APPENDIX-3

CSE-87-00004

A RUDIMENTARY DATABASE FOR THREE-DIMENSIONAL OBJECTS USING STRUCTURAL REPRESENTATION

James P. Sowers

A RUDIMENTARY DATABASE FOR THREE-DIMENSIONAL OBJECTS USING STRUCTURAL REPRESENTATION*

James P. Sowers

Computer Science and Engineering Department
University of South Florida
Tampa, Florida

* This project was supported by NASA-Langley grants NAG-1-772 and NAG-1-632.

I. INTRODUCTION

A database which enables users to store and share the description of three-dimensional objects in a research environment is presented. The main objective of the design is to make it a compact structure that holds sufficient information to reconstruct the object. The database design is based on an object representaion scheme which is information preserving, reasonably efficient and yet economical in terms of the storage requirement. The determination of the needed data for the reconstruction process is guided by the belief that it is faster to do simple computations to generate needed data/information for the construction than to retrieve everything from memory.

The next section discusses some recent techniques of three-dimensional representation that influenced the design of the database. Section III gives the schema for the database and the structural definition used to define an object. Section IV contains the user manual for the software developed to create and maintain the contents of the database.

II. BACKGROUND

Most of the three-dimensional object representation schemes [1-4] can be classified into three major categories: surface or boundary, sweep, and volumetric, where a representation scheme is a formal system for descibing shape or some aspect of shape along with rules that specify how that scheme is to be applied to a shape. The description resulting from applying the scheme to a given shape is the representation in that scheme. When deciding on a representation scheme for modeling objects the following properties should be considered [3,5]:

domain - a clearly defined descriptive power for the scheme,

validity - the representation for an object is in the range of

representations for the scheme,

unambiguity - the representation corresponds to a single object in the range of valid representations,

uniqueness - the ability to easily assess the equality of two objects in that for a given object a single representation is formed,

consistency - the same representation is always produced for a given object and scheme,

conciseness - the size, verbosity, or redundancy of the representation,

ease of creation - the ease with which valid representations may be created with the modeling system, and

efficacy for application - the representation is condusive to good, efficient algorithms for computing useful functions.

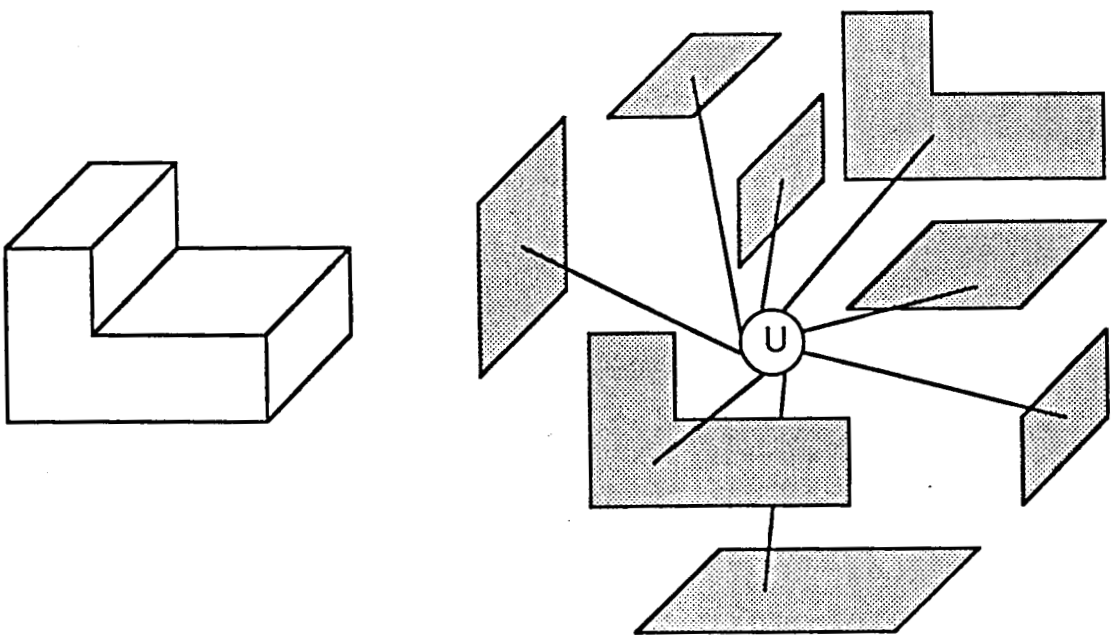

Figure 1. Surface Representation

Surface or boundary representation of an object [6,7] is represented by a set of "faces" or "patches" that are bound

together by a set of rules to make up the object [see figure 1]. Some current approaches include Coons patches [8], bicubic surface patches, Bezier methods [9], and B-splines. Even though boundary representation is unambiguous, it is not unique and the validity is not guaranteed.

Sweep representation of an object is represented by a two-dimensional set that is translated along a line. Two common methods are generalized cylinders [10,11] and symmetric axis transform, also known as the medial axis transform, which was introduced by Blum [12]. Nackman and Pizer present a theory to expand the symmetric axis transform to three dimensions in [13]. The definition of the two-dimensional symmetric axis transform also applies in three dimensions, except that maximal disks become maximal spheres and the symmetric axis becomes the symmetric surface, see figure 2. Generalized cylinders (generalized cones), is analogous to symmetric axis transform in three dimensions. A generalized cylinder is a solid whose axis is a three-dimensional space curve, and its cross sections are orthogonal to its axis, see figure 3. Also, the two-dimensional set defining the generalized cylinder may be allowed to rotate about the axis, while it is translated along the axis.

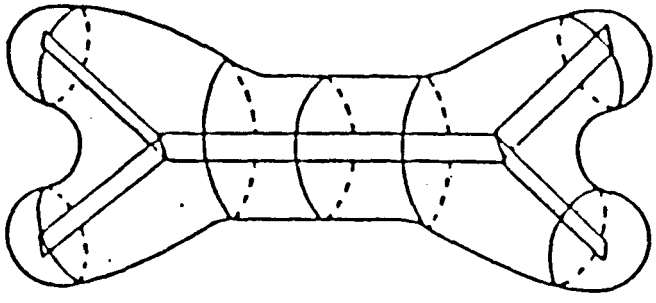

Figure 2. Example of 3-D Symmetric Surface

Sweep representation works well with manmade objects that have an axis of symmetry. It is concise, but in general, it is not

unique.

Volumetric representation of objects [3,14-17] is accomplished by representing an object in terms of more primitive solids. The three representations are: 1) spatial occupancy - values are represented as a three-dimensional array of cells which may be marked as filled or not with matter [15], 2) cell decomposition - cells are more complex in shape but still do not share volumes, so the only combining operation is "glue" [16,17], and 3) constructive solid geometry - complex solids are represented as various ordered operations of simpler objects (primitives), by means of psuedo-Boolean set operations. The primitives used could be simple geometric solids such as prisms, cylinders, ellipsoids, and boxels [18] to more complicated primitives such as superquadrics [19].

Volumetric representation is adequate to comprise most conventional, unsculptured objects and is unambiguous but is not unique.

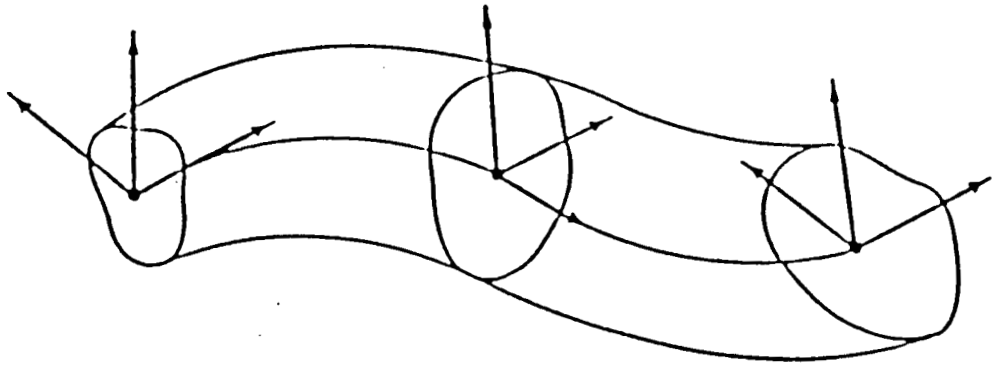

Figure 3. A Generalized Cylinder with some Cross-Sectional Coordinate Systems

III. DATABASE DESIGN

A general model, similar to the one described in [20], is used for the object description in the database, since it allows for the

ability to save properties about the object, such as global attributes and parameters. An object is defined as a 6-tuple O={C, N, A, P, R, PA}, where C is the class of the object, which defines a set of similar objects, N is the name of the object, A is the set of attributes for the object, P is the set of primitives used to make up the object, R is the set of relationship functions used to describe the associations between primitives in set P, and PA is the set of parameters. Parameters in this structure are basic characteristics about the object, such as material composition.

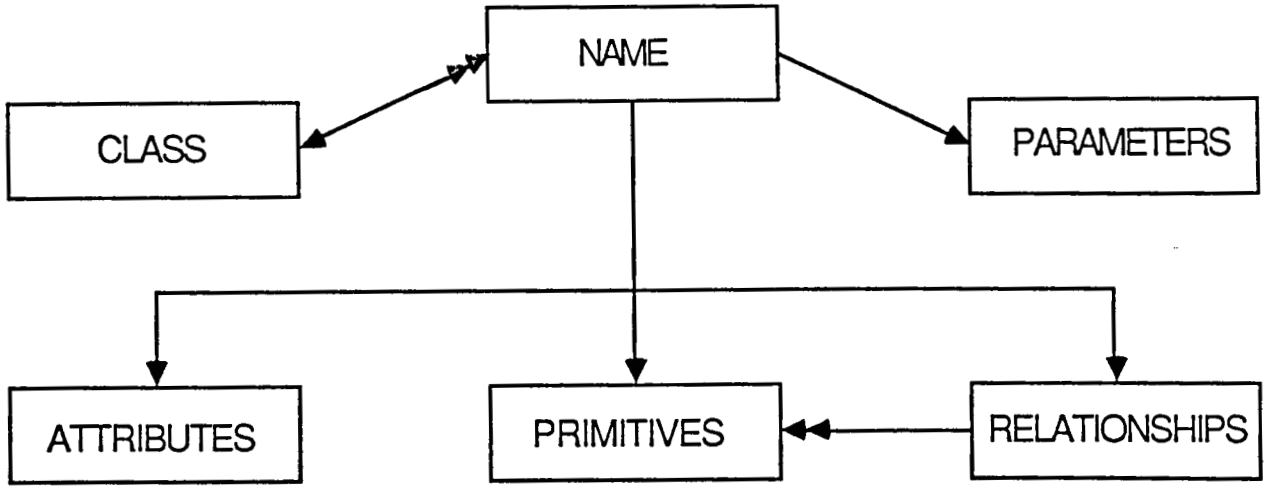

Figure 4. Database Schema

Figure 4 shows the database schema, with the notation borrowed from Martin [21]. The database file structure consists of a main file and a set of attribute, primitive, and relationship files for the objects in the database. For every object in the database a record that contains the name, class, parameters, and pointers to the attribute, primitive, and relationship files for the object is contained in the main file. The attribute file for an object consists of records that contain the attribute name and quantity.

The primitive file for an object consists of records that contain id tag, primitive id type, and three parameters, discussed further in Section III.A. The relationship file for an object consists of records that contain the operator, the two primitive id tags, and six parameters defining connection points and orientation, discussed further in Section III.B.

III.A PRIMITIVES

The present set of volumetric primitives consist of a four-sided prism, a right-wedge, a left-wedge, an ellipsoid, a cylinder, and a cone, which is shown in the appendix. The primitives as shown in the appendix give the "home" position for each primitive, and the format for the type id and the three parameters. The three parameters simply give the overall width, depth, and height of the primitive.

A record in the primitive file contains the id tag, primitive id type, and the three parameters. This information is used to determine the configuration of the primitive. The id tag is an unique integer value to distinguish the primitive from the others in the set. The primitive id type is used to classify the primary shape of the primitive and the three parameter values give the dimension for the final shape of the primitive. For example, if the id type is a cylinder and the values of the parameters are X=5, Z=1, and D=3, the shape of the primitive would be as shown in figure 5.

III.B RELATIONSHIP FUNCTIONS

The relationship describes how the primitives relate to each other so that the object can be reconstructed. The relational operations used are analogous to the ones employed by constructive solid geometry [22], hence the same limitations as mentioned under volumetric representation apply. A record in the relationship file

contains an operator, two ids, two connection points, and two rotational values. The general format for the relationship is as follows:

$$(OP, ID_i, ID_j, X_i, Y_i, Z_i, X_j, Y_j, Z_j, Rx_i, Ry_i, Rz_i, Rx_j, Ry_j, Rz_j)$$

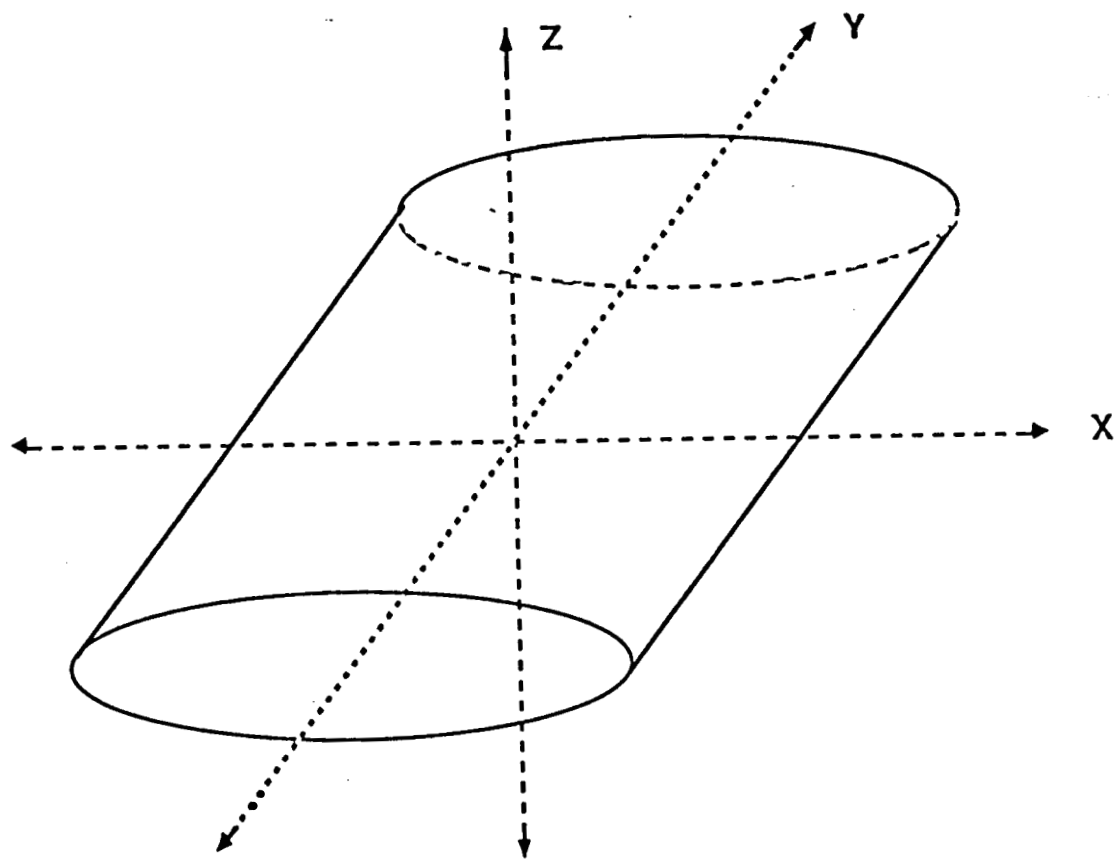

Figure 5. Cylinder with X=5, Z=1, and D=3 in home position

Where ID_i and ID_j are set to id tag values to identify which primitives, from the set of primitives, are to be used for the operation. The (X_i, Y_i, Z_i) and (X_j, Y_j, Z_j) points are values to describe where the two primitives will be connected. The valid range of values, relative to the "home" position and specific primitive type are defind as $-P_i/2 <= V_i <= P_i/2$; P_i is the

parameter value for the primitive in the i^{th} position, where i=1,2, or 3, and V_i is the value for the i^{th} position. (Rx_i, Ry_i, Rz_i) and (Rx_j, Ry_j, Rz_j) are the angles that the primitives are rotated after being connected; the rotation is about the connection point.

Union, intersection, and subtraction are the three operators employed to operate on the primitives. The operations, with their present restrictions, are defined below.

union operation:

$(+, ID_i, ID_j, X_i, Y_i, Z_i, X_j, Y_j, Z_j, 0, 0, 0, Rx_j, Ry_j, Rz_j)$

This is the union operation which connects the surface point (X_j, Y_j, Z_j) of primitive j to the surface point (X_i, Y_i, Z_i) of primitive i, with primitive j being rotated about its point of connection by (Rx_j, Ry_j, Rz_j) and the rotation angles for primitive i are set to zero.

intersection operation:

$(*, ID_i, ID_j, X_i, Y_i, Z_i, 0, 0, 0, 0, 0, 0, Rx_j, Ry_j, Rz_j)$

This is the intersection operation which places primitive j in primitive i, with the origin point of primitive j being located in primitive i at point (X_i, Y_i, Z_i) and primitive j being rotated about its connection point (origin) by (Rx_j, Ry_j, Rz_j). The rotation angles for primitive i are set to zero.

subtraction operation:

$(-, ID_i, ID_j, X_i, Y_i, Z_i, 0, 0, 0, 0, 0, 0, Rx_j, Ry_j, Rz_j)$

This is the subtraction operation which is identical to the intersection operation except primitive j is removed from primitive i.

The next section contains the user manual for the database that has just been described.

IV OPERATIONS ON THE DATABASE

IV.A Entering an object

Name:

After the main menu appears and selection of the CREATE OBJECT option, the following prompt will appear:

Please enter object name:

Enter any string up to 80 characters in length, but only the first 15 significant characters are kept, and the string must contain at least one non-blank character. Leading and trailing blanks are stript, however embedded blanks are not stript from the string. If an invalid name is entered the user will be prompted again for another name. Also if object already appears in the database the user will be asked for another name. The following examples show valid and invalid names.

Example 1.

VALID ENTRY	
NAME	COMMENT
1	one non-blank
A	one non-blank
A1C$	any character legal
bbbALPHAbbb	leading and trailing blanks ignored
ALPHA	recognized same as above
Alpha	different by case of lettering
abcd1234efgh5678ijk	only a through 7 is considered
good name	embedded blank left in name
	INVALID ENTRY
<cr>	null string
bbbbbbb	non-blank rule broke

Class:

The following prompt will appear for entry of object class:

Please enter object class:

The criteria for a valid class name is the same as for a valid object name. If an invalid class is entered the user will be prompted again for entry of class name.

Parameters:

Currently the parameters for an object have not been fully defined and the only parameter presently being looked for is the object's material composition. The following prompt will appear:

Please enter primary material composition of object:

The criteria for a valid material entry is the same as for a valid object name. If an invalid material entry is made the user will be prompted again for entry of material.

Attributes:

The present structure for object attributes is for name and quantity of attribute. One or more attributes can be entered fo any one object. The following prompt will appear:

Please enter attribute:

The criteria for a valid attribute entry is the same as for a valid object name. If an invalid attribute entry is made the user will be prompted again for entry of an attribute. After valid attribute is entered, the quantity (integer value) will be asked for and the following prompt will appear:

Enter quantity of this attribute:

No error correction is offered. After entry of quantity the user will be prompted, by the following, for continuation:

Enter another attribute (y/n)?

To continue entering attributes, enter either y or Y. To sto entering attributes, enter either n or N, if any other character is entered, the default is to stop.

Primitives:

There are presently six primitives available for composing an object as seen in appendix A. The following menu will appea for entering primitives comprising the object:

	PRIMITIVES	QTY
1)	Four-sided prism	#
2)	Right-wedge	#
3)	Left-wedge	#
4)	Ellipsoid	#
5)	Cylinder	#
6)	Cone	#
0)	Quit	

Choice:

The QTY column reminds the user of the number of primitives entered for each type. Depending on which primitive is selected the user will be prompted to enter the following dimensions fo that primitives: enter the X-axis diameter or length, the Z-axis diameter or height, and the Y-axis diameter or depth of the primitive. The values expected for each is a real number, n error correction is offered. The 0 option is used to qui entering primitives.

Relationship:

The relationships describe how the primitives entered above relate to each other so that the object can be reconstructed. The format for the relationship is as follows:

$$(OP, PRIM_i, PRIM_j, X_i, Y_i, Z_i, X_j, Y_j, Z_j, Rx_i, Ry_i, Rz_i, Rx_j, Ry_j, Rz_j)$$

union operator:
$(+,PRIM_i,PRIM_j,X_i,Y_i,Z_i,X_j,Y_j,Z_j,0,0,0,Rx_j,Ry_j,Rz_j)$

This is the union operator which connects the surface point (X_j,Y_j,Z_j) of primitive j to the surface point (X_i,Y_i,Z_i) of primitive i, with primitive j being rotated about its point of connection by (Rx_j,Ry_j,Rz_j).

intersection operator:
$(*,PRIM_i,PRIM_j,X_i,Y_i,Z_i,0,0,0,0,0,0,Rx_j,Ry_j,Rz_j)$

This is the intersection operator which places primitive j in primitive i, with the origin point of primitive j being located in primitive i at point (X_i,Y_i,Z_i) and primitive j being rotated about its origin point by (Rx_j,Ry_j,Rz_j).

subtraction operator:
$(-,PRIM_i,PRIM_j,X_i,Y_i,Z_i,0,0,0,0,0,0,Rx_j,Ry_j,Rz_j)$

This is the subtraction operator which is identical to the intersection operator except primitive j is removed from primitive i.

For entry of relationship the following prompt will appear:

Number of operations entered: #

Enter first primitive tag id:

The first prompt reminds the user of the number of relationship operations entered so far. The second one is looking for the tag id (integer value) for the first primitive involved. No error correction is offered. The next prompt to appear is:

Enter second primitive tag id:

Here enter the tag id (integer value) for the second primitive involved. No error correction is offered. The next prompt to appear is:

Enter desired operator (+,-,*):

Depending on the operator entered, the user will be prompted to enter the appropriate (real) values for the connection points and orientations. If the user does not want to enter a relationship, enter a character other than the operators and the current operation will be ignored. After entry of an operation the following prompt will appear:

Enter another operation (y/n)?

To continue entering relationships, enter either y or Y. To stop entering relationships, enter either n or N, if any other character is entered, the default is to stop.

IV.B Deleting an object

After selecting the DELETE OBJECT option in the main menu, the following prompt will appear:

Please enter object name:

Enter any string up to 80 characters in length, but only the first 15 significant characters are kept, and the string must contain at least one non-blank character. Leading and trailing blanks are stript, however embedded blanks are not stript from the string. If an invalid name is entered the user will be prompted again for another name.

If name entered exist it will be removed from the database along with all other files or information relating to it, otherwise an error message will appear stating no object with that name presently exist in the database.

IV.C Viewing the database

LIST OBJECTS - This option for viewing the database list the objects by name, a page at a time. To stop viewing the database, enter either n or N to the inquiry about continuing.

LIST CLASSES - This option for viewing the database list the classes and the objects associated with each class, a page at a time. To stop viewing the database, enter either n or N to the inquiry about continuing.

IV.D Changing an object

After selecting the UPDATE OBJECT option in the main menu the following prompt will appear:

Please enter object name:

Enter any string up to 80 characters in length, but only the first 15 significant characters are kept, and the string must contain at least one non-blank character. Leading and trailing blanks are stript, however embedded blanks are not stript from the string. If an invalid name is entered the user will be prompted again for another name.

Name change:
If the object exist in the database the following prompts will appear:

Object's name: <Hopefully the one the user entered>

Change object's name (y/n)?

To change name, enter either y or Y, then the following prompt will appear:

Please enter object name:

The criteria for name is the same as for entry of name for change. If an invalid name is entered the user will be prompted again for another name.

Class change:
For changing the class, the following prompt will appear:

Object's class: <class for object>

Change object's class (y/n)?

To change the class of the object, enter either y or Y, then the following prompt will appear:

Please enter object class:

The criteria for a valid class entry is the same as for a valid object name. If an invalid class is entered the user will be prompted again for a class.

Parameter change:
The present format for parameters is undefined and the only thing contained in this structure is the primary material composition of the object. The following prompt will appear for changing the material:

Object's material: <material of object>

Change object's material (y/n)?

To change the material, enter either y or Y, then the following prompt will appear:

Please enter primary material composition of object:

The criteria for a valid material entry is the same as for a valid object name. If an invalid material is entered the user will be prompted again for a material.

Attribute change:
For changing the attributes of the object, the following prompt will appear:

Change object's attributes (y/n)?

To change attributes, enter either y or Y, then the following prompt will appear:

Attribute is as follows:
name: <attribute>
quantity: #

Do you want to C)hange
D)elete
or get N)ext attribute

Change:
To change this attribute of the object, enter either c or C, the following prompt will appear:

Please enter attribute:

The criteria for an attribute is the same as for a valid object name. If an invalid attribute is entered the user will be prompted again for another attribute. Then the user will be asked for the quantity of this attribute by the following prompt:

Enter quantity of this attribute:

An integer value is being looked for and no error checking is offered. If only the quantity is desired to be changed, select the C)hange option and reenter the attribute then when asked for quantity enter the change.

Delete:
To delete the currently displayed attribute, enter either d or D, no prompt will appear for this option.

Next:
If no action is desired for the currently displayed attribute, to retrieve the next attribute, enter either n or N. No prompt will appear for this option. This allows a way to review the attributes associated with an object, without changing them.

Add:
After current attribute list is reviewed then you have the option to append more attributes to the list. The following prompt will appear for additions:

Add new attribute (y/n)?

To add another attribute, enter either y or Y, and the user will be prompted for attribute and quantity in the same manner as for changing attribute. Entering either n or N will stop changes to attribute list, also if any other character is entered the process will stop.

Primitive change:
For changing the primitives of the object, the following prompt will appear:

Change object's primitives (y/n)?

To change primitives, enter either y or Y, then the following prompt will appear:

```
Primitive is as follows:
                    Tag: #
                     Id: <primitive code>
          Length/X-axis: #
           Depth/Y-axis: #
          Height/Z-axis: #

Do you want to D)elete
        or get N)ext primitive
```

Delete:
To delete the currently displayed primitive, enter either d or D, no prompt will appear for this option.

Next:
If no action is desired for the currently displayed primitive, to retrieve the next primitive, enter either n or N. No prompt will appear for this option. This allows a way to review the primitives associated with an object, without changing them.

Add:
After current primitive list is reviewed then you have the option to append more primitives to the list. The following prompt will appear for additions:

```
Add new primitive (y/n)?
```

To add another primitive, enter either y or Y, and the user will be prompted as follows:

```
          PRIMITIVES
          ~~~~~~~~~~

      1) Four-sided prism
      2) Right-wedge
      3) Left-wedge
      4) Ellipsoid
      5) Cylinder
      6) Cone
      0) No creation of primitive


Choice:
```

Depending on which primitive is selected, the user will be prompted to enter the following dimensions for that primitives: enter the X-axis diameter or length, the Z-axis diameter or height, and the Y-axis diameter or depth of the primitive. The values expected for each is a real number, no error correction is offered. The 0 option is used to quit entering primitives.

Relationship change:
For changing the relationships of the object, the following prompt will appear:

Change object's relationships (y/n)?

To change relationships, enter either y or Y, then the following prompt will appear:

```
Relationship is as follows:
                    First primitive id: <tag #>
                   Second primitive id: <tag #>
                              Operator: <*,-,+>
 Connection point for first id(x,y,z): #,#,#

          Rotation of first id(x,y,z): #,#,#

Connection point for second id(x,y,z): #,#,#

         Rotation of second id(x,y,z): #,#,#

Do you want to D)elete
       or get N)ext relationship
```

Delete:
To delete the currently displayed relationship, enter either d or D, no prompt will appear for this option.

Next:
If no action is desired for the currently displayed relationship, to retrieve the next relationship, enter either n or N. No prompt will appear for this option. This allows a way to review the relationships associated with an object, without changing them.

Add:
After current relationship list is reviewed then you have the option to append more relationships to the list. The following prompt will appear for additions:

Add new relationship (y/n)?

To add another relationship, enter either y or Y, and the user will be prompted as follows:

Enter first primitive id:

Enter the tag id (integer value) for the first primitive involved. No error correction is offered. The next prompt to appear is:

Enter second primitive id:

Here enter the tag id (integer value) for the second primitive involved. No error correction is offered. The next prompt to appear is:

Enter desired operator (+,-,*):

Depending on the operator entered, the user will be prompted to enter the appropriate (real) values for the connection points and orientations. If the user does not want to enter a relationship, enter a character other than the operators and the current operation will be ignored.

IV.E Main menu

The following is the group of operations on the database, which is described in section IV.

1) Create object
2) Delete object
3) List objects
4) List classes
5) Update object
0) Quit

Choice:

ACKNOWLEDGEMENTS

Support for this project, from NASA-Langley Research Center grants NAG-1-772 and NAG-1-632, is gratefully acknowledged. Numerous discussions with and encouragement from the research group (COVIRT*) members at USF, Mike Goode and Karin Cornils of NASA-Langley Research Center is also acknowledged.

* COVIRT - COmputer Vision and Intelligent Robotics research Team.

REFERENCES

[1] J.K. Aggarwal, L.S. Davis, W.N. Martin, and J.W. Roach, "Survey: Representation Methods for Three-dimensional Objects," in *Progress in Pattern Recognition*, L.K. Kanal and A. Rosenfeld, Eds. North-Holland, 1981, pp. 377-391.

[2] N. Badler and R. Bajcsy, "Three-dimensional Representations for Computer Graphics and Computer Vision," *Computer Graphics*, vol. 12, pp. 153-160, August 1978.

[3] A.A.G. Requicha, "Representations for Rigid Solids: Theory, Methods, and Systems," *Computing Surveys*, vol. 12, pp. 437-464, December 1980.

[4] T.C. Henderson, "Efficient 3-D Object Representations for Industrial Vision Systems," *IEEE Trans. Pattern Anal. and Machine Intell.*, vol. PAMI-5, pp. 609-618, November 1983.

[5] C.M. Brown, "Some Mathematical and Representational Aspects of Solid Modeling," *IEEE Trans. Pattern Anal. and Machine Intell.*, vol PAMI-3, pp. 444-453, July 1981.

[6] A.R. Forrest, "On Coons and Other Methods for the Representation of Curved Surfaces," *Comput. Graphics Image Processing*, vol. 1, pp. 341-359, 1972.

[7] R.E. Barnhill and R.F. Risenfeld, "Surface Representation for Computer Aided Design," in *Data Structures, Computer Graphics and Pattern Recognition*, A. Klinger, K.S. Fu, and T.L. Kunii, Eds. New York: Academic, 1977.

[8] S. Coons, "Surfaces for Computer-aided Design of Space Forms," M.I.T. Project MAC, MAC-TR-41, 1967.

[9] P. Bezier, "Mathematical and Practical Possibilities of UNISURF," in *Computer Aided Geometric Design*, R. Barnhill and R. Riesenfeld, Eds., New York: Academic Press, 1974.

[10] T. Binford, "Visual Perception by Computer," Invited paper, *IEEE Systems Science and Cybernetics Conference*, Miami, December 1971.

[11] G. Agin, "Representation and Description of Curved Objects," Ph.D. Thesis, Stanford A.I. Memo, AIM-173, October 1972.

[12] H. Blum, "A Transformation for Extracting New Descriptors of Shape," in *Models for the Perception of Speech and Visual Form*, W. Wathen-Dunn, Ed., Cambridge, MA: MIT Press, 1967, pp. 362-380.

[13] L.R. Nackman and S.M. Pizer, "Three-dimensional Shape Description Using the Symmetric Axis Transform I: Theory," *IEEE Trans. Pattern Anal. and Machine Intell.*, vol. PAMI-7, pp. 187-202, March 1985.

[14] A.A.G. Requicha and H.B. Voelcker, "Solid Modeling: A Historical Summary and Contemporary Assessment," *IEEE Comput. Graphics Applications*, pp. 9-24, March 1982.

[15] L. March and P. Steadman, *The Geometry of Environment*, Cambridge, MA: MIT Press, 1974.

[16] C.L. Jackins and S.L. Tanimato, "Oct-trees and Their Use in Representing Three-dimensional Objects," *Comput. Graphics Image Processing*, vol. 14, pp. 249-270, Nov. 1980.

[17] J.L. Bentley, "Multidimensional Search Trees Used for Associative Searching," *Commun. Assoc. Comput. Mach.*, vol. 18, pp. 509-517, September 1975.

[18] E.P. Krotkov, "Thesis Proposal: Active Visual Perception for Determining Spatial Layout," Thesis, Dept. Computer and Information Science, Univ. of Penn., Spring 1987.

[19] R. Bajcsy and F. Solina, "Three-dimensional Object Representation Revisited," Tech. Report MS-CIS-87-19, Dept. Computer and Information Science, Univ. of Penn., March 1987.

[20] L.G. Shapiro, "A Structural Model of Shape," *IEEE Trans. Pattern Anal. and Machine Intell.*, vol. PAMI-2, pp. 111-126, March 1980.

[21] J. Martin, "Computer Data-Base Organization," Series in Automatic Computation, Prentice-Hall, 1975.

[22] A.A.G. Requicha and H.B. Voelcker, "Constructive Solid Geometry," Tech. Memo 25, Production Automation Project, Univ. Rochester, Rochester, N.Y., November 1977.

C-2