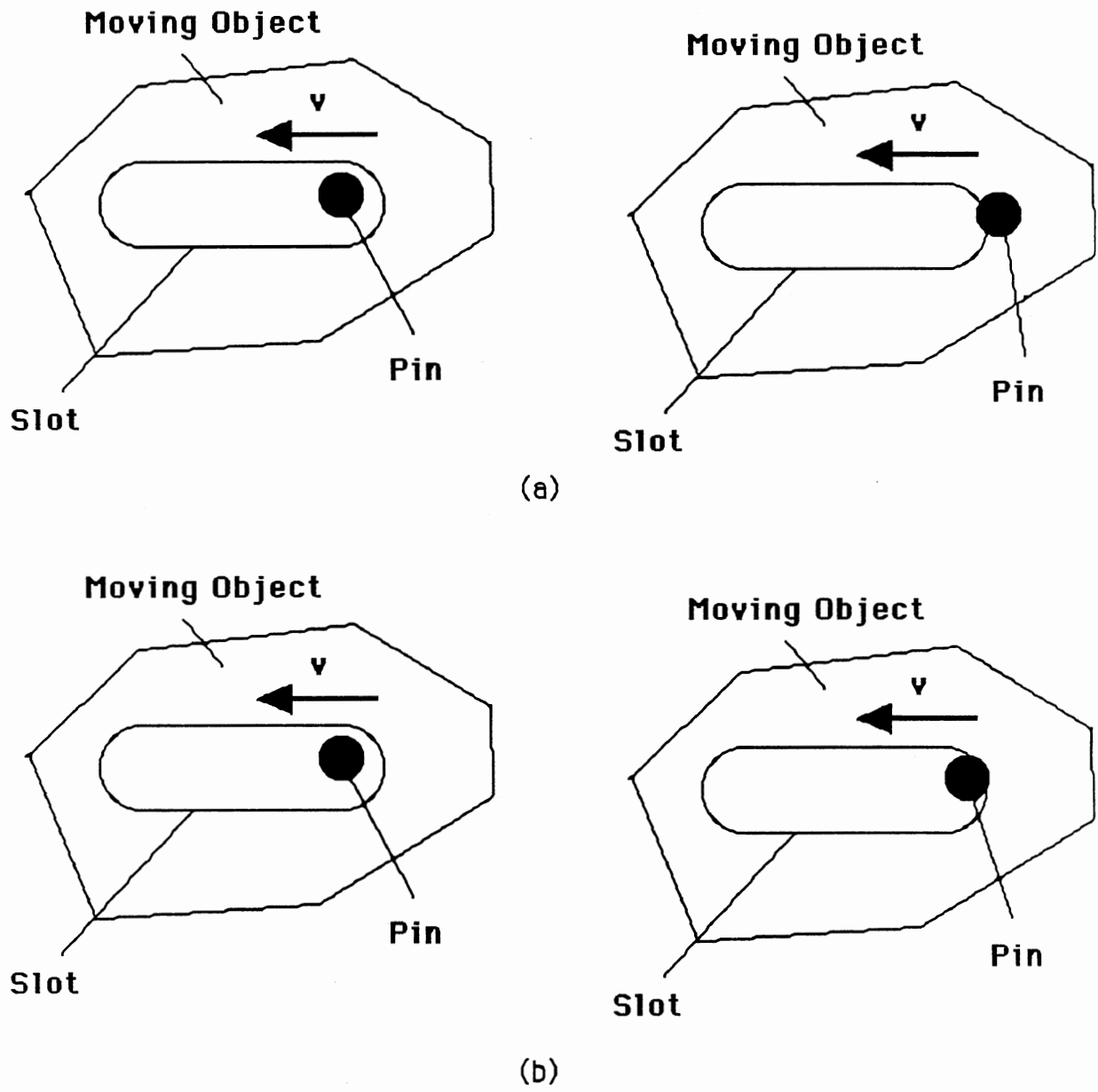


Figure 28. Movement with Large Collision and Small Collision

Component Data Structure

The data structure of the component is defined as follows:

```
struct OBJ
{
    double    outloop[MAX_VERTEX][3];
    int       outloop_vertex_number;
    double    pin_center[MAX_PIN][3];
    double    pin_r[MAX_PIN];
    int       pin_number;
    double    slot_center[MAX_SLOT][3];
    double    slot_dir[MAX_SLOT][3];
    double    slot_w[MAX_SLOT];
    int       slot_number;
    double    mass_center[3];
    double    spring_fix_pointer[3];
    struct OBJ *next;
};
```

where the pin and the slot are defined in Figure 29.

By using this data structure, we can define the planar mechanical part.

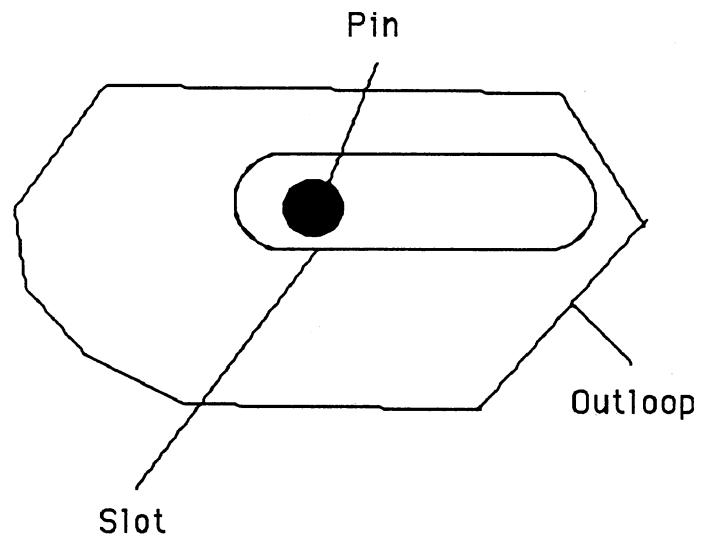


Figure 29. Pin, Slot and Outloop

CHAPTER IV

DEVELOPING A SOLID MODELING SYSTEM

Introduction

In order to use a computer to help people design and manufacture products, we must install some information to describe the products since the computer can not actively get information on its own. It is not an easy job to input all the data which is used to define the object. So, we should develop an application program. This program is called Solid Modeling System.

A Solid Modeling System is a program which allows the user to input basic information to describe the object and manipulate this information and store in a data structure for further applications.

Solid Representation in Solid Modeling System

Since all rigid real-life objects are solid, we should find a way to express them as models in the computer. As we described above, many models express the objects, such as CSG Model, Boundary Representation Model, Decomposition Model, etc. Every model has its own character, for example, the CSG Model is an easy way to express the object through the Boolean Set Operation. The Boundary Representation Model is a good model for display. So, in different situations, we should choose the appropriate model.

In our Solid Modeling System, we choose the CSG Model for user interface since this is the easiest way to express the solid. We choose the Boundary Representation Model as an internal solid model since we can develop more applications based on this method,

such as displaying the object and generating finite element mesh, etc.

The Solid Modeling System Configuration is depicted in Figure 30.

CSG Primitives

In CSG method, we should encourage the user by implementing default primitives and Boolean Set Operations in order to create the object. In this Solid Modeling System, we support the three basic primitives, Block, Cylinder, and Cone. Also, we support three Regularized Boolean Set Operations, Union*, Difference* and Intersection*.

Since all the CSG primitives are defined in its local coordination, we also support the translation and rotation function in order to allow users move the primitives to their proper place to do Regularized Boolean Set Operations.

Boundary Representation Data Structure

The data structure which is used to express the solid is the heart of Boundary Representation Model. Choosing a good data structure can save memory space and calculation time.

Below is my data structure used in Boundary Representation to express a solid:

```
struct SOLID
{
    int          solid_id;
    int          solid_type;
    struct FACE  *face_pointer;
    struct S_VERTEX_T *svt_pointer;
    struct SOLID *next;
}
```

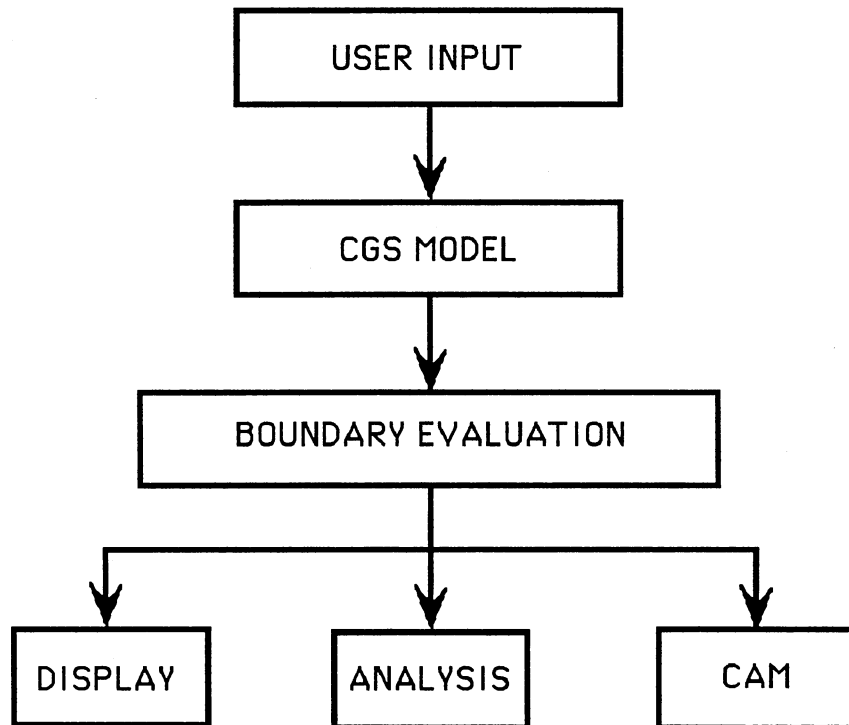


Figure 30. Solid Modeling System Configuration


```

struct FACE
{
    int            face_id;
    int            inter_loop_number;
    double         face_normal_x, face_normal_y, face_normal_z;
    struct LOOP    *loop_pointer;
    struct LOOP    inter_loop_pointer[MAX_INTER_LOOP_NUMBER];
    struct SOLID   *solid_pointer;
    struct FACE    *next;
};

struct LOOP
{
    int            vert_id;
    int            vert_type;
    struct LOOP    *for_pointer, *back_pointer;
    struct FACE    *face_pointer;
};

struct S_VERTEX_T
{
    int            index;
    double         x, y, z;
    struct S_VERTEX_T *next;
};

```

This data structure has three advantages:

- 1) It keeps all the information to describe the solid. There is little redundancy information.
- 2) By using the vertex table, we can save a lot of memory space.

3) By using "feed back" pointer, like `face_pointer` in loop structure, `solid_pointer` in face structure, we can quickly get information. For example, suppose we are in the loop structure and want to know vertex values x, y, z . We can use `face_pointer` to go back to face structure and use `solid_pointer` to go back to solid structure. From solid structure, we can find `solid_vertex_table_pointer` to get vertex values of x, y , and z .

Using Data Structure to Express CSG Primitives

Since the CSG primitives are basic solids, we should first use the data structure which we define above, to express these primitives. Then use these primitives which are expressed in the Boundary Representation Model to do boundary evaluation.

For example, in Figure 31, we use this data structure to model a block primitive.

Boundary Evaluation Procedure

The Boundary Evaluation Procedure is an important part in a solid Modeling System. Its function is to change the geometric information and topology information to create the new object depending on the Regularized Boolean Set Operations.

When two objects do Regularized Boolean Set Operation, sometimes new vertices and edges will be created, and the topology of some faces will be changed. For example, the rectangle face will be changed into a concave surface by doing some Regularized Boolean Set Operation. See Figure 32.

Creating the new vertexes and new edges, changing the topology of some faces correctly are the major objectives of the Boundary Evaluation Procedure.

The Boundary Evaluation Procedure in this Solid Modeling System is decomposed step by step. The Procedure is:

- 1) Choose the reference object and candidate object.
- 2) Change all the faces of the candidate object properly based on calculation and

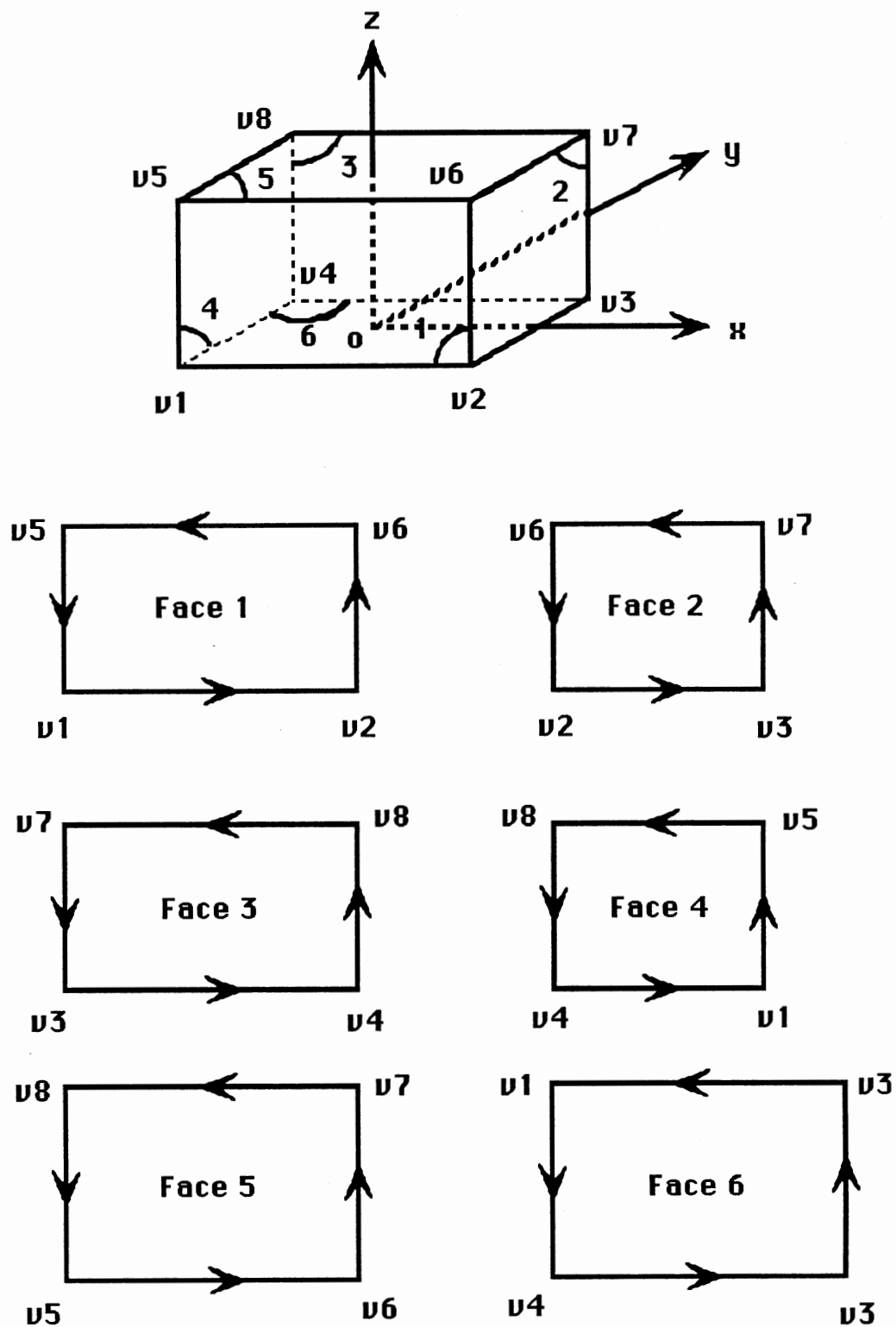


Figure 31. Using Boundary Representation to Model Block Primitive

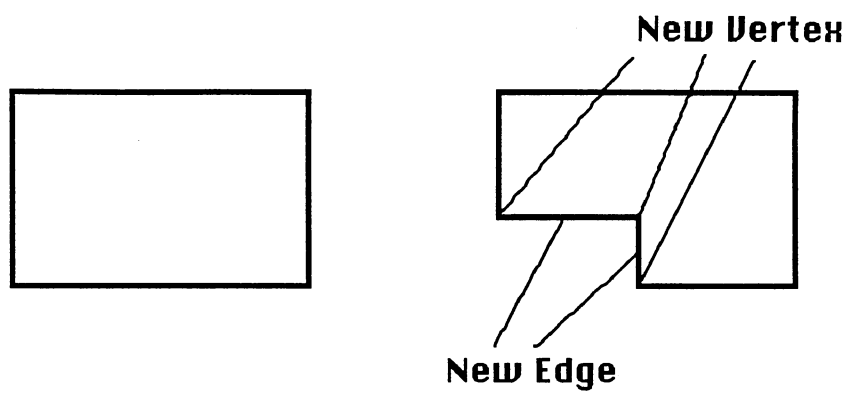


Figure 32. Create New Edges and Vertexes

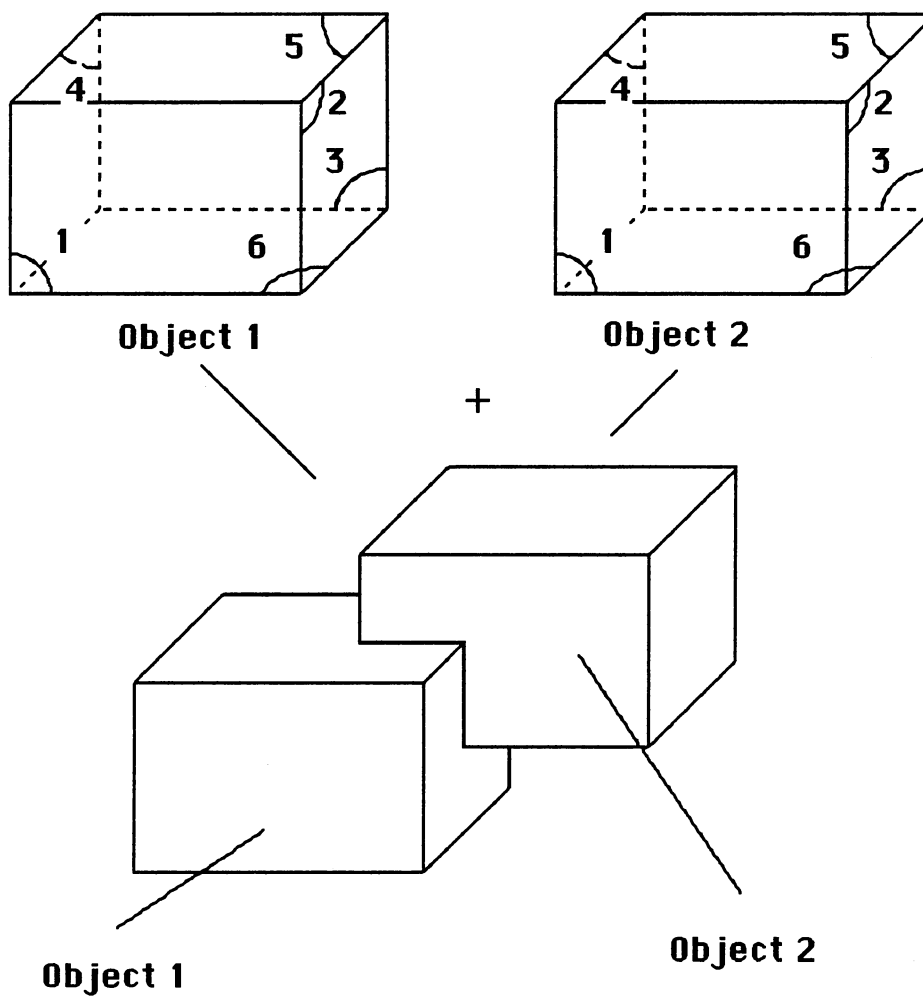


Figure 33 Two Blocks Do Union Boolean Set Operation

restore these changed faces.

3) Swap the reference object and candidate object and repeat step 2.

4) The final object is composed of the changed faces.

Here is an example in Figure 33. Suppose object 1 and object 2 do union Regularized Boolean Set Operation. We define object 1 as a reference object, object 2 as a candidate object. So face 2, 3, 5 of object 2 should not be changed and face 1, 4, 6 of object 2 are changed to face 1', 4', 6'. By swapping the reference object and candidate object, we know that face 1, 4, 6 of object 1 should not be changed, and face 2, 3, 5 of object 1 should be changed to face 2'', 3'', 5''. See Figure 34.

The final object is composited of face 1, 4, 6 of object 1, face 2, 3, 5 of object 2, changed faces, face 2'', 3'', 5'' of object 1 and changed face 1', 4', 6' of object 2.

By this example, we know that after defining the reference and candidate object, we should test every face of candidate object to check whether it should be kept, deleted or changed based on Regularized Boolean Set Operation. Usually, if we get one face and one object, we can classify which part of the face is inside object and which part of the face is outside the object. Changing each face of the candidate object becomes easy if we know which part of each face of the candidate object is inside the reference object and which part of each face of candidate object is outside the reference object. There is an example in Figure 35. Face 1 is one face of candidate object -- object2. we can calculate what part of face 1 is inside the reference object and what part of face1 is outside the reference object. So if object 1 and object 2 do Union*, we should change face 1 into " outside object 1 part of face 1 ". If object 1 and object 2 do Difference*, we should change face 1 into " inside object 1 part of face 1". If object1 and object 2 do Intersection*, we should change face 1 "inside object 1 part of face 1".

Table XII illustrates how to change the face based on Regularized Boolean Set Operation.

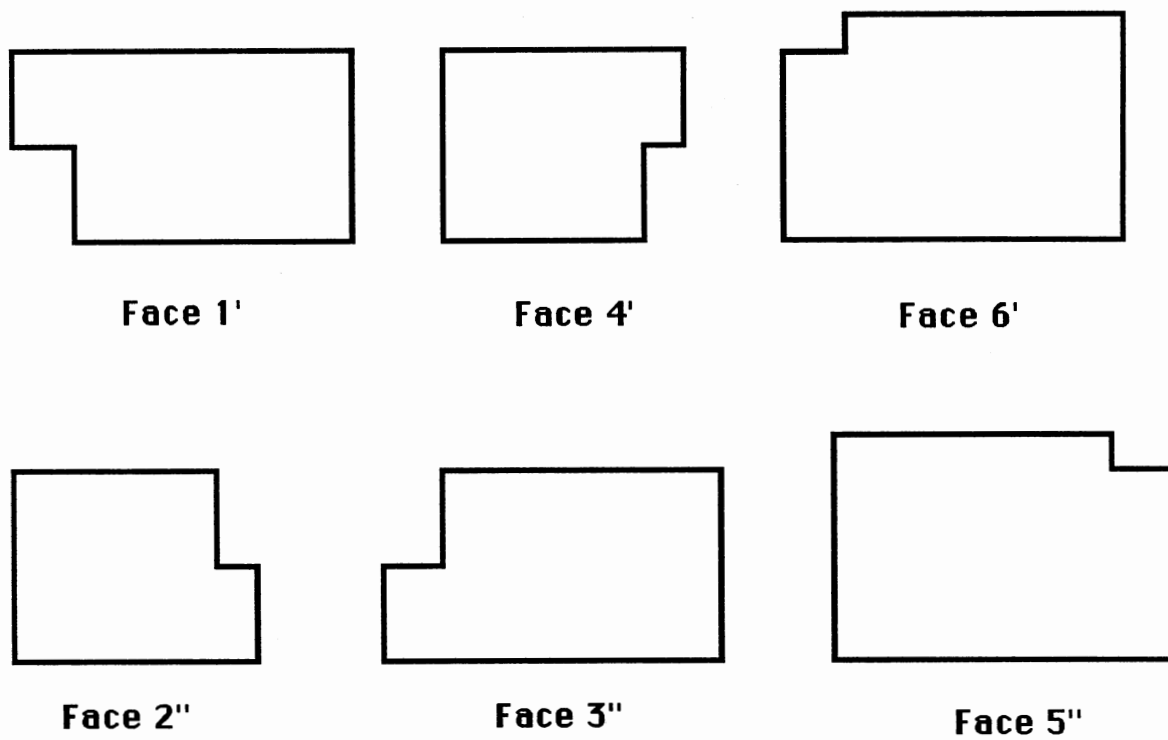
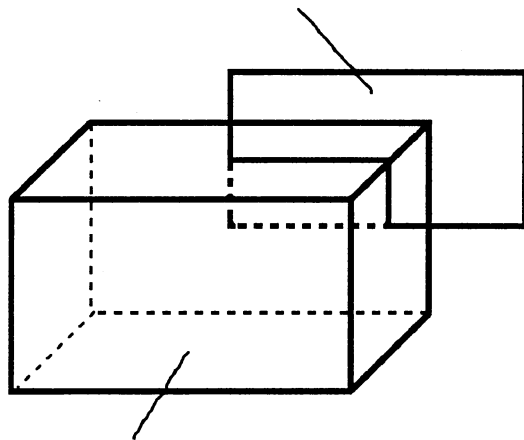
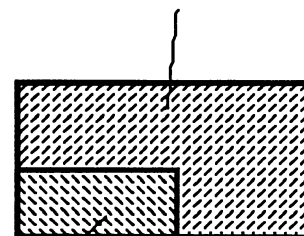


Figure 34. Changed Faces by Doing Regularized Union Boolean Set Operation

Face 1 is one face of candidate object



Outside object 1 part of face



Object 1 as reference object

Inside object 1 part of face 1

Figure 35. Relation Between Face and Object

TABLE XII

CHANGING FACE RULES

	Changed Face of Object1	Changed Face of Object2
Union*	Outside Object2 Part	Outside Object1 Part
Difference*	Outside Object2 Part	Inside Object1 Part
Intersection*	Inside Object2 Part	Inside Object1 Part

Geometric Calculation

When we use the Boundary Evaluation Procedure, we first should calculate the intersection line of the two faces. Usually we can write the following equations to describe the two planes. See Figure 36.

$$\text{Plane 1} \quad a_1 * x + b_1 * y + c_1 * z + d_1 = 0 \quad (3.1)$$

$$\text{Plane 2} \quad a_2 * x + b_2 * y + c_2 * z + d_2 = 0 \quad (3.2)$$

By using the following formula, we can determine whether the two planes intersect or not.

$$\cos \varphi = \frac{a_1 * a_2 + b_1 * b_2 + c_1 * c_2}{\sqrt{a_1^2 + b_1^2 + c_1^2} * \sqrt{a_2^2 + b_2^2 + c_2^2}}$$

When $\cos \varphi = 1$ or $\cos \varphi = -1$, two planes could parallel or coincident. Choose one pointer (x_1, y_1, z_1) at plane one, and calculate the distance between pointer (x_1, y_1, z_1) and plane two by using the following formula.

$$d = \frac{|a_2 * x_1 + b_2 * y_1 + c_2 * z_1 + d_2|}{\sqrt{a_2^2 + b_2^2 + c_2^2}}$$

If $d = 0$, then two planes are coincident, otherwise the two planes are parallel.

if $\cos \varphi \neq 1$ and $\cos \varphi \neq -1$, then two planes intersect.

The direction of intersection line1 is:

$$p = \begin{bmatrix} b_1 & c_1 \\ b_2 & c_2 \end{bmatrix}$$

$$q = \begin{bmatrix} c_1 & a_1 \\ c_2 & a_2 \end{bmatrix}$$

$$r = \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix}$$

where $a_1, b_1, c_1, a_2, b_2, c_2$ are coefficients of plane equations (3.1) and (3.2).

Now we should check whether this intersection line of two planes intersects two polygons or not. See Figure 37. By choosing one segment of the polygon in plane one, we test whether this segment intersects the intersection line. If the intersection line

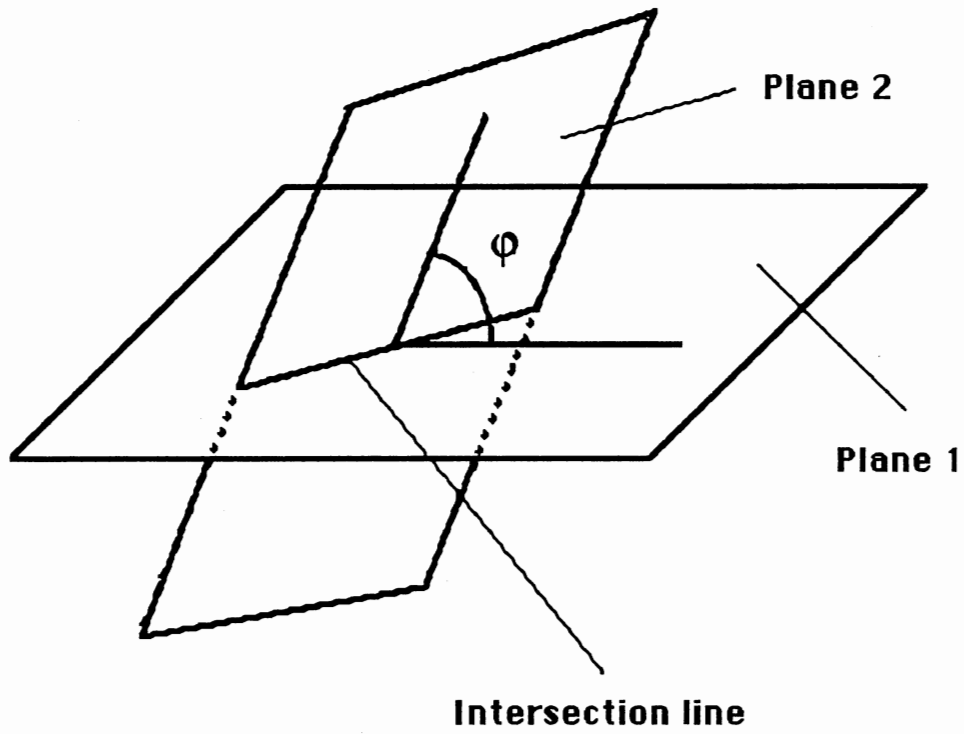


Figure 36. Two Planes

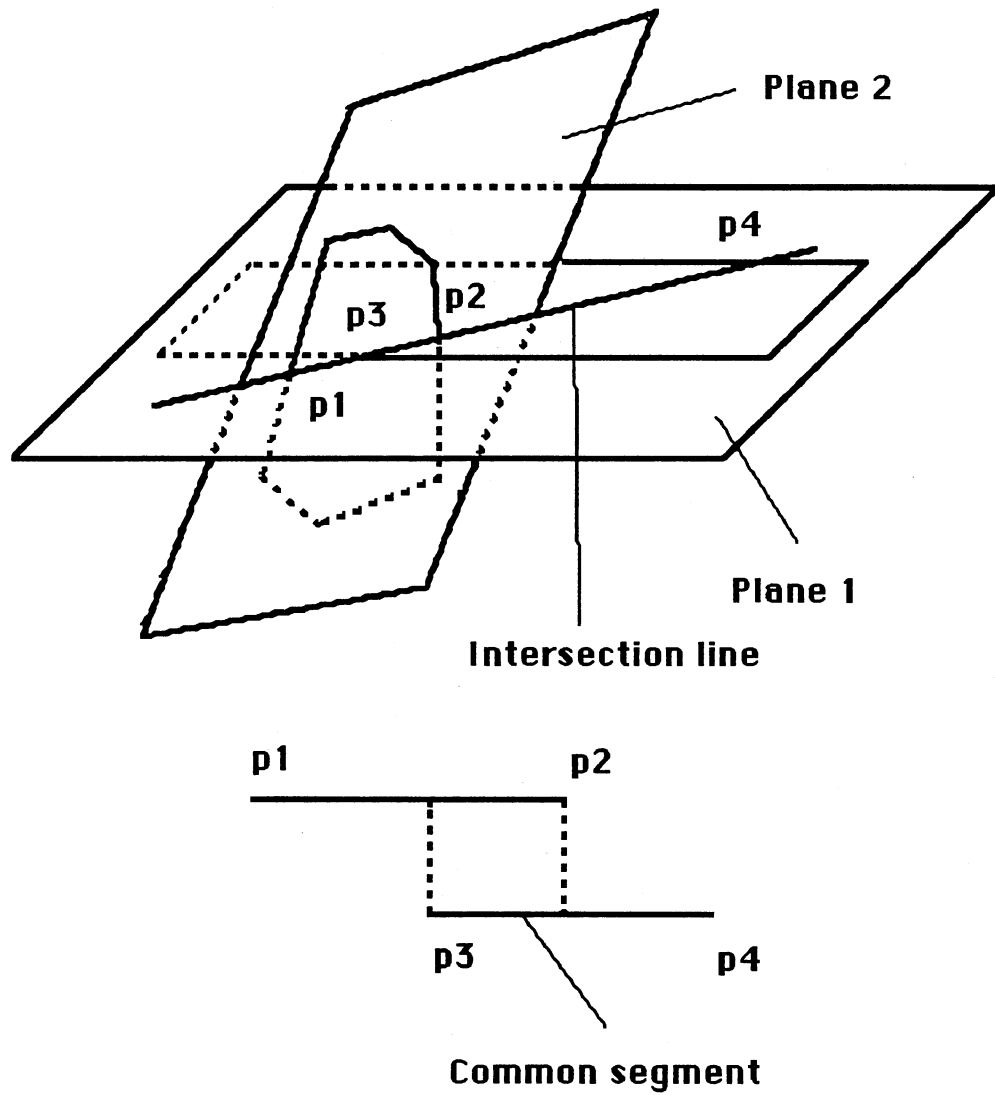


Figure 37. Calculate Common Intersection Segment

intersects the segment of polygon in plane one, restore this intersection pointer. After finishing to check each segment of the polygon in plane one with intersection line, we can get two intersection pointers, p1 and p2. Using the same method, we can check the polygon in plane two and get two intersection pointers p3 and p4. Choosing the common part of the segment p1p2 and segment p3p4, and we get the two polygon intersection segment p2p3.

If the intersection line does not intersect any segment of the polygon in plane one or polygon in plane two, we can deduce that the polygon in plane one and the polygon in plane two have no intersection line. See Figure 38.

The intersection formula of two planar lines is as following:

If we know that line1 direction is p1, r1, q1 and line2 direction is p2, q2, r2, we can calculate the angle between the two lines as,

$$\cos\varphi = \frac{p_1 * p_2 + q_1 * q_2 + r_1 * r_2}{\sqrt{p_1^2 + q_1^2 + r_1^2} * \sqrt{p_2^2 + q_2^2 + r_2^2}}$$

If $\cos \varphi = 1$ or $\cos \varphi = -1$, we know two lines are parallel or coincident. Choose one pointer at line one and then test whether this pointer is in line two or not. If this pointer is in line two, the two lines are identical. Otherwise the two lines are parallel. If $\cos \varphi \neq 1$ and $\cos \varphi \neq -1$, two lines intersect. The line intersection pointer is as following,

Line Equation:

Line one

$$\begin{cases} x = x_1 + p_1 * t_1 \\ y = y_1 + q_1 * t_1 \\ z = z_1 + r_1 * t_1 \end{cases}$$

Line two

$$\begin{cases} x = x_2 + p_2 * t_2 \\ y = y_2 + q_2 * t_2 \\ z = z_2 + r_2 * t_2 \end{cases}$$

if $q_1 * p_2 - q_2 * p_1 \neq 0$, then we can get

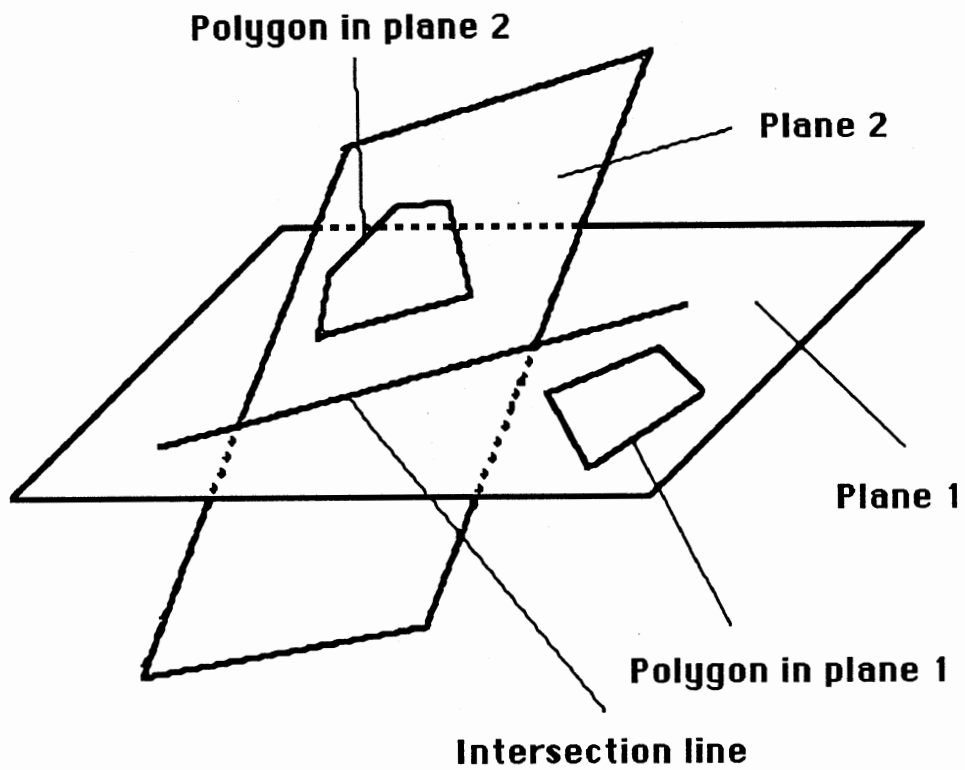


Figure 38. Two Polygons Do Not Intersect

$$t_1^* = \frac{y_2 * p_2 + q_2 * x_1 - q_2 * x_2 - p_2 * y_1}{q_1 * p_2 - q_2 * p_1}$$

if $r_1 * q_2 - r_2 * q_1 \neq 0$, then we can get

$$t_1^* = \frac{z_2 * q_2 + r_2 * y_1 - r_2 * y_2 - q_2 * z_1}{r_1 * q_2 - r_2 * q_1}$$

if $p_1 * r_2 - p_2 * r_1 \neq 0$, then we can get

$$t_1^* = \frac{x_2 * r_2 + p_2 * z_1 - p_2 * z_2 - r_2 * x_1}{p_1 * r_2 - p_2 * r_1}$$

Intersection pointer

$$\begin{cases} x^* = x_1 + p_1 * t_1^* \\ y^* = y_1 + q_1 * t_1^* \\ z^* = z_1 + r_1 * t_1^* \end{cases}$$

CHAPTER V

CONCLUSIONS AND RECOMMENDATIONS

Results

We have implemented the computer-aided preliminary design software based on the concept and objectives stated in Chapter I. We have used this software to design and evaluate various configurations of the camera shutter system referred to previously. By interacting ^{with} the model, we can simulate the planar part movement and test whether design functions properly.

Figure 39 shows a camera shutter system which includes a shutter with a slot, a shutter actuator, a pin, a lens, a spring and two stop blocks. At the initial position, the shutter is kept stationary by the force of a compression spring, the force of the stop block1 and the pin reaction force. When the shutter actuator moves to the right, the shutter rotates around the pin and the lens is opened. When the shutter actuator reaches the far right position, the shutter returns to its initial position by the counterclockwise moment created by the force of the compression spring. Then the shutter actuator moves from right to left to reset. The lens should always remain closed when the shutter actuator resets.

In Figure 40, we see that this design does not function properly. The shutter actuator pushes the shutter until the lens is opened. At this point, the compression spring causes the shutter to rotate around the pin in the clockwise direction. The shutter will never return to its initial position. By changing the location of the spring, we can guarantee that the force of the compression spring creates a counterclockwise moment on the shutter, so that it returns to the stable initial position.

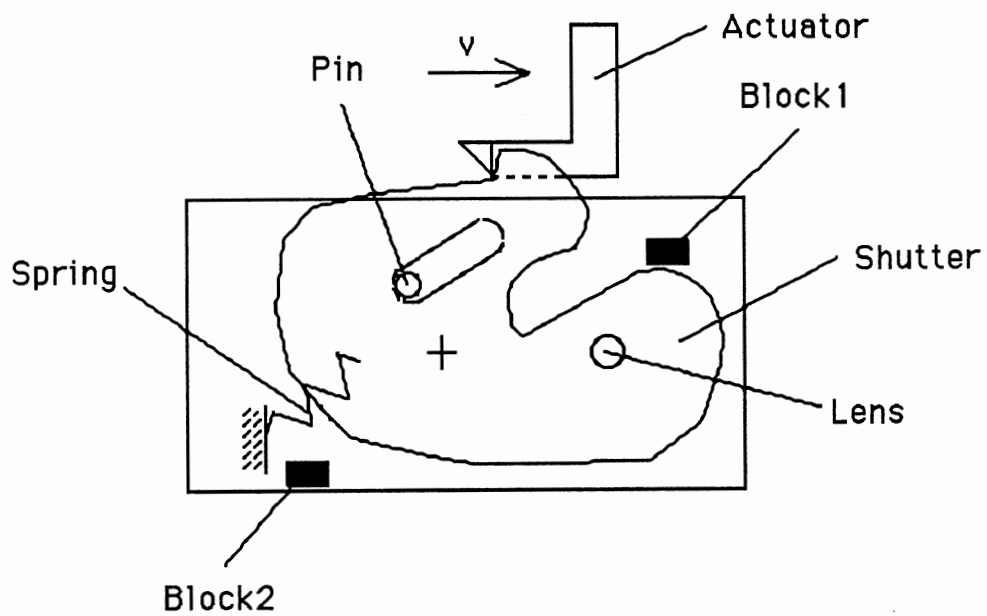


Figure 39. Design 1 Initial Position

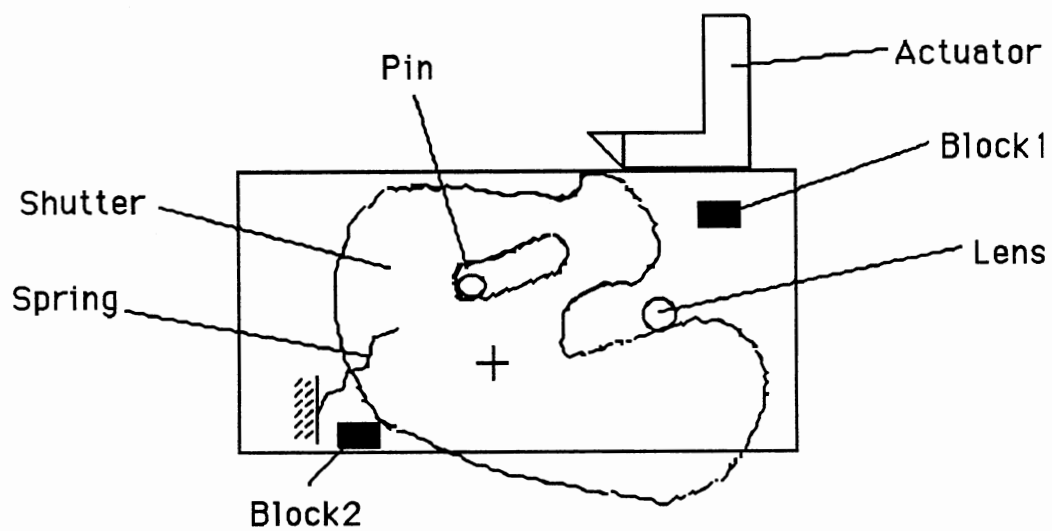


Figure 40. Design 1 in Second Stable Position

In Figure 42, the design also does not function properly. The lens will be opened when the shutter actuator moves from right to left. We can redesign the shape of the shutter to guarantee that the lens is always closed when the shutter actuator returns to its initial position.

By taking advantage of computer-aided preliminary design software, we can easily test the performance of our designs. When problems are found, we can correct them before we have made a large investment of time and money.

Conclusions

In Chapter I, a new concept for preliminary design modeling was developed. According to our objectives, we have explored this concept by developing prototype computer-aided preliminary design software.

In order to get realistic and interactive motion simulation, a dynamic model has been established. Also a method for automated detection of constraint change has been found by using the polygon collision technology. The collision geometry is used to calculate the reaction forces which are applied to the components to maintain the constraints.

We have implemented a user interface by which we can define the components and connect these components together in an assembly. A force block has been created in order to interactively apply a force. Also a slider has been developed by which the model parameters can be changed.

We have developed a solid modeler. By using this solid modeler, we have provided improved visual understanding of the design.

Finally, we have tested the system and found that even though the motion is slow, the user is able to interact the model and detect design problem.

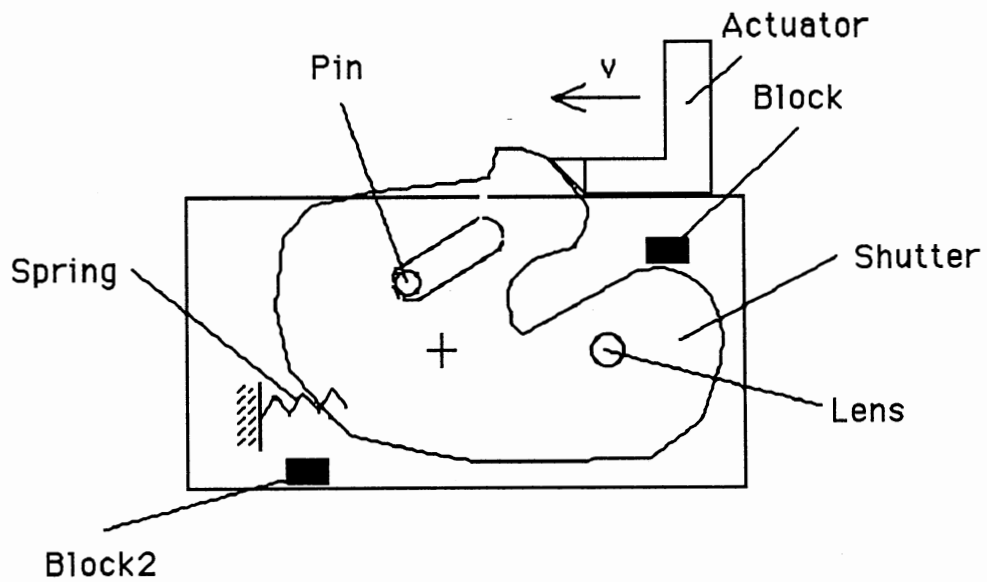


Figure 41. Design 2 Initial Condition

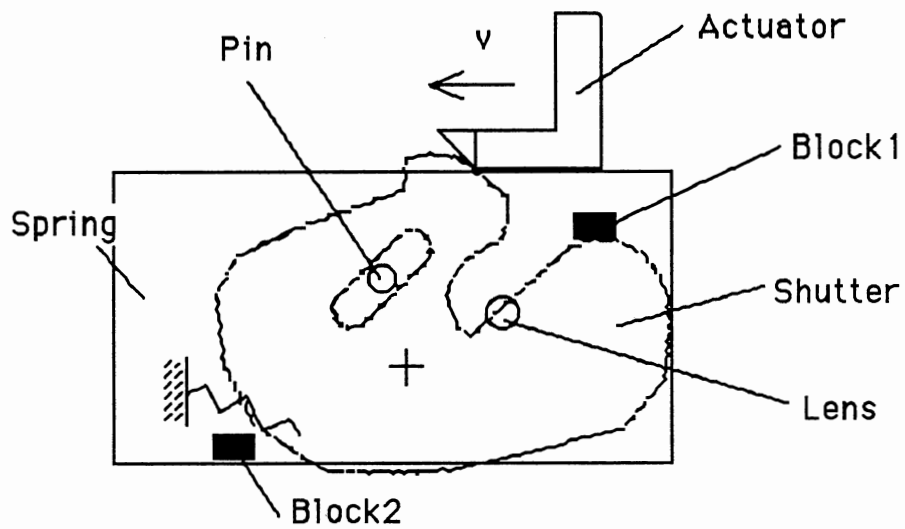


Figure 42. Design 2 With Lens Opening During Shutter Actuator Goes Back

Recommendations

Computer-Aided Preliminary Design

Although we have made a good user interface to define the parts, it uses a fixed sequence to define the parts. We should modify this interface so that we can define "geometric features", such as the outline of a part, define "constraint features" such as slots and pins and associate these features together to make a part. After completing the definition all individual parts, we should connect these parts together in an assembly.

Different mathematical models can be used to represent fixed constraints and variable constraints. We should add to our modeler the ability to handle both fixed and variable constraints.

Even though we have developed a way to simulate the movement of the parts, it may not be the fastest method. More investigations should be done in order to find better ways to perform the simulation as fast as possible, such as high speed parallel computation. For example, one processor could be assigned the task of computing collisions for a single component.

At this point, we have only implemented planar computer-aided preliminary design. We should continue the research to include preliminary design in three dimensions.

Solid Modeling System

This solid modeling system developed in this project is useful in its own right and should be improved. Now we use a data file for user input of geometry. We should write a graphical user interface to interactively define the primitives and Regularized Boolean Set Operations.

We should add "features" such as holes, steps and slots etc. to the solid modeling system and make it intelligent to other application such as manufacturing and assembly.

An investigation should also be done on the data structure used to represent the

mechanical part. Currently, different data structures are used at each phase of the production cycle. A single consistent data structure should be developed which contains the information necessary for the complete production cycle.

REFERENCES

- Baer, A., Eastman, C. M. and Henrion, M. "Geometric Modeling: A Survey." Computer-Aided Design, Vol. 11, No. 5, 1979, pp. 253-272.
- Baumgart, B. "A Polyhedron Representation for Computer Vision." AFIPS Conf. Pro.. In National Computer Conference, 1975, pp. 589-596.
- Baumgart, B. "Geometric Modelling for Computer Vision." PhD Thesis, Stanford University, 1974.
- Bezier, P. Numerical Control-Mathematics and Applications. A. R. Forrest (trans), Wiley, London, 1972.
- Chiyokura, H. and Kimura, F. "A Method of Representing the Solid Design Process." IEEE Computer Graphics and Applications, Vol. 5, Apr. 1985, pp. 32-41.
- Christessen, A. H. J. "Approximation of A Donut." Computer Graphics, Vol. 14, No. 3, July, 1980.
- Coons, S. A. "Surfaces for Computer Aided Design of Space Forms." MIT Project Mac, TR-41, June 1967.
- Dubowsky, S. and Freudenstein, F. "Dynamic Analysis of Mechanical Systems With Clearances - Part I: Formation of Dynamic Model." ASME Journal of Engineering for Industry, Vol. 93, Feb. 1971, pp. 305-309.
- Dubowsky, S. and Freudenstein, F. "Dynamic Analysis of Mechanical Systems With Clearances - Part II: Dynamic Response." ASME Journal of Engineering for Industry, Vol. 93, Feb. 1971, pp. 310-316.
- Elliott, W. S. "Interactive Graphical CAD in Mechanical Engineering Design." Computer-Aided Design, Vol. 10, No. 2, 1978.
- French, M. J. "Conceptual Design for Engineers." Design Council, London, 1985.
- Gilmore, B. J. and Cipra, R. J. "Simulation of Planar Dynamic Mechanical Systems With Changing Topologies - Part I: Characterization and Prediction of the Kinematic Constraint Changes." ASME Journal of Mechanical Design, Vol. 113, March 1991, pp. 70-76.
- Gilmore, B. J. and Cipra, R. J. "Simulation of Planar Dynamic Mechanical Systems With Changing Topologies - Part II: Implementation Strategy and Simulation Results for Example Dynamic Systems." ASME Journal of Mechanical Design, Vol. 113, March 1991, pp. 77-83.

- Gordon, W. J. and Riesenfeld, R. F. "B-Spline Curves and Surfaces." Computer Aided Geometric Design, Academic Press, New York, 1974.
- Jackins, C. L. and Tanimoto, S. L. "Octrees and Their Use in Representing three-dimensional objects." Computer Graphics and Image Processing, Vol. 14, 1980, pp. 249-270.
- Kajiya, J. T. "New Techniques for Ray Tracing Procedurally Defined Objects." ACM Transactions on Graphics, Vol. 2, No. 3, July.1983, pp. 161-181.
- Kawaguchi, E. and Endo, T. "On a Method of Binary Picture Representation and Its Application to Picture Compression." IEEE Transactions on Pattern analysis and Machine Intelligence, Vol. 2, No. 1, 1980, pp. 27-35.
- Light, R. and Gossard, D. "Modification of Geometric Models Through Variational Geometry." Computer-Aided Design, Vol. 14, No. 4, July 1982, pp. 209-214.
- Meagher, D. "Geometric Modeling Using Octree Encoding." Computer Graphics and Image Processing, Vol. 19, 1982, pp. 129-147.
- Orlandea, N., Chace, M. A. and Calahan, D. A. "A Sparsity-Oriented Approach to the Dynamic Analysis and Design of Mechanical Systems - Part I." ASME Journal of Engineering for Industry, Vol. 99, Aug. 1977, pp. 773-779.
- Pfeifer, H. "Methods used for Intersecting Geometrical Entities in the GPM Module for Volume Geometry." Computer-Aided Design, Vol. 17, No. 7, Sept. 1985, pp. 311-317.
- Requicha, A. A. G. "Representations for Rigid Solids: Theory, Methods, and Systems." Computing Surveys, Vol. 12, No. 4, Dec. 1980, pp. 437-465.
- Requicha, A. A. G. and Chan, S. C. "Representation of Geometric Features, Tolerances, and Attributes in Solid Modelers Based on Constructive Geometry." IEEE Journal of Robotics and Automation, Vol. RA-2, No. 3, Sept. 1986, pp. 156-166.
- Requicha, A. A. G. and Tilove, R. B. "Constructive Solid Geometry." Tech. Memo. 25, Production Automation Project, University of Rochester, Rochester, NY, Nov. 1977.
- 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, Rochester, NY, March 1978.
- Samet, H. "The Quadtree and Related Hierarchical Data Structures." ACM Computing Surveys, Vol. 6, 1984, pp. 187-260.
- Shah, J. J. and Rogers, M. T. "Expert Form Feature Modelling Shell." Computer-Aided Design, Vol. 20, No. 9, Nov. 1988, pp. 515-524.
- Smith, D. A. "Reaction Force Analysis in Generalized Machine Systems." ASME Journal of Engineering for Industry, Vol. 95, May 1973, pp. 617-623.

- Thatch, B. R. and Myklebust, A. "A PHIGS - Based Graphics Input Interface for Spatial - Mechanical Design." IEEE Computer Graphics & Applications, Vol. 8, March 1988, pp. 10-30.
- Tilove, R. B. "Set Membership Classification: A Unified Approach to Geometric Intersection Problems." IEEE Transactions on Computers, Vol. C-29, No. 10, Oct. 1980, pp. 874-883.
- Voelcker, H. B. and Requicha, A. A. G. "Geometric Modeling of Mechanical Parts and Processes." Computer, Vol. 10, Dec. 1977, pp. 48-57.
- Wehage, R. A. and Haug, E. J. "Dynamic Analysis of Mechanical Systems With Intermittent Motion." ASME Journal of Mechanical Design, Vol. 104, Oct. 1982, pp. 778-784.
- Winfery, R. C., Anderson, R. V. and Gnilka, C. W. "Analysis of Elastic Machinery With Clearances." ASME Journal of Engineering for Industry, Vol. 95, Aug. 1973, pp. 695-703.

VITA

XIANG HONG

Candidate for the Degree of

Master of Science

**Thesis: CONCEPTS FOR COMPUTER-AIDED PRELIMINARY DESIGN AND
MODELING OF ASSEMBLIES WITH MOVING PARTS**

Major Field: Mechanical Engineering

Biographical:

**Personal data: Born in Shanghai, P. R. C., October 18, 1963, the son of Mr. and
Mrs. Xingxia Hong.**

**Education: Graduated from Yucai High school, Shanghai, P. R. C., in July, 1982;
Received Bachelor of Science in Engineering Degree from Shanghai Jiao Tong
University, Shanghai, P. R. C. in July, 1986; Completed requirements for the
Master of Science degree at Oklahoma State University in December, 1991.**

**Professional Experience: Research Assistant, School of Mechanical and Aerospace
Engineering, Oklahoma State University, Oklahoma, June, 1990, to May,
1991; CAD/CAM Lab Assistant, College of Engineering, Architecture and
Technology, Oklahoma State University, Oklahoma, January, 1990 to May,
1991.**