VEGA
A Geometric Modelling System

by

Robert F. Woodbury

EDRC-48-03-87

April 1986

VEGA
A GEOMETRIC MODELLING SYSTEM

Robert F. Woodbury
Carnegie-Mellon University
4 April 1986

ABSTRACT: VEGA is a program which models rigid solid objects in three dimensions. Specifically, its domain is assemblies of planar faced polyhedra. VEGA supports a variety of operations to create, modify, query and delete these assemblies.

VEGA is intended to serve two purposes, that of a new medium of representation for the design process and of a programming package to support geometric applications in a wide variety of domains. Here we address primarily the first of these purposes, that of a new medium for design.

Designers of physical objects use an external medium, traditionally paper or physical models, not only to record their work, but to provide information which assists in the understanding of implications of design decisions. Designers proceed by performing operations, which reflect internal design decisions, on this external medium.

The operations used in design are generally reflective of these physical media. For example, models built of clay tend to formed by a subtractive processes, whereas models built of wood tend to be additive in nature. Designers who use drawings as their medium still tend to use operations which reflect operations on physical models.

Computers provide the fascinating potential to provide a much wider variety of operations at a much greater speed than is available with the traditional means of representation. In addition, a computer based representation can provide quantitative information not easily accessible from traditional forms. This opens the potential for the inclusion of formal means of evaluation in the design process; something which is generally almost absent in traditional design teaching. A computer program which effectively and "naturally" models physical objects and operations on them would be a valuable assistance to both the teaching and practice of design.

VEGA has been designed with these objectives in mind. It is currently undergoing redesign and additional implementation at the Center for Arts and Technology at Carnegie-Mellon University. VEGA represents physical objects with a scheme known as boundary representation and provides a wide variety of operations on these objects. VEGA also provides means to associate other, non-geometric, information with the objects it represents. VEGA is implemented under the

Table of Contents

List of Figures

ANDREW system. It communicates to ANDREW through a graphics package, also developed by our group. VEGA is intended to serve as a medium for ftiture studio courses in the Architecture, Industrial Design and Arts education. We have recently embarked on a project to make VEGA truly useful for these purposes.

KEYWORDS: assemblies, interaction, modelling, polyhedra.

This paper contains excerpts from VEGA: A Geometric Modelling System, Graphics Interface 83, Edmonton, Alberta.

1 Introduction

ITic typical architecture student, indeed any student whose task it is to design an object, spends a great deal of time and effort in the construction of physical models and in the creation of drawings which are record their work. These models and drawings serve two purposes, the first and most important as an ongoing medium for development of a design and the second as a record of the final designed artifact. Both of these functions are necessary for design work to proceed and to be presented. However, as means to achieve these ends, manual media such as physical models and drawings have very significant drawbacks in their veracity and in the speed with which they can be created.

Computer modelling of the geometry of objects provides an alternative representation to drawings and models. The basic idea is to create an internal representation which is in one-to-one correspondence with the real world object that is represented. Several significant issues arise in the creation of such a representation.

2 Issues for Geometric Modelling

2.1 Definition of an Assembly

It is a rare occurrence in architectural design to be concerned with a single object. Almost every architectural solution is a composite of parts, related together into an assembly.

An assembly is a collection of parts. When viewed from a designer's point of view these parts are complexly interdependent. The conceptual forces that determine a shape are many: location relative to other shapes, size dependent upon geometric constraints, functional requirements(strength, connectivity, mass) machining technology and materials standards to mention a few. Eastman [Eastman 80] makes a distinction between two types of dependencies; internal and external dependencies. He also states that the recognition of external dependencies is a necessary step to the creation of geometric modelling systems that strongly support the definition of assemblies.

One type of external dependency is relative location. An assembly of parts, for example a precast concrete panel, is composed of many pieces which maintain constant spatial relationships to each other independent of the location of the overall assembly. Some of these pieces may in turn be subassemblies themselves consisting of many pieces.

A modelling system for assemblies should support the groupings of parts into an assembly. An assembly may be defined as being composed of a part or of assemblies of parts. It is no coincidence that this definition of an assembly is similar to the definition of a tree. The data structure which represents an assembly of parts is a tree of relative locations. A part may have a parent, another part to which it is always spatially associated. Similarly a part may have children, which are other parts which are dependent for their spatial location on the location of the parent part

2.2 Definition of **a Part**

The part is the basic unit in an assembly. In the real world a part is usually associated with a physical entity which has a boundary between inside space and and outside space. This physical entity is called a solid or polyhedron. Any component of a boundary, for example a face, edge or vertex is located relative to some datum that is used to define the geometry of the boundary. This datum can be thought of as a coordinate system for the boundary of the part The existence of this coordinate system allows the specification of a boundary independent of the spatial location of the part itself.

A part may have numerous other properties which may be described in non-spatial ways. These properties or attributes of the part do not generally depend explicitly on a description of the boundary of the part.

A part must also have a location in space. For example, it would be meaningless to say that a screw existed without the concept of its existing somewhere. The location of a part is given by specifying a coordinate system somewhere in space at which the part can be defined in its own coordinate system.

These three properties of a part, a spatial boundary, non-geometric attributes and spatial location together define a part. Any model of a part must include capabilities for representing and manipulating these three types of properties.

2.3 Definition of Operations

The process of drawing or of building a model can be viewed as a process of performing operations on a set of objects. In drawing the objects are marks on paper and the operations involve placing or erasing various types of marks. In building models the objects are the raw components of the model, for example, clay, wood, chipboard, paper or plastic, and the operations are physical transformations, such as cutting, molding or glueing, on those objects. Computer models are not subject to the same physical constraints in their operations as are drawings and physical models. The potential for operation types is limited only by the imagination and mathematical skill of the implementor of an operation. A solid modelling system should support a wide variety of operations on the objects that it models. The limitations of drawings and models notwithstanding, one very valuable source for ideas concerning the operations that should be provided in modelling systems is the operations that may performed on these physical media. We have identified a set of operations that should provide us with a rich modelling environment for representation of architectural objects. These include: the spatial set operations, sweep operators, modelling operators for sites, parametric variations and a weak, but general modelling operator as a means of last resort

3 The VEGA Geometric Modeller

3.1 Solids Model

VEGA is based upon models of rigid, solid objects, or "solids". The usefulness of solids modelling for the representation of geometric components is solidly established in computer-aided design. There are many strong reasons to use a solids model. These include:

- The existence of only one data type for all solid geometric artifacts.
- The maintenance of well-formedness conditions.
- The completeness of information necessary for the calculation of all geometric properties of a polyhedron.

A solids model representation may be used as the general representation for the generation of views into any design system.

There are several types of representations for solids models. Requicha [Requicha 81] a has outlined the following different types: pure primitive instances, spatial enumerations, cell decompositions, simple sweep representations, boundary representations, and constructive solid geometry trees.

Of these, the boundary model, specifically the winged-edge data structure [Baumgart 72] [Braid 75] and its variants evolved at Carnegie-Mellon by Eastman et al. [Eastman 79] has served as the primary representation scheme for solids modelling in our lab. The advantages of the the boundary representation scheme for architectural use include the readiness of display of parts for interactive viewing, the ability to use a large number of primitive shapes without a concomitant large extension of code, and the ability to directly access components of a shape's boundary.

3.1.1 The Winged Edge Data **Structure**

The winged-edge data structure represents a solid by defining its boundary as a network of bodies, faces, rings, edges and vertices. [Eastman 79] [Braid 75] Each entity in the data structure carries two types of information; information describing the connectivity of entities to other entities of the same or of different types and information describing the location of the entity in euclidean space. These two types have come to be known respectively as the topology and geometry of a shape. [Eastman 79]

3.1.2 Euler Operators

A particular instantiation of the winged edge data structure as a solid or polyhedron is created by use of a set of operators known collectively as the euler operators. These operators guarantee that the resulting graph partially conforms to the well-formedness conditions of non-self-intersection, closure and orientability. [Eastman 79]

3.1.3 Definition Operators

The euler operators are used to create other higher-level operators which define shapes. These operators create the primitive shapes which the system uses. The primitive shapes currently supported by the modeller are cube, cuboid, cone, frustum, cylinder and extrusion. A new primitive shape can be created by using the euler operators to write a function which returns the required shape. Primitive shapes can be defined parametrically. Figure 1 gives an example of an extruded primitive shape.

Two shapes may be combined to form a resultant. There are four shapes possible from a combination of two shapes. These are the shapes resulting from union, intersection and the two difference operators. Figure 2 gives an example of shape operations on two polyhedra.

Shapes may be modelled. They may be scaled in each of the principal euclidean axes or rotated about the axes. Translation of a shape by changing its local origin is also supported. Figure 3 gives an example of scaling of a shape along one axis.

A set of modelling operators for sites (landscapes) provides the ability to define a site through a number of input points. These points are used to provide a spatial net which approximates the surface of the site.

A single shape may be used many times in an assembly of parts. An instance which defines a part may point to a shape which is already "being used" by other instances. When a shape which is used by multiple instances is changed, all of those instances automatically change. This allows a designer to quickly modify all parts which refer to a given shape by modifying only one shape. Since no canonical form exists for a boundary representation the use of a single shape by many instances provides the only means to maintain a notion of identity between parts. Use of a single shape representation for multiple instances of the same shape also results in a significant reduction of the storage used for a model of given complexity.

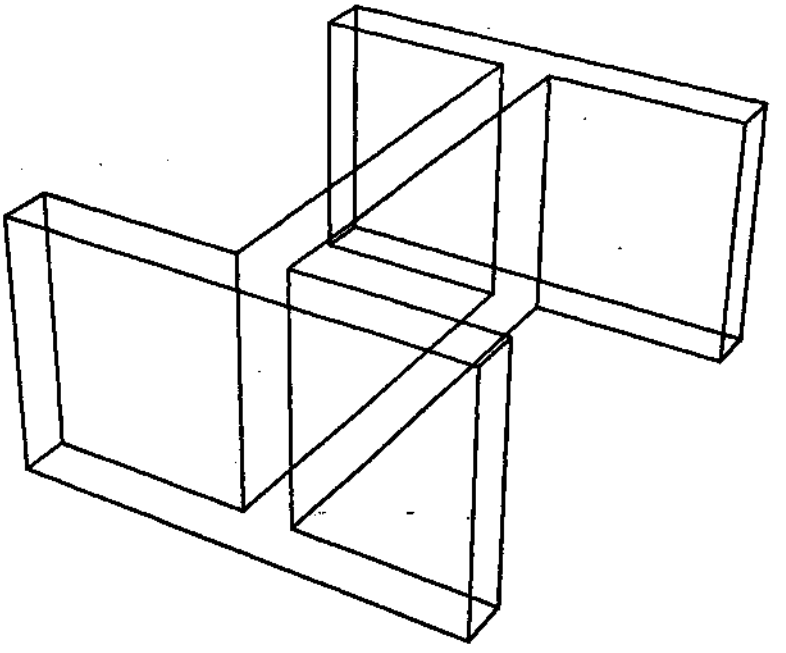

Figure 1: Example of an extruded primitive shape

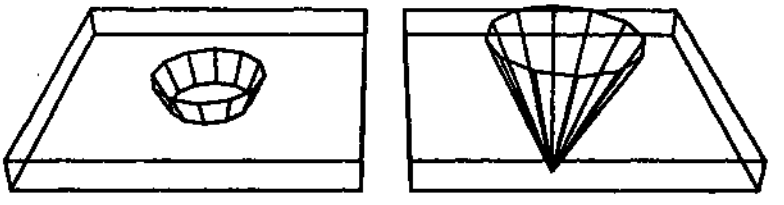

Figure 2: Example of a difference operator on a cone and a cuboid

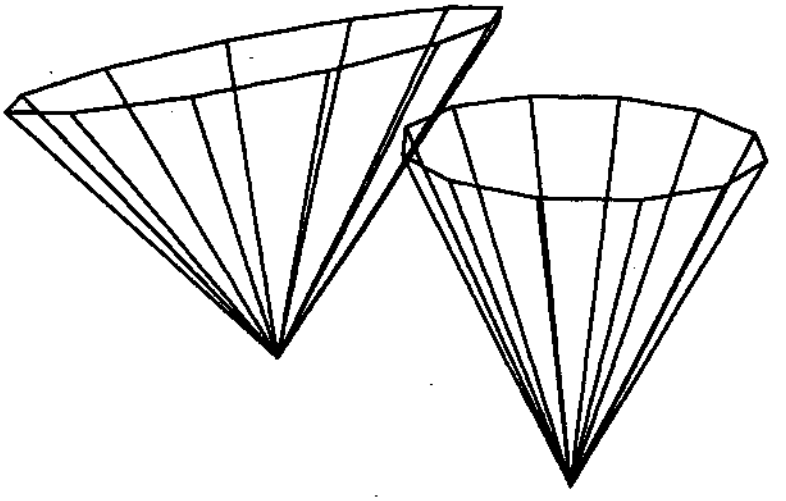

Figure 3: Example of a shape scaled along a single axis

3.2 Non-geometric Attributes

A part is not fully described by its location and physical boundary. Other information must be provided to provide complete information. Much of this information can be described textually. Currently, non-geometric information is stored as pairs of text strings referring to the name and value of a part property respectively. Multiple name-value pairs may be used by a single part or instance. In this case, they are grouped together into lists of attributes. An instance may point to one list of attributes. As with shapes multiple instances may point to a single attribute list

3.3 The Location Property

The location of any part in an assembly is determined by specifying a transformation relative to some other location. The resulting tree of transformations is called an instance tree or a location graph [Eastman 80]. In each node of the tree there resides a transformation relative to the parent of the node. For the sake of computational convenience, there is also a transformation which locates the node in global space, or the space defined by the root of the instance tree. The operators on the instance tree automatically maintain data integrity between the two transformations.

Each node in the instance tree may have associated with it two types of properties, a shape and non-geometric attributes. Either of these properties may be a null property. The composite of the three components, an instance or node in the instance tree, a shape and a collection of non-geometric attributes provides enough information for a complete description of some physical part

A set of operators is used to create, query and modify the instance tree representation. These operators ensure the well-formedness of the instance tree.

4 Interactive Modelling in VEGA

VEGA is presented to the student user as an interactive program which communicates through a computer screen, a pointing device and a keyboard. The fact that the models in VEGA are in three dimensions and the display screen and input device are for practical purposes limited to two dimensions creates the most difficult issues in the design of an effective and intuitive interface. Another problem which occurs lies in the large number of operators and which must exist to make a modelling system useful (there are over 110 operators in VEGA). The complexity of operators creates a third problem in that operators which are long and complex can involve the user in a mode of operation which is difficult to remember.

Our response to the first issue, that of three dimensions in a two dimensional world, has been twofold: to reduce the three dimensional world to two dimensions whenever a location changing operator is to be performed and to provide orienting three dimensional "marks" which emphasize certain objects or processes. An example of dimensional reduction occurs when the user wishes to move or to rotate an object: the specification of that operation occurs in some plane taken through the space of the object. An example of the use of orienting marks is the use of coordinate systems of various sizes used to identify arguments for operations.

We responded to the second and third problems, those of the large number of necessary operations and of the complexity of operations through the use of modeless interaction techniques as proposed by the implementors of Smalltalk80 [TESLER 81]. The idea of modeless interaction is that a user should never be put into a state where a sequence of actions are required before that state may be successfully exited. Thus we use a reverse polish notation for our operators. Prospective operands are placed on a stack* which is then used

by all operations as the source of operands. If sufficient operands of appropriate type are on the stack, then a requested operation will be performed otherwise nothing will occur. In geometric modelling there are a large number of primitive data types for operations, including numbers, strings, solid objects and assemblies. This led us to use a number of different operand stacks. The adoption of a modeless form of interaction allows a very simple and concise input syntax, making it easy to deal consistently with a large set of operators. Similarly the stack discipline allows a user to prepare each operation completely before committing to it, thus reducing requirements for long and complex operators.

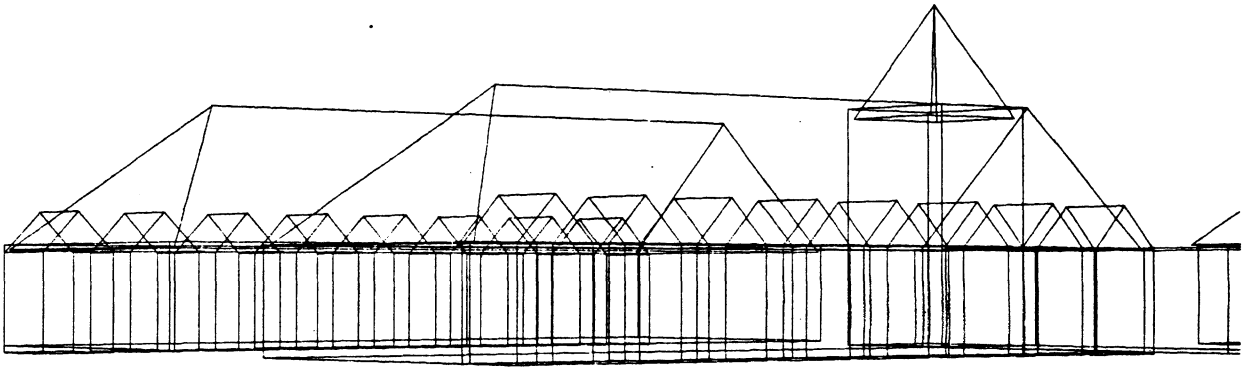

Figure 4: A Simple Building Model

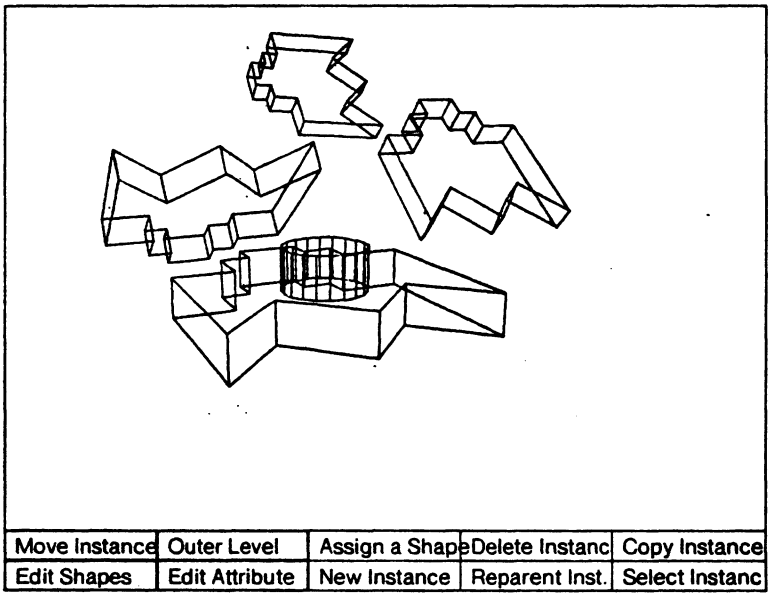

Figure 5: An Example of the Menu and Display

5 Current Uses

VEGA is currently being used in a computer modelling course in the Department of Architecture and for graduate student projects. The purpose of the modelling course is to introduce students to the use of various modelling tools as media for representation in architecture. The course requires a minimal background in computing, particularly an understanding of the concepts of algorithm and data structure. Typical assignments include the demonstration of cubic symmetry, wood framing layout and site planning studies. Students are also asked to prepare a critique of the user interface of the VEGA. Graduate students are using

VEGA to write specialized geometric modelling systems. These systems attempt to imbed the semantics of specific disciplines or design activities into a geometric modelling system. The models produced by one special system are data compatible with the VEGA system as a whole. To date, several such projects have been completed; a robot arm manipulation package, (see Figure 6), a drawing based modeller [McKelvey 84] **and** a site manipulation package (see Figure 7)^;

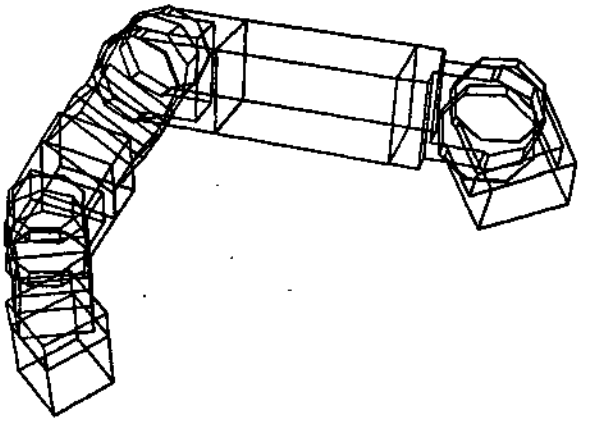

Figure 6: Robot Arm **Manipulator**

6 Work in Progress

In the Fall of 1985 a project to fully develop VEGA for instructional use was initiated. Over the **next two** years we intend to complete a number of additions to the VEGA modeller and to fully implement it **in the** ANDREW system and the **C programming language.**

6.1 Graphics

The current graphics package in which VEGA is implemented is not sufficient for the complexity of **objects** which we hope students will eventually model. We are rewriting the graphics package programming interface to conform with the draft 3D-GKS standard in anticipation of ANDREW having a first class **3D-GKS** implementation in the near fliture. We are also making incremental changes to the our local graphics package to improve efficiency for specific **tasks**

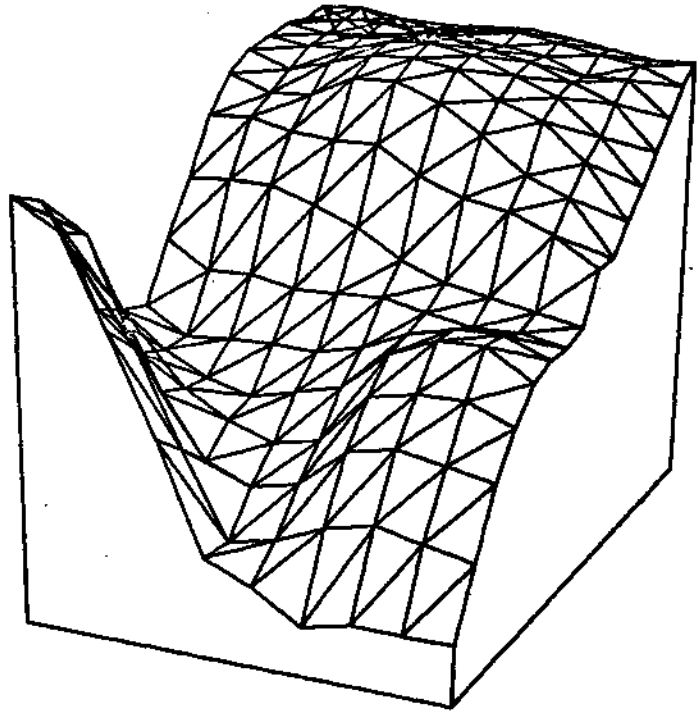

Figure 7: A Site Representation

6.2 Code Modifications

We are making a number of changes to the code which will not affect the basic functionality, but which serve to make the code more robust and efficient Primary amongst these changes is a rewriting of the spatial set operators. In the process of making these modifications, VEGA sill be translated into the C programming language.

6.3 User **Interface**

We have begun design of a user interface which takes advantage of the interaction capabilities of ANDREW. In particular, we intend to use multiple windows for increasing the visibility of VEGA models and layouts for organizing stack interactions and for forms based interaction.

The effective graphics display of completed VEGA models is important if these models are to be used for presentations in architectural design studios. We currently use commercially available graphics software for this purpose, a practice which we hope to continue. The main issue here is political, not technological; we need to find a ways to get high quality graphics rendering software ported to the ANDREW system.

Finally, we are experimenting with gesture based input of three dimensional models through the use advanced sensing technology.

6.4 Site Modelling

An important component of architectural modelling is the modelling of site information. We are building a modeller which will interpret either contour lines or point elevations and convert them to the solid modelling representation used inside of VEGA. Interaction with site information may then occur in three ways, through contours, points or by directly manipulating the resulting solids model.

6.5 Polyhedra Definitions

The capabilities of the VEGA to define and alter shapes are now limited to a small number of primitive shapes, some simple modelling operations and the spatial set operators. Much more is possible. We intend to define a group of functions each of which produce a family of shapes. The first of these functions will be the generalized extrusion function, which will be capable of generating simple extrusions, pocket and face extrusions, extrusions with integral holes, shapes of rotation and swept shapes with variable dimension orthogonal to the sweep axis. Other types of special shapes include primitives for the platonic and archimedean solids, for general spatial connectors for linear elements and for parametric shapes. Parametric shapes give the ability to define and manipulate shapes which can be scaled locally. For example, any wide-flange beam can be modelled using the same topological model, which is then scaled appropriately to give a particular instance of a wide-flange beam.

7 A Postscript

VEGA is amongst other things a tool; a tool for building future educational applications for architecture. It is one of a set of tools that I consider basic to architectural educational software, as basic as window managers and editors are to other types of endeavours. Two other very necessary tools are powerful and fast graphics packages for two and three dimensional interactive graphics and spreadsheet building tools. Tools of this nature will provide great power to future developers of educational software and should be a priority for all of us.

References

[Baumgart 72] Baumgart, B.G. *Winged edge polyhedron representation.* Stanford Artificial Intelligence Report CS-320, Stanford University, October, 1972.

[Braid 75] Braid, I.C. The Synthesis of Solids Bounded by Many Faces. *Communication of the ACM,* April, 1975.

[Eastman 79] Eastman, Charles M. and Weiler, Kevin. *Geometric Modeling Using the Euler Operators.* Technical Report 78, Institute of Physical Planning, Carnegie-Mellon Univ., February, 1979.

[Eastman 80] Eastman, Charles M. *The Design Of Assemblies.* Technical Report 11, Institute Of Building Sciences, Carnegie-Mellon Univ., October, 1980.

[McKelvey 84] McKelvey, Roy, Woodbury, R.F. A Drawing Based Modeller. In *Graphics Interface 84.* Canadian Man-Machine Communication Society, May, 1984.

[Requicha 81] Requicha A.A.G., Voelcker H.B. An Introduction to Geometric Modelling and its Applications in Mechanical Design and Production. *Advances in Information Systems Science.* Plenum Publishing Corporation, 1981, pages 293-328, Chapter 5.

[TESLER 81] Tesler, L. The Smalltalk Environment. *BYTE* 6(8):90, August, 1981.