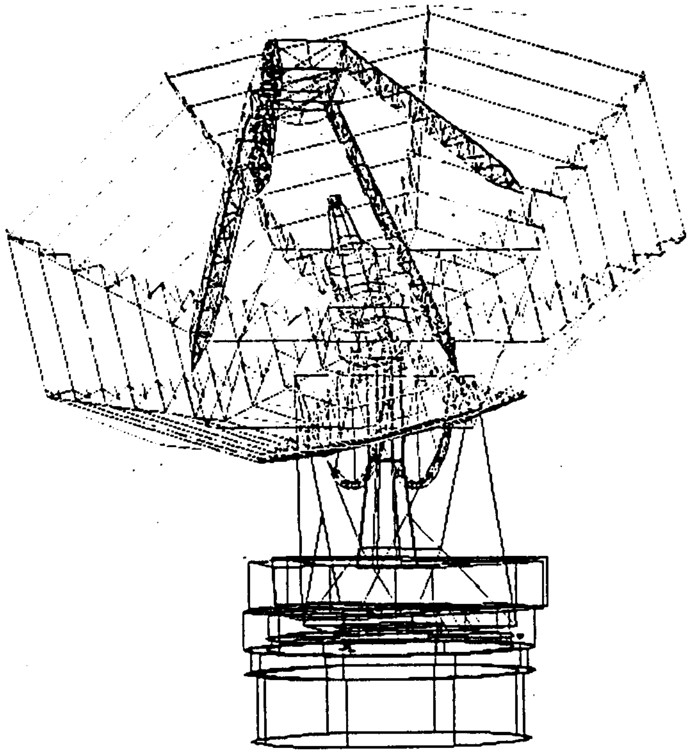
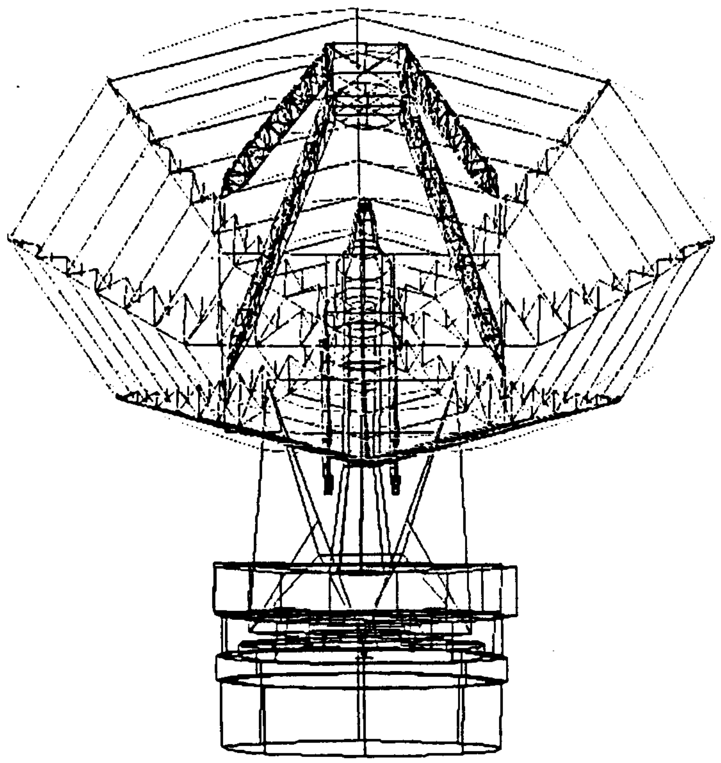
IAR - November 1986

RENSSELAER POLYTECHNIC INSTITUTE

CENTER FOR INTERACTIVE COMPUTER GRAPHICS





INDUSTRIAL ASSOCIATES REVIEW

NOVEMBER 6th & 7th 1986

OVERVIEW OF DATABASE PROJECTS

David L. Spooner

Center for Interactive Computer Graphics
Rensselaer Polytechnic Institute
Troy, New York 12180

OUR APPROACH

- We are exploring the use of entity and object oriented data modeling techniques for managing CAD data.

  – *entity oriented*: data is organized into hierarchical structures which are strongly typed.

  – *object oriented*: strict control over the operations that can be applied to a particular type of entity (possibly with inheritance of operations from related types).

- These data modeling techniques are appropriate for CAD data because they are based on the notions of creating and manipulating hierarchically structured objects and this is exactly what is done during the mechanical design process.

  – therefore, they provide the designer with a conceptually natural data structure

ADVANTAGES OF THIS APPROACH

- Objects are an appropriate paradigm for CAD
  - both deal with creating and manipulating objects
- Representation of data semantics
  - operational semantics is defined by the class operators
  - structural semantics can be defined within the operators
- Enforcement of data integrity
  - small well-defined set of operators for manipulating objects of a particular type
- System modularity
  - flexibility from the modular organization of the system around objects
- Inheritance of properties and operators
  - reduced programming time
  - reduced space requirements for the system
- Cleaner DBMS/programming language interface

CURRENT WORK

- Modeling System for Solids

  – study applicability of the object-oriented paradigm to data management for mechanical CAD

  – identify the requirements for an object-oriented data model for CAD

  – serve as an application driving the research data management system

- Distributed Object Manager

  – experimental system to move objects between different object databases

  – data translations is done with CDIL

  – explore distributed data management issues such as management of metadata, secury, integrity, and concurrency control

- ROSE - Relational Object (Entity) System for Engineering (M. Hardwick)

  – extended relational model similar to nested relations

  – extended relational algebra to deal with nested objects

  – interactive application development environment

MODELING SYSTEM PROJECT RESULTS

- The first version of the modeling system is complete.
  - implemented on a Tektronix 4044 AI workstation
  - implemented using Smalltalk-80
  - it includes boundary representation and CSG object classes
  - wireframe images of models are produced
  - low resolution monochrome graphics
  - tested for simple (small) models

- Future work will expand the system's capabilities.
  - move from Smalltalk to a version of C with objects
  - add assembly hierarchies and the concept of a part
  - explore more sophisticated graphics capabilities
  - explore interfaces to other applications such as finite-elements

- The modeling system will be used as an application for testing the lower levels of the data management system

- We will be continually re-evaluating the requirements fo: the data model

PRELIMINARY REQUIREMENTS FOR THE DATA MODEL

- We find the notion of object classes with instances useful for the data model (it is analogous to data types in traditional DBMSs).

- There must be strong typing throughout the data model to achieve all the potential advantages for integrity enforcement.
  - instance variables in Smalltalk do not have a type
  - parameters in messages do not have a declared type in Smalltalk

- The built-in *Collection* object classes in Smalltalk are very useful for modeling CAD data.
  - useful for association abstractions (aggregations of sets of subobjects)
  - need several type of collections such as Set. Bag. and OrderedCollection

- The notion of aggregation is weakly supported in Smalltalk and needs to be improved.

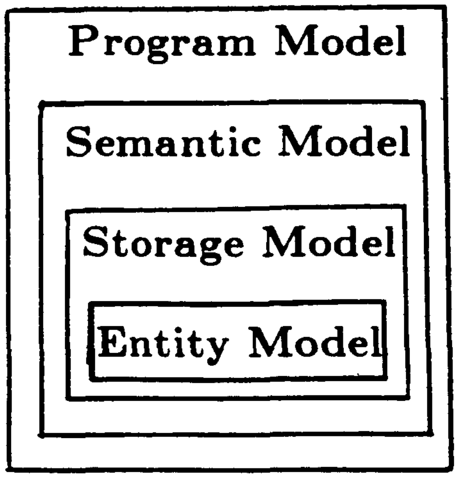
- Need cleaner interfaces to other. more traditional programming languages (C and FORTRAN).

- Much of the meta class mechanism in Smalltalk seems to be unnecessary for our purposes.

- Multiple inheritance would be convenient - though as yet has not proven to be absolutely required.

PRELIMINARY DATA MODEL

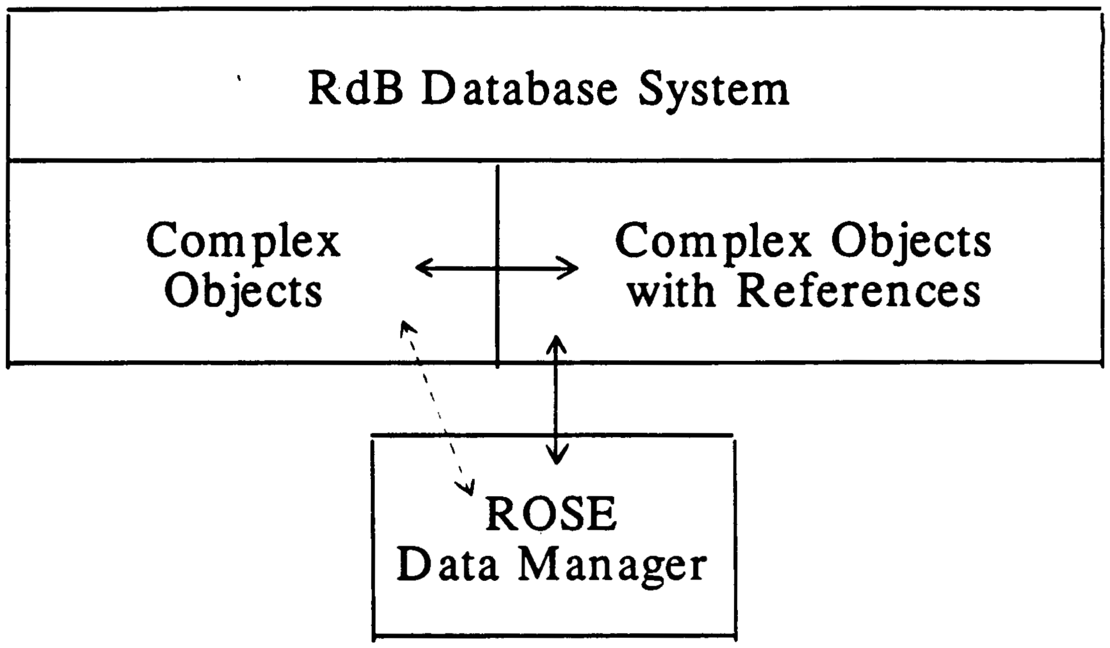
- The data model will be divided into four layers.



- The Entity Model is responsible for the basic data structures used to represent instances of object types.
  - based on AND/OR trees where AND nodes represent aggregation and OR nodes represent generalization
  - objects represented using constrained S-expressions
- The Storage Model is responsible for mapping objects to secondary storage.
- The Semantic Model is responsible for integrity enforcement.
  - this is where inheritance will be usefu'
- The Program Model maps between objects in the database and the data structures of a particular programming language.

DISTRIBUTED DATABASE - CURRENT WORK

- Building a prototype distributed object management system.
- Using CDIL (Common Data Interchange Language) to perform all necessary data translations.
- The current status of the project is shown below.



1. CDIL is now running on the VAX under VMS

2. Modifications have been made to CDIL to improve its usefulness

3. The object exchange paths indicated above have been implemented for fixed object schemas

DISTRIBUTED DATABASE - FUTURE WORK

- Generalize CDIL to eliminate the restriction to fixed object schemas and allow it to work automatically for a fixed data model
  - substantial modifications to CDIL
  - will be metadata driven
  - will be developed initially for the RdB environment
  - preliminary version available in December 1986
  - full version available in May 1987

- Expand the distributed system to include more nodes
  - finish the final link between RdB Complex Objects and ROSE
  - molecular objects using RdB segmented strings
  - include DATATRIEVE in system
  - include DBMS in system
  - include Britton-Lee IDM 500 database machine in system
  - two of these will be done by December 1986 and the last two by May 1987
  - in addition, the metadata driven version of CDIL will be migrated to these other systems by May 1987

NEW PROJECTS

- Virtual Persistent Object Manager (VPOM)

  – support data management in a workstation

  – support efficient interface between application programs and the object database

- Search Engine (Extended SQL)

  – need to be able to search an object database just like can search a relational database

  – need a high-level, easy to use query language to do this

  – should be upward compatible with SQL (an unofficial standard)

  – must be able to handle recursive data structures

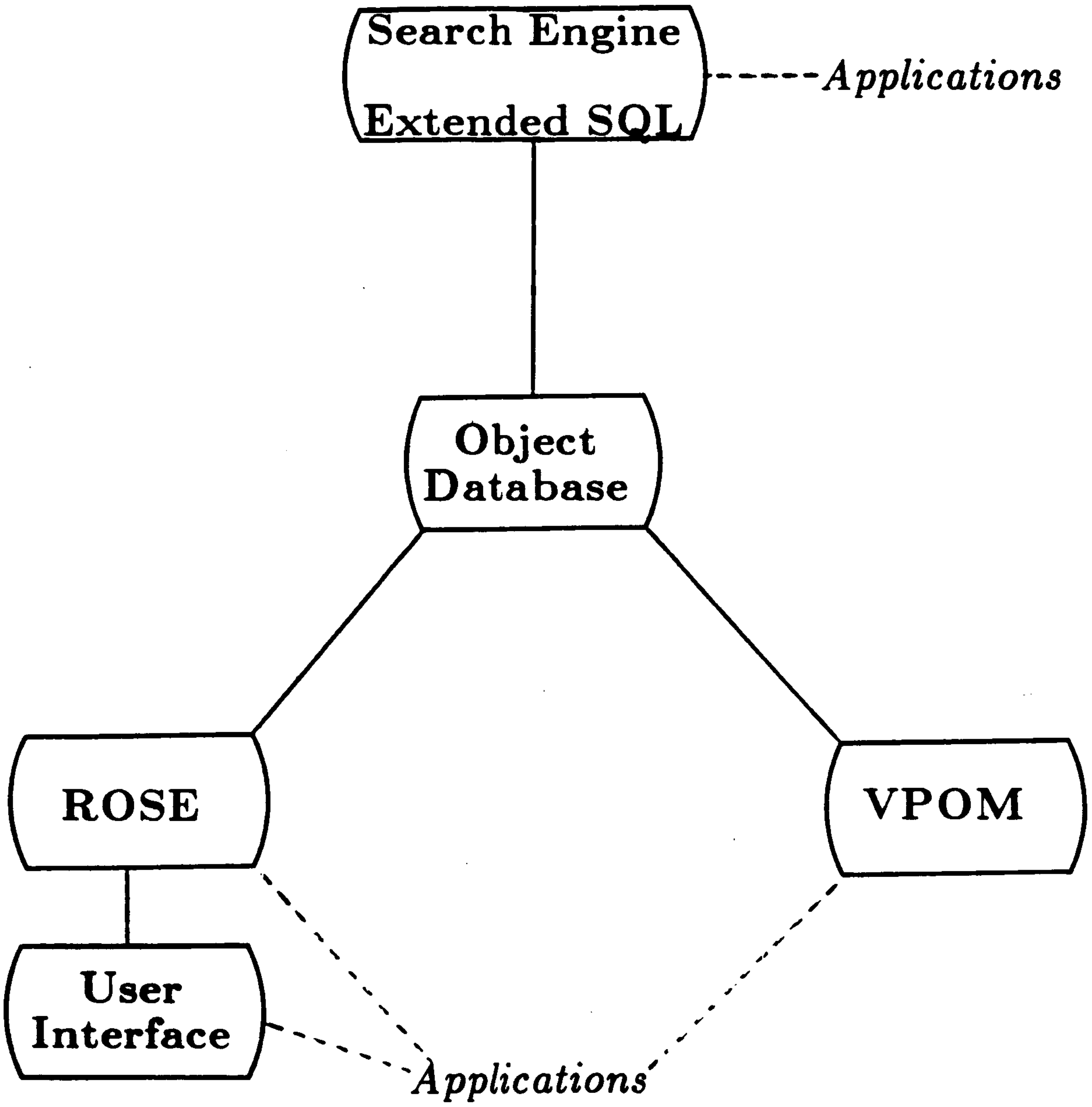
- Inspection Planning Workstation User Interface

  – develop a high-level object-oriented user interface for workstations

  – develop a generic set of tools for managing windows and menus

  – explore use of an object-oriented database in managing the user interface

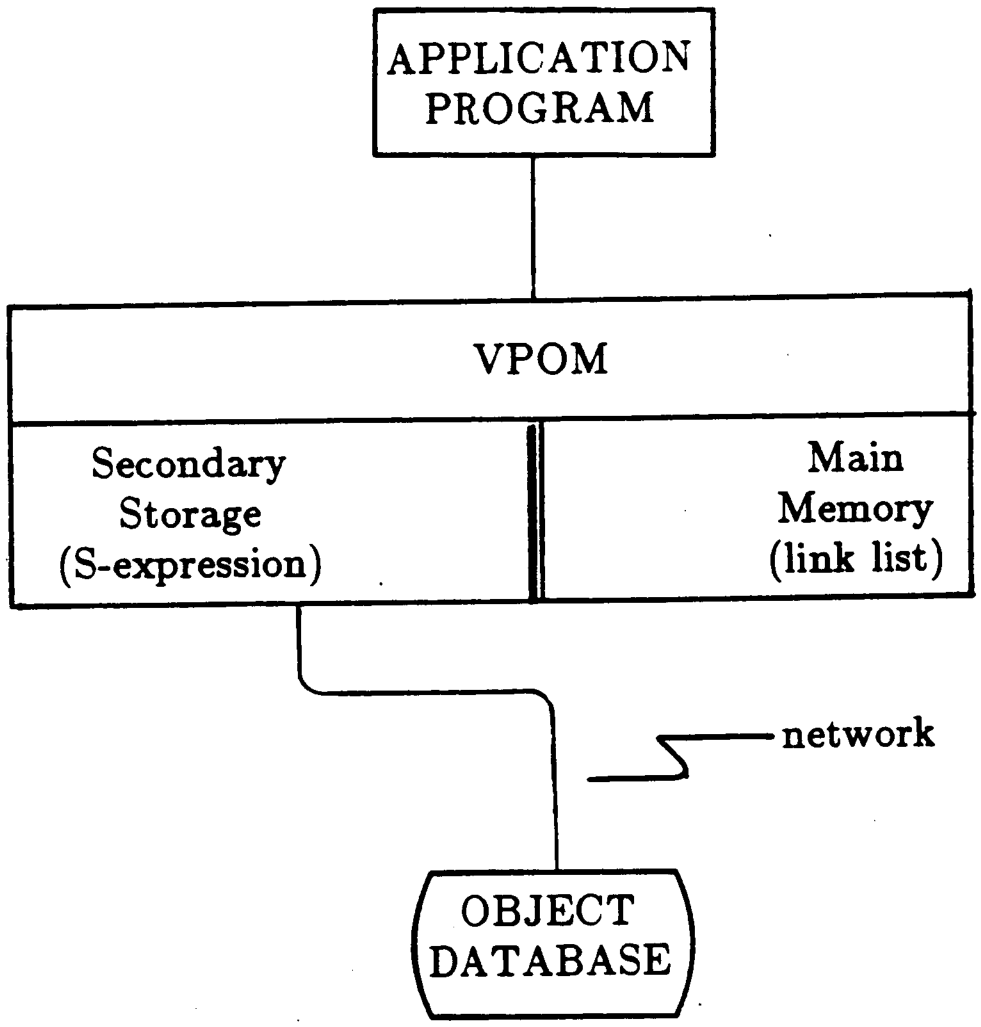
DATA MANAGEMENT FRAMEWORK



VIRTUAL PERSISTENT OBJECT MANAGER

- The VPOM is intended to manage data in a user workstation that is part of a large distributed CAD environment.
- It is intended for a scenario in which a designer copies data from a central database system into his workstation and then works primarily with the data in the workstation.
- It will allow persistence of data to be implemented for application programs. This capability is useful for CAD since the design process may span multiple sessions.
- The VPOM must act as an efficient interface to the Object Database from Application Programs (similar to a programming language interface to a traditional DBMS).
- The VPOM will be optimized for efficient retrieval of data while the Object Database system can be optimized for efficient searching.
- The VPOM is intended for applications written using an object-oriented language, but must work for other languages as well.
- The VPOM will use the four-layered data model based on constrained S-expressions described before (similar to ROSE).
- Concurrency control in the VPOM will be a difficult problem. Even if there is only one user on a workstation, he may have multiple processes in multiple windows.
- Current Status: A protptype has been implemented with an interface to the COOLE programming language (an object-oriented extension to C).

ARCHITECTURE OF THE VPOM



- The VPOM will support such commands as:

  – move objects between the Object Database and the VPOM

  – load/store objects between main memory and secondary storage in the VPOM

  – compose/decompose objects in main memory

  – move objects between the VPOM and Application Programs

EXTENDED SQL LANGUAGE

- Need a high-level language for specifying complex searches over the object database.

- SQL is such a language for relational databases.

  – SQL is an unofficial standard for relational databases

  – therefore our new language should be compatible with SQL to as large an extent as possible

- SQL deals only with relations and hence has a major weakness for CAD - it cannot deal with recursive data structures.

  – there are many examples of recursive data structures in CAD such as assembly hierarchies and CSG trees

- Therefore, an extended SQL is needed for CAD.

- The Extended SQL will be used both by ROSE and the VPOM to perform associative searches over the object database.

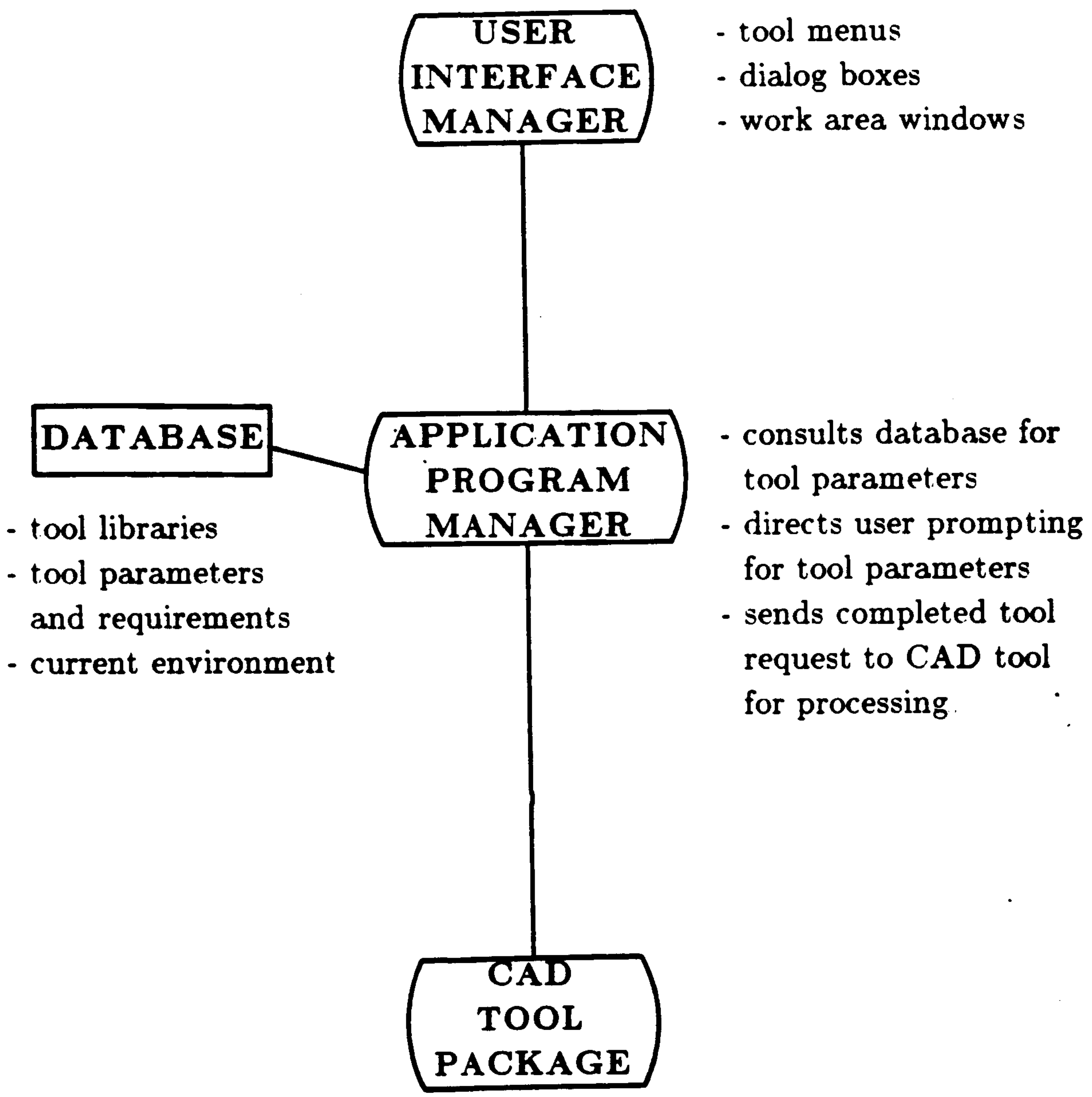
ISSUES TO BE RESOLVED

- Arguments over which Extended SQL operates will be deeply nested.
  - this includes recursive structures
  - Extended SQL will search these nested structures using extended projection operations
- Extended SQL must provide the capability to easily move from one object to another based on relationships between objects.
  - simple notation to do this
  - user must be made aware of such transitions when they occur (i.e., the boundary of objects must be clearly identified)
- Extended SQL must allow semantic functions to be defined and used in specifying searches over the object database.
  - application and data type specific
  - defined by the DBA and used by users of the database
- Initially, the language will strive more for usefulness than completeness.

INSPECTION WORKSTATION USER INTERFACE

- The interface consists of two major parts:
  - object-oriented tools for managing windows and menus
  - object database of commands and help information
- The window/menu tool package is implemented using C and DSM (Data Structure Manager - Rumbaugh GE CRD). The database is managed using ROSE (Hardwick - RPI)
  - generic set of tools
  - will be used as a high-level interface to IPDE
  - currently based on MicroVax GPX hardware running VMS and using UIS software for the low level graphics
  - provides a Macintosh style user interface for a MicroVax

USER INTERFACE ARCHITECTURE


USER INTERFACE MANAGER
- tool menus
- dialog boxes
- work area windows
DATABASE
- tool libraries
- tool parameters and requirements
- current environment
APPLICATION PROGRAM MANAGER
- consults database for tool parameters
- directs user prompting for tool parameters
- sends completed tool request to CAD tool for processing
CAD TOOL PACKAGE

LONG RANGE GOALS

- We will have the components for comprehensive and integrated data management environment for CAD.

  – Object database based on the data model under development

  – ROSE for interactive application development on workstations using the object database

  – VPOM for managing data in a workstation and serving as a programming language interface to the object database

  – Extended SQL query language for use with the object database

  – Distributed object management system

- Future work will integrate all these pieces into a distributed data management system for CAD.

- We will also continue to explore high-level database-driven user interfaces for CAD applications using these tools.

PROJECT OVERVIEW

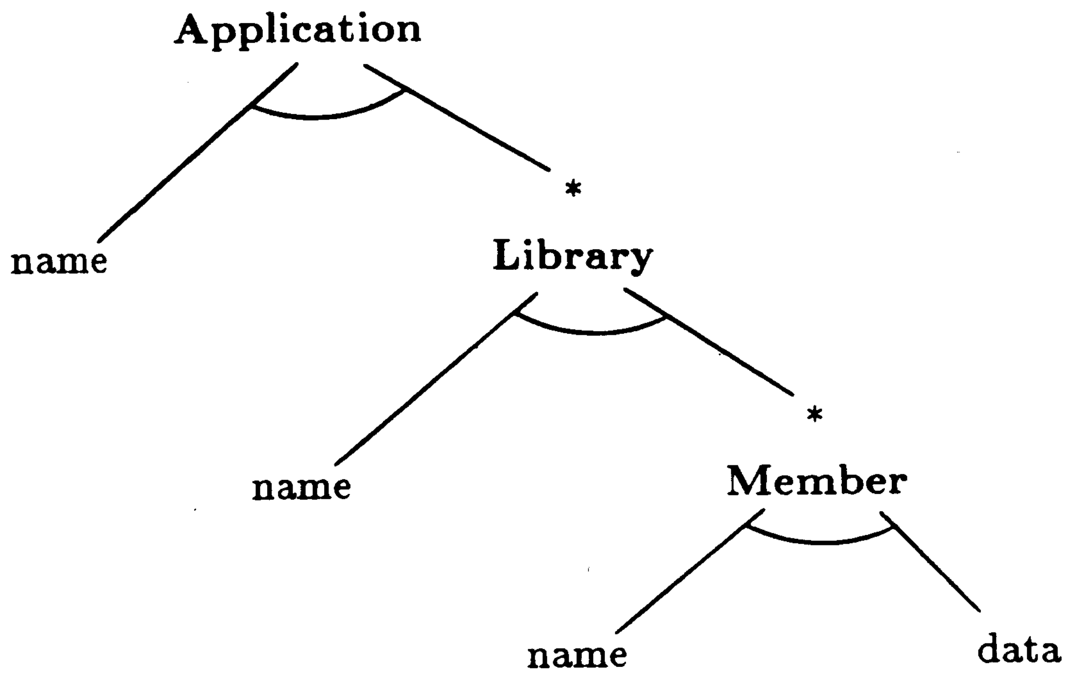
- The goals of the project are:
    - To study different approaches to modeling and storing engineering data in a relational database system.
    - To evaluate the performance of data storage and retrieval with a relational database system for engineering applications.

- The approach being taken in the project is to study the design and implementation of a general-purpose data repository using the SQL/DS relational database system.

- The design is based on an existing file system used to manage large quantities of data in engineering environments such as CAD.

LOGICAL DATABASE ORGANIZATION

- All data is logically stored in one or more **applications**.
- Within an **application**, all data is stored in one or more **libraries**.
- Within **libraries**, all data is stored in one or more **members**.



In addition, there are **composite libraries** for accessing data from several existing libraries together.

PROJECT SCHEDULE

- Implementation of the system is nearing completion.
- The next step is to generate a large database with both small and large members for the performance evaluation.
- The final step will measure system performance for various sizes of members and numbers of libraries and applications.
- It is expected that the project will be complete in early 1987.

List of Current Database and User Interface Projects

David L. Spooner
Martin Hardwick

Center for Interactive Computer Graphics
Rensselaer Polytechnic Institute
Troy, New York 12180-3590

1. Introduction

The database research group within the RPI Center for Interactive Computer Graphics consists of two faculty, Professors David L. Spooner and Martin Hardwick, one visiting faculty, Don House, one full time staff member and eight graduate students. This group is actively involved in the development of data models and database systems for CAD and development of high-level database-driven user interfaces for CAD tools. The approach being taken in most of these projects is to use entity and object oriented techniques for modeling CAD data. The projects address many of the problems in data management for mechanical CAD, including:

- data models for CAD objects,
- management of data in design workstations,
- distributed data management,
- performance of database systems for CAD, and
- user-interfaces to CAD tools and databases.

Each of the projects listed below addresses some specific aspect of the data management and user interface problem. Figure 1 shows the research framework that ties these projects together.

2. Object-Oriented Solid Modeling System and Data Model

A major database project last year was the development of an object-oriented solid modeling system. Work is continuing on this project. In particular, we intend to add assembly hierarchies to the system and to improve the graphics. We also intend to translate the system to another programming environment such as C++ or COOLE so that it can be more readily used as an application to test our object data manager software. Smalltalk has been an appropriate tool for development of the concepts. However, it is a self contained environment and hence inappropriate for continued development of the data management system.

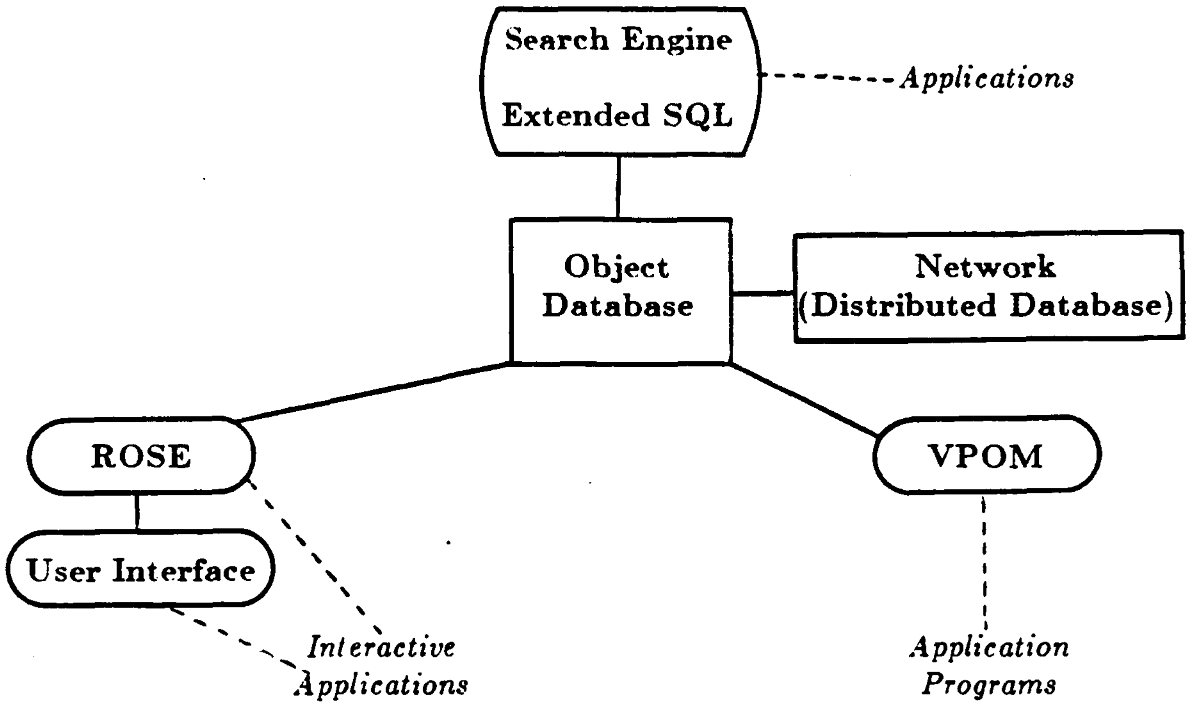


Figure 1: Database/User Interface Research Framework

The object-oriented solid modeler project identified several important requirements for an object-oriented data model for CAD. including:

1. The importance of an object paradigm based on the concept of object classes and class instances.

2. The need for strong support of both aggregation and generalization abstractions.

3. The need for strong type checking of instance variables in objects.

4. The usefulness of built-in *Collection* object classes.

From these requirements. a preliminary data model has been defined. This data model consists of four levels. The lowest level is the Entity Model responsible for the basic data structures used to represent instances of objects. The schema for the Entity Model is based on AND/OR trees with object instances represented using constrained S-expressions. The next level is the Storage Model responsible for mapping objects to secondary storage. The third level is the Semantic Model and is responsible for integrity enforcement in the data model. The highest level is the Program Model which maps between objects in the database and the data structures of a particular programming language.

3. A Workstation Application Development Environment

ROSE is an experimental database system for CAD/CAM applications. Its title is an acronym for the "Relational Object System for Engineering". As the title suggests, ROSE has been built by borrowing ideas from both Relational Database Systems and Object-Oriented Programming languages.

An analogy can be drawn between the role of ROSE in CAD/CAM applications and the role of a text editor in software engineering applications. ROSE allows CAD data to be edited, but unlike a normal editor it allows this to happen in a menu driven graphics environment.

ROSE is an unusual database system for many reasons. One reason is its "open architecture". A ROSE database stores objects such as integrated circuits and mechanical assemblies. These objects are large units of data. Therefore. ROSE stores them as files in conventional operating systems. Hence. many traditional database system services are implemented in ROSE using operating system services. For example, object security can be managed using file security, object version control can be managed using a code management system, and so on. ROSE is able to store its objects as files because they are large. Therefore. the overhead of using this organization is much less than it would be for a traditional database system where the equivalent organization would store every tuple as a file.

In effect. ROSE is a database system for design files. It does for design files what traditional database systems did for data processing. That is, it encapsulates the good ideas of design file systems and makes them available to the database programmer in an easy to use way. For example:

1. Design files represents data on multiple layers of detail.
2. Design files are identified using a name, not a key.
3. A design file is organized to suit its current application.

This last point is particularly important because the organization of a design file must change as a design moves through its life cycle. For example, when the design is first created, it must be possible to edit all of the data in the file for the design. But when a design file is used as a component in another design file, only the "interface" data of that file should be included. And when the design has been manufactured, the file should be archived in a way that ensures that its data will not change even though future versions of the design may may be created.

ROSE is a practical system as well as an experimental system. A version of ROSE have been implemented for SUN workstations under the UNIX operating system with a CORE graphics interface. Another version has been implemented on a MicroVAX II under the VMS operating system with an UIS graphics interface. New versions can be implemented very quickly on any system that gives good support to the C programming language. A graphics interface is desirable, but not necessary if a ROSE application does not need to use graphics.

4. A Workstation Data Manager for Application Programs

Another project is exploring the design and implementation of a workstation data manager for application programs called the Virtual Persistent Object Manager (VPOM). The VPOM is intended to manage data in a workstation that is part of a large distributed CAD environment. It is intended for a scenario in which a designer copies data into his workstation and works primarily with this data at his workstation. The VPOM must operate as an efficient interface to the object database from application programs and will be optimized for efficient retrieval of data. It is intended for applications written using an object-oriented language, but must work for other languages as well.

The VPOM will use the object-oriented data model discussed above. That is, the data model will be based on AND/OR trees. Instances of objects will be represented as constrained S-expressions. This will allow ROSE and the VPOM to operate from the same object database.

The VPOM will support commands to:

1. Move objects between the object database and the VPOM.
2. Compose/decompose objects in a main memory workspace.
3. Move objects between the VPOM and application programs.

A prototype implementation of the VPOM has begun. This prototype provides an interface to the COOLE programming language. A second prototype is also under way to provide an interface to the C programming language. When complete, these prototypes will be used as models for the design of the full VPOM. It is expected that the prototypes will be completed in December 1986 and the first version of the full VPOM will be finished in mid to late 1987.

5. A Query Language for an Object Database

Extended SQL is a query language for an entity or object oriented database. The Extended SQL language is needed to allow ROSE and the VPOM to perform associative searches over the object database and to extract objects from the database. The language can also be used directly by users. Since SQL is becoming an ANSI standard for relational query languages, it is desirable for a query language for an object database to be upward compatible with SQL.

Examples of the queries that can be expressed with the Extended SQL query language include:

- Find all the parts in an assembly.
- Compute the weight of an assembly from the weights of its parts.
- Find all of the assemblies that use a particular part.

Initially, the language will strive more for usefulness than completeness. Issues to be addressed in the development of the language include:

- Handling deeply nested and recursive object structures.
- Handling relationships between disjoint objects.
- Definition of semantic functions for specifying searches over the object database and implementing transitive closure types of operations.

The schedule for development of Extended SQL calls for the definition of the first version of the language to be completed by December 1986. The year 1987 will be devoted to completing a prototype implementation and testing it with the VPOM and ROSE. The prototype implementation is expected to be done in the early Fall of 1987.

6. Distributed Object Management

Another project is addressing issues of distributed object management in a mechanical CAD environment. Since objects represent a semantically meaningful clustering of data for application programs, they also represent a reasonable clustering of data for assignment to nodes of a distributed system and for data translation and exchange in a distributed system.

We will use CDIL (Common Data Interchange Language) to translate objects from one form to another. CDIL was developed in the RPI Center for Interactive Computer Graphics as a means to facilitate exchange of CAD data. Several modifications have proven necessary to enable CDIL to better handle exchange of objects between databases. The most important change is to make CDIL metadata driven so that object definitions can be modified and new objects defined without requiring major reprogramming of exchange routines. These modifications are expected to be completed in the Spring of 1987.

An experimental distributed system is being implemented to test the concepts. Currently this system has the ability to move assemblies defined as complex objects between two different RdB databases and ROSE. Future work will expand the number and types of databases included in the system. Completion of the modifications to CDIL discussed above will allow the system to more easily handle a wider variety of objects.

The long range goals of the project are to use the experimental system to study issues in distributed data management. This includes such things as: metadata management, concurrency control, integrity, security and other related issues.

7. User Interfaces to CAD Tools and Databases

A large project is looking at object-oriented. database-driven, graphical interfaces to CAD applications. An object database is used to store information about commands and tools provided by a particular CAD application system. A Macintosh-like user interface is used to allow the user to invoke these commands and tools. Menus are generated from the data in the database. Icons representing each tool are stored in the database for this purpose.

When an icon is selected, the database is consulted to determine what parameters are necessary. A dialog box is displayed to prompt the user to enter these parameters. This allows the user interface software and the database software to perform many types of integrity checking for the application system and greatly simplifies its use. Addition of new tools is also simplified and requires only adding the appropriate information to the database. Since menus are generated from the data in the database, the new tool will automatically appear in future menus.

Future work will address icon programming for the interface where a user can graphically string together icons for tools and commands to produce a flow chart for new high-level operations. Optimization of the icon programs is also an important consideration.

A related project is adapting these ideas to development of special-purpose, database-driven user interfaces. For example, a three dimensional graphics interface is being constructed for a solid modeling system. The modeling system has a command line interface to which we are adding the graphical interface. To be useful this interface must imitate some of the functionality of the solid modeling system in a non CPU-intensive way. For example, it must give a reasonable graphical representation of the object being constructed. Also, a goal is to allow connected objects so that if one object is moved all connected objects are moved with it.

8. Performance of a Relational Database System

A final project is addressing the performance of a relational database system. SQL/DS, for CAD applications. The database system is being used to implement the functionality of a file system that supports a solid modeling system. When complete, the database will be evaluated under a variety of conditions to determine its performance relative to the original special-purpose file system.

Data in the database is organized into applications. libraries and members. Each application contains libraries and each library contains members. The data for each member is stored in 32K byte segments, one segment per tuple.

The database implementation is expected to be complete by December, 1986, with the evaluation of the system taking place in early 1987.

9. Long Range Goals

Collectively, these projects provide the components for a comprehensive and integrated data management environment for CAD. Future work will integrate these pieces to build such a system. We will also continue to explore database-driven user interfaces for CAD applications.

AN APPLICATION PROGRAM OBJECT MANAGER

David L. Spooner
Martin Hardwick
Kelvin W. Liu

SUMMARY

A major database project last year was the development of an object-oriented solid modeling system. That system was used to develop a set of requirements for an object-oriented data model for CAD. From those requirements, a preliminary data model is being developed. This data model is intended for a workstation data manager also under development. This workstation data manager will support an efficient interface between application programs and an object database.

RESULTS

The object-oriented solid modeler project identified several important requirements for an object data model for CAD, including:

1. The importance of an object paradigm based on the concept of object classes and class instances.
2. The need for strong support of both aggregation and generalization abstractions.
3. The need for strong type checking of instance variables in objects.
4. The usefulness of built-in Collection object classes.

From these requirements, a preliminary data model has been defined. This data model consists of four levels. The lowest level is the Entity Model responsible for the basic data structures used to represent instances of objects. The schema for the Entity Model is based on AND/OR trees with object instances represented using constrained S-expressions. The next level is the Storage Model responsible for mapping objects to secondary storage. The third level is the Semantic Model and is responsible for integrity enforcement in the data model. The highest level is the Program Model which maps between objects in the database and the data structures of a particular programming language.

This data model is being used to design and implement a workstation data manager called the Virtual Persistent Object Manager (VPOM). The VPOM is intended to manage data in a workstation that is part of a large distributed CAD environment. It is intended for a scenario in which a designer copies data into his workstation and works primarily with this data at his workstation. The VPOM must operate as an efficient interface to the object database and will be optimized for efficient retrieval of data. It is intended for application programs written using an object-oriented language, but must work for other languages as well.

The VPOM will support commands to:

1. Move objects between the object database and the VPOM.
2. Load/store objects between main memory and secondary storage within the VPOM.
3. Compose/decompose objects in the main memory of the VPOM.
4. Move objects between the VPOM and application programs.

PROJECTED PLANS

A prototype implementation of the VPOM has begun. This prototype provides an interface to the COOLE programming language. A second prototype is also under way to provide an interface to the C programming language. When complete. these prototypes will be used as models for the design of the full VPOM. It is expected that the prototypes will be completed in December 1986 and the first version of the full VPOM will be finished in mid to late 1987. Future plans call for continued enhancements to the VPOM.

AN EXPERIMENTAL DISTRIBUTED OBJECT MANAGER

Michael Webb
Charudatta Rudrakshi
David Spooner

SUMMARY

In this project, an experimental distributed object manager for the support of heterogeneous CAD databases is being implemented. This system will be an experimental environment for the examination of distributed data management issues for heterogeneous CAD databases. The object manager is intended to support data exchange between object oriented databases. Certain object oriented concepts also appear to provide solutions to some of the issues of implementing heterogeneous distributed databases. In particular, objects appear to be a natural unit for data partitioning, location, exchange and translation.

CDIL (the Common Database Interface Language) is being used for data translation (TR-83001). CDIL is a language tool which is intended to facilitate data exchange between heterogeneous databases for CAD applications. CDIL consists of a precompiler to FORTRAN 77 and a run time support environment. Modifications and extensions are being made to CDIL to improve its ability to support the exchange of object structures.

RESULTS

Currently, there are two different RDB databases implementing recursive assembly/part hierarchies on the VAX system. The first database uses a standard implementation of complex objects. The other database also uses the complex objects data model, but allows shared object references. Objects can be exchanged between these two databases using CDIL. An interface also exists to allow objects to be transferred from either of these two databases into a file format which can be used by ROSE (see the other project summaries in this section). It is also possible to transfer data using the ROSE format into the shared object references RDB database.

Work is under way to complete the existing interface to the ROSE format, allowing the direct transfer of objects between the ROSE format and either of the RDB databases. This system is also to be extended to include other database systems. The next database to be included in this system will be a database using the molecular objects data model, and the segmented string features of RDB. Other systems which may be connected to this experimental system include the Britton Lee IDM 500 database machine, support for VAX files (possibly through RMS), DATATRIEVE, and DBMS.

Another area of current work is the implementation of routines based on a fixed data model, as opposed to being restricted to fixed objects structures. This will eliminate the need for reprogramming of the database dependent routines for minor changes in object structures, and allow these routines to be used for several different applications. Modifications and extensions are being made to CDIL to allow the inclusion of semantic information about the object structures being manipulated, and to support the use of a metadata database which will contain this information. The first such database dependent routines will support a generalized complex objects data model.

PROJECTED PLANS

In the future, further databases could be included in the distributed object manager system. It should be possible to include databases using other data models, or databases which are implemented on other computer systems which may be connected to the VAX system. Since the distributed object manager is intended to be an experimental environment, it will eventually be used for the examination of general issues in heterogeneous distributed databases for CAD. These issues include the management of metadata, data integrity and consistency, concurrency control, system recovery, data location and duplication, security, and the effects of different system architectures.

GRAPHICAL USER INTERFACE/DATABASE MANAGEMENT PROJECT

Shawn Javid
Blair Downie
Alok Mehta
David Spooner
Martin Hardwick

SUMMARY

This project involves the design and implementation of a graphical user interface and database system for a CAD application. The current application is text based, offering a wide variety of commands, with each command having several options. This produces a very general and flexible interface: the user, however, must remember the name and syntax of each command. The graphical user interface will assume most of this responsibility for the user. Also, extending the system (for example, by adding a new command) will become an easier task with the new interface.

The goals of the project are:

1. To accomodate a variety of levels of users.
2. To support textual and graphical script and procedure development (via iconic programming).
3. To support interactive experimentation (via iconic simulation).
4. To provide the user with an intelligent help environment.
5. To provide the user with immediate feedback if errors occur during the execution of a command or procedure. Ideally, this feedback should be graphical whenever possible. Also, the user should have the ability to modify certain parameters (without having to re-specify all of the parameters) and resubmit the command or procedure for execution when errors occur.
6. To be able to verify and optimize scripts of commands and procedures.
7. To simplify the task of installing new commands and procedures into the system.

The project has been divided into several phases:

1. The first phase consisted of identification of the characteristics necessary for the database system and user interface. This phase also involved evaluation of software tools available and selection of development environments for the user interface and database.
2. The second phase involved the implementation of generic graphical user interface tools. At the same time, the database storage scheme was defined and procedures were written to allow commands and icons to be entered into the database in a straightforward manner.

3. The third phase involved tailoring the user interface (using the generic tools developed in phase 2) to the CAD application and inserting sample commands into the database. This phase also involved combining the interface, database, and CAD application into one cohesive system.
4. In the fourth phase, advanced features such as iconic programming and intelligent help systems will be examined.

RESULTS

The object oriented paradigm was determined to be well suited for an application such as the user interface. Several environments were examined and an object oriented extension of C was chosen as the environment in which the user interface would be developed. The ROSE database management system (see other summaries in this section) was found to be very powerful and flexible, and thus was chosen for the base database.

Several graphics tools have been implemented including a hierarchically layered "pull-down" menu system and a "dialog box" system. A "dialog box" is a tool that allows a program to communicate with the user. Dialog boxes can be used for providing feedback to the user and for requesting input from the user. The advantages of using dialog boxes are: (1) they allow users to enter data using sophisticated input tools such as sliding bars for integers; (2) they guarantee that all data entered will be valid and in the specified range, thus freeing the program from having to parse input and check for errors; and (3) they give the user the ability to examine all inputs, enter inputs in any order, and modify inputs before submitting them to an application program.

The ROSE database system has been modified and extended to allow it to run on the vaxstation GPX workstation. The format for commands and icons in the database has been defined. Several commands and icons have been entered into the database; each command and icon is treated as a separate object. Work on the creation and representation of scripts and procedures in the database has begun.

Currently, phase 3 of the project is nearly complete. The user interface, the database, and the application have been implemented as separate processes. Communication protocols necessary to allow these three processes to interact as one cohesive system have been defined and are being implemented. This phase of the project is expected to be completed by December.

PROJECTED PLANS

The design and development of an iconic programming environment for this application will begin this spring. A sophisticated and intelligent help system will also be designed. The user interface and database manager will be extended to allow the user to textually and graphically create, save, modify and execute scripts and procedures. We will study issues in automated procedure checking and procedure optimization.

EXTENDED SQL QUERY LANGUAGE FOR CAD/CAM

David L. Spooner
Martin Hardwick

SUMMARY

An entity or object oriented database system must be able to search an object database using a high-level query language just like a relational database system searches relations. However, relational query languages are not sufficient for use with an object database since they are unable to handle deeply nested and recursive data structures. Since SQL is becoming an ANSI standard for relational query languages, it is desirable that a query language for an object database be upward compatible with SQL. Thus, there is a new database project under way in the Center to extend SQL for use with object databases.

RESULTS

The extended SQL project is designing a query language for an object-oriented database system to allow expression of complex searches over the objects in the database. This query language will allow queries such as the following to be expressed in a single query to the database:

1. Find all assemblies that use a particular part.
2. Compute the total weight of an assembly from the weights of the parts it contains.
3. Find all of the parts in an assembly that are supplied by a particular supplier.

This extended SQL query language (ESQL) will be used by both ROSE and the VPOM (see other project summaries in this section) to perform associative searches of the object database. In addition, it can be used directly by users.

ESQL must operate over arguments that are deeply nested and possibly recursive object structures. ESQL will use an extended projection function to search these arguments. Extended projection is a function that searches a hierarchical object for occurrences of a particular attribute and returns all occurrences of that attribute. It differs from the usual projection function of a relation database in that it "extends" the projection function to operate on all tuples nested within other tuples in the hierarchical description of an object.

ESQL must provide the capability to easily move from one object to another based on relationships between objects. A simple and clear notation is needed to do this. However, it must be clear to the user that such transitions have occurred so that boundaries of objects can be clearly identified.

ESQL must allow semantic functions to be defined and used in specifying searches over the database. These functions will be application and data type specific and are defined by the database administrator. These functions will also be used to implement common recursive queries and transitive closure types of operations.

Initially the language will strive more for usefulness than completeness.

PROJECTED PLANS

The syntax for ESQL is currently being designed. It is modeled after other extensions to SQL where appropriate. The design of the first version of the language is expected to be completed in December 1986. A prototype implementation of the language will begin in 1987. It is anticipated that this prototype implementation will be finished in late 1987. Future work will continue to enhance the language.

ROSE; AN EXPERIMENTAL DATABASE SYSTEM FOR CAD/CAM

Martin Hardwick
David Spooner
George Samaras
Blair Downie
Jay Hersh

SUMMARY

ROSE is an experimental database system for CAD/CAM applications. Its title is an acronym for the "Relational Object System for Engineering". As the title suggests, ROSE has been built by borrowing ideas from both relational database systems and object oriented programming languages.

An analogy can be drawn between the role of ROSE in CAD/CAM applications and the role of a text editor in software engineering applications. ROSE allows CAD data to be edited, but unlike a normal editor it allows this to happen in a menu driven, graphics environment.

ROSE is an unusual database system for many reasons. One reason is its "open architecture". A ROSE database stores objects such as integrated circuits and mechanical assemblies. These objects are large units of data. Therefore, ROSE stores them as files in conventional operating systems. Hence, many traditional database system services are implemented in ROSE using operating system services. For example, object security can be managed using file security, object version control can be managed using a code management system and so on. ROSE is able to store its objects as files because they are so large. Therefore, the overhead of using this organization is much less than it would be in a traditional database system where the equivalent organization would store every record as a file.

In effect, ROSE is a database system for design files. It does for design file systems what traditional database systems did for data processing file systems. That is, it encapsulates all of the good ideas of design file systems and makes them available to the database programmer in an easy to use way. For example:

1. Design files represents data on multiple layers of detail.
2. Design files are identified using a name, not a key.
3. A design file is organized to suit its current application.

The last point is particularly important because the organization of a design file must change as a design moves through its life cycle. For example, when the design is first created it must be possible to edit all of the data in that file. But when a file is used as a component in another file only the "interface" data of that file should be included in the new file. And when the design has been manufactured, the file should be archived in a way that ensures that its data will not change even though future versions of the design may change.

RESULTS AND PROJECTED PLANS

ROSE is a practical system as well as an experimental system. A version of ROSE has been implemented on SUN micro-computers under the UNIX operating system with a CORE graphics interface. Another version has been implemented on a Micro-VAX under the VMS operating system with an IUS graphics interface. New versions can be implemented very quickly on any system that gives good support to the C programming language. A graphics interface is desirable, but not necessary, if a ROSE application does not need to use graphics.

1. An Object Oriented User Interface for an Image Processing System. In this project ROSE is being used to store a database describing the commands available in an image processing system. For each command, it stores data describing a Macintosh style dialogue box that can be used to get the parameters of the command from the user. After the data has been fetched from the user, the database performs checks and then generates an image processing command that will perform an action for the user. In this project, the hope is to evolve the role of the database system to the point where it has complete knowledge of the current state of the image processing system. The database will then be able to drive an intelligent interaction with the user in which most data values are entered by pointing at icons instead of typing at the keyboard.
2. An Interface/Version Design Methodology. In CAD applications the characteristics of a design are frequently known before that design has been implemented. These requirements can be encapsulated as an "interface". In this project, we are experimenting with the design of a database where a component in an assembly can be instantiated as either an interface or a version. If a component is instantiated as a interface than it is called a "socket" and it must be "plugged" with a version at a later time. Currently, the experiments are being applied to an electrical database. In the future, the ideas will be applied to a mechanical database.
3. A Three-Dimensional Graphics Interface. This project invokes the implementation of a solid modeling system. The modeling system currently has a command line interface and sponsors are very interested in giving it one of the new style Macintosh interfaces. Hence, the goal of the project is very similar to that of the image prosessing project. However, to be intelligent, this interface must imitate some of the functions of the solid modeling package in a non-CPU intensive way. For example, it should give a reasonable graphical representation of the object being constructed. Also a goal is to allow objects to be connected so that if one object is moved all of the objects connected to that object will be moved.

PERFORMANCE STUDY OF A RELATIONAL DATABASE FOR CAD/CAM

Amit Pradhan
David Spooner

SUMMARY

This project involves studying the performance of a relational database for engineering and CAD/CAM data. The database design is based on a file system used to store data for a solid modeling system. The design is implemented in the form of a small database containing dummy data. Data storage and retrieval operations have been implemented and are performed on the dummy data using the SQL/DS relational database system. Other operations involving queries on the database will be implemented within the next month. The performance of data management operations will then be studied by using a realistic and larger database.

RESULTS

A relational database has been designed with relations defined for application names, libraries (within each application) and library attributes, and members (within each library) and member attributes. A small (dummy) database has been generated for testing of data storage and retrieval functions. Within a relation, member data are stored in sequentially ordered (32k byte) blocks. A program interface has been designed and implemented to allow a user's application program to store/retrieve data and query information from the database. At present, this interface consists of a complete implementation of member management functions, and a partial implementation of library and application management functions (as defined for the file system). All the implemented functions have been tested on the dummy database.

PROPOSED PLANS

The implementation of the remaining functions which interact with the database will be completed within the next month. Consequently, a relational database with a realistic amount of data will be generated and used to study the performance of the relational database version of the file system.

USER INTERFACES

Martin Hardwick
David Spooner
Don House
Blair Downie
Shawn Javid
Alok Mehta
George Samaras
Jay Hersh

OVERVIEW

- Our approach to user interfaces.

- Description of Applications.

OUR APPROACH

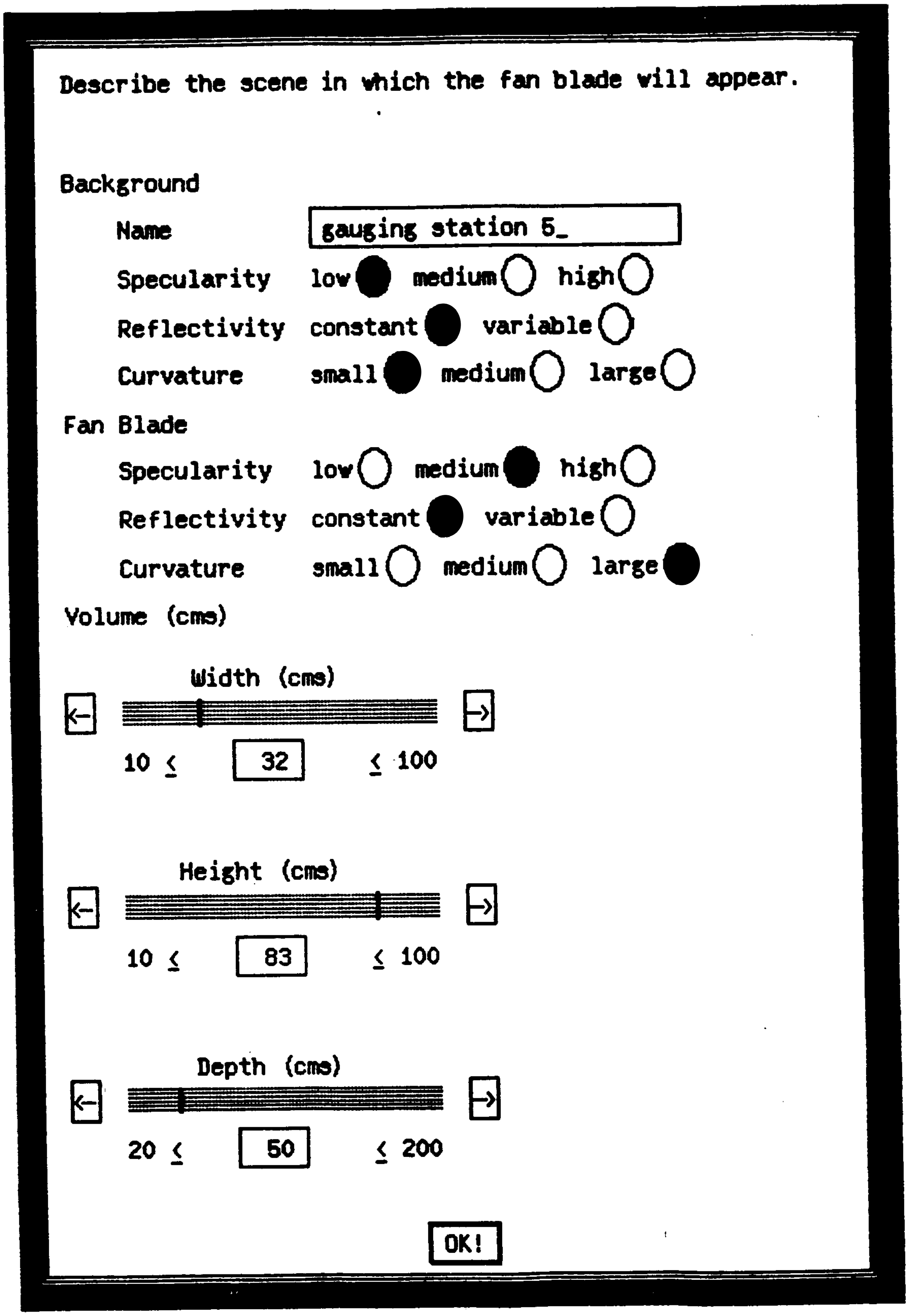
- Knowledge based.
- Database driven.
- Experimental.

USER INTERFACES

- ICONS, Pop-up menus, Dialogue boxes, Gadgets.
- Use a mouse not a keyboard.
- For example

  Do this
  On this
  And this

  Done in three hand movements.



Describe the scene in which the fan blade will appear.

Background

Name	gauging station 5_
Specularity	[✓ low] medium high
Reflectivity	[✓ constant] variable
Curvature	[✓ small] medium large

Fan Blade

Specularity	low [✓ medium] high
Reflectivity	[✓ constant] variable
Curvature	small medium [✓ large]

Volume (cms)

Width (cms)

10 ≤ 32 ≤ 100

Height (cms)

10 ≤ 83 ≤ 100

Depth (cms)

20 ≤ 50 ≤ 200

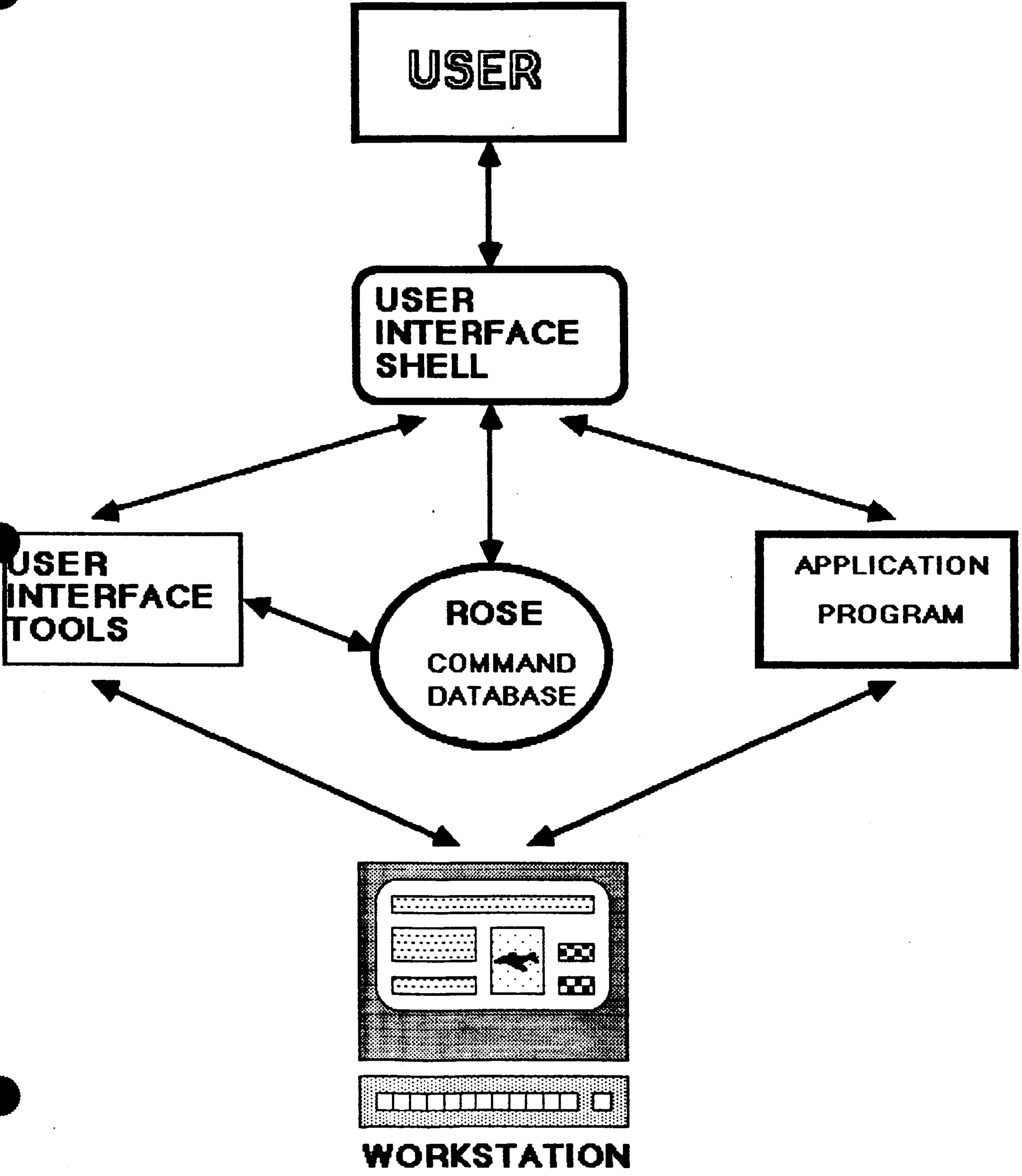
WHY DATABASE

- Need knowledge about what the user is doing and the state of the application.
- Will want to *edit* data until the right results are obtained.
- A database system for this type of application has been implemented.

ROSE

- Relational Object System for Engineering.
- Lets CAD entities be edited using an entity algebra.
- Can capture knowledge as "mappings".
- Used to implement menu driven, graphics applications.
- Designed to operate like a Design File database, not a Data Processing database.

ARCHITECTURE

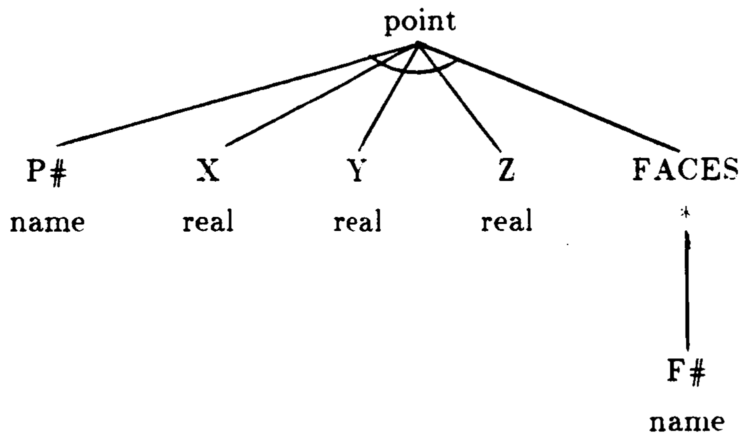


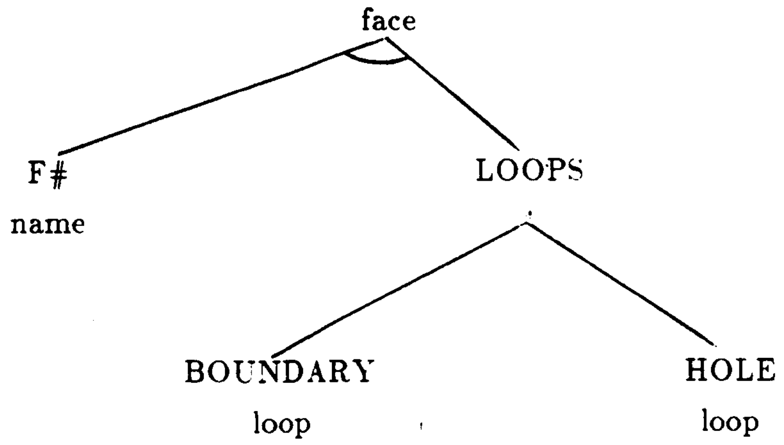
TECHNICAL CHARACTERISTICS

- Describes entities using DEEPLY nested relations.
- Defines entities using AND/OR attribute trees.
- Stores entities in entity sets.
- Manipulates entities using an entity algebra.

THE "STICKY" PART DATABASE

- Model solids as lists of connected faces.





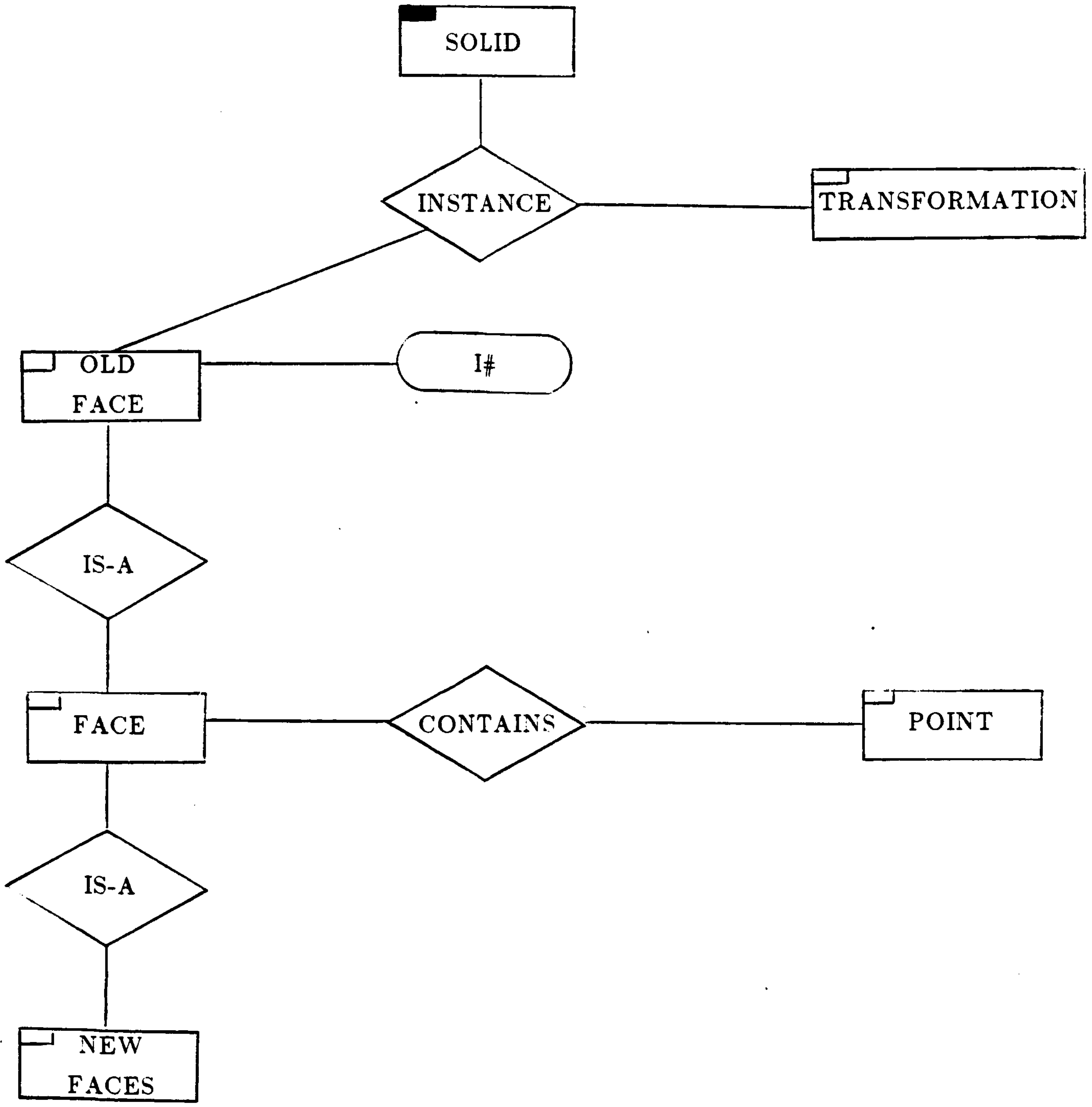
DATA INSTANCES

- Described as (ATTRIBUTE value).

```
(POINT    (P# p1)
          (X 1.0)
          (Y 1.0)
          (Z 1.0)
          (FACES    (F# f1)
                    (F# f2)
                    (F# f3)))
```

```
(FACE   (F# f1)
        (LOOPS    (BOUNDARY    (POINTS    (P# p1)
                                          (P# p2)
                                          (P# p3)
                                          (P# p4))
                               (CURVE line))))
```

THE STORAGE MODEL

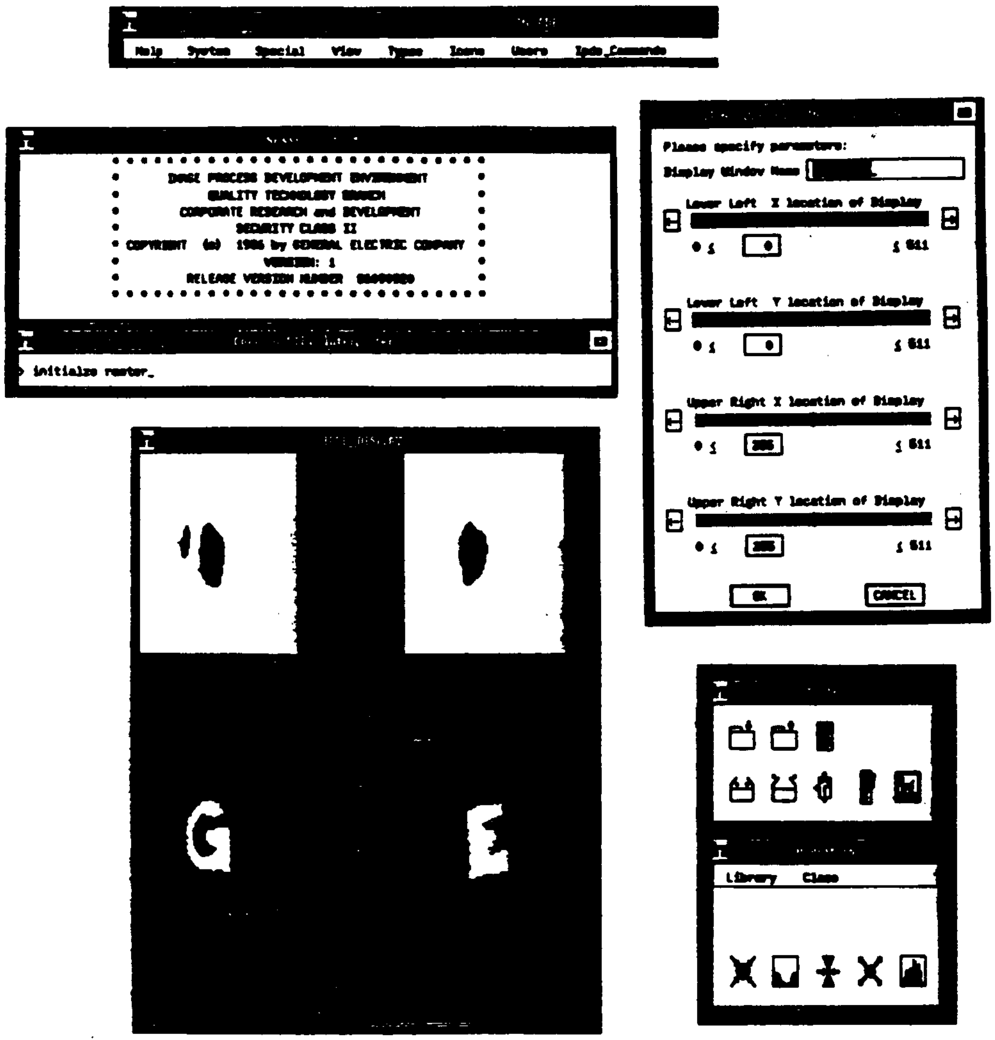


ROSE AND USER INTERFACES

- Gives intelligent feedback.
- Lets the user build up a description of some action.
- Let's data be edited until the right result is obtained.
- Interaction can be closely coupled or loosely coupled.

AN IMAGE PROCESSING APPLICATION

- Image processing system for Inspection Applications.
- Current interface is similar to VMS.
- Want ICON interface where icons represent commands and image processing buffers.
- Want to capture sequences of commands as new icons (iconic programming).



```
* * * * * * * * * * * * * * * * * * * * * * * * * * *
*       IMAGE PROCESS DEVELOPMENT ENVIRONMENT         *
*           QUALITY TECHNOLOGY BRANCH                 *
*       CORPORATE RESEARCH and DEVELOPMENT            *
*              SECURITY CLASS II                      *
* COPYRIGHT  (c)  1986 by GENERAL ELECTRIC COMPANY    *
*                 VERSION: 1                          *
*       RELEASE VERSION NUMBER  [illegible]           *
* * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
> initialze raster_
```

Please specify parameters:

Display Window Name

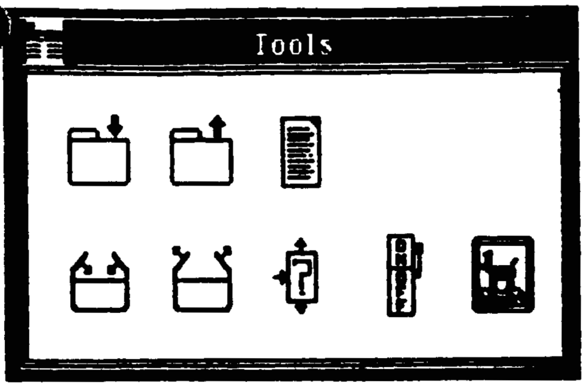
Lower Left X location of Display: 0 ≤ 0 ≤ 511

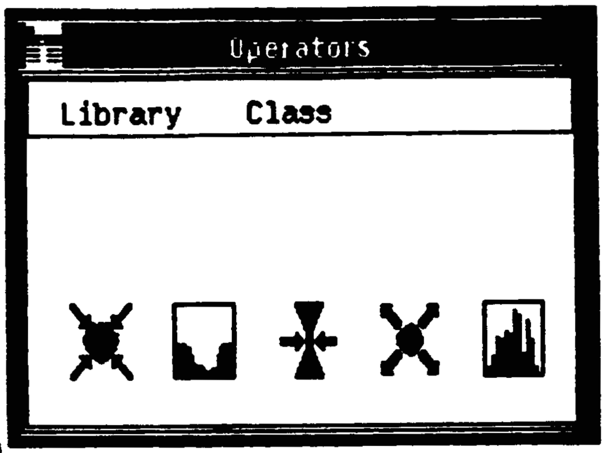
Lower Left Y location of Display: 0 ≤ 0 ≤ 511

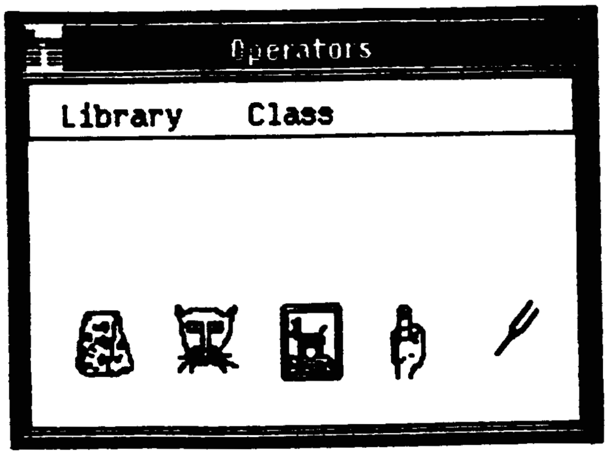
Upper Right X location of Display: 0 ≤ 255 ≤ 511

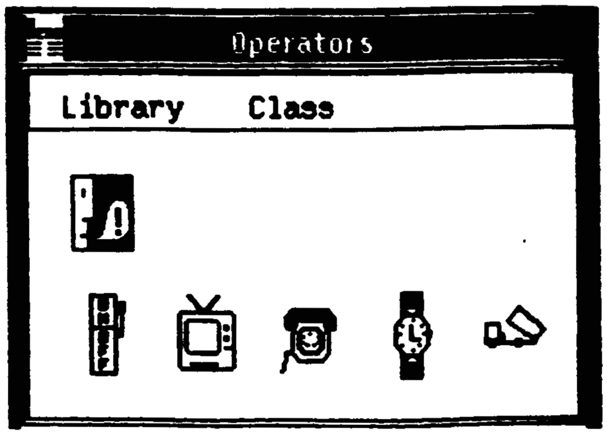
Upper Right Y location of Display: 0 ≤ 255 ≤ 511

ICONS

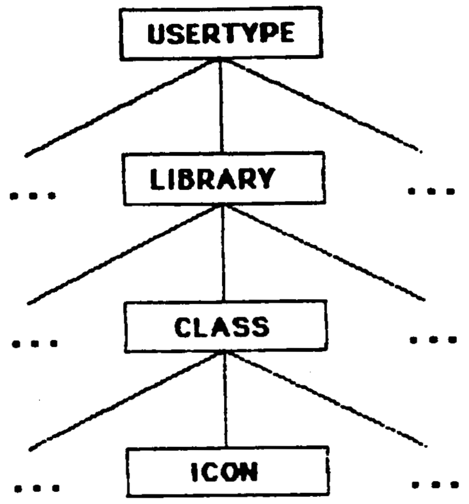








- HIERARCHICAL ORGANIZATION IN ROSE COMMAND DATABASE



- LIBRARY AND CLASS CAN BE CHANGED INTERACTIVELY USING MENU BAR

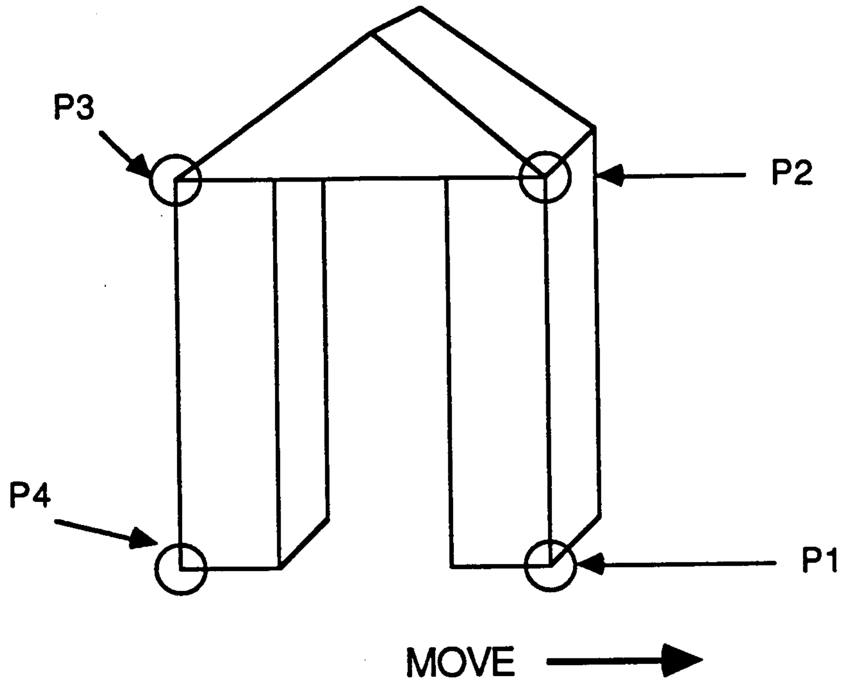
- IF ICONIC COMMAND REQUIRES PARAMETERS THEN DIALOG BOX GENERATED BY ROSE TO COMPLETE COMMAND

CURRENT STATUS

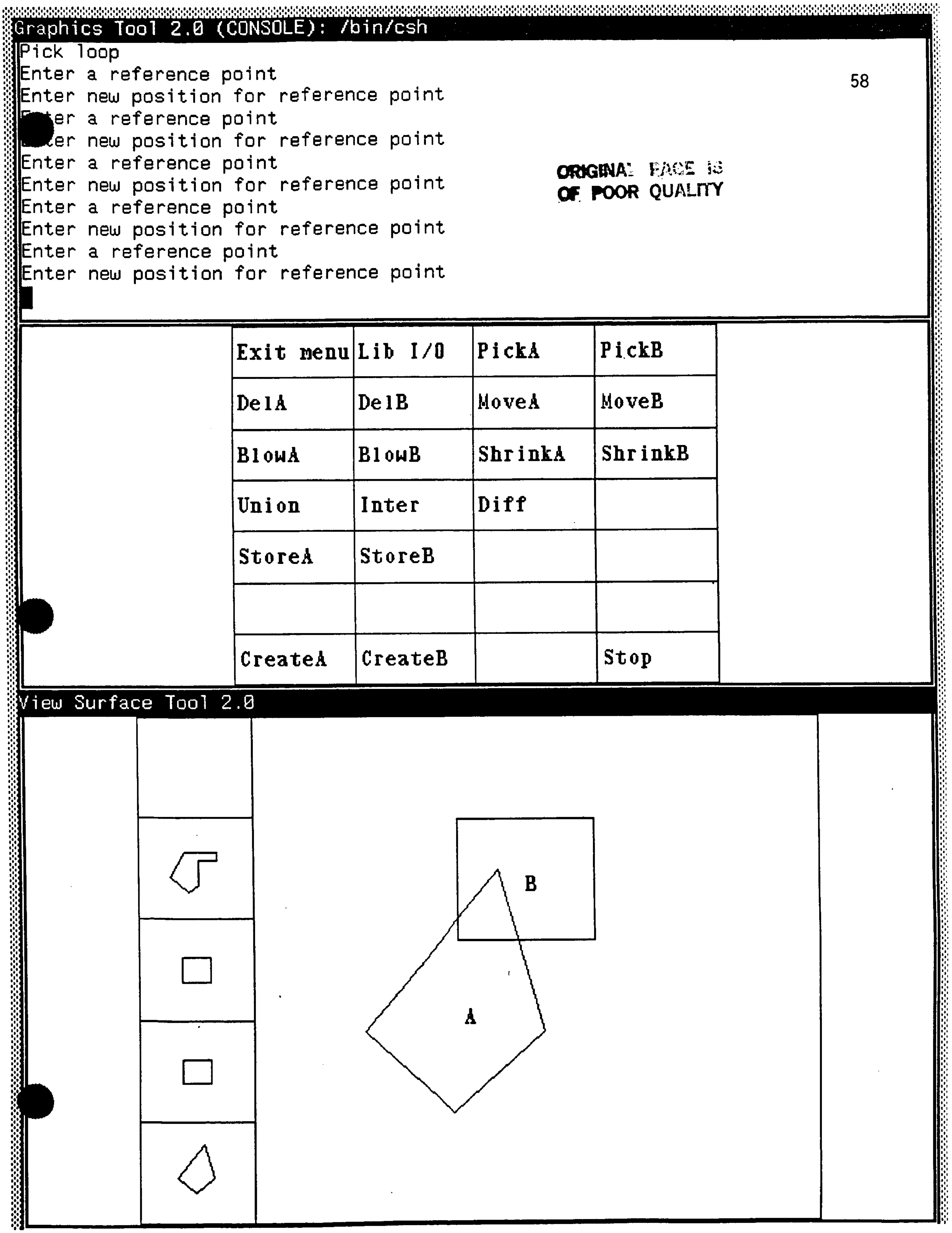
- Command icons have been programmed, but buffer icons have not.
- Debating role of direct input.
- Have a "strawman" architecture for iconic programming.

SOLID MODELING

- Solid modeling system.
- System used to manufacture and analyze three dimensional parts.
- Current interface is command line input that spans multiple lines.
- New interface will give more feed back.
- Be intelligent so that if two entities are "connected" they will both move when one is moved.


P3
P2
P4
P1
MOVE

ORIGINAL PAGE IS
OF POOR QUALITY



```
Graphics Tool 2.0 (CONSOLE): /bin/csh
Pick loop
Enter a reference point
Enter new position for reference point
Enter a reference point
Enter new position for reference point
Enter a reference point
Enter new position for reference point
Enter a reference point
Enter new position for reference point
Enter a reference point
Enter new position for reference point
```

CURRENT STATUS

- A "strawman" architecture has been designed.
- A two dimensional system with similar goals has been implemented.
- Currently, working on implementing a three dimensional interface for ROSE.
- Biggest problem is finding a workstation with three dimensional graphics.

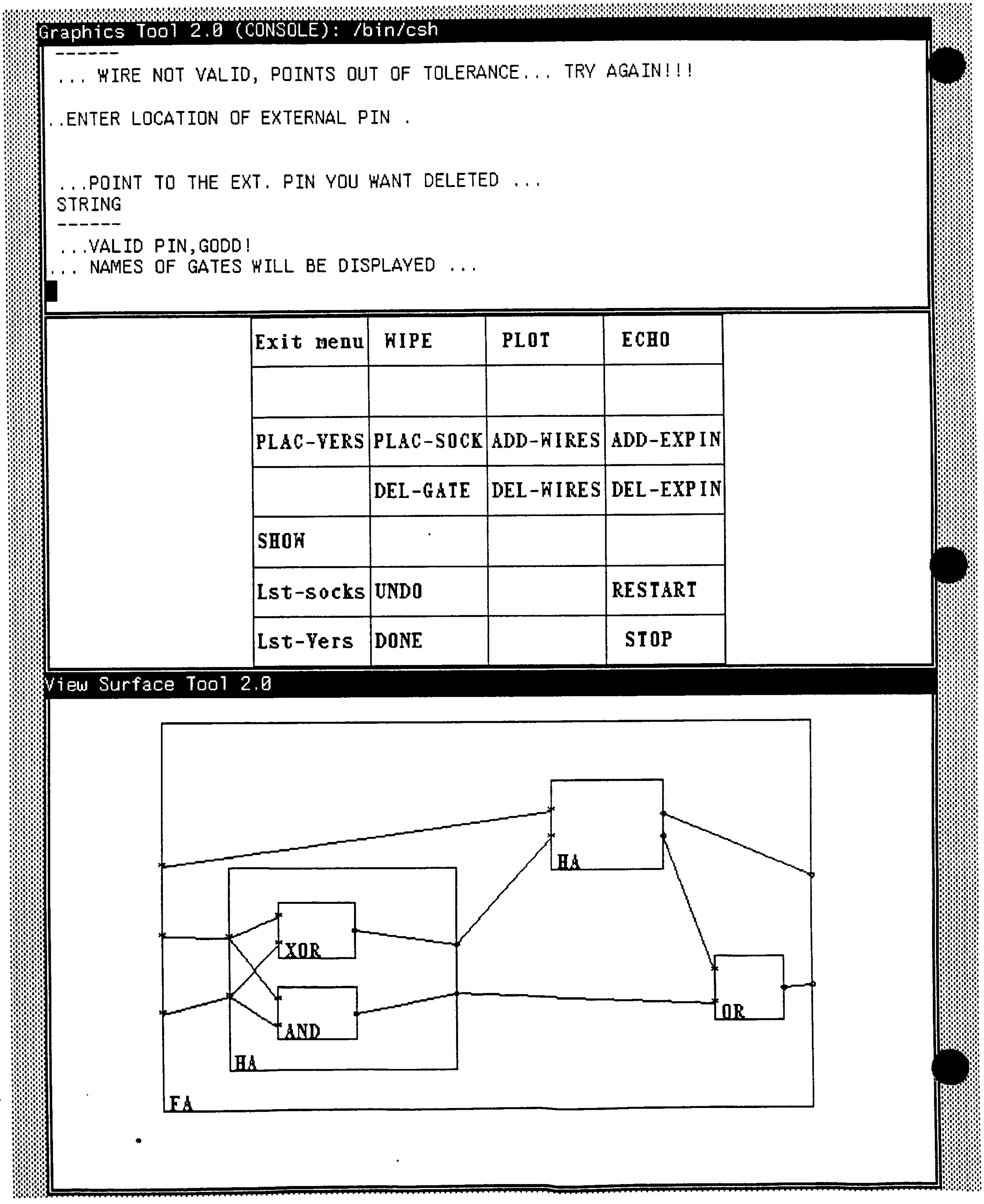
CIRCUIT DESIGN AND SIMULATION

- Goal is to build a graphics interface for the VHDL circuit simulation language.

- System will allow circuits to be instantiated on the graphics screen and connected with wires.

- System will build a VHDL program that simulates the new circuit.

- A strong distinction is made between circuit interfaces and circuit implementations (versions).



Graphics Tool 2.0 (CONSOLE): /bin/csh

```
 ------
 ... WIRE NOT VALID, POINTS OUT OF TOLERANCE... TRY AGAIN!!!

..ENTER LOCATION OF EXTERNAL PIN .


 ...POINT TO THE EXT. PIN YOU WANT DELETED ...
 STRING
 ------
 ...VALID PIN,GODD!
... NAMES OF GATES WILL BE DISPLAYED ...
```

Exit menu	WIPE	PLOT	ECHO
PLAC-VERS	PLAC-SOCK	ADD-WIRES	ADD-EXPIN
	DEL-GATE	DEL-WIRES	DEL-EXPIN
SHOW			
Lst-socks	UNDO		RESTART
Lst-Vers	DONE		STOP

View Surface Tool 2.0

CURRENT STATUS

- A fairly sophisticated design system for circuit logic has been implemented.
- The system lets circuits be created and edited and includes functions such as UNDO.
- System needs to be linked into VHDL so that it can be used to generate VHDL programs.

SUMMARY

- We have implemented a remarkable number of applications very quickly.
- Each interface has a minimal impact on the underlying application program.
- Goal is to build a Computer Integrated Manufacturing (CIM) shell for CAD/CAM applications.

COOLE - C OBJECT ORIENTED LANGUAGE EXTENSION

John Bremser

SUMMARY

In September, 1985, a project within the Graphics Center was started to look into ways in which humans interact with (computer/software) systems. In preparation for this research (formally user interfaces) a project to bring the power of advance computing environments within the VAX-station environment was started. This project involves identifying those tools that make certain other computing environments more powerful, and decisions on an appropriate strategy to implement them in the VAX-station environment easier. The following is a (partial) list of those tools:

1. An object-oriented approach to computing (i.e., LOOPS from Xerox).
2. Multiwindowing - hardware/software that allows the user to interact with the system using a keyboard, mouse, and large format screen.
3. Multitasking - specifically, the ability to edit and execute code without the intermediate steps of editing, compiling, and linking. In short, the abiality to use windows for various tasks associated with the program development process.
4. Multiprocessing - a distributed object-oriented network is planned.
5. Compiled codes in many different languages including C, the software will be written predominatly in C and the first tests of the system will be preformed in C.
6. The VMS operating system. This is due to the familarity with VMS, rather than any other reason. It should be noted that there is no reason why other operating systems could not be used.
7. A C interpreter, the product currently being considered is the interpreter from Oasys.

The whole system (all of the components listed above), will be packaged together, as a unit, and will be used in the VAX-station environment. This software/hardware combination will form the basis of the advanced computing environment.

RESULTS

The C-Lisp and COOLE software have been ported to the VAX-station environment. Additionally, several new functons have been added to the various libraries, as well as new data type (ARRAY) to the COOLE software. Timing analyses have been performed on the various modules, and increases in software throughput have been achieved.

Programmers's reference manuals have been written for the C-Lisp and COOLE software, and are available on request through the Center. These manuals are constantly being updated with the addition of new functions to the libraries added on an as needed basis.

Work has begin on the EXCALIBUR knowledge-base browser. This system is highly dependent on the graphical aspects of the VAX-station and is, therefore, not portable. There are no plans at present to port this code to any other system. It should be noted that EXCALIBUT is related to COOLE in the same fashion that the Smalltalk browser is related to Smalltalk.

An implementation of a subset of the PHIGS graphics (proposed) standard has been implemented using COOLE on the VAX-station by Craig Shelley. For more details, see his summary.

PROJECTED PLANS

Work will continue on the development of EXCALIBUR and its associated knowledge-base. This knowledge-base will also form the basis of a general purpose window management system on the VAX-station.

Another area of interest is in the development of a knowledge-base "compiler". This program should be able to take an environment created using COOLE and EXCALIBUR and create an executable program without the overhead associated with message passing the COOLE environment. This is philosophically possible, but there are details that need to be worked out.

A third area of interest lies in the area of message passing. The central issue is throughput. Data must be taken to determine how much time each of the various routines require for execution. It is also necessary to analyze several "typical" applications using COOLE in order to understand exactly what types of messages are typical and how these can be improved in terms of time and space efficiency.

Current Status

C-Lisp

Functional as of December, 1985
38 callable functions
13 auxilliary functions
1400 lines of C code

CURRENT STATUS

COOLE

FUNCTIONAL AS OF MAY, 1986
55 CALLABLE FUNCTIONS
15 AUXILLIARY FUNCTIONS
3000 LINES OF C CODE

Current Status

COOLE

Basic Data Types

- Text
- Bitmap
- Object
- Expr
- Code
- Array
  - Integer
  - Float
  - String
  - Pointer

CURRENT STATUS

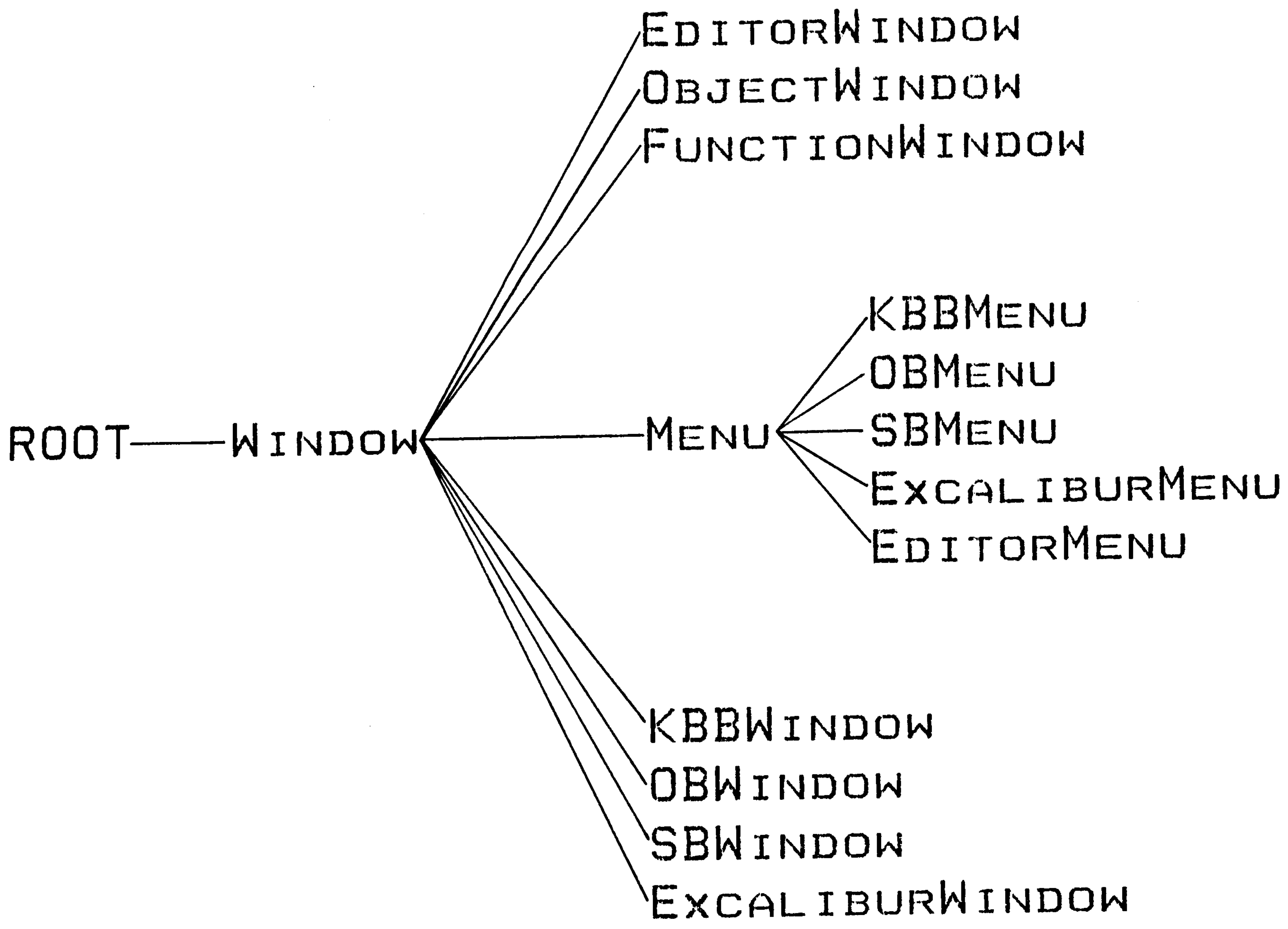
PHOENIX

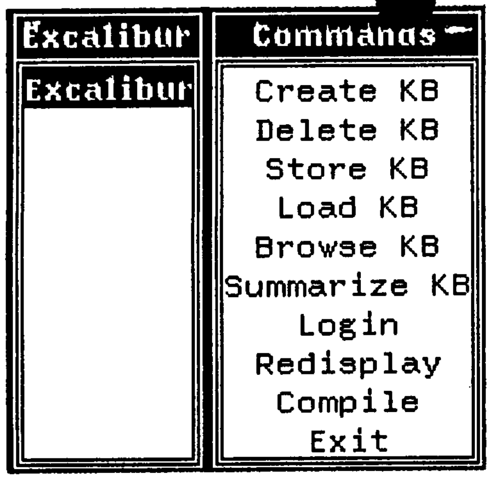
FUNCTIONAL AS OF MAY, 1986
15 FUNCTIONS
700 LINES OF C CODE

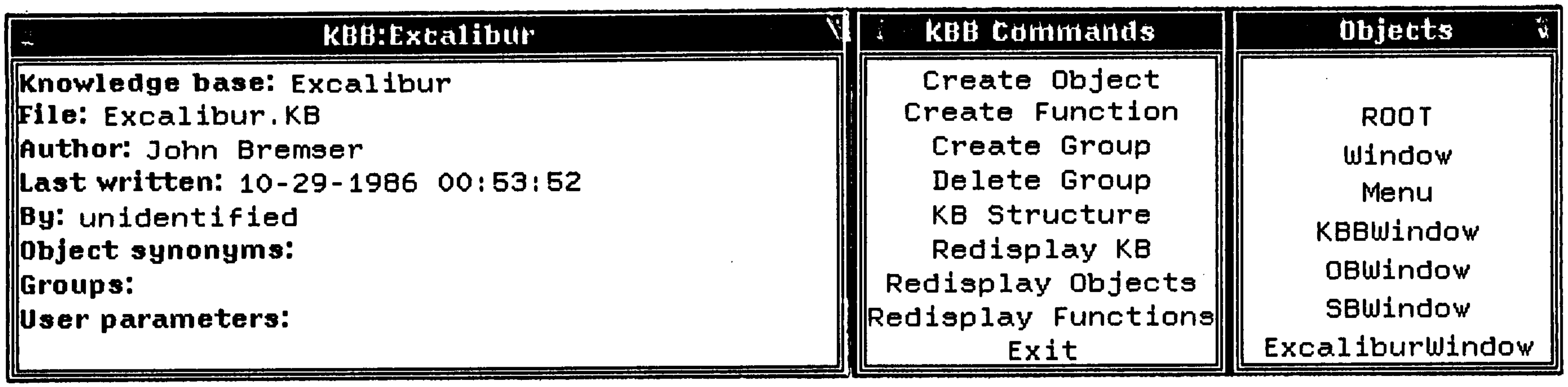
CURRENT STATUS

EXCALIBUR

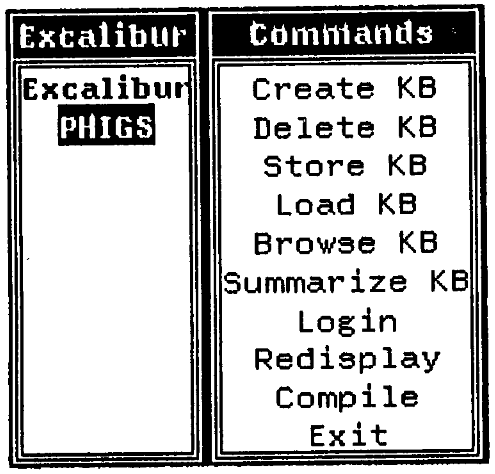
PARTIAL FUNCTIONALITY
20 FUNCTIONS
1400 LINES OF C CODE
1 COOLE KNOWLEDGE BASE

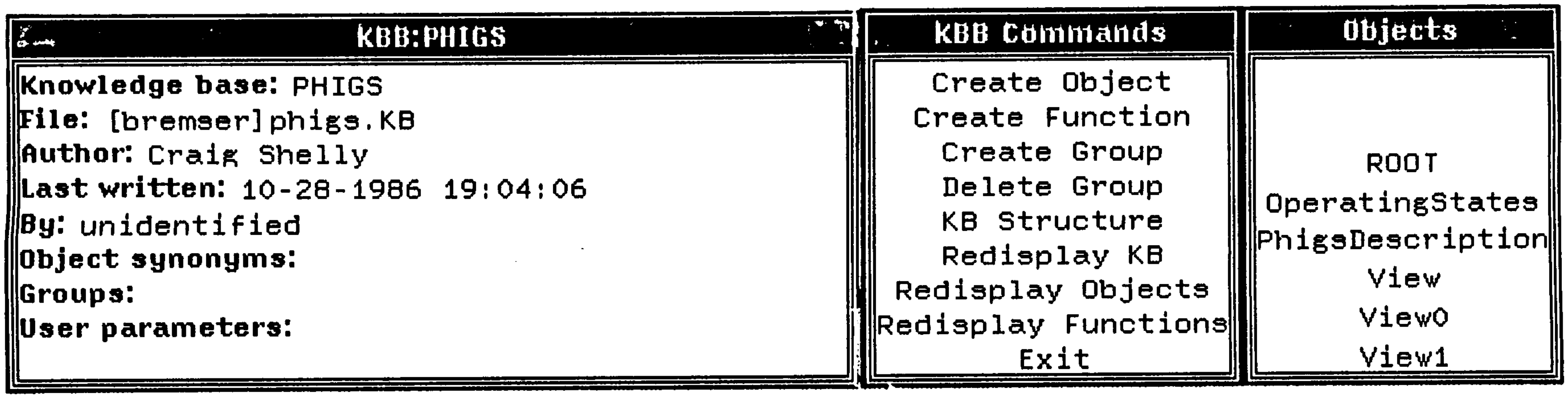

ROOT
WINDOW
EDITORWINDOW
OBJECTWINDOW
FUNCTIONWINDOW
MENU
KBBMENU
OBMENU
SBMENU
EXCALIBURMENU
EDITORMENU
KBBWINDOW
OBWINDOW
SBWINDOW
EXCALIBURWINDOW





Knowledge base: Excalibur
File: Excalibur.KB
Author: John Bremser
Last written: 10-29-1986 00:53:52
By: unidentified
Object synonyms:
Groups:
User parameters:





Knowledge base: PHIGS
File: [bremser]phigs.KB
Author: Craig Shelly
Last written: 10-28-1986 19:04:06
By: unidentified
Object synonyms:
Groups:
User parameters:

CURRENT STATUS

VAX-STATION

INCREASED THROUGHPUT

MANUALS

Work In Progress

Excalibur

C interpreter

FUTURE DEVELOPMENT

EXCALIBUR

MESSAGE PASSING

DISTRIBUTED MESSAGING

GRAPH PLOTTING

SUGGESTIONS

A PHIGS KNOWLEDGE BASE

COMPILER

SYSTEM PERFORMANCE

SOURCE CODE BROWSER

DISTRIBUTED PROCESSING

??????

NON-MANIFOLD GEOMETRIC BOUNDARY MODELING

Kevin Weiler

SUMMARY

Geometric modeling representation techniques, of which solid modeling is a more recent example, have found increasingly wider use in applications ranging from industrial mechanical part design to motion picture production.

One kind of geometric modeling representation that has found especially wide application is the *boundary representation* technique, where the surfaces, edges, and corner vertices of objects are represented explicitly. Wireframe and surface, as well as some varieties of solid model representations, are examples of boundary modeling techniques. Many of the more sophisticated boundary representations explicitly store topological information about the positional relationships among surfaces, edges, and vertices.

Until recently, boundary-based solid modeling techniques which explicitly stored topological adjacency relationships were restricted to representing only manifold domains. This disallows such conditions as two surfaces touching at a single point, two surfaces touching along an open or closed curve, two distinct enclosed volumes sharing a face as a common boundary, and a wire edge emanating from a point an a surface. Yet common modeling operations, such as the boolean set operations (both standard and regularized versions), gluing operations, and others, can produce results with these conditions, even with strictly manifold input.

These conditions have come to be known in the geometric modeling field as *non-manifold* conditions. A representation which allows such conditions is therefore called a non-manifold representation, since it can represent non-manifold as well as manifold conditions.

There are many advantages to a true non-manifold geometric modeling representation. Non-manifold representations provides a single unified representation for any combination of surface, solid modeling and wireframe modeling forms. Composite objects can be modeled directly. Analysis applications such as finite element analysis can be directly supported in the modeling representation environment, allowing direct access by both modeling and analysis functions. Closed form Boolean operations are possible in a non-manifold representation. While some non-boundary based representations do allow non-manifold conditions, they do not allow unrestricted representation of wire edge and surface elements; non-manifold boundary representations therefore also have a representational advantage over other representation forms.

The talk discusses recent advances in non-manifold boundary modeling techniques which explicitly store topological adjacencies. In particular, a new data structure, the Radial Edge structure, which provides access to topological adjacencies in a non-manifold boundary representation, is discussed. A general set of non-manifold topology manipulation operators is also described which is independent of a specific data structure and is useful for insulating higher levels of geometric modeling functionality from the specifics and complexities of underlying data structures.

Greater detail relating to this and other work concerning the use of topology in geometric modeling can be found in the PhD thesis, "Topological Structures for Geometric Modeling", RPI/CICG, August 1986, TR-86032.

MASS PROPERTY ANALYSIS ON NON-MANIFOLD TOPOLOGICAL MODELS

Peter Alduino

SUMMARY

The new radial edge data structure developed by Kevin Weiler provides access to all the topological adjacencies in a non-manifold boundary representation. This structure combines wireframe, surface, and solid modeling in a unified representation which allows a geometric modeler based on the radial edge structure to incorporate a variety of object types in a single model.

The feasibility of using the radial edge data structure as the foundation for geometric modelers used in the design, analysis, and manufacture of solid mechanical parts requires the ability to perform mass property analysis. This project will develop a method for computing the mass properties of a model based on the radial edge data structure.

RESULTS

The radial edge data structure uses a hierarchical structure to relate the topological elements of models. Listed top-down these elements are: models, regions, shells, faces, loops, edges, and vertices. The mass property attributes of a high-level element are generated by calculating the same attributes for the lower dimensional elements and aggregating these results as the hierarchical structural is traversed bottom-up. Thus, a models mass is the sum of the masses of the regions in that model. Likewise, the mass of a particular region is broken down into the masses of the individual shells which compose that region. Partitioning the mass property attributes into the lower dimensional components enables the calculations to be efficiently generated. Any element unchanged since the previous mass property computations does not have these values regenerated.

The algorithm calculates the mass properties by the appropriate integration of the model elements. Since only polyhedral models are used as input to the analysis routine, these integrals are reduced to simple summations. All the faces in the model are planar and are defined by loops consisting of a sequence of straight, directed edge segments connecting vertices. The area of such a loop is calculated by choosing a projection point P in the plane of the loop (defined as the first vertex of the first edge in the sequence of edges defining the loop) and summing the signed areas of the triangle formed by P and each directed edge of the loop in sequence. Similarly, the volume of a shell defined by a sequence of planar faces enclosing an area is calculated by choosing a projection point 0 in three space (defined as the origin) and summing the signed volumes of the tetrahedrons formed by 0; and, in each set of three vertices formed by a fixed vertex and each of the directed edges in sequence.

The lengths, area, volumes, masses, and centroids of the individual edges, triangles, and tetrahedrons are used to evaluate the mass property attributes of the higher level elements. The computed attributes are stored in the database associated with the particular model element being analyzed as the hierarchical structure of topological elements is being traversed. These intermediate results are aggregated along the way to form the final mass property attributes of the original model element. The final results are displayed to the user interactively.

PROJECTED PLANS

The goal of this project is to automatically generate the mass properties of a geometric model described by the radial edge data structure. Computed will be the attributes of volume, mass, surface area, centroid, and moments which will be stored in the model database along with the other model information (topology, geometry, etc.).

The analysis program will be written in the high-level computer language C and will be developed and run on a Digital VAXstation II.

LIST OF CURRENT FINITE ELEMENT MODELING PROJECTS

Mark S. Shephard

August 20, 1986

Center for Interactive Computer Graphics
Rensselaer Polytechnic Institute
Troy, New York 12180-3590

1 INTRODUCTION

The finite element modeling group within the RPI Center for Interactive Computer Graphics consist of two faculty, Professors Mark S. Shephard and Joseph E. Flaherty, two full time research staff and twelve graduate students. This group is actively involved in the development of modeling procedures aimed at automating the finite element modeling process. The projects being worked on are concerned with the automation of all aspects of this process including the development of;

1. geometrically-based model generation procedures,
2. fully automated mesh generators,
3. efficient adaptive analysis procedures, and
4. prototype automated finite element modeling systems that address specific application areas.

Each of the individual projects listed below address one specific aspect of this entire process. Only current, or just completed, projects are listed below.

2 MODIFIED-QUADTREE IMPROVEMENTS

The modified-quadtree program is a two-dimensional automatic mesh generator that is capable of generating mesh topologies in two-dimensional domains. It is the most mature of our automatic mesh generators with the current version having been sent to nine Associates. The following list gives the current projects related to the modified-quadtree mesh generator and its interactive graphic frontend program:

1. Improved mesh control module - The mesh control module is being rewritten to improve the layout of the menu page and add a number of new mesh control capabilities. The added mesh control capabilities will include a absolute mesh control point to go with the current relative mesh control point, referred to as a refine point, absolute and relative mesh control curves, and absolute and relative mesh control regions. Other features will be added to allow the generation of any of the basic element topology types (all triangles, all quadrilaterals, and mixed meshes) based on a region-by-region specification.

2. Addition of an analysis setup module - This module will contain menu items to indicate the polynomial order (an thus the placement of all but corner nodes) of the elements in the finite element mesh. Functions will be available to specify different orders in different parts of the model. The node and element reordering procedures will also be invoked from this menu page.

3. Addition of a material property editor - A material-library based material property module has been developed and is currently being tested.

4. Improvements to the geometry and mesh databases - Both the geometry and mesh databases in the modified-quadtree program are being improved to allow the efficient handling of analysis and mesh attribute information. These capabilities are necessary for many of the new features being added and will play an important role in supporting the advanced applications of adaptive analysis and automated metal forming analysis.

5. Addition of automated mesh parameter setting procedures to insure that the mesh generator will always produce a valid mesh - It can be proven that the modified-quadtree mesh generator can produce a valid mesh as long as the topological structure of the mesh is identical to that of the original geometry. By the addition of special checks and procedures, the mesh generator can check and resolve any problems on a quadrant by quadrant basis as part of the meshing process. For example, the current version will automatically reduce the mesh control parameters around a loop if they are too high to insure the loop will enclose a finite area in the mesh that is generated. In addition to being able to devise checks and procedures to always insure a valid mesh, additional checks can be added to perform automatic refinements to control the shapes of the resulting elements. For example, the current version can properly mesh a quadrant with 50 internal vertices, but one can be sure the resulting mesh will yield poorly shaped elements. In this case, it would be a simple test to automatically refine such quadrants

to reduce the number of vertices to a more reasonable number.

6. Rewrite of interior nodal smoothing procedures - The current interior smoothing is a standard Laplacian procedure which does not do a good job on all quadrilateral modified-quadtree meshes, and, in sever cases of a sharp reentrant corner with a caurse mesh, pull a node outside the domain making the mesh invalid. Therefore an interior smoothing procedure that is based on a more direct measure of the element shapes must be developed.

3 MODIFIED-OCTREE MESH GENERATOR

The modified-octree mesh generator is a fully automatic mesh generator for three-dimensional geometries. This procedure has been under development for nearly five years. Although we have had a prototype capability for nearly three years, the change from integer to real intersections and the algorithmic complications associated with them has made the task of producing a working version of the program that can be released to the Industrial Associates difficult. At this time, the last basic algorithmic procedures for the real intersection version are being implemented and the program will be ready to be released to the interested Associates by January 1987. Although this will be far from the final version of the program that we will be producing, that version will be capable of automatically generating meshes for most geometries. Improved versions will be released at specific intervals after that. (Note - the initial release of this program will be more on the order of the current releases of the modified-quadtree program in terms of a set structure and updating procedure.) The areas of the modified-octree program currently receiving consideration include;

1. Properly representing geometric features that fall on the octant boundaries - The algorithms that properly convey information to the appropriate neighboring octants when a geometric feature falls exactly on the boundary of an octant have been developed and implemented. They are currently being debugged. This functionality was imperative to making the modified-octree mesh generator a useful tool for general application.

2. Improvements to the Bezier geometry front end - A number of improvements have been made to the Bezier surface patch bottom-up modeler developed to test the modified-octree technique. Although the long range plans are to have users link the mesh generator to their commercial solid modelers, the functionality needed to do this is typically not

currently available (most solid modeler vendors are working on such functionality). Thus it is desirable to add some additional capabilities to the current modeling procedures. It is not currently planned to develop a solid modeler that can support Booleans. Instead some reasonably straight forward modeling capabilities are being added. The first group of procedures added automatically develop the topological relations that define the adjacencies of the geometric entities in the obvious cases. This drastically reduces the user's efforts needed in those cases. The non-obvious cases, when faces are to be concatenated for example, still require user selection. The smoothing procedures have been extended so the nodes are not constrained to the interior curves of concatenated faces. A simple set of built-up type shapes that can be glued to others is currently being developed. These paramertized shapes will be used by an Associate for control parameters in shape optimization.

3. Improvements to the octant tetrahedronization algorithm - There are improvements being made to these algorithms to insure the generation of the best shaped elements within an octant.

4. Development and implementation of algorithms to collapse small features within octants - As we learned in the modified-quadtree mesh generator, better meshes are obtained if small geometric features caused by intersections close to a quadrant boundary are 'collapsed' to that boundary. The same is true in the modified-octree. We are currently considering the algorithmic procedures needed to carry out this 'collapsing' in three dimensions such that the topology is unaltered (This is needed to insure validity of the mesh) and no additional approximation to the geometry is introduced in the final mesh.

5. Development and implementation of algorithms to allow the generation of surface only meshes - The additional algorithmic procedures that will allow the modified-octree mesh generator to produce surface only meshes are being developed and implemented.

6. Improvements in the geometry and mesh data bases - A number of alterations are being developed and implemented in both the geometry and mesh data bases of the modified-octree mesh generator. These improvements are needed to support current and future algorithmic developments and will allow for a more efficient integration of the program with solid modeling systems.

4 ADAPTIVE ANALYSIS PROCEDURES BASED ON THE MODIFIED-QUADTREE AND MODIFIED-OCTREE MESH GENERATORS

One of the goals of our research program is to develop efficient adaptive analysis procedures based on the modified-quadtree and modified-octree mesh generators. Current efforts in this area are involved with;

1. Development of local remeshing - A local remeshing algorithm has been developed for the modified-quadtree mesh generator which allows flagged quadrants to be refined and remeshed such that only those portions of the mesh that must be modified are affected.

2. Consideration of modified-quadtree based analysis on a parallel processor - RPI has recently received two Sequent computers, and we have just begun considering the development of parallel analysis procedures for these machines that directly employ the modified-quadtree, and then modified-octree, structures to coordinate the analysis process and balance the computational loads on processors.

3. Develop a three-dimensional adaptive analysis procedure based on modified-octree meshes that can produce a near optimal mesh in one adaptive analysis step - This is a joint project with Professor Zienkiewicz, University College Swansea, in which we plan to use the three-dimensional version of his new error estimator to first predict where and how much a mesh should be refined and then to estimate the errors in the final mesh. The modified-octree mesh generator will be used to generate the original mesh and a modified-octree based local remeshing procedure will be responsible for carrying out the refinements. Current efforts are concerned with a two-dimensional version based on modified-quadtree meshes.

4. Investigation and development of multigrid-solution procedures for use with modified-quadtree and modified-octree meshes - Effort on the development and implementation of multigrid procedures operating off the tree structures to obtain iterative solutions to adaptively refined meshes has recently been initiated.

5. Error estimators for second order elements - Efforts of the development of such estimators for both two and three dimensional element types is based on some recent work of Professor Babuska, University of Maryland who is collaborating on these efforts.

Note - Much of the adaptive analysis work is supported by National Science Foundation Grants.

5 TOPOLOGICALLY-BASED AUTOMATIC MESH GENERATOR FOR P-VERSION FINITE ELEMENTS

The goal of this project to produce an automated modeling procedure suited for use in P-version finite element analysis. Babuska has recently proven that the optimal meshes for these cases are extremely coarse, exponentially graded meshes. Since the modified-quadtree and modified-octree mesh generators are, by design, unable to generate such meshes, we are developing topologically-based meshing procedures that are capable of this. A two-dimensional algorithm has been developed and a procedure is being developed in cooperation with Professor Babuska to have it generate the proper initial mesh for the predictive analysis and then the optimum mesh for the P-version analysis. Since a topologically-based mesh generator places much higher demands on the interface to a geometric modeling system, this area of the project is receiving special consideration. In particular, we are investigating the use of Kevin Weiler's advanced data structures (see Weiler's 1986 thesis, CICG technical report) and topologically-based geometric modification operators to carry out this interface. Since the real complexities of this interface arise in three-dimensional geometries, we plan to develop an initial, simple version of a three-dimensional topological mesh generator to test these concepts before developing the complete set of three-dimensional mesh generation and mesh control procedures.

6 MODIFIED-QUADTREE BASED AUTOMATED METAL FORMING ANALYSIS -

This project is concerned with the development of a geometrically based automated finite element modeling system for the analysis of large, general deformation metal forming analysis. The current efforts are concentrating on a two-dimensional system that employs the modified-quadtree mesh generator to generate all the meshes needed during the meshing process. A geometrically-based program procedure to developed and is currently being implemented. This modeling capability is being integrated with both the MARC and ABAQUS analysis programs which will carry out the nonlinear finite element analysis steps.

7 MODIFIED-QUADTREE AND MODIFIED-OCTREE BASED TRANSONIC FLOW ANALYSIS

Procedures are being developed to solve the Euler equations on modified-quadtree trianqular meshes. These procedures can be used to solve realistic compressible flow problems through the various flow regimes. A finite volume procedure for both linear and quadratic triangles has been developed and successfully solved using both explicit and implicit time marching algorithms. Extensions to three-dimensions using a similar finite volume approach is just beginning.

8 EXPERT SYSTEM APPROACHES TO THE GENERATION OF INDIRECT FINITE ELEMENT ELEMENT MODELS

This project addressed the area of developing finite element models for problems where the finite elements used are of a lower geometric dimension than the geometric entity they represent. Since the use of these indirect element types is dictated by rules of thumb that are based on the analyst understanding of structural behavior, it is not possible to automate the generation of such finite element models using only analytically-based algorithms. However, expert system based procedures are ideally suited for this task. Thus a prototype system was developed for the generation of airframe structural models that can extract a geometric model form a geometric modeling system, in this case CATIA, classify it, pass it to an expert system, in this case PRISM, to have the classified geometry flagged as to how it should be modeled with indirect finite elements. After that, all the information can be passed to a set of application routines to have the actual finite elements generated.

MODIFIED-QUADTREE MESHING ENHANCEMENTS

Peggy Baehmann

SUMMARY

The ever evolving modified-quadtree mesh generator has again been targeted for changes. The main areas receiving modification are the geometric and the finite element databases. The databases are being upgraded to be able to meet the needs of various modeling situations. For example, they are being revised to contain information needed for analyses that deal with evolving geometries and contact surfaces such as found in metal forming analyses.

Enhancements have also found their way into the meshing algorithms of the modified-quadtree mesh generator. One enhancement is the capability to generate higher order elements. Other improvements still on the way include a new method to smooth the finite element mesh, and the capability to create splines in the interactive geometry portion of the modified-quadtree mesh generator.

RESULTS

The geometric modeler requires several enhancements to allow for the advanced features in many applications. By utilizing the topological concepts outlined by Kevin Weiler (see Weiler Thesis August 1986, TR-86032), as well as geometric modeling extensions, the geometry database will be able to support these areas. The geometric modeler is often required to define a model edge with more than one curve: this is called a concatenated edge. A concatenated edge would have as its geometric definition a set of continuous curve segments and no vertex would be required at each curve segment definition, only at the ends of the edge. Another useful capability is the ability to constrain, but not fix a vertex along a closed edge. For instance, a closed edge geometrically defined by a circle must have one vertex, yet if no other intersections occur along the edge, the vertex need only lie somewhere along the edge. In virtually all applications, the ability to modify the model and later return to its original form or even return to some intermediate form is useful. In simple load applications, the endpoint of a load should be represented by a vertex along the original edge. Placement of the vertex along the edge would break up the edge into two edges. If the load were later removed, the edge would be returned to its original state. In metal forming, similar situations occur when the workpiece comes in contact or out of contact with the die. Here not only the topology changes in the die, but also the geometric definition of the work piece as deformation occurs.

The work on improvements in the finite element database are currently in progress. In terms of higher order elements, the finite element database will contain information allowing any type of higher order element to be created. The higher order nodes that fall along a finite element edge will be associated with the original finite element edge as an attribute. The additional higher order finite element nodes that fall within the interior of the finite element will be associated with the finite element as an attribute.

The process for finding the higher order nodes along an edge is to interrogate the geometry to find the additional nodes, instead of just subdividing the linear finite element edge. Currently the additional nodes needed for second order elements have been generated, but until the existing mesh generation program has been converted to the new finite element database, they are not being renumbered.

Another improvement that will be seen in the new finite element database is the capability to have more than one node number for a given finite element node. This capability is needed for contact problems such as in metal forming at the interface between a die and a workpiece.

PROJECTED PLANS

The finite element database will be implemented as soon as possible. There are a few areas that still need a little thought such as where load and boundary condition information will be stored. As soon as the finite element database is implemented, second order elements and nodes will be able to be reordered and a tape will be sent out to the Industrial Associates the modified-quadtree mesh generator as well as an updating routine to be used for converting old examples to the new format.

The geometric database will also be implemented in the very near future as it is essential to the metal forming project. It also needs a little more thought in the areas of load and boundary conditions.

MODIFIED-QUADTREE FRONT END IMPROVEMENTS

Bessem Jlidi
Marek Garbos

SUMMARY

The two-dimensional finite element model generation software package is a collection of several program modules. In recent years, considerable improvements in computer programming data structures, and hardware capability have initiated development of new menu structures and better algorithms. These changes are now being reflected on a number of these modules.

A new material property attribute editor has been developed and implemented. It is designed to be "user friendly" avoiding cumbersome menu pages and keeping the number of steps to get the job done to a minimum. The material property editor allows the user to select a material type, from either a previously created user library or an external material library. In addition, the MPAE has a useful set of features like editing, sorting, verifying, selecting and assigning material types.

A new mesh generation attribute editor is currently being developed. It will incorporate all the latest mesh improvements. The use of robust and efficient algorithms will have a great impact on the overall program performance. A totally new Setup for Analysis module will serve the purpose of element reordering, and other various operations to be performed on the final mesh database.

The geometry generation and problem class modules are being redesigned to include new functions and account for new changes undergone by the whole program. In addition, the Loads and Boundary Conditions are at the development phase.

RESULTS

The material property editor module is now fully implemented and integrated with the current version of the modified-quadtree program. It takes full advantage of whatever system the program is running on. It uses different colors for more visibility and functionality. It also alerts the user in case of an improper menu item selection by the use of bells. Moreover it has horizontal and vertical scroll capabilities that allow the user to select a material or its corresponding properties very easily.

Sorting of materials can be done according to either their industry or internal names. Protection against accidental reassignment of a material to a region has also been provided. Finally the user has four choices: to quit without saving, to go to the previous or next menu page, or to go to the main page. In the last three choices saving of the information generated during that session is done automatically.

Currently, work is being done on updating the old functions and the addition of new features to the modified quadtree mesh generator at the user interface level in terms of user requirements, software specification, and algorithm design. These new features can be categorized under four groups of active menu items:

1. Features to assign mesh size control parameters which can be grouped under size parameters.

2. Features to assign mesh control parameters in general as to improve mesh gradation and levels of refinement which can be grouped under meshing attributes.
3. Features to assign element attributes which can be grouped under that same name.
4. Features to display the geometry, mesh, size control parameters, meshing attributes, and element attributes which can be grouped under display.

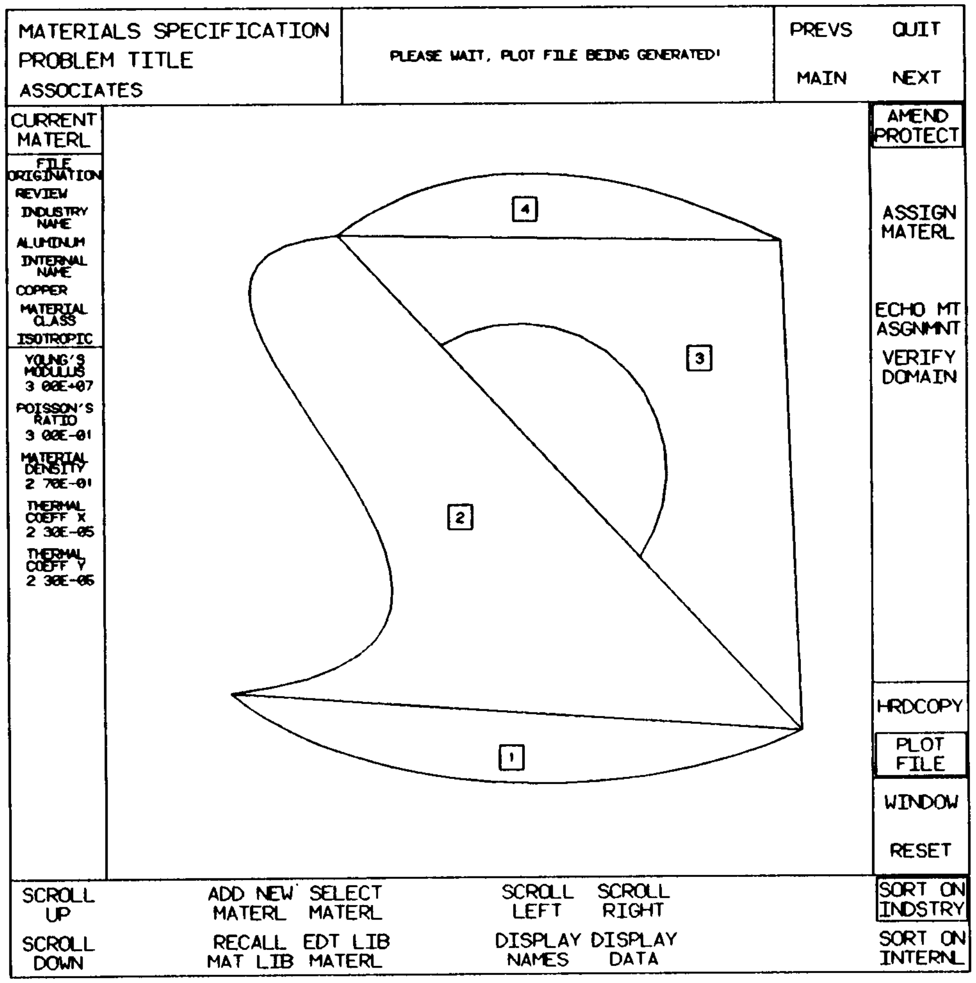
The old functions that are to be updated will consist of a current status display of pertinent information and a new option for the mesh invoking process.

The updating of the old functions and the addition of the new features will accomplish two objectives. The first one is to increase the functionality of the mesh generator. The second one is to make the interaction between the user and the software go smoother and friendlier. These features will have also a positive impact on the analysis capability carried outside the mesh generator, since incorporating these new changes will result in a richer and more complete input finite element database.

Work is also being done at the theoretical level on the loads and boundry conditions attribute editors.

PROJECTED PLANS

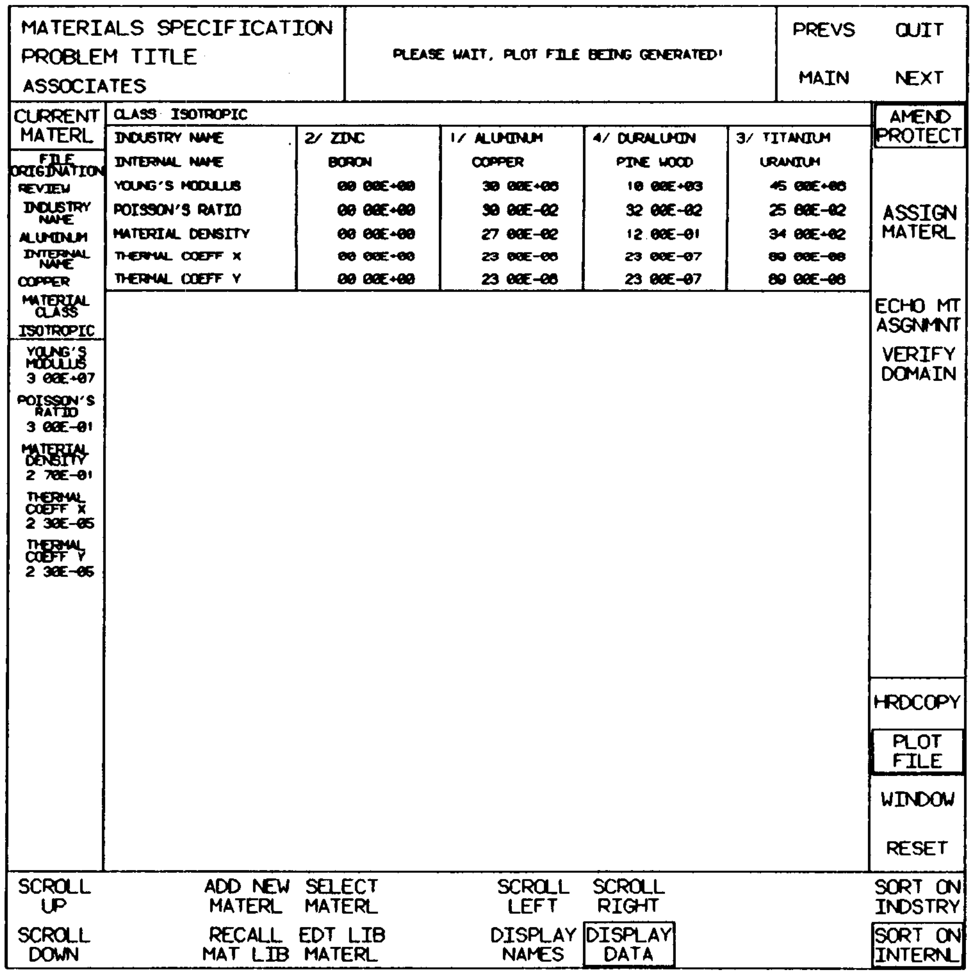
Following the same approach, the geometry and problem class modules will be redesigned and implemented accordingly. A setup for analysis module will also be added. This module will perform the last step in constructing the final database to be used by any finite element analysis package.



PLEASE WAIT, PLOT FILE BEING GENERATED!

FILE ORIGINATION
REVIEW
INDUSTRY NAME
ALUMINUM
INTERNAL NAME
COPPER
MATERIAL CLASS
ISOTROPIC
YOUNG'S MODULUS
3 00E+07
POISSON'S RATIO
3 00E-01
MATERIAL DENSITY
2 70E-01
THERMAL COEFF X
2 30E-05
THERMAL COEFF Y
2 30E-05

ORIGINAL PAGE IS
OF POOR QUALITY



MATERIALS SPECIFICATION
PROBLEM TITLE
ASSOCIATES

PLEASE WAIT, PLOT FILE BEING GENERATED'

PREVS QUIT
MAIN NEXT

CURRENT MATERL
FILE ORIGINATION
REVIEW
INDUSTRY NAME
ALUMINUM
INTERNAL NAME
COPPER
MATERIAL CLASS
ISOTROPIC
YOUNG'S MODULUS
3 00E+07
POISSON'S RATIO
3 00E-01
MATERIAL DENSITY
2 70E-01
THERMAL COEFF X
2 30E-05
THERMAL COEFF Y
2 30E-05

CLASS ISOTROPIC

INDUSTRY NAME	2/ ZINC	1/ ALUMINUM	4/ DURALUMIN	3/ TITANIUM
INTERNAL NAME	BORON	COPPER	PINE WOOD	URANIUM
YOUNG'S MODULUS	00 00E+00	30 00E+06	10 00E+03	45 00E+06
POISSON'S RATIO	00 00E+00	30 00E-02	32 00E-02	25 00E-02
MATERIAL DENSITY	00 00E+00	27 00E-02	12 00E-01	34 00E+02
THERMAL COEFF X	00 00E+00	23 00E-06	23 00E-07	80 00E-08
THERMAL COEFF Y	00 00E+00	23 00E-06	23 00E-07	80 00E-08

AMEND PROTECT
ASSIGN MATERL
ECHO MT ASGNMNT
VERIFY DOMAIN
HRDCOPY
PLOT FILE
WINDOW
RESET

SCROLL UP
SCROLL DOWN
ADD NEW MATERL
RECALL MAT LIB
SELECT MATERL
EDT LIB MATERL
SCROLL LEFT
DISPLAY NAMES
SCROLL RIGHT
DISPLAY DATA
SORT ON INDSTRY
SORT ON INTERNL

AUTOMATED ADAPTIVE FINITE ELEMENT MODELING BASED ON MODIFIED-QUADTREE TECHNIQUES

Peggy Baehmann

SUMMARY

An automated finite element modeling program is being implemented which consists of an adaptive finite element analysis process in conjunction with the two-dimensional automatic modified-quadtree mesh generation. The goal of an adaptive finite element analysis process is for the analysis functions to be able to interact with a mesh such that it can refine (or derefine) the areas of the mesh needing further refinement (or derefinement) and then proceed with the analysis without further user intervention. The analysis is finished when the solution is within a user prescribed accuracy tolerance. The only input needed for such an analysis process is the geometric model, the load and boundary conditions, and an accuracy tolerance for the solution.

The modified-quadtree mesh generator is ideal for h-type local remeshing because the mesh generator is based on a quadtree. Since the mesh is generated from the modified-quadtree, all that has to be done to change the density of the mesh is to change the level of the terminal quadrants in the quadtree in the areas needing refinement or derefinement. For example, Figure 1a contains the original mesh of an object. For the areas needing remeshing, first the elements are traced back to the quadrant from which they were formed and all information associated with the quadrants are deleted. A transition zone is then constructed around the new quadrants, and the old elements are deleted Figure 1b. Next the quadrants are refined (or derefined) to a level as determined from the adaptive analysis procedures. The boundary edges are then reintersected with the new quadrants, Figure 1c. Finally, in the areas of the new quadrants and the transition zone, the finite elements are created and smoothed, Figure 1d.

Currently work is underway on two approaches to adaptive finite element analysis are being worked on. One method uses a multi-grid solution procedure which employs a succession of meshes to solve the finite element equations at a given level of mesh refinement. This solution technique is well suited for use with meshes generated by the modified-quadtree technique because the successive meshes (coarser or finer) needed by the multi-grid solver can be obtained from different levels of the modified-quadtree. There is an efficient re-solution per mesh update using this method.

The second method which follows more standard finite element solution techniques is the minimum analysis step approach. In this method, few solution analyses are needed and it is easy to implement with existing finite element programs. Basically, this method performs a finite element analysis, measures the error in the solution, and refines the mesh only in the areas with the high error. By refining the mesh to the levels indicated by the error estimator, an attempt is made to minimize the number of subsequent finite element solutions required.

RESULTS

The minimum analysis step approach is being used in a finite element modeling system

C-2

in conjunction with the modified-quadtree mesh generator. The minimum analysis step approach uses an a posteriori error estimator developed by Zienkiewicz and Zhu (1986). The error in the energy norm is obtained based on the difference between the discontinuous stress field calculated from the finite element solution and a continuous stress field obtained from an efficient global type stress smoothing

$$e = S - F$$

where S = improved continuous stress field
F = discontinuous stress field from finite element solution

The finite element mesh is examined and the elements are refined when

$$r = e/m > 1$$

where e = error on element
m = maximum element error

to a level n defined as

$$n = c/r \text{ for } r > 1$$

where c = current element size
n = predicted element size
r = ratio of element error/maximum element error

The modified-quadtree mesh generator contains information enabling the finite elements to be traced back to the quadrants from which they were formed. The tree is then locally refined n levels at the site of the quadrants which contain the elements needing refinement. Figure 2 contains a set of meshes that were adaptively refined by the automated finite element modeler using the error estimator developed by Zienkiewicz and Zhu.

PROJECTED PLANS

Efforts in the area of adaptive finite element modeling is to be continued. Work on the multi-grid approach for finite element solutions has been started. The multi-grid approach will use an iterative solution method, namely a conjugate gradient solver, in conjunction with locally remeshed meshes obtained from different tree levels of the modified-quadtree mesh generator to efficiently solve the finite element equations.

Changes are also being implemented in the modified-quadtree mesh generator to allow for derefinement during the adaptive remeshing used by either solution approach.

Efforts on the three-dimensional version of these procedures based on the modified-octree mesh generator will also be initiated during this reporting period.

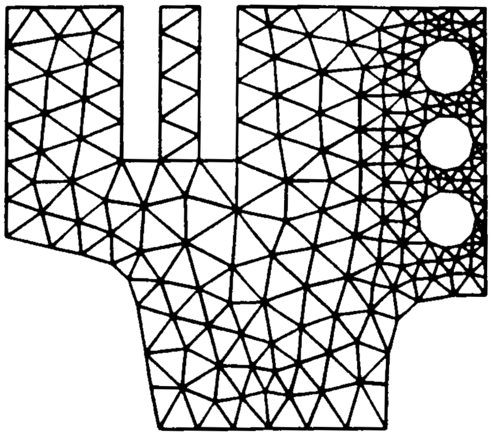


Figure 1a. The original finite element mesh.

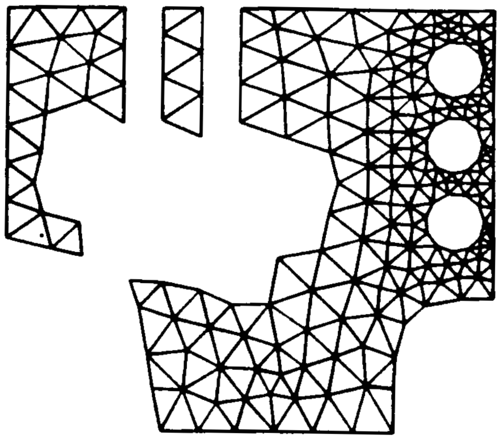


Figure 1b. Deletion of elements in area to be refined and in the transition zone.

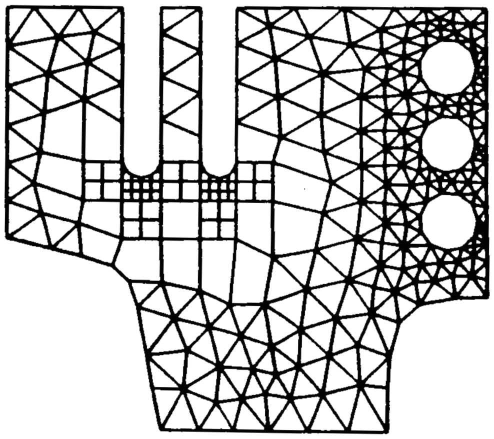


Figure 1c. Refinement of quadtree and reintersection of boundary edges in area to be refined and the in transition zone.

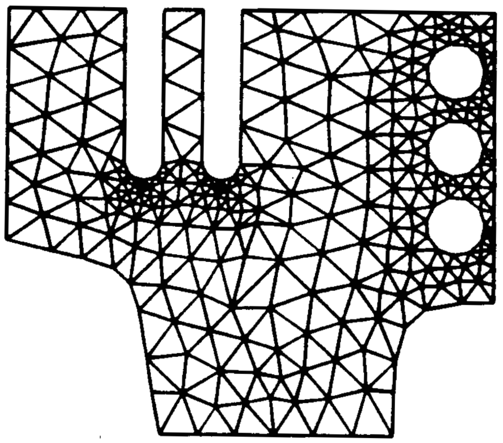


Figure 1d. Locally refined mesh.

Figure 1. The sequence of local refinement in the modified-quadtree mesh generator.

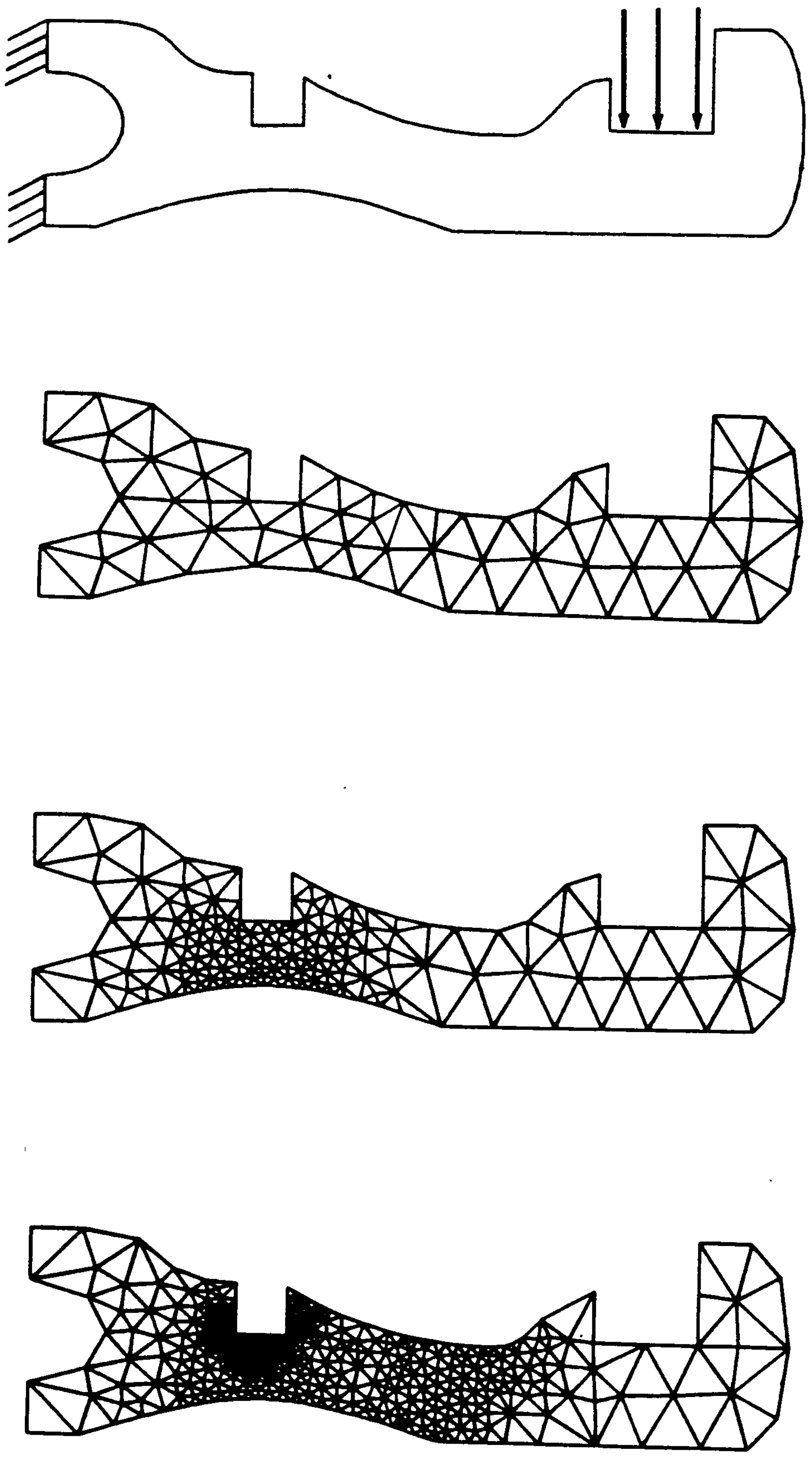


Figure 2. A set of adaptively refined finite element meshes using the modified-quadtree mesh generator and a-posteriori error estimation techniques developed by Zienkiewicz and Zhu.

SOLVING THE EULER EQUATIONS WITH FINITE VOLUME TECHNIQUES ON MODIFIED-QUADTREE MESHES

Joseph Flaherty
Mark Shephard
Ray Ludwig
Fabio Guerinoni

RESULTS

We have been using modified-quadtree to investigate a finite volume method for solving the Euler equations of gas dynamics in two dimensions. The basic method uses the "smooth" flux vector splitting due to B. van Leer (1).

During the past year, research has been concentrated on basically two aspects: to improving the quality of the steady state solutions and to developing new methods of solution, (all of them based on the van Leer splitting) that would improve the time required for convergence to steady state.

Early experiments were run on symmetric NACA 0012 airfoil with freestream Mach number of 0.85. Experiments predict a shock a few centimeters left of the trailing edge. The basic method captured the shock in the right position and resolved it sharply. There were, however, some oscillations in the supersonic preshock zone that could not be explained by the numerical method alone.

In order to isolate the causes of the spurious oscillations, we examined near and far field boundary conditions. In particular, we implemented two type of non-reflecting boundary conditions for the far field: Bayliss-Turkel (2) and Hedstrom (3). Both of the methods failed to yield significant results.

When dissipation was added to the scheme, the solution became smoother and the oscillations disappeared, but the shock was less defined and the peak values werre reduced. After additional experimentations, we discovered a positive correlation between mesh uniformity on certain regions of the domain and the existence of oscillations. The use of well graded meshes reduced or eliminated oscillations and yielded good solutions, both transonic and supersonic.

To try to speed up convergence, we used three types of schemes: Switched Evolution/Relaxation (SER), a varying time step scheme developed by W. Mulder (4) to use on finite difference schemes; Immediate Updating (IU), that takes advantage of the reordering feature of the modified-quadtree mesh generator; and finally, a Locally Implicit (LI) method, developed by the Center using the smoothness flux vector splitting. Unfortunately, the SER method did not converg, even though parameters were introduced to minimize the negative effects of varying time step. The results of the IU scheme were roughly comparable to the basic method, although only one renumbering scheme was tried. It was concluded that IU schemes work and may be useful when space becomes critical.

The Locally Implicit method was successfully implemented, showing improved convergence rates over the standard explicit method. Here again, a well graded mesh seems to be of importance.

RESULTS

Because the LI method was the most successful, it is being more closely evaluated. Three specific topics are being investigated: performance on finer well graded meshes; performance in terms of work, as opposed to iterations needed to reach the steady state; and behavior of explicit/implicit hybrid methods and methods of varying time step, with minimization of work as a goal.

New and better graphics equipment acquired by the Center permits the specification of more challenging problems, since it also permits the specification of better mesh characteristics to use with the specific problem. Problems can now be handled that are two to four times larger. At the present time, angle of attach problems on very fine meshes can routinely be solved.

However, these problems require a large amount of computation and test the limitation of the machines. For this reason, work has be started on the implementation of the modified-quadtree mesh generator and the Euler equations solver on a parallel computer.

PROJECTED PLANS

This work has opened a number of areas of research. To further improve the numerical methods, investigation on the following areas is possible:

1. Piecewise linear approximations as opposed to piecewise constant.
2. Higher order approximations for time integration.
3. Error estimation and analysis.
4. Fully implicit methods.

The following software-related issues will be investigated:

5. Parallel implementation of the basic and derived methods.
6. The use of tree structure of the generated mesh in numerical algorithms.

REFERENCES

1. Van Leer, B., "Flux Vector Splitting for the Euler Equation", *Lecture Notes in Physics*, 170, pp. 507-512, Springer Verlag: Berlin, 1982.
2. Bayliss, A. and Turkel, E., "Outflow Boundary Conditions for Fluid Dynamics", *SIAM Journal of Sci. and Stat. Computing*, Vol. 3, pp. 250-259, 1982.
3. Hedstrom, G.W., "Non-Reflective Boundary Conditions for Non-Linear Hyperbolic Systems", *Journal of Comp. Physics*, Vol. 30, pp. 222-237, 1979.
4. Mulder, W., "Dynamics of Gas in a Rotation Galaxy", thesis, Lyden, Netherlands, 1985.

AUTOMATED METAL FORMING ANALYSIS

Humphrey Chow
Richard Ashley
Peggy Baehmann

SUMMARY

The ability to automatically analyze metal forming processes with the finite element method will reduce the amount of time, effort, and cost in producing new die sets and forming tools. Furthermore, the finite element analysis can also predict areas of high stress and wear on the tools, and identify highly stressed areas in the workpiece.

However, the analysis of metal forming processes is hindered by the lack of an automatic finite element modeling capabilities with remeshing capability. This is because a forming process usually results in drastic changes taking place in the geometry of the workpiece, which may cause the finite element mesh of the workpiece to degenerate to the point where it is no longer valid. To continue this analysis any further requires the remeshing of the workpiece. Therefore, many finite element meshes may have to be created to completely analyze a single forming process. As one can imagine, the finite element analysis could be greatly enhanced through the use of an automatic mesh generator with automatic remeshing capability.

Another critical aspect is the proper geometric modeling of the workpiece and the tracking of its shape during the analysis. This is important because accurate analysis of interaction between the workpiece and dies depends on their proper geometric representations.

The goal of this work is to develop the automatic geometry tracking and meshing system for metal forming analysis. This system will be independent of particular analysis program used.

RESULTS

The general integration scheme to obtain an automated metal forming analysis as applied to isothermal forming of plane strain and axisymmetric parts is shown in Figure 1. The problem definition is defined by running an interactive graphic program. The modified-quadtree mesh generator then creates the mesh and transfers the necessary information to the model control procedures. The model control procedures will interact with both the modified-quadtree mesh generator and the analysis program to allow a fully automated metal forming analysis. Current efforts are employing the MARC program as the analysis engine with efforts on the specific interface needed for ABAQUS beginning.

An initial version of the interactive graphic program to define the problem definition has been implemented within the framework of the modified-quadtree front program. A standalone version of the modified-quadtree mesh generator is used to generate all-quadrilateral element meshes and transfer the necessary information to the model control procedures, and to generate initial input file for the MARC analysis program.

The present version of the model control procedures can perform the following automatically: (i) generate all-quadrilateral mesh and prepare initial MARC input file; (ii) submit input file to MARC analysis to perform all increments prescribed; (iii) process MARC output file, determine the increment number at which unacceptable quadrilateral elements (internal angles lie outside 45 to 135 degrees range) exist, and update the workpiece-die geometry for the previous increment which has no unacceptable elements. Further analysis can be continued by invoking the modified-quadtree mesh generator to remesh the workpiece globally and to prepare a MARC rezone input. Figure 2 shows that a cylinder is upsetted with 50% reduction in height.

Although the MARC analysis program will not allow a node to lie outside the die boundary, it is still possible for an element boundary to lie outside the die material. A remeshing of the workpiece at that time will result in the loss of this material. The material loss will be eliminated by identifying the portions of the die boundary at which this problem may occur, and then examining these areas during the analysis for workpiece-die intersections. In order to prevent this loss of workpiece volume due to the die surface overlaping the workpiece, the problem should be examined at the geometric model level, not at the finite element model level. Figure 3 illustrates the motivation for viewing the die penetration problem from a geometric standpoint. Figure 3.1 shows a portion of the boundary of a workpiece approaching a die surface. In this figure, both the actual geometric boundary of the workpiece, and the finite element representation of this boundary are depicted.

After an increment of die displacement, the relative position of the die and workpiece may have been changed to a position similar to that shown in Figure 3.2. In this figure, the finite element model does not predict any loss in workpiece volume. However, a loss in workpiece volume has occurred because the geometric boundary of the workpiece has penetrated the die surface. Therefore, it is the recognition of when the actual boundary of the workpiece has come in contact with the die surface that is the critical factor in preventing workpiece volume loss.

The resulting volume loss due to the die surface penetration by the workpiece boundary can be eliminated by backing up the analysis until the workpiece has just come in contact with the die surface, as illustrated in Figure 3.3. It should be noted, though, that this backing up of the analysis is not a straight forward procedure due to the nonlinearity of this analysis itself. However, the desired position is bounded by the starting and ending points of the current analysis increment (i.e., Figures 3.1 and 3.3).

When the analysis has been re-incremented until the point at which the die surface and the workpiece boundary have just come in contact is reached, the analysis will be temporarily stopped, and the geometric and finite element model of the die and workpiece will be updated. The geometric model will be updated by inserting a vertex at the point of contact between the die and the workpiece to insure that this critical point is properly constrained and represented. The finite element model will then be updated to insure that the proper mesh gradation are applied in the area of this point of contact. This updating of the finite element model may be performed by remeshing the entire domain of the model, or it may be performed by localized remeshing in the area of contact.

As the analysis is now continued, this portion of the die surface will no longer be a critical area where workpiece volume loss may occur. This is because with further die motion, the workpiece boundary will roll along the die surface as it makes contact.

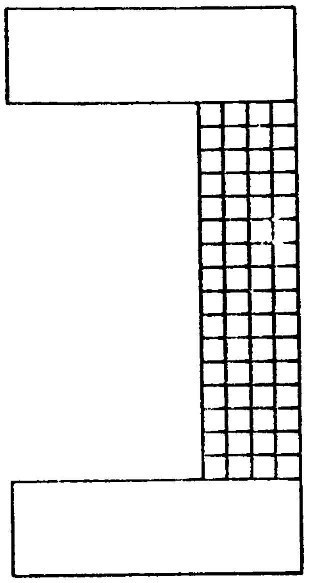
PROJECTED PLANS

1. Implement parametric cubic spline to represent the free surface of the workpiece: it is able to provide slope and curvative continuities along the free surface.
2. Implement the algorithms on die-workpiece geometry overlap checking, and backing up and restarting the analysis based on geometry considerations. This implementation will follow once the free surface representation by parametric cubic spline has been implemented.
3. Develop the interface to use ABAQUS as the analysis program within the system.
4. Determine when local or global remeshing is required so that remeshing can be done by modified-quadtree mesh generator during the analysis.
5. Adaptive analysis will be used for error estimate and to determine region in the workpiece where finer mesh is desired.

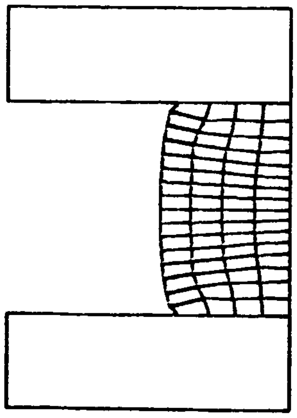
```
 -------------------
| PROBLEM DEFINITION |                                          EXIT
 -------------------                                             ^
        |                                    END OF ALL  |  LOADSTEPS
        V                                                |
   -----------          ---------------         -------------
  |   MESH     |-------->|    INITIAL      |------->| EXECUTE      |<-
  | GENERATION |         | ANALYSIS INPUT  |        | ONE LOADSTEP |  |
   -----------          ---------------         -------------     |
                      -------------------           |    ^    ^    |
                     | UPDATE DIE & WORK- |<---------     |    |    |
                     | PIECE GEOMETRIES   |               |    |    |
                      -------------------                |    |    |
                              |                          |    |    |
                              v                          |    |    |
              YES     -------------------                |    |    |
 --------------------| DIE-WORKPIECE        |             |    |    |
|                    | GEOMETRIES OVERLAP?  |             |    |    |
|                     -------------------                |    |    |
|                             | NO                       |    |    |
|                             v                          |    |    |
|                NO       ---------      YES              |    |    |
|           ---------------| MESH OK? |-------------------    |    |
|           |              ---------                          |    |
|           |                 | NO                            |    |
|           v                 v                               |    |
|      -------------       -------------                      |    |
|     | GLOBAL REMESH |    | LOCAL REMESH |                    |    |
|      -------------       -------------                      |    |
|           |                 v                               |    |
|           |              -------------                      |    |
|           ------------->| REZONE INPUT |---------------------    |
|                          -------------                           |
|                    -------------------       ----------          |
|                   | BACKUP ONE LOADSTEP; |   | RESTART &  |        |
 ------------------>| SCALE LOADSTEP SUCH  |--->| EXECUTE    |        |
                    | THAT DIE & WORKPIECE |   | PARTIAL    |        |
                    | WILL BARELY CONTACT  |   | LOADSTEP   |        |
                     -------------------       ----------          |
                                                   |               |
           ------------------------------------------              |
           v                  v                                    |
      -------------       -------------                            |
     | GLOBAL REMESH |    | LOCAL REMESH |                           |
      -------------       -------------                            |
           |                  v                 ----------         |
           |               -------------       | EXECUTE    |       |
           -------------->| REZONE INPUT |--------->| REMAINING  |-----
                           -------------       | LOADSTEP   |
                                                ----------

  MODIFIED-QUADTREE      MODEL CONTROL PROCEDURES     ANALYSIS PACKAGE
                                                      (MARC OR ABAQUS)
```

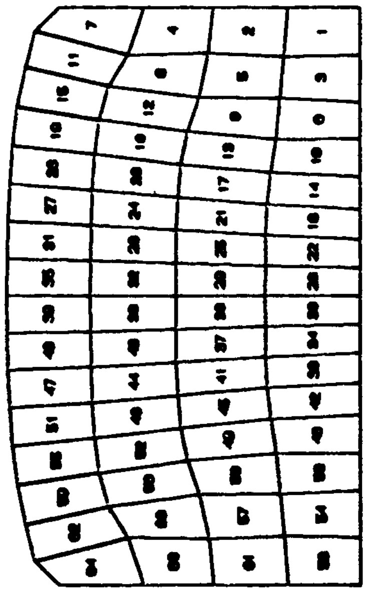
Figure 1. Integration scheme



zero increment

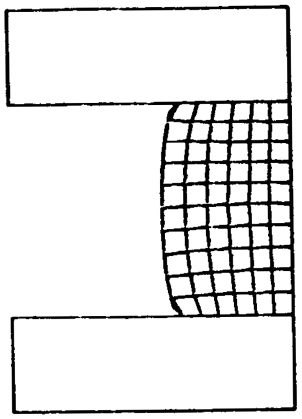


increment #35

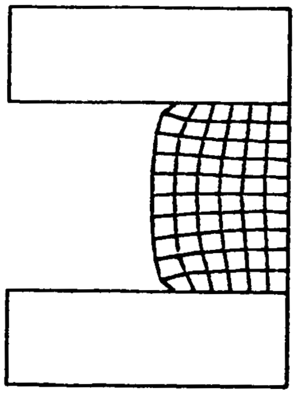


increment #35 with enlargement

Figure 2. Cylinder Upsetting



increment #35 after remeshing



increment #40 (restart/rezone at increment #35)

Figure 2(cont'd). Cylinder Upsetting

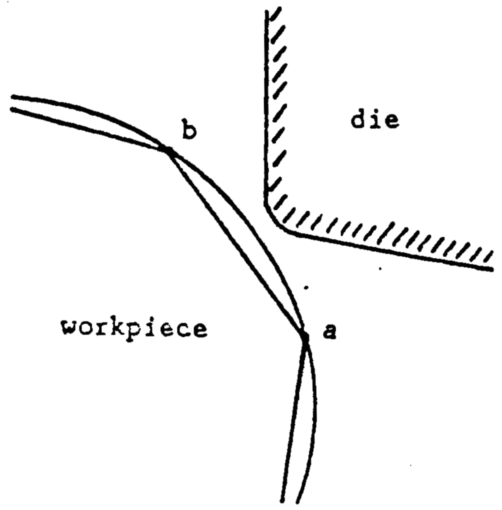


Figure 3.1. Workpiece boundary approaching a portion of the die surface.

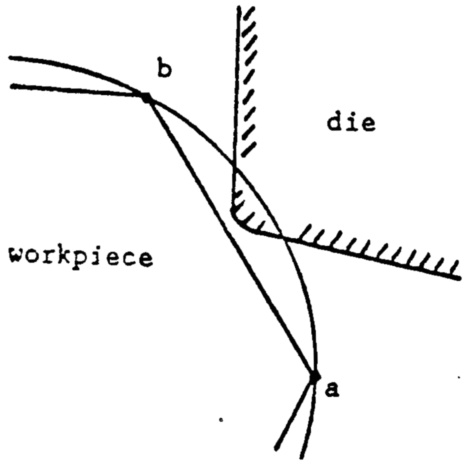


Figure 3.2. The geometric boundary of the object penetrating the die surface.

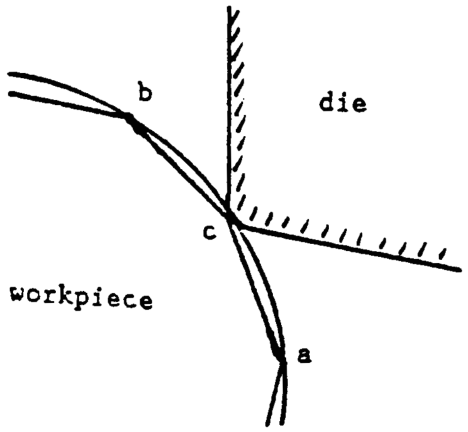


Figure 3.3. Initial point of contact between the workpiece boundary and the die surface.

AUTOMATIC TOPOLOGICALLY BASED MESH GENERATOR

Marcel Georges

SUMMARY

The goal of developing an automatic topologically based mesh generator is to produce extremely coarse, exponentially graded meshes suited for use in p-version finite element analysis. Geometrically based mesh generators are not capable of producing controlled, extremely coarse meshes which require the introduction of elongated elements.

It has been demonstrated that for elliptic problems with singularities, near optimal convergence can be obtained using complete p-refinement on properly constructed meshes. The meshes needed in these cases are coarse meshes with an exponential gradation factor of .15 emanating from all active singularities. For this class of problems, an element removal mesh generator, which operates by removing one element at a time from the object, is the best discretization procedure within an automated finite element modeling system. Although the two-dimensional topologically based mesh generator, is capable of producing exponentially graded meshes by exponential placement of nodes on the boundary curves emanating from a singular point, the resulting meshes were sometimes deficient for optimal due to either lack of control of elements at singularities or the production of higher number of elements than desired. By the addition of a fourth operator, SINGULARITY REMOVAL, to the three existing operators, optimal meshes are obtained. Optimal meshes for hp adaptive finite element procedures require an initial proper mesh to carry out an initial analysis with low p-level. Based on a posteriori error estimator and indicators, which have been developed by Babuska and Rank, the mesh is refined around the appropriate singularities. The mesh refinement is carried out by increasing the number of elements around singularities by dividing the surrounding triangles in geometric progression.

RESULTS

A new meshing operator, denoted as SINGULARITY REMOVAL, was developed for use in generating near optimal mesh configurations. This operator is used to isolate and remove each singular point before the start of meshing process. SINGULARITY REMOVAL isolates singular points by removing one or more layers of elements working into the interior of the domain. The number of layers around singularities can be specified either before meshing starts where the specified number of layers are removed or after the meshing process is completed with the triangular edges emanating from the singular points which are split in geometric progression. Triangles which have the singular point as a vertex will have the new introduced nodes as vertices and quadrilateral elements are formed by joining the two old and two new vertices.

The first layer of elements is the minimum number of triangles that can remove a singular point and yet maintain a prespecified degree of element shape control. Studies have been conducted to determine the requirements on the shape of these elements and the minimum number of elements required to guarantee near optimal convergence when p mesh refinements is used. These studies have showed that the number of elements around singularities is not important as much as the shape of these elements. Angle of elements at the singularities should not exceed 90 degrees to maintain near optimal convergence when hp refinements are carried out. It was also shown that, quadrilateral elements are more appropriate to be used in the second layer and higher, because it gives better accuracy with lower number of finite elements (i.e. lower number of degrees of freedom, and less solution time) and make the refinement process much easier.

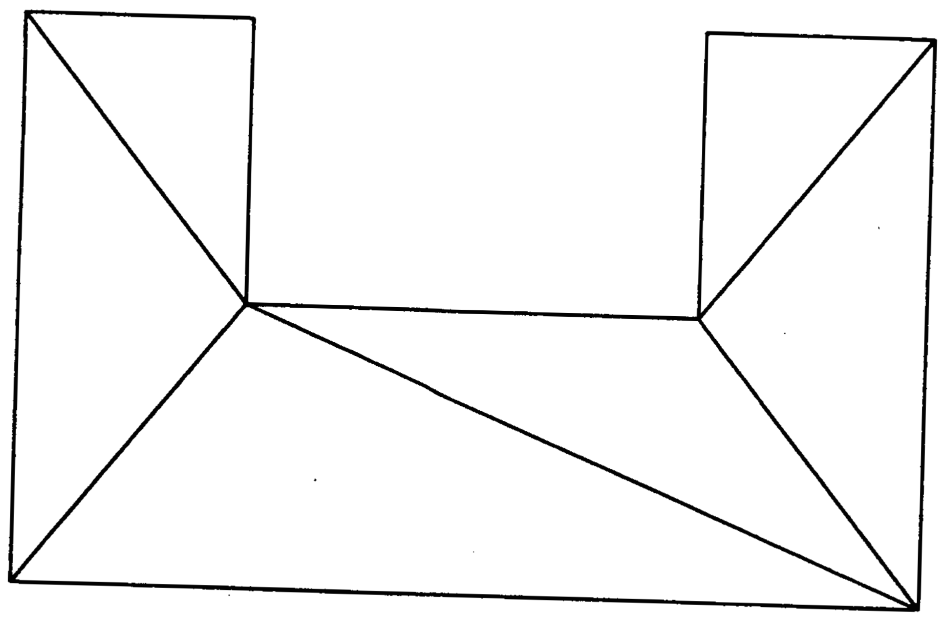
This meshing algorithm has been implemented and added to the program which has been modified and transferred to the CICG VAX computer. This program is still written in FORTRAN and running under the VMS operating system.

PROJECTED PLANS

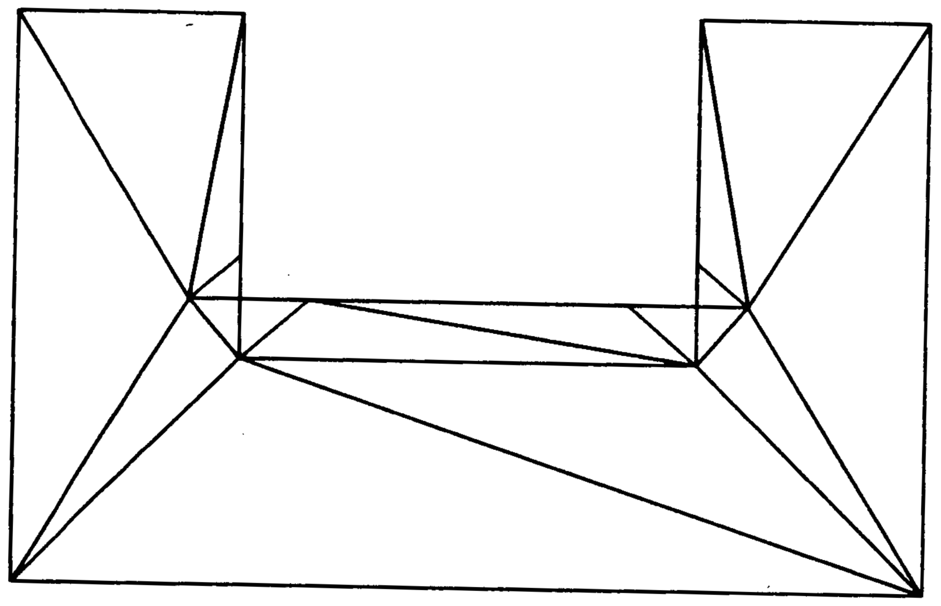
The goal of this project is to develop a fully automated hp finite element modeling system capable of discretizing any geometry into a valid finite element mesh without any user intervention. The input to the system is the geometric representation of the domain, the analysis attributes, and the desired level of accuracy. The output will be the results, to the prescribed accuracy, tied back the original geometry. The development of the proposed system requires the understanding of the requirements on the proper initial mesh design. This system will be capable of getting feedback from a posteriori adaptive finite element procedures about required mesh refinements.

Currently efforts are focusing on qualifying the detailed requirements of the meshes needed for optimal p-version adaptive analysis. This allows carrying out of the appropriate algorithmic alterations to the current mesh generator. It will then be extended to handle multiply connected regions with curved boundary edges and to begin the development of procedures for the meshing of three dimensional domains.

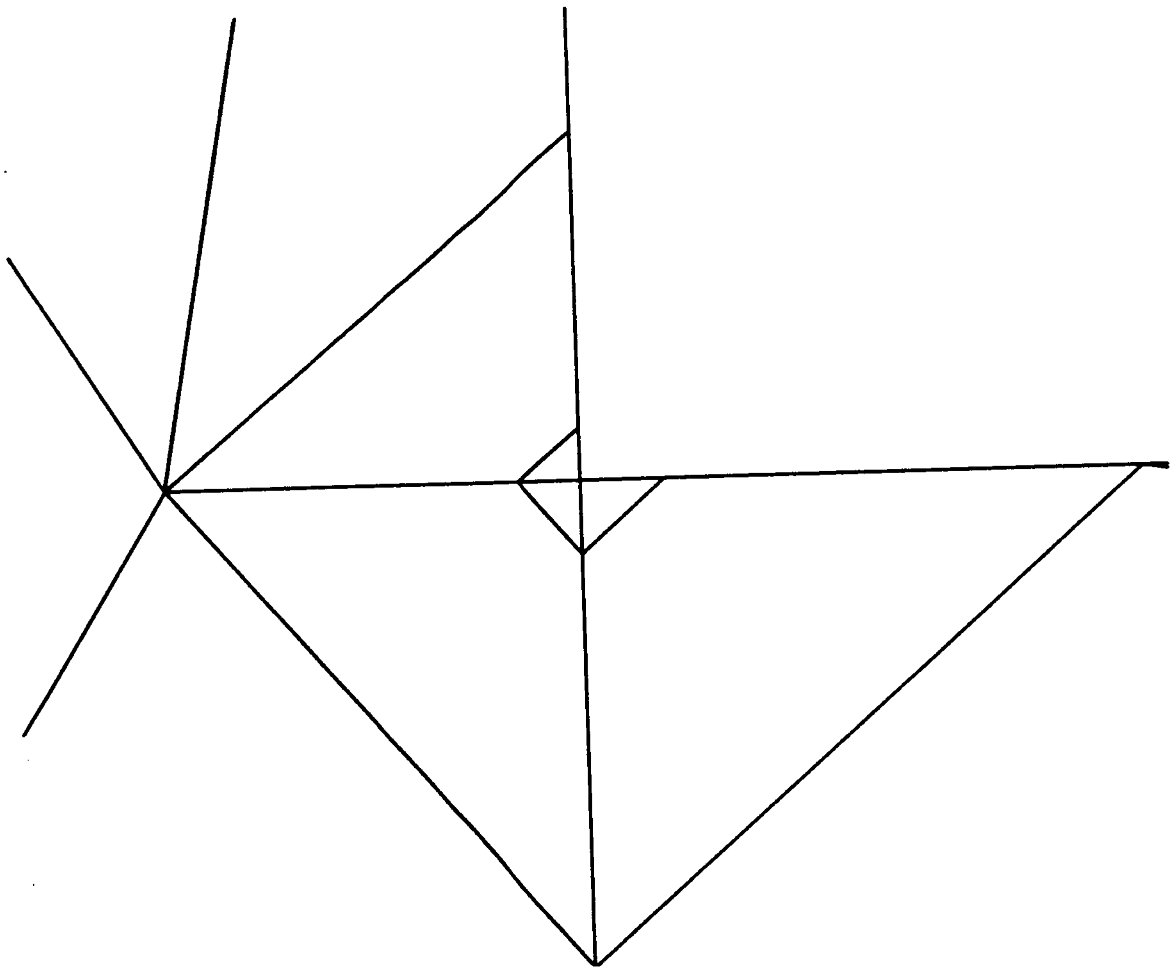
Efforts toward the development of a three-dimensional hp automatic finite element mesh generator will begun during the upcoming reporting period. Development of the three-dimensional capability will require the development of the meshing operators and the interface to a geometric modeling system. Unlike the two-dimensional case or the three-dimensional geometrically based mesh generator, the interaction of this type of mesh generators with geometric modeling is a complex issue. The operators will be designed to operate on the data structure of the solid modeler, which should store enough geometric and topological information to avoid the need for repeated searching or geometric interrogation. This is very important because topological integrity of the object has to be maintained throughout the paring process. These operators will be designed to use primitive boundary operators. The advantages of developing an algorithm based on boundary operators are that they can hide the actual data structures used and can be made more portable for interfacing with different kinds of solid modelers. Several forms of boundary-based data structures have been developed with the majority of them oriented for two-manifold solid objects. In a strict sense, a finite element model is not two-manifold. These are not the most appropriate one to house the finite element mesh. The new radial edge data structure, which has been developed by Kevin Weiler at GE for non-manifold geometric modeling, removes these constraints and can store both the geometric and finite element model in a consistent form. Thus the three dimensional algorithm will be integrated with the radial edge data structure which will be the basis structure used in the system.



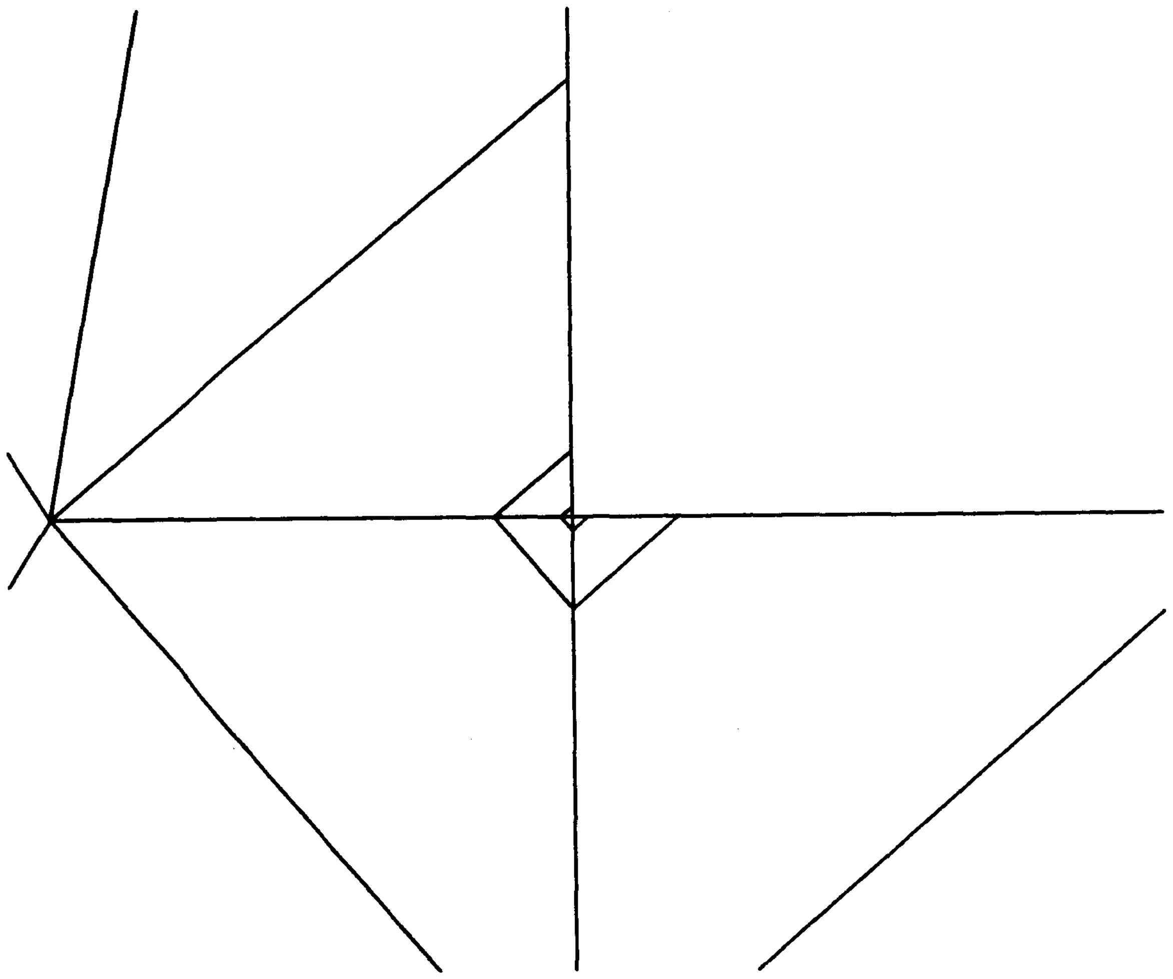
Mesh generated without applying SINGULARITY REMOVAL



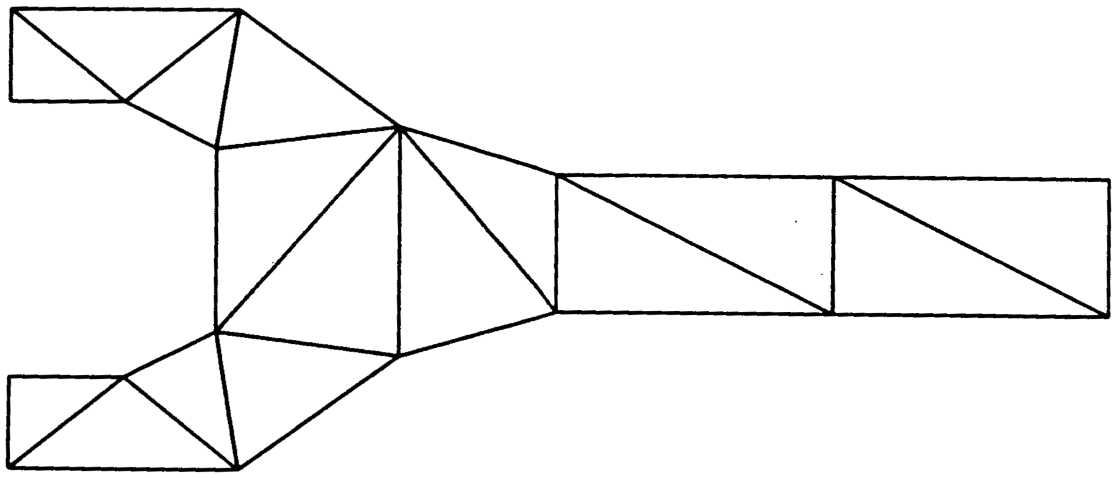
SINGULARITY REMOVAL has been applied at two vertices
and one layer of elements has been removed



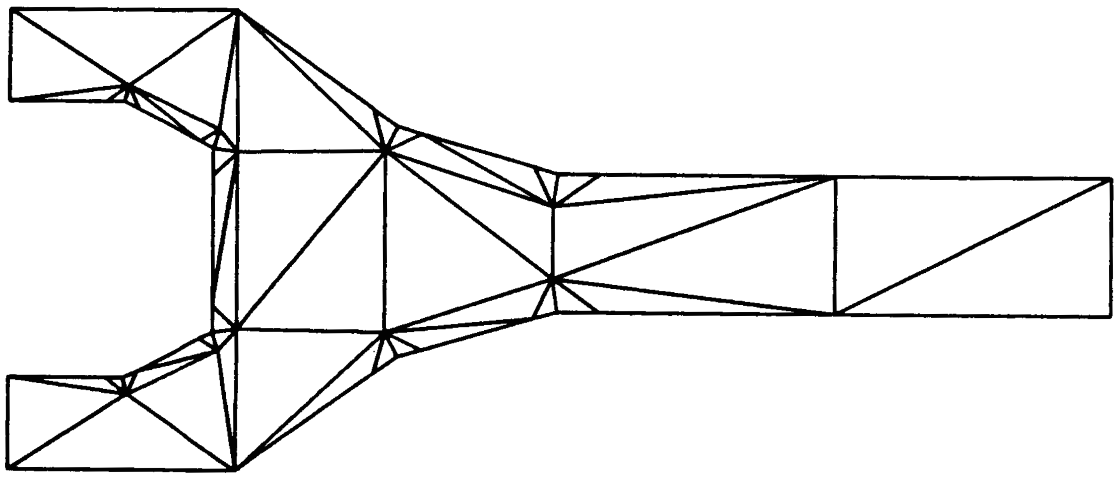
One layer of elements has been added to refine mesh around one of the singular points



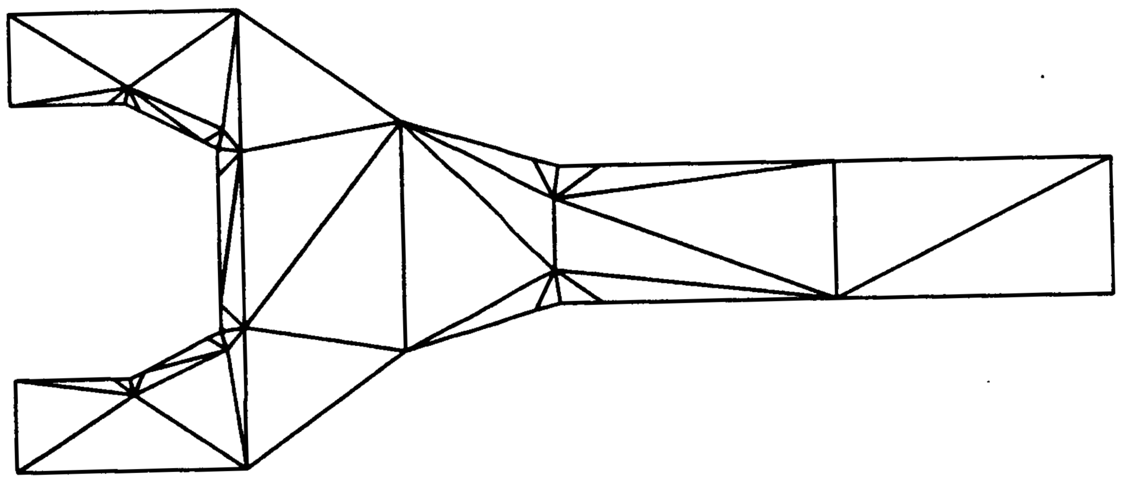
Mesh refined with two layers of elements has been added



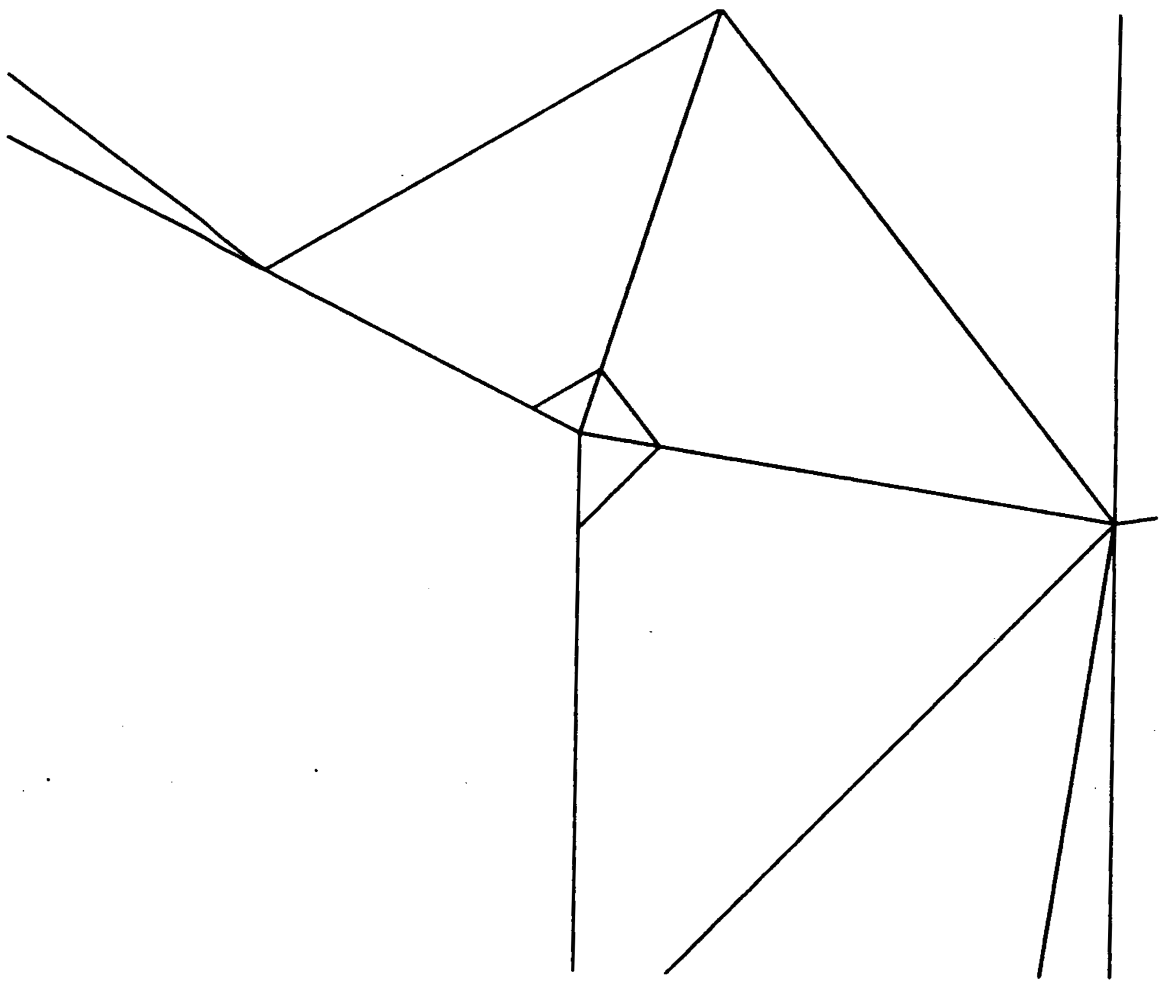
Mesh genrated without applying SINGULARITY REMOVAL



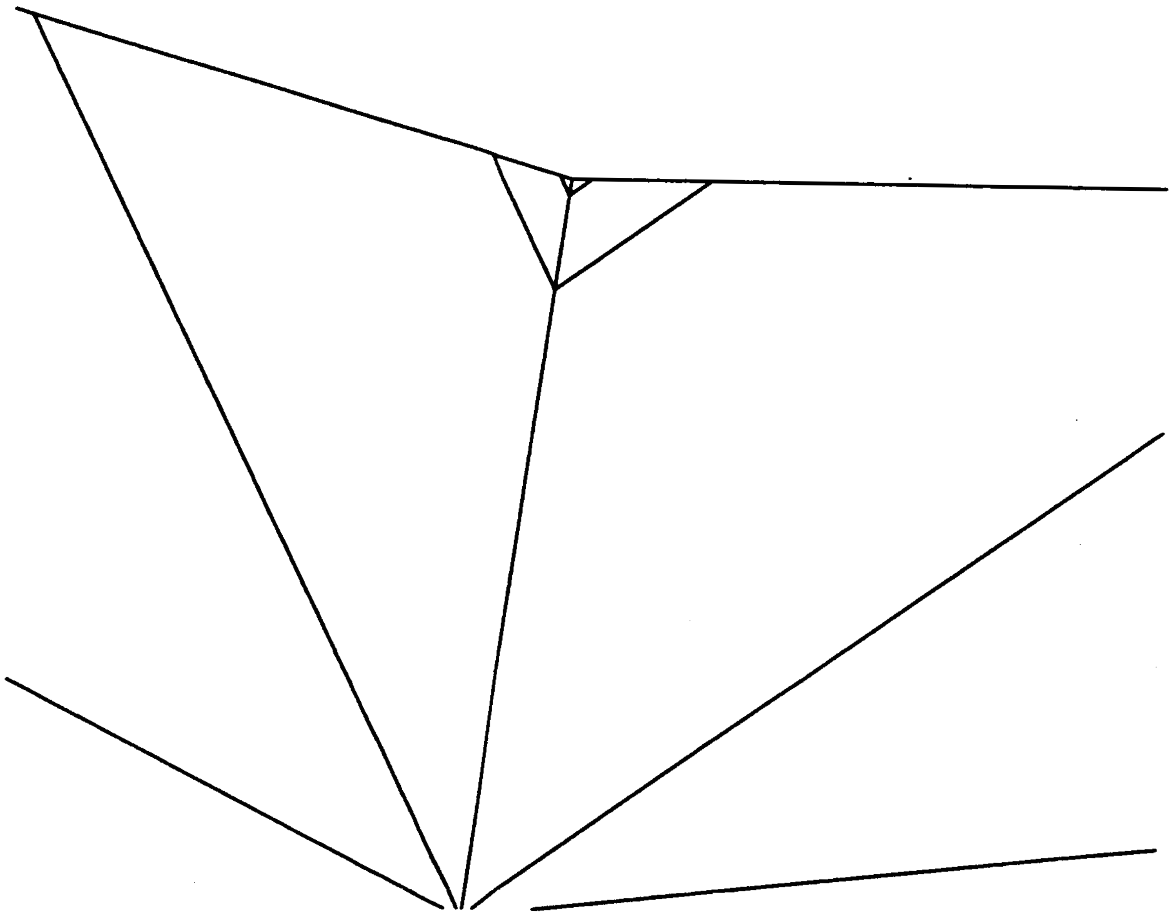
Mesh generated with SINGULARITY REMOVAL has been applied to all singular points



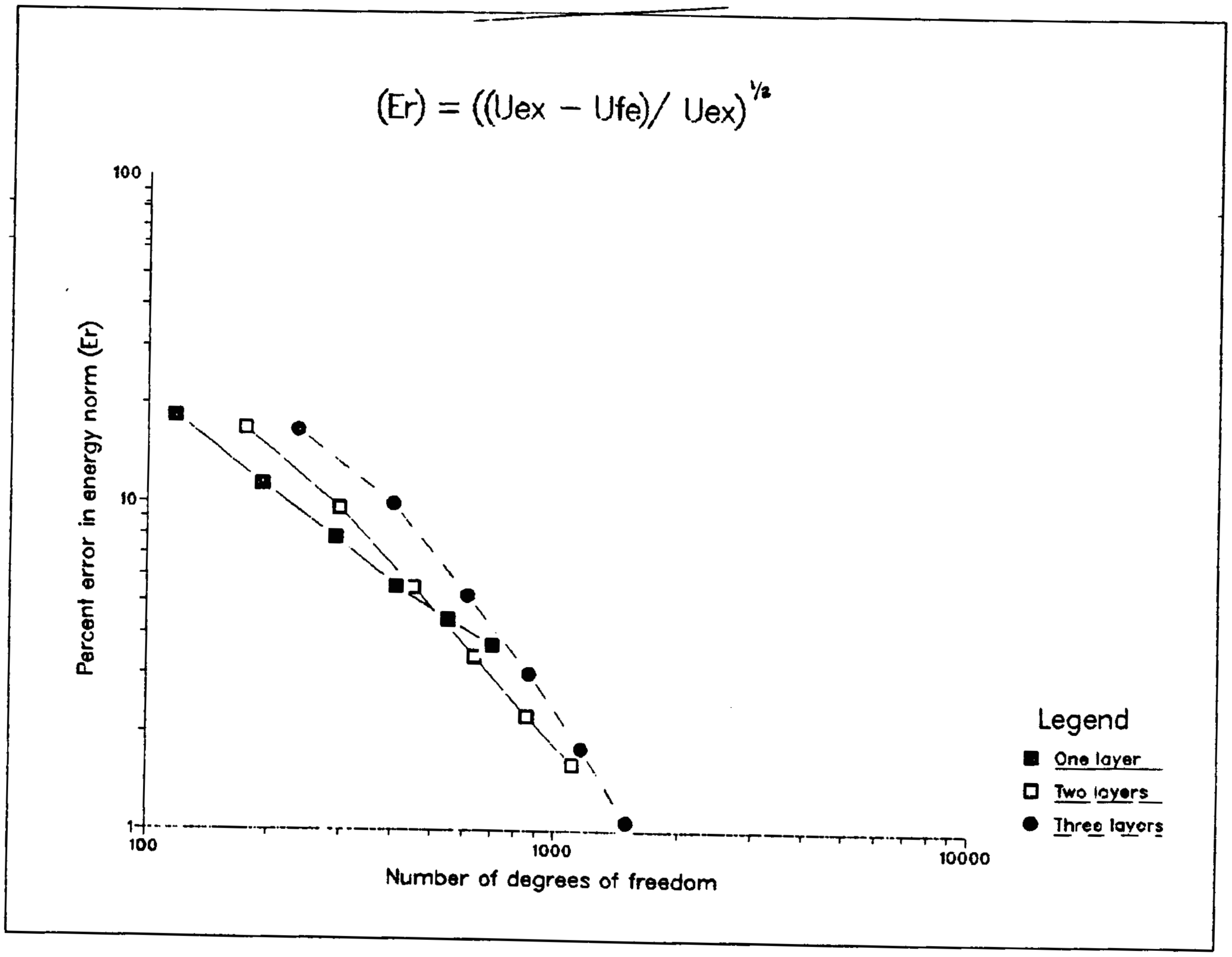
SINGULARITY REMOVAL has been applied only to indicated active singular points



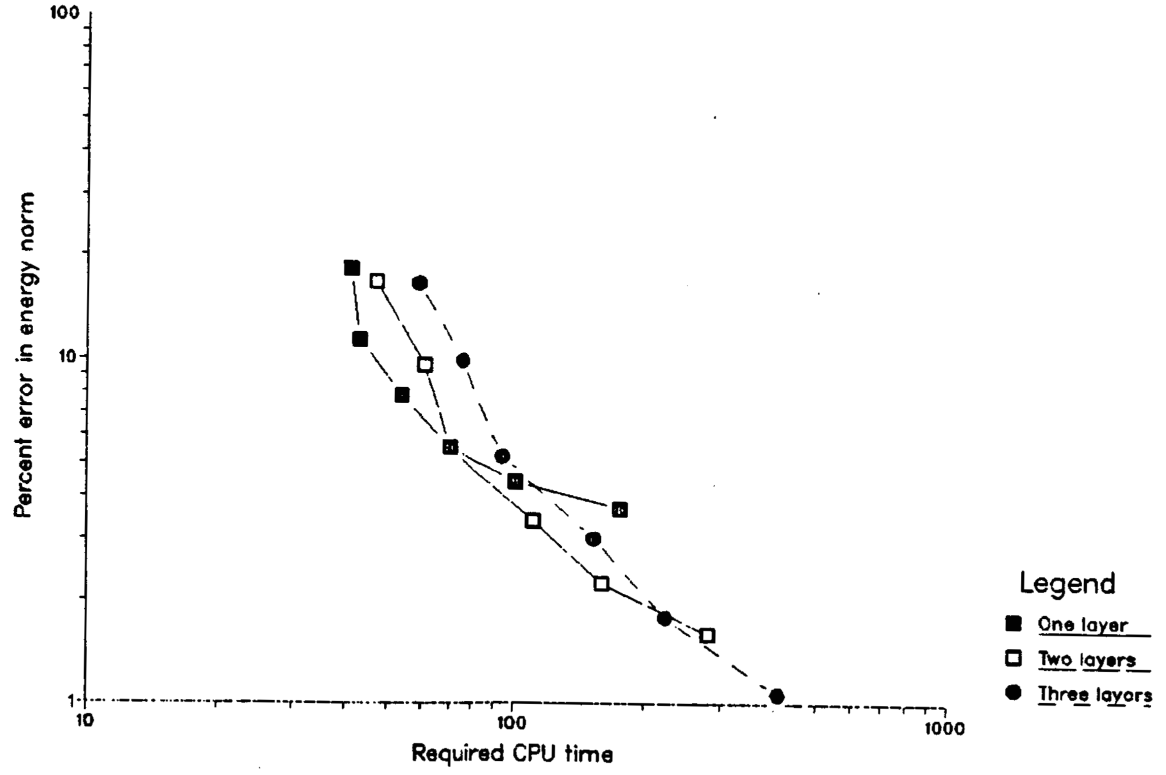
One layer of elements has been added to refine mesh around one of the singular points



Two layers of elements were added to refine the mesh
around different singular point



Percent error in energy norm versus degrees of freedom
for different number of layers at different p-level



Percent error in energy norm versus required solution time for different number of layers at different p-level

AUTOMATIC THREE-DIMENSIONAL MESH GENERATION USING THE MODIFIED-OCTREE TECHNIQUE

Kurt Grice

SUMMARY

The modified-octree technique is used as a tool for automatically creating solid, shell and wireframe finite elements from any physically realizable geometric model. Past efforts concentrated on a first pass through using the existing data structure and identifying the limitations of both the algorithms and the data structure itself. Since April, work continued on the design and implementation of the new data structure and algorithms that will eliminate the restrictions identified on the first pass.

RESULTS

The most significant restriction found in the previous version of the mesh generator was the limitation on the position of the model within the octree universe. Intersections of the various model topologies could not fall along boundary features of an octant, such as a model vertex being placed at a corner, edge or face of an octant. The more complex the model, the more difficult it was to insure this. In order to eliminate this restriction, both algorithmic and data structure changes were necessary.

In any octant, it was necessary to identify all discretized entities that in some way exist in it. However, as the process of discretization continues, additional entities would be added to the octants that may actually use the lower order topologies. Simply adding the new entities to the octants would cause a database explosion. Therefore, only "non-connected" discretized entities are stored in the octants list. For instance, if a model edge were discretized, the vertices for that edge (already inserted into the octree) would no longer be considered "non-connected" for the octants where the model edge intersects. The discretized edge, however, becomes a "non-connected" entity for each of the octants until the model faces are inserted. This process continues until the actual finite elements are formulated. The finite elements are not considered "non-connected" entities but are stored such that the octant to element links exist for later use in an adaptive analysis environment.

The above illustration identifies a need for specific storage of the various discretized entities. The discretization process builds a topological structure beginning with the vertex and ending with a finite element. Therefore, a complete topological structure is required with not only sufficient links between adjacent topological entities, but also adequate links between the octants and the original topological entities. To handle the adjacencies between the topological entities, a data structure based on Kevin Weiler's radial edge data structure was used (see Weiler Thesis August 1986, TR-86032). The actual implementation was a restricted set though, since certain assumptions about the topology of the finite elements could be made, such as all elements would be hexahedral or less, and that all element faces would be quadrilateral or triangular. This substantially reduced the storage requirements needed by the full implementation. The data structure for linking the octants to the topological entities was described above with the use of "non-connected" entities in the octant. However this only provides a one way link.

During the discretization process it is necessary to know what octants "own" the various topological entities. Keeping a linked list would be storage intensive. For instance, a vertex alone could be in as many a eight octants. Utilizing the fact that the octree is not just a convenient data structure but also a spatial discretization, and that there exists a correspondence between the two provides a means of identifying octants about selected positions in space. Therefore, the use of tree traversals by sending appropriate coordinates into the tree and returning the list of octants will supply the topological entity to octant information. The final finite elements, however, will include a direct link to the octant where they were formulated. This will assist in many applications requiring a local remeshing capability including adaptive analysis and metal forming.

Finally, many of the algorithms were enhanced to both handle the new data structure as well as other modeling issues. For instance, it is necessary to handle not just a solid that represents a volume, but also shells and wireframes as well. The need to handle non-manifold objects is very desirable for real applications. Previously, the modified-octree mesh generator would only handle true closed solids. Intersections of the octants with the faces of the model was partially done with in/out testing. Although efficient with implicit primitive expressions, this is a time consuming process for the Bezier patch surface representation used in the current modeler. In addition, it would not allow surface only meshes unless a closed volume was created. The enhancement to the modified-octree mesh generator not only allowed for surface only meshes on non-closed volumes, but also improved the throughput by reducing the number of intersection with the surface of the model during face insertion.

PROJECTED PLANS

By the beginning of 1987, the first release of the modified-octree mesh generator will be distributed to the Industrial Associates. Along with the work required to maintain our release schedule, projected plans include:

1. The use of real intersections representing the discretized points in the octree may cause small segments or cuts in the octants. These segments would be used as the edges and faces of the finite elements. Typically, the resulting shape of these elements is unacceptable. By removing the small segments from the definition of the octants, the poorly shaped elements can be avoided. The process is referred to as collapsing since typically intersections are collapsed down to know octant features such as corners. Scott Wittchen's work in collapsing for the modified-quadtree program (see Wittchen Thesis May 1986, TR-86003) will be used as a general guide for the octree procedures.
2. Interfacing with other geometric modelers including PADL and our own enhanced modeler being developed by graduate student Cathy Graichen. This work requires the alteration of the geometric communication operators that are dependent on the model definition. The operators have already been identified.
3. Integrate with an adaptive analysis to utilize the capabilities within the modified-octree for adaptive local remeshing.

4. Nodal repositioning along the boundary of the object and on the interior provide better overall element shape. and can also be used to enhance element gradations. The ability to reposition nodes. however. requires complex geometric operators as well as the ability to verify optimal element shape. This requires additional effort to make the process robust.

5. Graduate students Curt Soechtig and Jim Lo are currently working on improving tetrahedronizing in two areas. First. the algorithm is greedy by nature. That is, during finite element creation. given various choices of elements to create, the algorithm will take the best shaped element despite what could happen to the final elements. Second. the algorithm requires an a priori triangulation of the octant faces. The triangulation virtually defines the resulting tetrahedrons. Improved element shape could be achieved if triangulation could be altered during the process or developed as needed.

FINITE ELEMENT GENERATION FOR THE MODIFIED OCTREE AUTOMATIC MESH GENERATION TECHNIQUE

Curt Soechtig

SUMMARY

The current work effort entails the automatic surface triangulation of open and closed shells and the tetrahedronizing of solids using the modified octree spatial discretization technique. The finite elements for shells and solids are created on an octant by octant basis. The triangulation of shell surfaces is a non-greedy algorithm because it looks to remove the triangle with the best aspect ratio which prevents the creation of a triangle with the worst aspect ratio. For solid models the octant face and surface loops are triangulated using the shell triangulation method before any tetrahedrons are created. The current tetrahedronizing process is a greedy algorithm because it removes the tetrahedron with the best aspect ratio without looking at the possibility of tetrahedron with the worst aspect ratio.

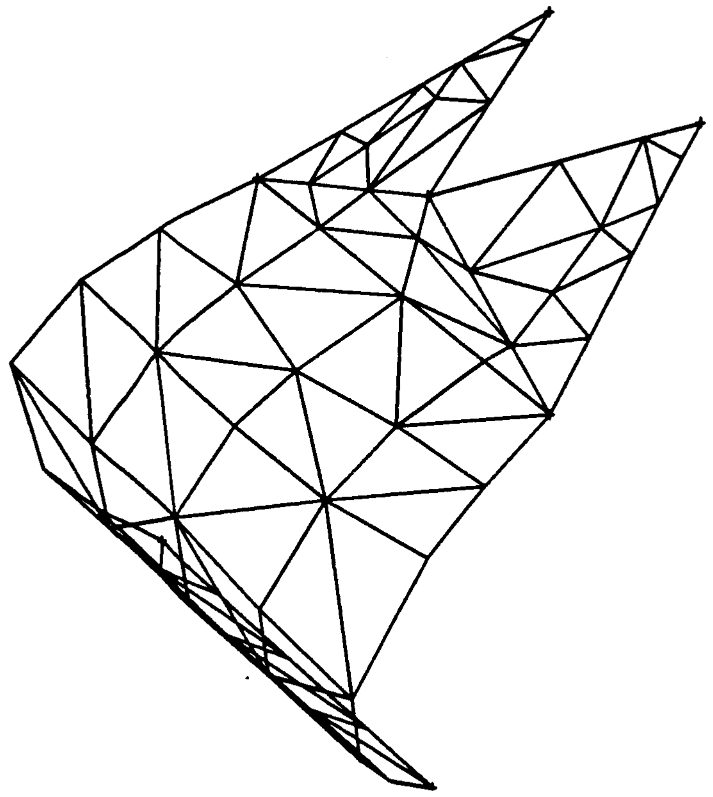
RESULTS

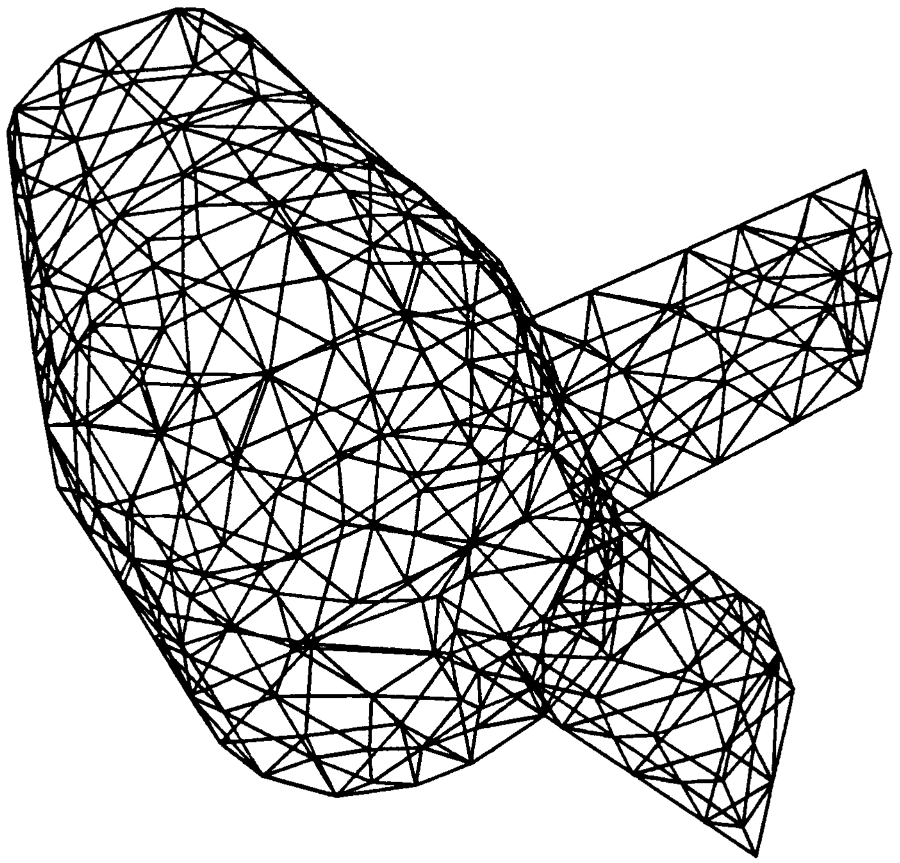
The main problem to be over come for the triangulation of shells is to detect for the creation of a triangular element which violates the model geometry. For solid models a connection which could cause a geometric violation can be detected by querying the solid modeler for in/out/on of any point on the new connection line which has been projected to the model surface. For the shell cases if the shell is open an inside does not exist for the model so the geometric modeler can not answer the query for in/out/on of a point. This problem is resolved by taking any new connection which is in question and projecting it to a linear discretized topologically equivalent representation of the shell surface. If the point lies on the discretized surface the new connection does not violate the model geometry else a geometric violation occurs. This method for determining geometric violations can be used for solids because a solid is just a special case of a shell. Figure 1 is an example of an open shell mesh and figure 2 is an example of a closed shell mesh.

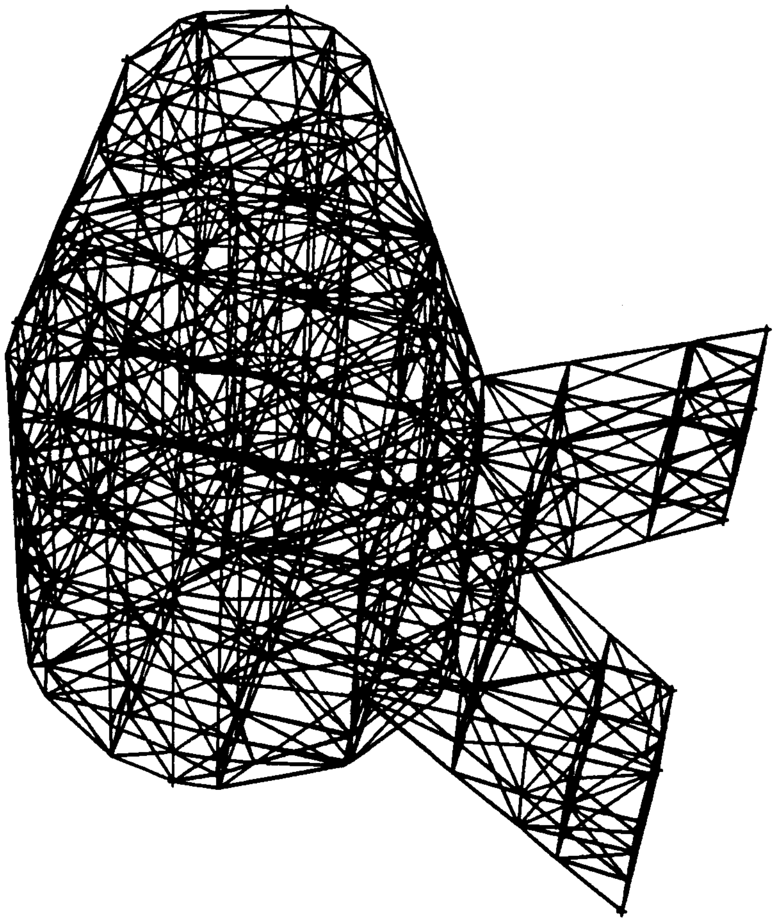
The tetrahedronizing routines currently remove tetrahedrons from a previously triangulated octant one at a time. The criterion for removal is based on two factors. The first factor is the amount of decrease in octant complexity caused by the removal of a particular tetrahedron. The second factor is the aspect ratio of the tetrahedrons ((volume)^2/(surface area)^3). The aspect ratio is used only when a number of tetrahedrons exist which decrease the octant complexity by the same degree. Checks have been made for special cases to disallow tetrahedron removal which have been known to cause tetrahedrons with extremely poor aspect ratios. Also, if no satisfactory tetrahedron can be found a point is placed at the centroid of the octant and the tetrahedrons are defined by the connections of the end points of all the surface and face loop triangles with the centroid. Centroid tetrahedron creation is not preferred because it generates more tetrahedron elements. Figure 3 is an example of a tetrahedron mesh for a solid.

PROJECTED PLANS

1. Rewrite the tetrahedronizing routines so they do no require a completely triangulated octants, but can employ a partial triangulation.
2. Handle special cases which are currently tetrahedronized using the octant centroid differently so as to create less elements.
3. Finish the conversion of the current tetrahedronizing routines into a non-greedy algorithm.
4. Placement of mid-edge nodes for higher order elements.
5. Creation of different shell finite element topology types such as quadrilateral, mixed triangles and quadrilaterals, etc..
6. Creation of different solid finite element topology types such as hexagons, pyramids, wedges and mixes of these with tetrahedrons.
7. Conversion of the overall modified octree mesh generation process into a form suitable for parallel computation.







GEOMETRIC MODELING CAPABILITIES FOR MODIFIED OCTREE FINITE ELEMENT ANALYSIS MESH GENERATION

Catherine Graichen

SUMMARY

In order to generate meshes using the modified octree technique, there are several requirements on the geometric modeling capability. These include complete boundary file information and the ability to interrogate the modeler about relevant features of the model using geometric operators. These geometric operators are:

1. Determine if a point is inside, outside or on the surface of the model.
2. Determine where (if at all) a line segment intersects a given surface.
3. Determine where an edge intersects an arbitrary planar surface.
4. Determine the nearest point on a specified surface to a given node.
5. Determine the normal to a specified point on a given surface.
6. Convert the parameter value of a point on a curve or surface to the xyz coordinates.
7. Convert the xyz coordinates of a point to the parameters for the curve or surface on which it lies.
8. Return the first and second derivative values of the surface at a given point.

The last three geometric operators are necessary for various sections of the smoothing algorithm.

The geometric capability provided with the mesh generator is intended to meet these needs to test the mesh generator and integrate some of the concepts of automatic mesh generation into other larger application areas such as shape optimization. However, the goal is not to implement all the abilities of a solid modeler, since the mesh generator will eventually be linked to commercial geometric modeling systems.

Currently, the geometric modeling capabilities include a bottom-up modeler which uses Bezier curves and Bezier triangular and rectangular patches to define the surfaces. All the necessary geometric operators have been written for the Bezier geometry.

To enhance the user interface, recent efforts have emphasized the addition of primitive element definition. The goal is to provide higher level capabilities and reduce the tedious task of determining several Bezier control points for complex objects and spending significant effort and time to form the curves and patches properly. Primitives should allow object definition to occur in a more natural manner. For instance, a cylinder is defined by the center points of the two circular faces and the radius. The remaining information such as points on the circles is derived from the specified parameters.

For applications, such as shape optimization, the modification of primitives will allow only the original parameters to be changed. Further work will allow primitives to be joined to form more complex objects in a straightforward manner. These efforts should dramatically improve the user interface for defining geometric models.

RESULTS

1. The definition of cylinders through the use of circles for curves, planar circular patches, and cylindrical patches. The user specifications for a cylinder are the center points of the two circular faces and the radius. This information is used to define a point on each circle, the circles that bound the faces, and the patches which describe the surfaces (two planar circular patches for the top and bottom and the cylindrical patch). The associated topology is also defined.
2. The intersection routines for the circular and cylindrical patches. To determine whether an intersection occurs on one of the patches, the patch information and the line segment information are transformed to a specific coordinate system. The intersections are then determined analytically and the results are transformed back to the original coordinate system and returned.
3. Design and implementation of a new menu structure which will allow more flexibility as more features become available for geometric modeling and during the mesh generation process. The new menu structure and implementation provide a more logical breakdown of functions by providing submenus in each work area (geometry and mesh). The tasks in the submenus are divided into two classes, utilities and functions. Utilities include viewing and I/O tasks. Functions consist of geometry definition and modification in the geometric work area and mesh control specification, octree insertion, and mesh generation in the mesh work area.

PROPOSED PLANS

1. Definition of additional primitives and the new curves and patches that are needed for these definitions. Additional primitives will include items such as cylindrical wedges, spheres, spherical section and boxes. New curves to support the above items will include arcs and line segments. New patch types will include pie shapes and spherical patches.
2. Develop methods to combine primitives to form more complex objects. These methods must include a method to determine the appropriate intersection curve and a method to create the new surface description as well as updating the appropriate topological features. In general, complex objects will use the new types of patches and curves to form the objects. More than one curve may be associated with an edge and several patches together may describe the surface associated with a face. The multiple patches and curves may be of several types or all the same.
3. Implement remaining geometric operators for circular and cylindrical patches and the geometric operators for new curve types and new patch types as they are developed.
4. Implement a more extensive data structure. The new data structure supports extensive attribute information. The extension will allow the topology to be more flexible and will enable the support of more complex analyses in the future, such as metal forming analysis. The data structure will allow the addition of other capabilities such as automatic shell definition.

EXTENSION OF MODIFIED OCTREE AND MODIFIED QUADTREE GRAPHIC INTERFACES TO SUPPORT IBM graPHIGS DEVICES

Richard Brunner

SUMMARY

As a continuing effort to extend the unique capabilities of modified octree and modified quadtree mesh generators to users on a variety of systems, the graphic interfaces of these programs are being supplemented with new routines that support IBM users with graPHIGS capability. Thus any IBM VM/CMS user with graPHIGS capability can run modified octree and modified quadtree mesh generators using any graPHIGS supported IBM graphics terminal (IBM 5080, IBM 3250, etc.).

At present, the graphic interface routines already support some graphic terminals on DEC and PRIME systems. In addition, there already exists routines that support the IBM 5080. However, with the availability of graPHIGS, the opportunity to support a wider range of IBM terminals with one set of routines is now possible.

RESULTS

The project has been broken up into two phases. The first phase will involve IBM graPHIGS emulation of the subset of the PRIME HGP graphics functions used in the original graphic interface routines. This will provide the initial capability of running the mesh generators on any graPHIGS supported terminal on an IBM system. From analysing their performance and usage during the first phase, it will be possible to determine which of the graphic interface routines need to be rewritten directly in graPHIGS during the second phase, thus taking advantage of graPHIGS's 3 dimensional functions and improving overall performance.

In addition, the functionality of the modified-octree three-dimensional graphics routines will be extended by utilizing other graPHIGS procedures.

As a result of performance issues and dissimilarities between the PRIME HGP package and IBM graPHIGS, only the subset of HGP used in the graphic interface routines are being emulated during the first phase. This constraint is crucial if the three-dimensional aspects of the programs are to run at any significant speed. Much of the FORTRAN source code for phase one has been completed.

PROPOSED PLANS

It is expected that testing of phase one will begin near the end of this month. From an analysis of phase one, the graphic interface routines needing alteration will be rewritten during phase two to directly utilize the 3 dimensional capability of graPHIGS, thus yielding higher performance. Furthermore, investigation will begin into extending the functionality of the modified-octree and modified-quadtree frontend programs by utilizing additional capabilities of graPHIGS that were not available in the HGP package when the programs were originally written. Phase two is expected to begin in January of next year.

STATUS OF PROJECTS ON HIGH LEVEL HIGH PERFORMANCE GRAPHICS SYSTEMS (PHIGS)

Salim Abi-Ezzi

SUMMARY

The group involved in developing and evaluating high level/performance graphics systems consists of one full time research staff and six graduate students. The group uses PHIGS, a proposed graphics standard, as a focal point. PHIGS is suitable for that purpose due to its powerful concepts and high level functionality. The initial project that the group worked on was an experimental implementation of PHIGS (xPHIGS) which was completed in August 1985 (1). xPHIGS provided an environment to study and evaluate PHIGS which on many occasions resulted in feedback to the standard development process.

Internally, xPHIGS resulted with a strong understanding of implementation issues of high performance systems, and triggered a new phase of projects in the following directions:

1. Adding extra utility functions to PHIGS
2. Developing applications on PHIGS
3. Designing and building a PHIGS machine

UTILITIES

Although PHIGS constitutes a solid base for high level graphics systems, a need was still felt for extending it with a number of extra utility functions. These utilities treat requirements of sufficient commonality among graphics applications. The functions were initially developed as extensions to xPHIGS and eventually migrated to graPHIGS (2), the IBM implementation of PHIGS. This was straightforward since the utilities are developed on top of PHIGS and hence are portable across PHIGS implementations. Most of the functions listed below are completed, however, some are being enhanced and some are still being developed.

1. The interactive shell: The shell allows entering PHIGS commands interactively. This utility may be used for two purposes: first, it is handy to test the effect of PHIGS routines which helps in developing applications, second, it may be called from an application in which case structures and tables may be inquired via PHIGS inquiry routines. This possibility makes the shell a handy debugging tool for PHIGS applications.
2. Archival functionality: This functionality allows the storage and retrieval of structures from files. It is part of the standard specification but it isn't supported by graPHIGS yet.
3. Hidden Line/Hidden Surface Removel: A technique based on the binary space partitioning (BSP) tree is developed to support hl/hsr (3). adding such a feature to PHIGS enhances the structure's modeling capability considerably.

4. Modeling Clip: Modeling clip is convenient for clipping a model before viewing, making it possible to instanciate a portion of a part in an object. The technique involves clipping to a convex region, possibly unbounded, defined by a set of arbitrary planes specified in modeling coordinates. This set of planes behaves like the name set attribute and may be changed via structure elements. Modeling clip was easily supported by a simple extension of the hl/hsr technique, see (3).
5. Shading: Due to some limitations enforced by the PHIGS functionality, it is only possible to support constant shading. The application gets involved in the process by setting a portion of the color table with consecutive shades of a certain color. These shades then get applied to the primitives depending on their orientation relative to the light source.
6. The Menu Tool Kit (4,5): A requirement of any interactive application is a menu system. The menu tool kit saves an application the trouble of dealing with the graphics details of such a menu system. The content of the menus and the parameters controlling their appearance are specified in a text file. This file is then processed by a utility that generates PHIGS structures for these menus and stores them in an archive file. At run time, the application calls a routine to retrieve these structures and another to request selections from the menu system for there on, the menu tool kit deals with managing the menus. The tool kit supports some advanced features; for details consult the references.

APPLICATIONS

An application's view of PHIGS is totally different from that of an implementer's. While the later's primary goal is to provide the functionality at a high performance, the former's goal is to use the functionality to develop efficient and effective applications (6). Currently, three different activities in this area are taking place:

GEOMETRY MODELING PACKAGE

This is an interactive package for constructing 3D geometric models (7). One of the package's targets is supporting a friendly user interface that makes the hierarchical data grouping mechanism, supported by PHIGS, available to the user. Another goal is developing 3D input techniques out of the provided PHIGS logical input devices. The final intended product is a tool for developing and saving charts as well as illustrations of possibly annotated geometric models.

CONICS TUTORIAL

This tutorial illustrates the characteristics, properties, and applications of conics. The package relies heavily on the dynamical articulation of graphics to emphasize some of the conics concepts. A major feature of this application is showing the relationship between the algebraic and geometric forms of a conic. The user can interactively alter some parameters of the algebraic form and observe the changes to the corresponding curve. In some cases, a slight adjustment to a parameter results in a change in the type of the conic. For example, from a parabola to an ellipse. An interesting point observed from this application is that editing modeling matrices is sufficiently flexible to support a large class of graphical data manipulation. This is particularly true if the graphical data is properly grouped in PHIGS structure networks.

PHIGS TUTORIAL

The idea behind this tutorial is to use PHIGS for teaching PHIGS; not a trivial task given the complexity of some of the PHIGS concepts. However, such a tutorial is extremely useful for introducing PHIGS to application programmers. The application is still in its early stages now, and numerous ideas that draw upon computer aided instruction and user interfaces are being considered and evaluated.

CATIA™ BACKEND

This application includes a utility that converts CATI™M models, read from CATI™M files, into PHIGS structures. These structures then get used to color, shade and view the original model from different angles. The main idea is providing rendering features to be compounded to CATI™M's superior modeling capabilities.

THE PHIGS MACHINE

While PHIGS specifies the functionality of an advanced graphics system, what is still needed is the hardware that can support such functionality at an adequate level of performance. This leads to the PHIGS machine project (8). The idea behind the project is to use off-the-shelf components to build a host independent PHIGS machine suitable for a micro environment, whether it is workstation or PC based. The machine, called the PHIGS experimental and testing engine (PETE), is based on a pipelined architecture and consists of a set of cards bundled in a box. The box will drive its own raster display, and will have access to input devices. Its interface to the outside world will be at the PHIGS level via a certain standard physical interface, which has not been decided on yet.

The main hardware components of the machine are completed. In particular, the PHIGS transformation/clipping pipeline is fully supported. Preliminary tests and demos show that the pipeline can perform at the stated goal of 1,000 vectors in real time (30,000 vectors transformed and clipped per second). One important feature of the system is that it supports 32-bit floating point data, full 4x4 transformation matrices, and clipping in floating point. This provides a strong answer to the criticism raised on PHIGS, regarding the difficulty of supporting it on integer or fixed point hardware. There is no need to stick to such hardware anymore since technology is beyond that point.

The status of the major components of the system is as follows:

1. The traversal card is an Intel 80286 based card with 0.5 mb of memory. Involved programming is needed for the card to support the PHIGS routines through a host language. There is a very good chance that this card will be the bottleneck of the system. In the future there may be a need to replace it with a card based on some faster processor such as the 80386, the 68020, or the transputer processor. The code, however, will be written in C and hence will be portable to whatever processor comes next.

2. The transformation card is based on the Weitek 1032/1033 floating point chip set and is currently the horsepower of the system (9). This section is critical to system's performance so it was designed to operate at full speed most of the time. To do this, it has FIFO buffers on its front and back, absorbing the possible variations in thruput of the previous and next sections. Microcode is developed for this card to do the floating point operations required by PHIGS, such as matrix multiplies and trigonometric operations. Furthermore, the card has local memory for storing the viewing table, modeling matrices, and the matrix portion of the environment stack.
3. The 80286-based output stage processor is equipped with a floating point co-processor which is also based on a Weitek chip set (1164/1165). This part of the system deals with clipping which is typically done in 3D, but may need to be done in homogeneous coordinates (4D) in the case of perspective viewing. The output stage processor also deals with driving the graphics card that carries on the last step.
4. The graphics card is currently based on special graphics chips from Xtar. The card is specialized in rasterizing clipped 2D primitives specified in device space (integers). Consideration is being given to replace this card with one based on the TI 34020 GSP chip. This would primarily provide more general support which is essential in such an experimental system. The generality will be needed for the future support of higher functionality such as text, hl/hsr, and shading.

SUPER PHIGS

All the previous activities resulting in confirming the PHIGS direction as well as suggesting additions to its functionality. These additions are primarily triggered via application requirements and/or implementation considerations. One area of extensions that will be supported by the PHIGS machine is the indirect specification of both modeling transformations and geometric primitives.

The extensions have positive implications to both implementers and application programmers. In the first case, modeling transforms still get specified from structure elements via indices to a workstation table. In the second case, primitives get specified via indices to a list of points associated with the corresponding structure, which allows sharing of points by primitives.

The introduction of table driven modeling transforms is a first step in supporting local input functionality. Indirectly specifying primitives helps in supporting higher level primitives (arcs, curves, or surface patches), boundary representation modeling, and smooth shading. Other areas that still need to be investigated include raster primitives and operations, and the behavior of PHIGS in multi-windowing systems. These areas are becoming common features in the new generation of workstations, and should be considered by modern graphic systems.

As a conclusion, we're working in a wide scoped environment that gives us different points of view on the features and needs of high level high performance graphics systems. PHIGS is being used as a guideline but is being extended with new concepts and more functions. These ideas may very well impact the future extensions to PHIGS which will eventually result with the perfect system: Super PHIGS.

REFERENCES

1. Abi-Ezzi, S.S. and Bunshaft, A.J.: "An Implementer's View of PHIGS", *Computer Graphics and Applications*, Vol. 6, No. 2, Feb. 1986, also CICG-TR86004.
2. Abi-Ezzi, S.S. and Hersh, J.S. "RPI Extensions to graPHIGS", CICG-TM86004.
3. Abi-Ezzi, S.S.: "The Priority Tree, a HL/HSR Approach for PHIGS", CICG-TR86019.
4. Kader S.E.: "PHIGS Menu Creator - User Guide", CICG-TM86018.
5. Kader S.E. and Abi-Ezzi S.S. "A PHIGS Generalized Menu Tool Kit", CICG-TR86039.
6. Abi-Ezzi S.S. and Kader s.e.: "PHIGS in CAD", *Computers in Mechanical Engineering*, Vol. 5, No. 1, July 1986, also CICG-TR86040.
7. Wang S.L., "Interactive Geometric Modeling Tool Supported by PHIGS", CICG-TM86014.
8. Abi-Ezzi, S.S. et al, "PHIGS on a Microcomputer", CICG-TM86002.
9. Greenberg, M.H., "Hardware Design for the PHIGS Transformation Processor", CICG-TR86035.

A 'C' BASED PHIGS BINDING FOR THE PHIGS HARDWARE PROJECT

Salim Abi-Ezzi
Darryl Champagne
Jorge Molina
Haruo Shimizu
Michael Toelle

SUMMARY

This project involves the further development of the PHIGS Hardware Project [reference summary] by the introduction of a PHIGS level binding. This binding, to be written in the 'C' programming language (with certain critical portions written in Assembly Language), will serve as an interface between the application program and the graphics sub-system. Some of the basic responsibilities of the binder are:

- Receiving PHIGS commands issued from the application program.
- Structure editing and storage, memory management, and input handling.
- Traversal coordination, including attribute binding and pipeline 'feeding'.
- The basic objective of the design is the development of data structures and algorithms that provide (1) enough speed to make full use of the high-speed pipeline, (2) flexibility and modularity for experimentation and further development, and (3) portability.

Another key goal of this project is host independence. In software, this is achieved through the use of a subset of PHIGS' commands to interface to the application. In hardware, this is achieved by means of a standard communications interface.

RESULTS

The design of the data structures and the general guidelines for the binder has already begun.

The initial implementation of this section of the PHIGS system will reside on an Intel 80286/80287 based system, and in view of this, it has been decided to use the Microsoft 'C' Compiler and the Microsoft Macro Assembler. The design and implementation of the hardware interface to the 'host' system is a relatively minor task, and will be done in parallel with the software development.

It is planned to complete the design of the binding in December of 1986, and to have the system completed with a PHIGS interface by May of 1987.

THE PHIGS TUTORIAL

Steven Kader

SUMMARY

Current responsibilities revolve around creating an on-line PHIGS tutorial. The goal of this tutorial is to enhance the understanding of PHIGS and ease the writing of application programs using PHIGS. The tutorial is divided into individual lessons, which will use the explain, interact, summarize method of tutoring. The major emphasis will be on the interactivity aspect. This approach is greatly needed in trying to teach a subject such as computer graphics.

In the initial stages of development of the PHIGS tutorial, certain utilities were needed but had yet to be developed. One example is a well defined menu system, which is necessary in an application of the magnitude of the tutorial. Since most interactive applications need some type of menu system, it seemed a waste for every application programmer to reinvent the wheel. For this reason, the project was to create a generalized menu system. The complete system has since been called the PHIGS Menu Tool Kit (TR86019).

RESULTS

The purpose of the PHIGS menu tool kit is to allow an application programmer to easily create and utilize a robust menu system. The menu system uses a hierarchical approach to menu design. Menus consist of items which point to another node in the menu hierarchy or to nil. The use of hierarchies makes the flow of the menu system logically defined, which helps both the application programmer and the user. One nice feature of the menu system is that an experienced user may jump around the hierarchy (i.e., lateral movement). The only problem with this is that the application program may want to restrict this movement. The solution is a higher level grouping of menus called clusters. The menu hierarchy may be divided into a number of logical clusters, each being a smaller menu hierarchy.

The use of clusters makes it possible to limit the jumping from one menu to another within the same cluster.

There are two parts to the Menu Tool Kit. In the first section the application creates a menu description file. This is a data file containing the detailed information about each of the menus in the desired menu system. The Menu Tool Kit then reads this file and creates corresponding PHIGS structures to map to these menus. These structures are then archived. The hierarchy of the menu system, specifically the pointers to next menus are handled by application data elements. Each item in a menu structure has an associated application data element. Therefore, when an item is picked the menu that it points to (assuming it is not a leaf item) is readily available through a structure inquiry. The second part of the menu system resides as two routines which are called by the application program. One routine is used to initialize the menu system. Initialization includes the retrieval of the PHIGS menu structures. Once the menu system is initialized, all the application program has to do is request a menu selection. The Menu Tool Kit will take care of the details and then when a leaf item is selected, or there is a change in cluster identification, the tool kit will return these three values 1) cluster id, 2) menu id, 3) item id. The application then uses this information to process the request.

Since PHIGS allows dynamic attribute binding of primitives it is possible to change the shape, color, and position of the menus with very little modification. A menu structure is actually made up of two structures. The first is an appearance structure which is global to all of the menus. This appearance structure contains transformations and attribute selection elements which dictate the look of the menus. The second structure is a basic structure with the items of the menus. All menu structures have the same structure. Examples of modifiable menus include having the menu at the bottom rather than the top, or yellow menus instead of red menus. This feature has not been implemented yet but would not be difficult to add at a later date.

PROJECTED PLANS

Future plans include completing the majority of the PHIGS tutorial by March 1987. In the process of working on this tutorial, many new ideas and tools will be developed for PHIGS. One example of this is the Menu Tool Kit as PHIGS lacks good text processing, a tool has been developed a tool which takes text from a file and formats it into a PHIGS structure, which can be used in the tutorial was developed. This allows modification of the text to be extremely simple. Simply change the text in the data file and re-create the PHIGS structure, which is then archived. These and other tools will enable the PHIGS tutorial to be as comprehensive as possible.

PHIGS HARDWARE PROJECT

Jorge Molina
Michael Toelle

PROJECT DESCRIPTION

This project involves the development of a specialized hardware/software system to implement the proposed graphics standard PHIGS. The motivation for this project is the need for a system which is able to handle the large processing requirements of PHIGS at interactive speeds, requirements which clearly call for the use of specialized high speed hardware.

The main criteria in the design of the system are performance and flexibility, while the limiting factors are cost and development time. The degree of flexibility of the system should accommodate new developments in the proposed PHIGS standard, and provide a modular environment in which to test new algorithms and ideas.

The system consists of four modules which break down the PHIGS processing into functional blocks: system control, transformation, clipping, and rasterization. These modules are pipelined, with FIFO buffers as connectors.

The system control module is responsible for structure storage and editing. During traversal, it extracts the structure components and feeds them down the pipeline. This module is based on the 80286/80287 processor set, for which many software development tools are available.

The primary responsibility of the transformation module is to perform the coordinate transformations during traversal. It also computes and stores transformation and viewing matrices, and initiates the clipping. This module is a microcoded processor, with downloadable microstore, floating point multiplier and ALU, and limited integer processing capabilities.

The clipping module is a secondary floating point processor which performs the clipping of graphical primitives, and the conversion to the integer format required by the rasterization module. This module is a microcoded processor based the AMD 2910 sequencer, and Weitek 1164/1165 floating point chips.

The rasterization module is responsible for receiving graphical primitives and attribute selections, binding them together, and generating the image frames on the screen. Currently this module consists of two sub-modules, one based on the 80286 and the other on the XTAR VLSI Graphics Microprocessor chip set.

Two of the key aspects of this design as they apply to PHIGS are: (1) the use of floating point number representations in transformation and clipping, and (2) the ability to calculate and store transformation and viewing matrices locally on the transformation module. Floating point data representations are critical to maintaining the accuracy of the data through the numerous matrix-vector transformations implied by the hierarchical structure of the PHIGS model. The ability to carry this accuracy through the transformation and viewing calculations makes it equally important to use floating point in the clipping process, where primitives of widely varying extents may be incorrectly clipped if limited integer representations are used.

The use of the transformation module to calculate and store the transformation and viewing matrices acts to distribute processing between the system control module and the transformation module and decreases the volume of data transferred from the system control module to the transformation module during traversal.

This decrease in data transfer volume is accomplished in two ways. The first by allowing transformation and viewing matrices to be pre-calculated and indirectly referenced (via matrix indices) in the structure during traversal. The second and most important use of this storage is the concept of an environment stack, which locally stores the global, local, and view matrices during the traversal of a child structure.

The microcode programming was aided by a set of utility programs written for that purpose, such as a symbolic microassembler, loader, and linker. Testing programs were written to evaluate system integrity.

The microprogramming on the transformation module provides a PHIGS-like set of instructions to the system control module. This opcode set include the generation of utility matrices, such as rotation around any of the three main axes, scaling, translation, and view matrix generation. Also included are opcodes to control the environment stack, aiding in the execution of child structures.

RESULTS

A first iteration of the system hardware has been completed and the main microcode programming blocks are finished. A set of demonstration structures to test the performance of the system, with the following results has been programmed:

- A structure with about 700 vectors can be transformed, clipped, and rasterized within 1/30 of a second.
- Structures containing more than 2000 vectors were continously regenerated with a reasonable degree of interactivity.
- Programs displaying hierarchical structures proved to be easy to design and debug.
- Multiple-view of structures was tested. Again, the programming proved to be very easy.
- The current bottlenecks of the system are apparently found on the 80286 processors used. Especially the one in the rasterization module. More testing still needs to be done to confirm this supposition.
- Microcode development for the system, while not a trivial task, was shown to be manageable, given the tools designed for that purpose.

These results are encouraging. Specifically, it is known that certain portions of code can still be streamlined, to make the system even faster. The original target of 1000 vectors within 1/30 of a second seems a feasible goal.

FUTURE WORK

The next phase in the developemnt of the system concerns the PHIGS binder. This is a program running on the system control module which receives PHIGS commands from a host system and input from several input devices. It performs structure editing and coordinates the traversal.

Efforts are also continuing in identifying and removing the bottlenecks in the pipeline. The flexibility of the two microcoded processors and the general purpose processors makes it easy to redistribute processing throughout the pipeline.

Work is being done on the replacement of the rasterization module by another with more capabilities. This would simplify the implementation of some PHIGS-specific details. such as the various forms of handling text. fill area. and fill area set.

Eventually it would be desirable to include on the system some functions not currently provided by PHIGS. such as hidden surface elimination. shading. etc. This is another reason for a more powerful rasterizing module. The current decision is to build this module around the Texas Instruments TMS-34010 processor.

Evaluation of the proposal for replacing the clipping module by the board that is currently the transformation module. and building an even more powerful transformation processor is in process. This would allow the clipping module to aid in the preprocessing of rendering algorithms, increasing the system's performance.

Much work can be done to reduce the physical size, power consumption. and cost. by using programmable logic arrays. The next iterations of the hardware would use them, along with more powerful commercial integrated circuits.

Another area of experimentation is dynamically tying local input devices to structure elements (matrices). This would allow a PHIGS workstation to do interactive display without the intervention of an application program.

Finally. consideration for support of more complex graphical primitives. such as arcs. b-splines, general conics. etc. with the aid of some special-purpose hardware modules that would be inserted in the pipeline is being investigated.

CONCLUSIONS

Success in obtaining a relatively low-cost system which allows fast interactive performance has been obtained. The system was designed to make future additions easy, which provides an excellent environment for testing new algorithms and specialized pieces of hardware.

INCORPORATING MODELING CLIP AND SHADING INTO PHIGS

Robert O'Bara

SUMMARY

A hidden line/hidden surface removal, (hl/hsr), algorithm based on the bsp (binary space partitioning) tree was implemented on top of PHIGS. This project deals with the extension of these routines, which are written in VS PASCAL, in order to provide the additional utilities of model clipping and shading.

A model clip is the result of introducing a clipping plane into the modeling coordinates of a structure. This provides the ability to view different cross-sections of a structure which is composed of complex internal sub-structures. Introducing shading into PHIGS provides a more realistic and understandable display of a structure.

In addition to supplying these routines, the project must also redesign the data structures in the hl/hsr routines in order to increase the storage efficiency of the routines.

RESULTS

A new data structure composed of a linked list of arrays has been designed and integerated into the hl/hsr routines. The new structure increases memory efficiency and eliminates the routines' previous upper limit of fifty points per primitive.

The algorithm for model clipping creates a list of clipper nodes which is located at the root of the bsp tree. A clipper node contains the equation of the model clip plane in world coordinates. The clipper behaves like a filter. It allows only the parts of primitives that are located in the positive half-space of the clipping plane to be passed down to the next node. The information for a clipper is stored as an application data element in the structure. The information consists of the normal of the model clip plane and a point on the plane. Both the normal and the point are specified in model coordinates. The normal of the plane points in the direction of the positive half-space of the plane.

The model to be used in the shading algorithm is constant shading with a single light source located at the viewer's position. There are two reasons for choosing constant shading over smooth shading. First, information concerning common vertices between polygons is not stored in the PHIGS database. Second, there is no method provided in PHIGS to render points in the same primatives' different shades. The reason for placing the light source at the viewer's position is to avoid calculating shadows and occlusions.

Each polygon to be shaded in the bsp tree has associated with it a shade table index and a set of shading parameters. The parameters include the type of shading, (parallel or perspective), and the attenuation properties of the surface and the light source. This information is inserted into the structure as an application data element. Other information needed is the angle between the surface normal and the viewing direction. In the case of perspective shading, the distance from the light source to the center of the polygon is also required. Both the angle and distance can easily be calculated given the surface normal, which is stored in the bsp tree, and the viewing information.

A shade table stores the RGB values of the shades ranging from ambient light to saturation color. The table is a subset of PHIGS' color table and each shade table has an index associated with it so that it can be referenced during shading. A shade table is set by calling the routine GXSST. The parameters passed to the routine are (1) the shade table index, (2) the number of shades to be stored, (3) the starting index in the color table for the shade table, (4) the ambient color, (5) the saturation color, and (6) the relative shade ratio, (which specifies the relative changes between RGB components between shades).

FUTURE PLANS

The next stage of the project is to completely test out the new data structure. Following this step will be the implementaion of the model clip and shading algorithms. After intergrating shading into PHIGS, an investigation into possible modifications of the algorithm, which will allow multiple light sources, will be done.

PERFORMANCE IMPROVEMENTS IN GDP RAY TRACER

Elaine Korngold

SUMMARY

The Geometric Design Processor (GDP) is an interactive graphics system which models complex three-dimensional objects using Boolean combinations of primitives, such as cylinders, cuboids, and cones. GDP objects are ray traced using a high quality image renderer developed at RPI last year. Industrial designers at IBM use the ray tracer to view photographic-like images of computer models without building the models.

Ray tracing involves shooting rays (vectors) from the viewer into the model area, determining which object is hit by the ray, computing the shading of the object and displaying it. Naive ray tracing checks every object in the model for possible intersections. This checking takes up to 95% of ray tracing time. Two independent techniques exist for improving ray tracing performance: object coherence, and spatial coherence. they are both aimed at decreasing the number of objects each ray has to intersect, and they offer approximately a twenty time improvement over the naive method. The goal of this work is to implement both techniques in the GDP ray tracer to achieve more optimal performance.

RESULTS

The object coherence method for increasing ray tracing speed was implemented in GDP ray tracer. Twenty test models were created to monitor performance improvements. The models contain 8 to 1000 primitives objects. The object coherence technique consists of three steps:

1. During preprocessing complex objects consisting of a large number of polygons are bounded by tight extents that are inexpensive to intersect. The extents are parallelepipeds consisting of planes lined up with coordinate axes.
2. During preprocessing a hierarchical tree of bounding extents is created. The hierarchical tree is a copy of the GDP model tree. The leaves of the tree contain the actual objects. Each intermediate node of the tree contains a bounding volume of its children. During ray tracing the children of a given node are tested for intersection with the ray only if the ray intersected the bounding volume of the node.
3. During ray tracing the hierarchy is traversed for each ray to find which object at the leaf node level should be shaded. The traversal is implemented as a priority queue with a partial heap. During the traversal, the node that lies the closest to the ray is examined first.

As a result of implementing the object coherence technique, GDP ray tracer runs 20 times faster on large test models.

PROJECTED PLANS

Object Coherence

It has been proposed that the model hierarchy created by a designer is not optimal for ray tracing. Plans are underway to study various techniques for building a more optimal hierarchy.

Spatial Coherence

Spatial coherence method for complex polyhedrons will be implemented.

Animating GDP Models

The possibility of animating GDP models using CLOCKWORKS will be examined.

IMPLEMENTING TEXT AND FONTS IN IBM'S GEOMETRIC DESIGN PROCESSOR

David Shen

SUMMARY

The geometric design processor (GDP) is an interactive three-dimensional solid modeler which creates objects using combinations of various primitives such as cuboids and hemispheres. Unfortunately, GDP has no facility to insert filled text on to models. Now, a new primitive will be added to GDP called an FTEXT (short for "filled text"). Standard fonts used by IBM on their products will be entered into a database where a user can select the font and place it on a model. The purpose of implementing text and fonts in GDP is to facilitate the design process of control panels by enabling the industrial designer to view a design iteration before actually building the finished product.

RESULTS

A font character is a two-dimensional planar face which can be placed on the surface of a primitive. When ray-traced, FTEXT lines on surfaces will appear as the characters they contain. FTEXT lines are characterized by font name and its text string. The user can place an FTEXT in a specified position at which time the FTEXT will appear in wireframe in the model.

Currently, coding is in progress to create and manipulate FTEXT objects.

PROJECTED PLANS

Upon addition of FTEXT as a GDP primitive, full GDP function support will be given to FTEXTs. i.e. filing, fetching, erase, rotate, replicate, etc. A library of helvetica alphabets will also be completed. Next semester, FTEXT handling will be expanded to include a font editor and more advanced editing functions. Other fonts of different styles and sizes will be added to the font library.

PETER, PHIGS, PS300 AND SURFACE OBJECTS IN THE CLOCKWORKS

David Breen

SUMMARY

Since the last review, numerous objects have been added to the CLOCKWORKS. Several of these objects brought the interactive modeling capabilities of TIM to the CLOCKWORKS. These objects are PETER, PHIGS and PS300.

PETER (PEncil Test renderER) evaluates the GEOFF geometry of the CLOCKWORKS and creates PHIGS structures which are sent to an interactive device. The PHIGS structures consist of transformations, polylines, polyline attributes, instances of other structures and viewing information. PHIGS is the basis for the CLOCKWORKS' interactive device interface. The PHIGS' object contains device independent data such as state data, viewing data, currently open structure, currently posted root, etc.. The PHIGS' output model has been fairly well implemented. The PHIGS' input model has been fairly well ignored. Input is received through relatively high level commands, rather than low level device commands.

The Evans & Sutherland PS300 was the first interactive device interfaced to the CLOCKWORKS. The PS300 object handles the device specific parts of the methods defined in the PHIGS object.

The objects PETER, PHIGS and PS300 support an environment where a user can create geometry and view a vector representation of it using the real time capabilities of the PS300. The transformations of the model can also be interactively manipulated. Scripts can be previewed using these objects.
Other GEOFF objects have been added to the system, namely surface, stnr_srf, and bez_srf. Surface is an object which encapsulates all the data and procedures common to all surfaces. This object simplifies the task of adding a new surface to the CLOCKWORKS. Stnr_srf is an object which supports Steiner surfaces. A Steiner surface is a collection of Steiner patches. A Steiner patch is a quadratic triangular Bezier patch. Bez_srf is an object which supports arbitrary order rectangular Bezier surfaces. A Bezier surface is a collection of Bezier patches.

RESULTS

These objects are in working order and are currently being used to make models in the CLOCKWORKS.

PROJECTED PLANS

Bugs that require fixing or functionality that needs expanding are explained in the CLOCKWORKS' future plans.

RENDERING IN THE CLOCKWORKS

Phillip Getto

SUMMARY

The renderers in the Clockworks animation system provide abstractions of several types of output devices. *'PETER'* the pencil test renderer, outputs vectors in a hierarchical structure to a *"PHIGS"* device. Note, it does not evaluate the Boolean expression representing the object. but, displays all of the primitives.

"VIRA". in the strictest sense, is only a visible surface determination processor. It determines the list of surfaces which are behind each pixel. If the output device to which it is talking is an *"ABUFFER"* (Anti-aliased visibility data buffer). then the list of surfaces is output. If the output device is an image, then *"GESHIA"* is called, with the list of surfaces, to shade the pixel.

"GESHIA". is a set of methods (functions), implementing various shading models. It accepts an AVID (Anti-aliased VIsibility Data) list for a pixel and computes the intensity of the pixel with the shading models called for by the visible surfaces.

"ISABEL". the Interactive Shader of Abuffer Elements. allows a user to interactively select objects to be shaded. The user can select a primitive or a part to be shaded without recalculating every pixel in the image. Shading model parameters or even shading models can be changed. interactively. with results displayed rapidly.

"BART". the system's ray tracer provides the highest quality output. It is fairly mature. but is still being improved and fine tuned.

RESULTS

All of the renderers are up and running. Specific results and plans for *PETER* can be found in a separate summary.

VIRA uses a scan line algorithm to evaluate the CAS graphs from GEOFF and then determine the visible or possibly visible surfaces for each pixel. The AVID buffer can be stored on disk or *GESHIA* can be invoked directly to shade each pixel in the image. Input to *VIRA*, presently, can only be in the form of polygons. so superquadrics could be rendered by *VIRA* a method which approximates a superquadric with polygons.

A number of shading models have been implemented in *GESHIA*. including those proposed by, Gouraud. Phong, Blinn. Whitted, and Hall. However. the Whitted and Hall models are not fully implemented as they do not interface to a ray tracer. Both 2D and 3D procedural and look up table maps have also been implemented. Procedural textres are separate programs which return some value, via UNIX pipes. for a given point in space. Look up table maps come from image files. The values of a 2D map may be mapped onto a surface in several different ways, including spherical, cartesian and extruded mappings.

Work on *BART* has resulted in improvements in the lighting model and the speed of superquadric intersection calculations. Internal reflections were added to the lighting model calculations, giving more realistic glasses. Superquadric intersection calculations were sped up by: 1) transforming the ray into a unit superquadric space, and 2) faster determination of the intervals containing intersection points. The intersection point, because of the symmetry of a superquadric, must be in the intervals defined by the intersection of a ray and the planes $x = 0$, $y = 0$, $z = 0$, $x = +y$, $x = +z$, $y = +z$, and the points t_1 and t_1 to the interval of interest along the ray, if they exist. Thus, by intersecting the nine planes with the given ray and then evaluating the points in the implicit form of the superquadric's equation, the intervals containing roots can be immediately determined.

PLANS

The data structure used by *BART*, *VIRA*, and *ISABEL* to represent the surfaces behind a pixel are different. The structures are conceptually the same but use slightly different data in slightly different forms to represent the same information. Since they are not the same, *BART* cannot use the shading models in *GESHIA*, nor can it store ABUFFERs form use with *ISABEL*. Nor can *GESHIA* use the intersection routines in *BART* for those shading models which incorporate global illumination effects. There is also a duplication of the algorithm for evaluating the CSG graph in *BART* and *VIRA* since there is one version for each of the two data structures is needed. It is planned that the data structures be changed so that they are the same, thus allowing *BART* and *ISABEL* to use parts of each other and *BART* and *VIRA* to share the CSG expression evaluation code.

Recent work in the field has focused on increasing use of bounding volumes to accelerate ray tracing. However, all of the work has been for evaluated models, (models involving no Boolean operators e.g., a list of polygons). Work is in progress to extend these results to unevaluated CSG models. This work should be completed in the next few months.

INTERACTIVE SCRIPTING USING DIAGRAMS AND KEYFRAMES

Brion Sarachan

SUMMARY

An interactive user interface for scripting animations in the CLOCKWORKS has been implemented. The goal of the system has been to provide the user with a convenient environment for creating animations, while automating all scripting operations which are tedious or mechanical.

The user is able to interactively construct a script in diagrammatic form. Timelines represent cues containing actions. Marks on the timelines represent points of time at which the user will specify the value of some animated parameter.

An interactive keyframing system then allows the animator to set the values of animated parameters. In other words, the animator creates animation frames through a process of geometric modeling and an animation script is automatically created based on the diagrams and keyframes specified by the user. Splines are used to interpolate intermediate values of animated parameters and thus create in-between frames.

RESULTS

The foundation of the interactive user interface is an action editing system in which CLOCKWORKS language statements can be parsed and stored in parse tree form. The *parsetree* object has been added to the CLOCKWORKS environment which stores the parse trees of actions modifies these trees interactively.

Cue objects have been modified to store their actions in parse tree form. *Cues* contain the necessary looping constructs which update the geometry being scripted for each animation frame.

A new *keyframe* object processes information specified interactively by the user. The appropriate actions are written to the parse trees stored in *cues*. *Keyframe* also assembles *spline* objects according to the data specified by the user. These splines define the functions which animated parameters are to follow. The user may specify spline points interactively, rather than specifying explicit quantitative values.

The *Chronos* object allow the user to interactively create or modify a script at the topmost level, in diagrammatic form. Timelines representing the time extent of cues can be interactively manipulated, as can marks representing key frames of the animation.

Interactive functions have been added to the device driver for the Evans and Sutherland PS300. These allow transformable geometric objects to have their translations and rotation interactively manipulated with respect to their own local coordinate systems. This allows geometric objects to be scripted as autonomous entities. An object can be easily moved or rotated with respect to its own orientation.

Several experimental scripts have been generated. These have dealt with the motion of rigid objects. The interactive interface allowed these scripts to be created quickly, and to conform to the motion which was intended.

PROJECTED PLANS

Further experimentation involving articulated motion and shape transformations will be performed.

THE CLOCKWORKS' FUTURE PLANS

David Breen
Phillip Getto

SUMMARY

The CLOCKWORKS has become a software system which supports a wide variety of animation functions. Using the CLOCKWORKS, a user can create a hierarchical model comprised of several geometric primitives. The material properties of the model can be interactively specified. Using scripting primitives the model can be transformed through time. Given a script, the animation can be generated frame by frame using a variety of renderers.

Even though the CLOCKWORKS supports many animation features, there is still much work to be done. Provided with enough time and resources there is interest in working on the following projects:

1. Write a tutorial explaining how to use the CLOCKWORKS.
2. Implement fractal landscape, plants or other high order objects.
3. Investigate color coordinate systems.
4. Develop a menu driven interactive interface for the CLOCKWORKS.
5. Port the CLOCKWORKS to Sun, UVAX, VAX and IBM equipment.
6. Upgrade the CLOCKWORKS' object-oriented environment by implementing return addresses, tokens, multiple inheritance and other important features.
7. Implement a Raster Technologies 380 object in order to expand the number of interactive devices that interface to the CLOCKWORKS.
8. Develop pams and Steiner patch approximation methods for the new surfaces so they can be rendered.
9. Interface BBSRF to the CLOCKWORKS through the file system.
10. Unify the underlying structures of BART, VIRA and ISABEL.
11. Expand the GEOFF CSG graph representation.
12. Implement a conditional cue object. It will start and stop based on events rather than time.
13. Implement the evaluation of Boolean expressions in order to define events.
14. Represent rotations using quaternions.
15. Make movies - (e.g. atomic orbital hybridization, future flight).
16. Develop a grasp interface.
17. Develop a renderer which removes hidden lines.

18. Enhance the viewing calculations of PETER.
19. Study various techniques for accelerating BART.
20. Investigate point sampling methods for various geometric primitives.
23. Implement Kajiya's "rendering equation".

REMARKS ON ALGORITHMIC MODELING

Harry McLaughlin

SUMMARY

In engineering practice, geometric descriptions of objects are almost always given in terms of closed form expressions. In this process, two unwanted results can occur. First, the closed form expression may fail to capture intended geometric features, e.g., desired monotone cross sections of a surface. Secondly, extraneous geometric features may be introduced, e.g. ripples in a surface.

There is a growing recognition for the need for more control of geometry in geometric modeling. To this end, some study has been given to defining geometric objects procedurally (algorithmically). Each step of the procedure is defined using geometric descriptions so that the geometric characteristics of the modeled object can be deduced from the algorithm. This is a distinctly different approach from using analysis tools on parameters of a closed form expression.

Some of the technical issues which arise, and are being investigated, are:

1. What are appropriate definitions of curve and surface?
2. What is a good way to build human intuition of smoothness into appropriate (useable) algorithmic definitions for smoothness?
3. How expensive is algorithmic modeling?
4. How does the current state-of-the-art of evaluation algorithms translate into procedural thinking?

RESULTS

Geometrically defined algorithms for generating discrete curves have been proposed and are currently being implemented. A study of currently used B-spline evaluation algorithms has been undertaken and an attempt to extract their geometric features has been made. Work has begun on the problem of defining shape of 3-space data.

PROJECTED PLANS

The research project outlined above is a large undertaking. Its goals will be pursued vigorously with the anticipation that modeling techniques based on geometric intuition will evolve.

THE SUN DAGGER PROJECT

Diane Desenberg

SUMMARY

The motivation for this research stems from the computer modeling of an archaeoastronomical site in Chaco Canyon. Due to the many irregularities of the rocks and a desire for greater accuracy in the model, new photogrammetric data was collected in May, 1986. Since the new data set was much denser, the photogrammeters could not determine which face a given point was located. Without such face data, the previous method for reconstruction could not be used. A new method to formulate the surface connections from the data became a necessity. As a result, during the last six months, the primary focus of this work has been on modeling the slabs and spirals that constitute the site.

The goal of this research is to display and manipulate a surface that is only known by a set of unordered points. To do so, an edge relationship between these points must be developed. Since for any given data set there may be many possible solutions, a method is needed to construct a boundary appropriate for the object being modeled. Current efforts have pursued the subject in two dimensions, with an attempt to limit dealings to algorithms and concepts that have applicability in three dimensions.

The Delaunay triangulation was chosen as a method to define a general structure from which to obtain a solution. It is a structure that allows an approximation of the shape of the points. After this triangulation is performed on a given data set, there are two possible circumstances. Either all of the points from the data set are on the convex hull or there are points from the data set that are still interior to the convex hull. In the first case, a solution has been successfully determined. In the second instance, triangles must be eliminated until all of the points lie on the boundary.

In order to find a solution in the second case, the approach chosen was a greedy algorithm. It determines which triangle to remove next until all of the points are on the boundary. Triangles are eliminated in such a way as to preserve the polygonal quality of the boundary. The judgment of whether a particular algorithm is a "good" one is based on the following steps:

1. Define a closed curve.
2. Digitize it.
3. If the original topology is not reconstructed, digitize the curve at a higher density.
4. If at some point, the method produces the original topology and continues to produce the original topology for greater densities, then it is a good method.

If it reconstructs the original topology from a lower density of points the none method can be deemed better than another for a given point set. These methods should produce some intuitive notions about how the various algorithms process points.

RESULTS

Eight different criteria for removing triangles were chosen. One of the better and one of the worse criteria for the data sets chosen are outlined below.

With the overall goal of an object with maximum area, Alternative 1 consisted of removing the triangle with the smallest individual area first. Its behavior resulted in the worst reconstructions. It is currently hypothesized that for certain data sets no matter how densely the polygons are discretized, the results will not improve. This is a startling result. A more thorough analysis on a wider data set is needed.

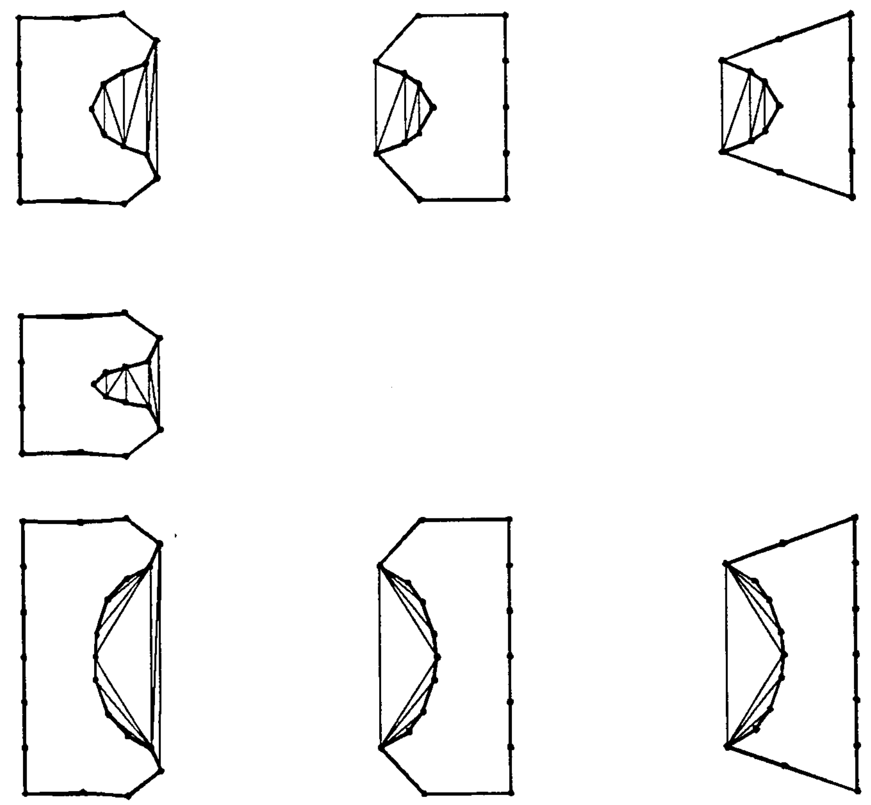
With the overall goal of minimizing the angular movement around the polygon. Alternative 2 consisted of removing the triangle with the flattest interior angle first. In general. as data points were chosen at a greater density. the results eventually coincided with the topology of the original curve.

The first alternative tended to produce shapes with narrow tunnels etched out of them. The second alternative tended to produce shapes with few major indentations from the outer edges. To test these observations further. data was chosen from a polygon with a narrow tunnel etched out of it. From the density of the points chosen. most people would have trouble distinguishing a coherent shape. However. Alternative 1 had no trouble determining the original shape. Alternative 2 did a much poorer job. Evidently. Alternative 2 needed a higher density than Alternative 1 for this particular data set.

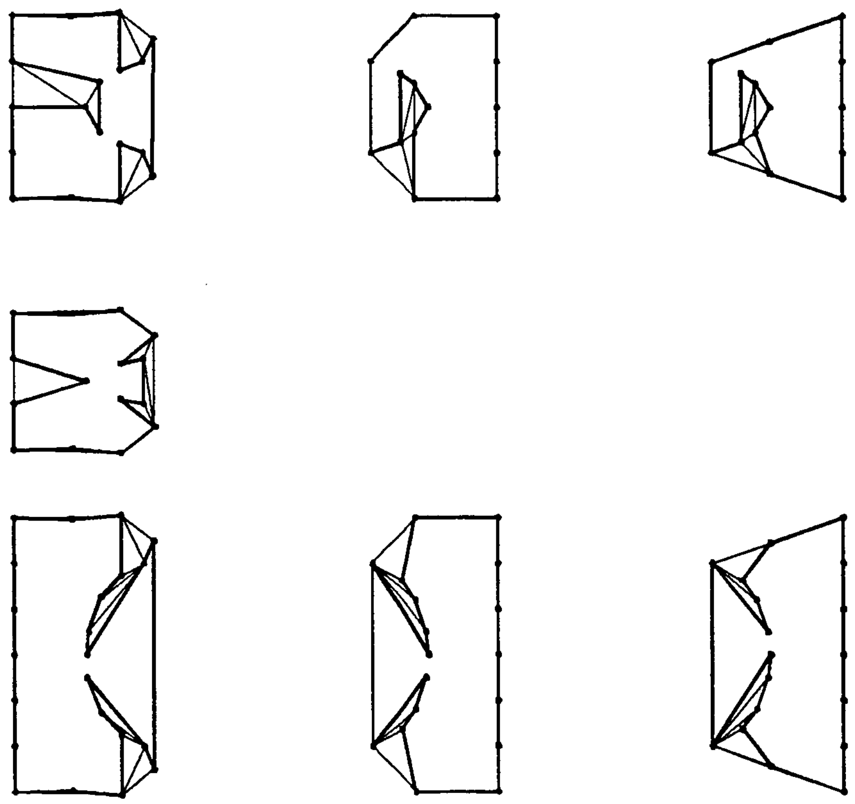
PROJECTED PLANS

This research still leaves many questions unanswered. How dependent are the results upon the original polygon? If a different underlying triangulation were chosen. to what degree would the criteria maintain the qualities observed here? Another valid approach might be to start someplace in the center and add triangles until all of the points lie on the boundary. These and many other efforts would add insight to this area of study.

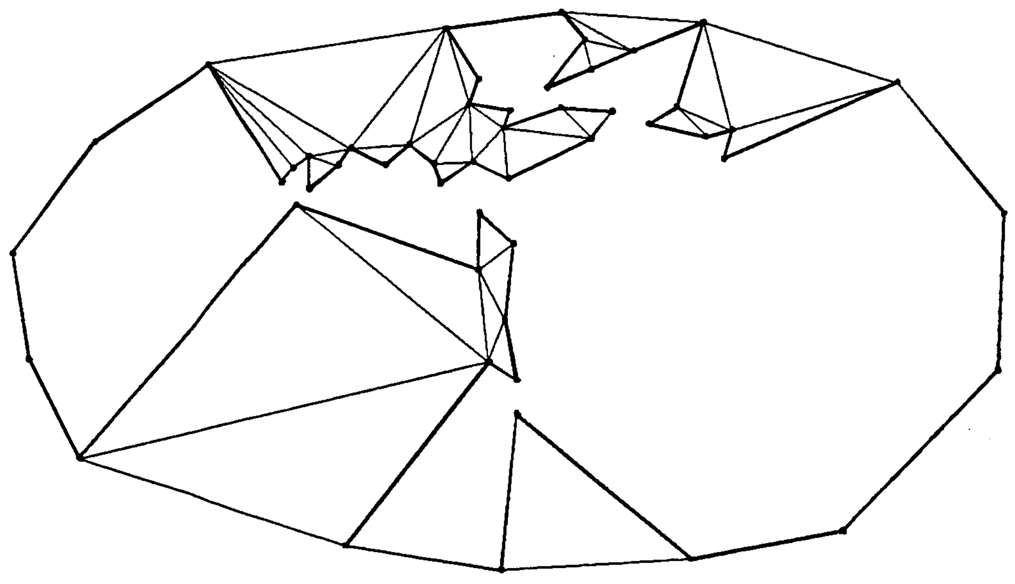
ALTERNATIVE 1: FINAL BOUNDARY WITH TRIANGLES REMOVED



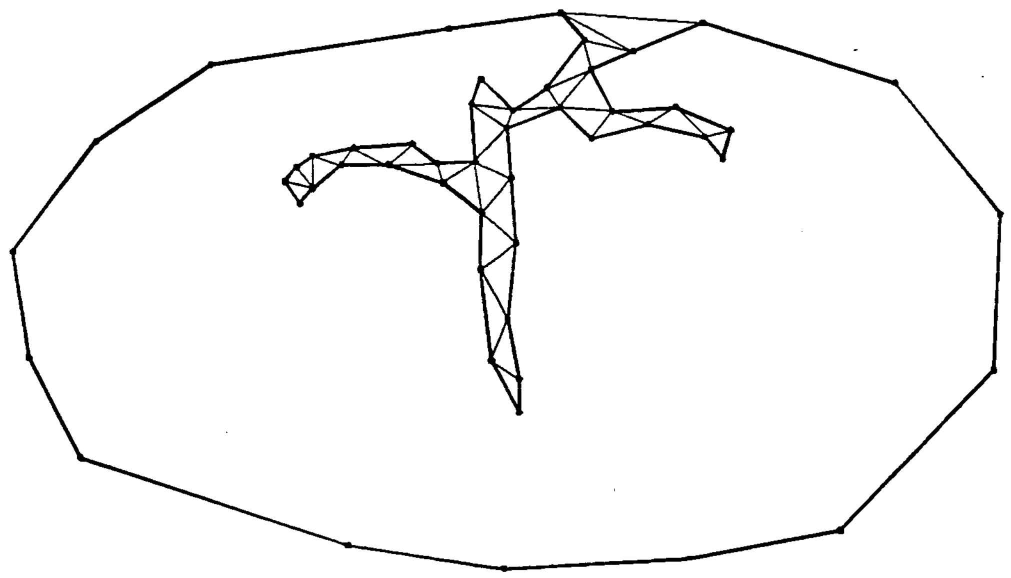
ALTERNATIVE 2: FINAL BOUNDARY WITH TRIANGLES REMOVED



ALTERNATIVE 1: FINAL BOUNDARY WITH TRIANGLES REMOVED



ALTERNATIVE 2: FINAL BOUNDARY WITH TRIANGLES REMOVED



COLLISION FREE PATH PLANNING FOR A MANIPULATOR

Kurt Dittenberger

SUMMARY

The purpose of this project is to develop an efficient algorithm that finds a collision free path for a manipulator. The input of this algorithm should be a description of the robot, its workcell, its payload, and the starting and ending position of its endeffector. The output of the algorithm should be a collision free path for the robot. More precisely, it should compute for each joint variable of the robot a function (depending on the time) such that if the joints are moved according to these functions there will be no collision between the parts of the robot, its payload, and the obstacles in its workcell.

To find a collision free path, the first step is to describe all the positions of the robot for which there is no collision. Depending upon how these positions are described there are two different approaches.

The collision free positions of the robot can be described by determining the locations and orientations of all its links with regard to the world coordinate system (computing the 'free space' for each link with regard to the world coordinate system). A collision free path for the last link (endeffector) can, now, be obtained by selecting a path that lies completely inside its 'free space' and connects the starting and ending position. Then, one has to calculate the paths for the other links, which depend on the path of the endeffector, and check whether they are collision free or not.

The collision free positions of the robot can also be described by its joint variables (computing the 'free space' for the robot with regard to its joint space). Again one has to select a path that lies completely inside this 'free space' and connects the starting and ending position. However, unlike the first method whenever a path in this 'free space' is selected the problem is solved.

RESULTS

Until now, most of the research has been devoted to the first method. For two-dimensional workcells where the links of the robot and the obstacles can be described as a finite union of convex polygons, there exists a method for calculating the 'free space'. One can also find a path inside this space algorithmically. The problem with this method is that for a collision free path of the endeffector (there are in general infinite many) the resulting path for the other links are generally not collision free.

The second method would avoid this problem but until now there does not exist any method that computes the 'free space' with regard to the joint space.

PROJECTED PLANS

Besides literature studies, future work will now concentrate on the second method. After getting some experience with this approach, it will be decided whether to keep going on with this method or to try to solve the problem of the first method.

AN ATTRIBUTING FACILITY OF GEOMETRIC DESIGN PROCESSOR (GDP)

Wei-Shan Chiang

SUMMARY

Current CAD systems just store geometrical and topological data about the solid modeler. as a collection of low level geometric components. It does not incorporate sufficient data to allow control. analysis. or sales system, etc. Additional level information in the CAD system is desirable.

As a results. there is a need to add attributes to the solid modeler. Those attributes can be assembly sequences. tolerance information. cost of part. material type. color. part description. etc.

The attribute of solid models were recognized as a central feature among many related design applications. GDP (Geometric Design Processor). is an interactive computer graphics tool used to build solid models. This project provides new functions for GDP user to create proper attribute information for each object node of the part and assembly level in the solid model tree.

RESULTS

A set of new functions was written and added to the GDP program. Those attributing interface functions form a new major group labeled "ATR". All GDP interactions with the attributing facility is via this set of functions. They are as follows: GETATR. GETRATR. PUTATR. PUTRATR, ALTATR. REFATR. ERSATR. EDITATR. UTILATR. LIB.

About 25 routines (4.500 lines) of PL/I code have been implemented to support this project. Through proper menu driven flow, GDP users can create. edit. save, and retrieve attribute information based on the GDP solid modeling representation.

The performance of those functions has been very encouraging based on tests with small models. The response time from the attributing functions is comparable to the performance of what GDP performs.

PROJECTED PLANS

Future work will focus on linking attribute data models with the database function, allowing the user to deal with attribute data more efficiently. Using the MERGE operation in GDP. one can group some of the lower level object nodes together and generate a new high level object node. But. the problem of propagating attribute information from the non-object node to its father-object node in the tree structure is also important to consider. Finally. a general and powerful attribute data format is needed. It then can be converted to meet engineering needs.

AUTOMATIC CONVERSION FROM 2D ENGINEERING DRAWING TO 3D SOLID MODEL

Hon-Yue Chou

SUMMARY

The traditional engineering drawing is based on the orthographic projections of an object in space. So it is a two-dimensional object. While modern design and manufacturing systems are based on the solid modeling, the engineering drawing is hard to integrate into the system. As a two-dimensional representation, human interpretation is necessary to convert it to three-dimensional solid model in all applications.

The purpose of this project is to provide a complete interface between a two-dimensional drawing system and a three-dimensional solid modeler system such that no human interpretation is needed. This interface should convert a traditional engineering drawing to a three-dimensional solid model as well as some associated information under a friendly and highly interactive enviroment. Once the solid model is built, one can manipulate this model and connect it to all kinds of applications.

RESULTS

Several algorithms have been developed for converting the two-dimensional projection to the three-dimensonal object. The surfaces which can be handled includes planar, cylindrical, conical, spherical and toroidal surfaces. The limitations in these surfaces are: the axes of all quadratic surfaces must be prependicular to one of the projection planes.

Implementation for planar faces objects according to the algorithm developed by Markowsky and Wesley is completed now. The input is assumed as a general two-dimensional three view drawing. The third angle projection is assumed and each view is identified by a name. Only points and lines are given in each view and the final results will show all possible objects constructed from the input drawing. Boundary representation is used for a solid model and results contain a list of faces and their outward directions. PASCAL is the implementation language resulting in a program is that about 5800 lines.

PROJECTED PLANS

A careful study shows that although the engineerng drawing is based on the orthographic projections, many times an actual drawing is not an exact projection of the object it represents. The algorithms previously developed can be applied to very limited cases and cannot be used in a practical sense. The new approach proposed here is to take care of the cross sectional views as well as some conventions and other notes appearing in the drawing. The new algorithm has been studied and is under development.

FEATURE EXTRACTION FOR GENERATIVE PROCESS PLANNING

Xin Dong

SUMMARY

Once the boundary representation (b-rep) of a machined part is generated, the automatic recognition of feature information becomes very useful. First of all, the description of a part in terms of features makes generative process planning much easier to perform. Secondly, design flaws can be discovered from such a description if the relevant design and manufacturing knowledge is applied. It is these two issues that this work has been focused on.

To accomplish these goals, shape patterns are first recognized from the b-rep of a part. A shape pattern is a set of faces which is sufficiently primitive and thus is the building elements of design or manufacturing features. The boundary of each shape pattern is then defined. Furthermore, a method has been under development to classify the relations among shape patterns.

Based on these shape patterns and their relations, various types of features, either simple or structured, are recognized, then, the following tasks can be performed:

1. design flaws;
2. determine initial workpiece;
3. determine tool accessibility to each feature;
4. generate feature volumes;
5. generate cutting volumes.

Since the entire task is very heuristic and knowledge-intensive, the application of expert system technology is highly desirable and currently being developed. Its uses include:

1. describe shape patterns;
2. describe the ordering in the shape pattern recognition process;
3. describe features;
4. resolve ambiguity in feature recognition process;
5. describe design flaws and corrections;

RESULTS

Most of the methods and ideas required have been developed. The current work is focusing on the development of data structures and the rule syntax for the expert system.

PROJECTED PLANS

The future work will be to implement a prototype system and to evaluate its performance. Some modifications are expected to be necessary when some experiment results are obtained.

GEOMETRY IN PROLOG

Randolph Franklin

SUMMARY

Wm. Randolph Franklin ia an Associate Professor in the Electrical, Computer and Systems Engineering Department and the Computer Science Department. His interests include computer aided aritifical intelligence techniques and Prolog. Some of these algorithms include planar graph overlay and hidden surface programs where the input might have thousands of edges. He also uses Prolog technqieus to debug expert systems.

The attached paper describes an area of his research.

in *Advanced Computer Graphics, Proceedings of Computer Graphics Tokyo 86,* edited by Tosiyasu L. Kunii, April 22-25, 1986, pp. 71-78, Springer-Verlag.

GEOMETRY IN PROLOG†

Wm. Randolph Franklin‡
Margaret Nichols *
Sumitro Samaddar
Peter Wu

Rensselaer Polytechnic Institute
Troy, NY, 12180, USA

October 1985

1. ABSTRACT

The Prolog language is a useful tool for geometric and graphics implementations because its primitives, such as unification, match the requirements of many geometric algorithms. We have implemented several problems in Prolog including a subset of the Graphics Kernal Standard, convex hull finding, planar graph traversal, recognizing groupings of objects, and boolean combinations of polygons using multiple precision rational numbers. Certain paradigms, or standard forms, of geometric programming in Prolog are becoming evident. They include applying a function to every element of a set, executing a procedure so long as a certain geometric pattern exists, and using unification to propagate a transitive function. Certain strengths and weaknesses of Prolog for these applications are now apparent.

2. INTRODUCTION

The fifth generation logic programming language Prolog[Clocksin81a, Coelho80a], appears appropriate for research in geometry and graphics. Some examples of its use in architectural design are given in [Swinson82a, Swinson83a, Swinson83b]. Its use in CAD has been evaluated in [Gonzalez84a]. Constructing geometric objects from certain constraints is described in [Bruderlin85a]. Over the past two years, the authors of this present paper have implemented several geometric and graphic problems in Prolog using assorted machines. This paper describes the experiences, including some paradigms of programming that have appeared useful, and finally listing the advantages and disadvantages of Prolog that we have experienced.

3. IMPLEMENTATIONS

Over the last two years we have implemented several graphics and geometric algorithms in Prolog, totally a few thousand lines of code, using four different Prolog interpreters on four different computers. The systems include:

† This was supported by the National Science Foundation under grant no. ECS-8351942 the Data Systems Division of the International Business Machines Corporation, and by the Rome Air Development Center under the postdoctoral development program via Syracuse University.

‡ Address for 1985-1986 academic year: Computer Science Division, 543 Evans, University of California, Berkeley, CA, 94720, USA. Telephone: 415-642-9955. Arpanet address: wrf@ernie.Berkeley.EDU

* Current address: North American Philips Lighting Co., Bloomfield NJ.

Machine	Operating System	Prolog Version
IBM 3081	Michigan Terminal System	York (U.K.)
IBM 4341	CMS	Waterlog
Prime 750	Primos	Salford
VAX 780	Unix bsd 4.3	UNSW

The implementations include:

- Graphics Kernal Standard subset
- Convex Hull
- Planar Graph Traversal
- Big Rational Numbers
- Polygon Intersection
- Recognizing Groupings of Objects

3.1. Graphics Kernal System Subset

This graphics addition to Prolog was implemented by Nichols [Nichols85a] on an IBM 4341 using Waterlog [Roberts84a], under the CMS operating system. This allowed us to draw lines and so on on the 3270 graphics CRT from a Prolog program. We implemented two classes of lines: *permanent* and *backtrackable.* If the Prolog procedure that drew a backtrackable line was backtracked over, then the line would be erased. This used a feature of the graphics package GSP.

The major problems were as follows. Waterlog, like most Prologs, lacks floating point numbers, and even four byte integers. (The latter was undocumented; large integers just didn't work.) However it has the powerful capability to be linked to programs in other languages such as Fortran. Thus we implemented a real number in Prolog as a data structure of the form *real(A,B)* where *A* and *B* are Prolog integers holding the upper and lower halfword, respectively, of the integer. The user never looks at *A* and *B*, but accesses the real numbers via procedures such as *addreal(X, Y, Z)* and *realtointeger(R, I)* that took real numbers in the stated form and did the obvious things.

3.2. Convex Hull

This Graham Scan algorithm was implemented by Wu [Franklin85a] on both the IBM 4341, and on the Prime in Salford Prolog [Salford84a]. The Salford system allows both real numbers and dynamic linking to Fortran routines. We also tested York Prolog [Spivey83a], which is written in Pascal. The York system has the advantage that it is portable to any machine that can compile a thousand line Pascal program that uses four byte integers. Unfortunately this did not include the official Pascal compiler available from Prime. (We have not evaluated third-party Pascal compilers for Prime computers.) We also tested York Prolog on an IBM 3081 running the Michigan Terminal System, but found the other computers' operating systems more flexible and cheaper to use.

3.3. Boolean Combinations of Polygons

A program to perform operations such as intersection, union, and difference on two planar polygons was implemented by Franklin and Wu [Franklin85a] on the Prime and IBM 4341. The algorithm was by Franklin [Franklin85a]. Wu first implemented a package to perform arithmetic using rational numbers in multiple precision. Each number, in life a quotient of an integer numerator and denominator, is implemented as a list of the numerator and denominator. Each of them is a list of groups of the digits of the number. For example, 123456789/987654321 is represented as [[56789, 1234], [54321, 9876]]. Rational numbers are used to avoid roundoff errors, as part of an ongoing investigation into their utility in geometry and the map overlay problem in cartography [Franklin84a].

3.4. Object Grouping in Photoreconnaissance

Assume that we recognize certain objects in a photograph, and know that they tend to be grouped in clusters. However we might have failed to recognize all of the objects, and some of them might be absent. We have built an experimental system in Prolog to help [Samaddar85a].

3.5. Planar Graph Traversal

At some point during an object space hidden surface algorithm, Franklin[Franklin80a], we have the set of the visible edges and must join them to find the visible polygons. This requires a planar graph traversal, sometimes called a tesselation. For example, in figure 1,

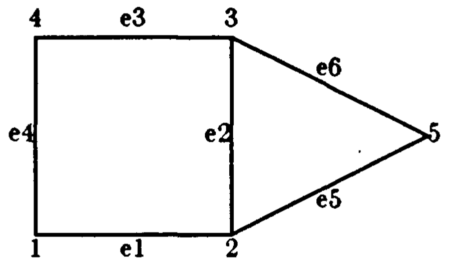


Figure 1: Finding the Faces of a Planar Graph

we are given the vertices and edges in the form

```
vert(vert-name, x-coord, y-coord)
edge(edge-name, first vertex, second vertex, angle)
```

for example

```
vert(v1, 0, 0)
edge(e1, v1, v2, 0)
```

The angle of the edge is supplied because of the difficulty of computing arctangents using only integers. The output is a set of facts of the form

```
polygon([v1, v2, v3, v4])
```

This was implemented in UNSW Prolog [Sammut83a] on a Vax.

4. STRENGTHS AND WEAKNESSES OF PROLOG

Certain advantages and disadvantages of Prolog for graphics and geometric applications are becoming evident from these implementations.

4.1. Advantages Of Prolog

- Prolog has same high level advantages of Lisp, as the equivalence of code and data and dynamic data allocation.
- There are the specific advantages of Prolog. Unification makes determining graph connectivity a primitive operation and in general is useful for propagating transitive properties such which occur frequently.
- The pattern matching fits with the form of expression of many algorithms. For example, our polygon combination algorithm proceeds as follows. Whenever the pattern of two edges intersecting, or one edge ends on the interior of another edge, occurs, then retract those edges and assert new smaller edges. When this pattern no longer exists, then we have a superset of the edges in the output polygon.
- Although many of the above features could be implemented in any language that is Turing equivalent, Prolog is somewhat standard so that different researchers can understand and use each others' extensions.

4.2. Disadvantages Of Prolog

However, there are some problems with using Prolog for geometry.

- There are software engineering problems with using Prolog for a large project because of its lack of nesting in the program and databases.
- Many geometry algorithms are more natural to a forward reasoning system than a backward reasoning system. That is, we are more likely to want the output from some given input than the reverse.
- The natural way of expressing pattern matching algorithms requires us to modify a database as we are searching through it. Thus in polygon overlay, whenever we find the pattern of two edges crossing, we retract them and assert four new edges. Backtracking and redoing a database that we are modifying does not work on all Prologs.
- In general Prolog in completely unstandardized around the fringes as some tests of cuts in [Moss85a] show.

5. PARADIGMS OF PROGRAMMING

Certain techniques have proven to be generally useful in our implementations, and may be useful to others also. They include the following paradigms.

5.1. Set Based Algorithms

Many algorithms such as polyhedron intersection and hidden surface algorithms, Franklin [Franklin82a, Franklin80a], are the alternation of two types of steps:

- Applying function to every element of a set, and
- Combining all the elements having a common key.

This is clearly easy in Prolog.

5.2. Pattern Matching

The second paradigm uses pattern matching to propagate certain properties. For example, in the planar graph traversal algorithm, the edges around each vertex are found and sorted by the angle at which they leave it. Then the edges around each vertex are paired to form *corners*. These corners can be considered to be fragments of the output polygons. Whenever two fragments exist such that the last edge of one is the same as the first edge of another, then these two fragments are retracted and a single longer fragment asserted. When such a pattern no longer exists, then we have the output polygons.

5.3. Unification

Frequently we wish to determine the closure of some transitive property, such as when we are given a set of graph edges *edge(u, v)*, and wish to determine the connected components. We have implemented the following short algorithm that uses unification and the set processing paradigm.

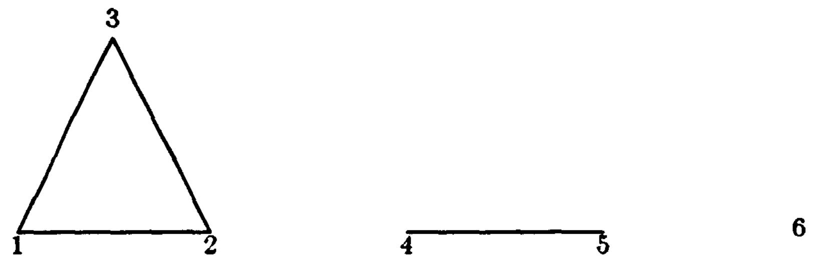


Figure 2: Determining Graph Connectivity

- Create a property list (*plist*) with one record per vertex, and the property of each vertex a free variable. For example in figure 2 we would have *[[1,_],[2,_],[3,_],[4,_],[5,_],[6,_]]*.
- Process the set of edges and for each edge unify the free variable properties of the endpoints. After this we will have *[[1,_1],[2,_1],[3,_1],[4,_2],[5,_2],[6,_3]]* with one unique free variable per graph component.
- Bind a name identifying each component to the free variables in the list to give something like *[[1,first],[2,first],[3,first],[4,second],[5,second],[6,third]]*.

A longer example of a simple hidden surface algorithm would go as follows.

- Wherever the pattern of two edges' projections' intersecting occurs, split the edges into four smaller edge segments.
- For each edge segment find the set of faces hiding its midpoint. Iff it is empty then the edge segment is visible. Draw them.
- Use a planar graph traversal algorithm such as described above to link the visible edges into polygons.
- For each polygon, find a point inside it and then find the set of faces whose projections contain the projection of that point. Find the closest such face; the polygon came from it. Color the polygon accordingly.

This illustrates all of the paradigms operating together.

6. REFERENCES

Bruderlin85a.
Beat Bruderlin, "Using Prolog for Constructing Geometric Objects Defined by Constraints," *Eurocal 85, Conference Proceedings*, Linz, Austria, 1985. Institut fü Informatik, ETH Zürich, CH-8092, Zürich, Switzerland

Clocksin81a.
W.F. Clocksin and C.S. Mellish, *Programming In Prolog,* Springer-Verlag, New York, 1981.

Coelho80a.
H. Coelho, J.C. Cotta, and L.M. Pereira, *How to Solve it With Prolog, 2nd edition,* Ministerio da Habitacao e Obras Publicas, Labatorio Nacional de Engenharia Civil, Lisboa, 1980.

Franklin80a.
Wm. Randolph Franklin, "A Linear Time Exact Hidden Surface Algorithm," *ACM Computer Graphics,* vol. 14, no. 3, pp. 117-123, July 1980. Proceedings of SIGGRAPH'80

Franklin82a.
Wm. Randolph Franklin, "Efficient Polyhedron Intersection and Union," *Proc. Graphics Interface'82,* pp. 73-80, Toronto, 19-21 May 1982.

Franklin84a.
Wm. Randolph Franklin, "Cartographic Errors Symptomatic of Underlying Algebra Problems," in *Proc. International Symposium on Spatial Data Handling,* vol. 1, pp. 190-208, Zurich, Switzerland, 20-24 August 1984.

Franklin85a.
Wm. Randolph Franklin and Peter Y.F. Wu, *Convex Hull and Polygon Intersection Implemented in Prolog,* Rensselaer Polytechnic Institute, Troy, NY, July 1985.

Gonzalez84a.
J.C. Gonzalez, M.H. Williams, and I.E. Aitchison, "Evaluation of the Effectiveness of Prolog for a CAD Application," *IEEE Computer Graphics and Applications,* pp. 67-75, March 1984.

Moss85a.
Chris Moss and Earl Fogel, *Tests to Distinguish Various Implementations of Cut in Prolog,* Imperial College and Logicware Inc., June 1985. Reported on Usenet in Net.lang.Prolog, message-id <1742@utecfa.UUCP>.

Nichols85a.
Margaret Nichols, *The Graphic Kernal System in Prolog,* ECSE Dept., Rensselaer Polytechnic Institute, Masters Thesis, Troy, NY, August 1985.

Roberts84a.
Grant Roberts, *Waterloo Prolog Users Guide,* Intralogic, Waterloo, Ont, Canada, 1984.

Salford84a.
University of Salford, *LISP/PROLOG Reference Manual,* March 1984.

Samaddar85a.
Sumitro Samaddar, *An Expert System for Photo Interpretation,* ECSE Dept., Rensselaer Polytechnic Institute, Masters Thesis, Troy, NY, August 1985.

Sammut83a.
Claude Sammut, *UNSW Prolog User Manual,* University of New South Wales (Australia), 1983.

Spivey83a.
J. M. Spivey, *University of York Portable Prolog System (Release 1) User's Guide*, York, U.K., March 1983.

Swinson82a.
P.S.G. Swinson, "Logic Programming: A Computing Tool for the Architect of the Future," *Computer Aided Design,* vol. 14, no. 2, pp. 97-104, March 1982.

Swinson83b.
P.S.G. Swinson, "Prolog: A Prelude to a New Generation of CAAD," *Computer Aided Design,* vol. 15, no. 6, pp. 335-343, November 1983.

Swinson83a.
P.S.G. Swinson, F.C.N. Periera, and A. Bijl, "A Fact Dependency System for the Logic Programmer," *Computer Aided Design*, vol. 15, no. 4, pp. 235-243, July 1983.

INTERACTIVE DESIGN AND ANALYSIS OF STRUCTURAL SYSTEMS

Rolf Wentorf

SUMMARY

The objective of this project is to integrate commercially available analysis and design packages in order to achieve a highly interactive three-dimensional structural design system. In particular. CADAM™, SAP IV and a decision table processor for design rule checking are linked together by a common interface. The interface. referred to as Preframe. is composed of several highly interactive modules and has been described in previous summaries. For a more detailed summary of the design rule checker. refer to the summary written by Jose Cronembold.

RESULTS

The following is a list of the latest accomplishments.

1. Debugging and other necessary modifications were completed on the Preframe package so as to allow eight modes of operation:

   - GEOMetry. a geometry editor for the structural skeleton:
   - DISPLacement. which assigns boundary conditions;
   - MEMBers. assigns cross-sections to the skeleton;
   - PROPerties. a member cross-section editor;
   - FORCE. a load macro library and editor;
   - LOADS. assign load macros to the skeleton;
   - ANalysis/DEsign. controls analysis and member design checking;
   - Input/Output. stores and retrieves program information.

2. Work on the last mode, interface to analysis and design check routines. was completed except for the features described below in 'PROJECTED PLANS'.
3. The AISC steel design code was implemented through the decision table processor.
4. The user may now view analysis and design results graphically.
5. The program has been upgraded from FORTRAN H and GSP (Graphics Subroutine Package) to FORTRAN VS and graPHIGS/GDDM. Software has also been written to allow operation under the GKS graphics standard.
6. The graphics have been improved to include color and additional icons.
7. The entire menu system has been generalized and expanded in capacity by a factor of 100. while cutting comparable response time in half.

8. The user may enter dimensioned data for distance and weights.
9. The mass properties of the structure may be specified by the user and calculated natural frequencies are available as results.
10. Programmer's documentation has been improved.
11. A user's manual has been written.
12. The program has successfully performed several test problems which exercise the system as an integrated package.

PROJECTED PLANS

Additional tasks consist of the completion of the displaced structure activity, minor improvements to user prompts and messages, and the completion of testing.

Possible extensions include software to help re-create CADAM™ drawings representing the analyzed structure, and the semi-automatic interpretation of CADAM™ drawings to retrieve member cross-sections, materials, support conditions and loadings. Other possibilities include the development of a library of new cross-sections and materials, the inclusion of temperature induced member forces, and a more automated member sizing procedure.

AUTOMATED PROCESSING OF DESIGN CONSTRAINTS

Jose Cronembold
Kincho Law

INTRODUCTION

This section describes a package: dtab, for automated processing of design constraints. The purpose of this module is to develop a general and flexible package for checking the design of structural components against specification codes. This software is designed to handle any given set of design constraints, and flexible enough to expand and/or link to other packages, such as preframe. The approach taken is through the use of decision logic tables which have been proven appropriate for documenting specification constraints and for developing a single general processor such as dtab.

This package is made up of separate, highly interactive, modules that give it an extremely dynamic characteristic and whose capabilities include: library editor, links different sets of decision tables to specific design specifications; editor, decision table editor; translator, translates decision tables into FORTRAN routines; processor, executes the decision tables.

STATUS

All modules are virtually complete, and operational and are in the process of being linked together, this involves automatic identification of ingredients from conditions and actions to generate the information required by the processor module to execute the decision tables. The processor module has also been linked to preframe and testing of results has already began.

CHAPTER NATURAL LANGUAGE PROCESSING AND COMPUTER GRAPHICS

Rolf Wentrof
Kincho Law

SUMMARY

Computer graphics and natural language processing may be combined to form an effective interface for engineering design. Although natural language processors have been created for such applications as information retrieval, language instruction and facial image rendering, their use in engineering design is relatively new. Natural language processing techniques may be useful for engineering design problems where a natural vocabulary already exists for the problem parameters, where conventional menu-driven graphic input would be complex, or where user training is to be minimized.

RESULTS

A natural language interface for the design of plate girders has been implemented. The natural language processor can be used to define a proposed design along with the loads acting onit, check for interference between components, calculate the weight of each component, perform adequacy checks according to current building codes, and to graphically display the efficiency of the proposed design. The language processing techniques used in this project, similar to those developed by Roger Shank at Yale University, utilize a conceptual parser which parses the input based on expectations, which in turn are based on knowledge of the specific problems, knowledge of the domain, and the basic functions available. Other techniques implemented include ellipsis and the resolution of ambiguous phrases by references to the database.

It was found that a natural language or strictly menu-driven graphics;

1. Natural language allows the entry of all attributes of a family of components in one compact sentence or phrase.
2. Natural language allows the user to specify the action in his/her commands and can adapt to individual idiosyncrasies.
3. The user's manual can be greatly reduced in size.
4. Natural language processing can be more transportable than computer graphics since it is less dependent upon particular hardware and software features.

The natural language parser implemented has been formalized using memory organization packets, which can be represented as a conceptual graph, and has the features described below.

Top down	Goal directed approach, based on concepts, not syntax and grammer.
Expectations	The concepts are organized so as to produce expectations as to what may be mentioned. The "meaning" of words are distributed throughout the conceptual graph, rather than lumped together in a dictionary.

Domain	MOPs provde a convenient way to organize and incorporate knowledge of the domain.
Communication	NLP places the burden on interpreting the desired result into a path through the decision tree on the machine. rather than the user.

PROJECTED PLANS

Work continues on the development of a more general and formal approach to this area. Formalization of the NLP for other geometry-related design problems and problems involving complex or abstract entitites will be continued. Another area of interest is the implementation of learning heuristics to increase the adaptibility of the natural language processor.

AN ON-LINE TUTORIAL FOR CATIA™

Tom Valente
Russ Nero

Work has been progressing on an on-line tutorial to teach people how to use CATIA™ a three dimensional computer design package. People using a manual to learn CATIA™ have difficulty understanding what they are being asked to do. An on-line tutorial offers considerably more interaction with the user. If the user makes a mistake, the tutorial can tell him immediately and ask him to re-execute the step. In more advanced lessons, the student will perform several steps on his own before the tutorial checks his work. He will then be given the opportunity to correct his mistakes before moving on through the lesson.

RESULTS

PHIGS (Programmer's Hiearchical eirarchical Interactive Graphics System) has been used to simulate CATIA'S™ functions and has been successful in producing early lessons for the beginner on CATIA™. In these lessons, the user is taught how to create points by keying in their coordinates, lines by connecting points, and three-dimensional wire-frame models by using CATIA'S™ Curve 2 depth function to stretch a figure along a translation vector.

In the future are plans to have lessons teaching how to create geometric primitives offered by CATIA™ and how to transform models into solid volumes. As the lesson plans progress, emphasis will be switched from our instructions to CATIA'S™ prompts. Once the user understands CATIA'S™ prompts, he will be able to learn CATIA'S™ intricacies without the on-line tutorial.

TOLERANCES IN COMPUTER-AIDED GEOMETRIC DESIGN

Joshua U. Turner

SUMMARY

The GEOTOL project has broken new ground by taking advantage of solid modeling technology to provide for the automatic assessment of tolerances in computer-aided geometric design. An existing solid modeling system has been extended with capabilities for the representation of various tolerances and part relationships as intrinsic properties of the solid model. This information has been used as the basis for an automated analysis of tolerance values for acceptability according to given design criteria.

RESULTS

Using the existing tolerance representation and relative positioning capabilities of the GEOTOL system. a fully-automated tolerance analysis capability has recently been completed. Using a Monte-Carlo method. the tolerance analyzer can simulate variations in the size and orientation of part features to perform either a worst-case or statistical tolerance analysis. Individual assembly-instances can be displayed. and summary data for the analysis can be graphed.

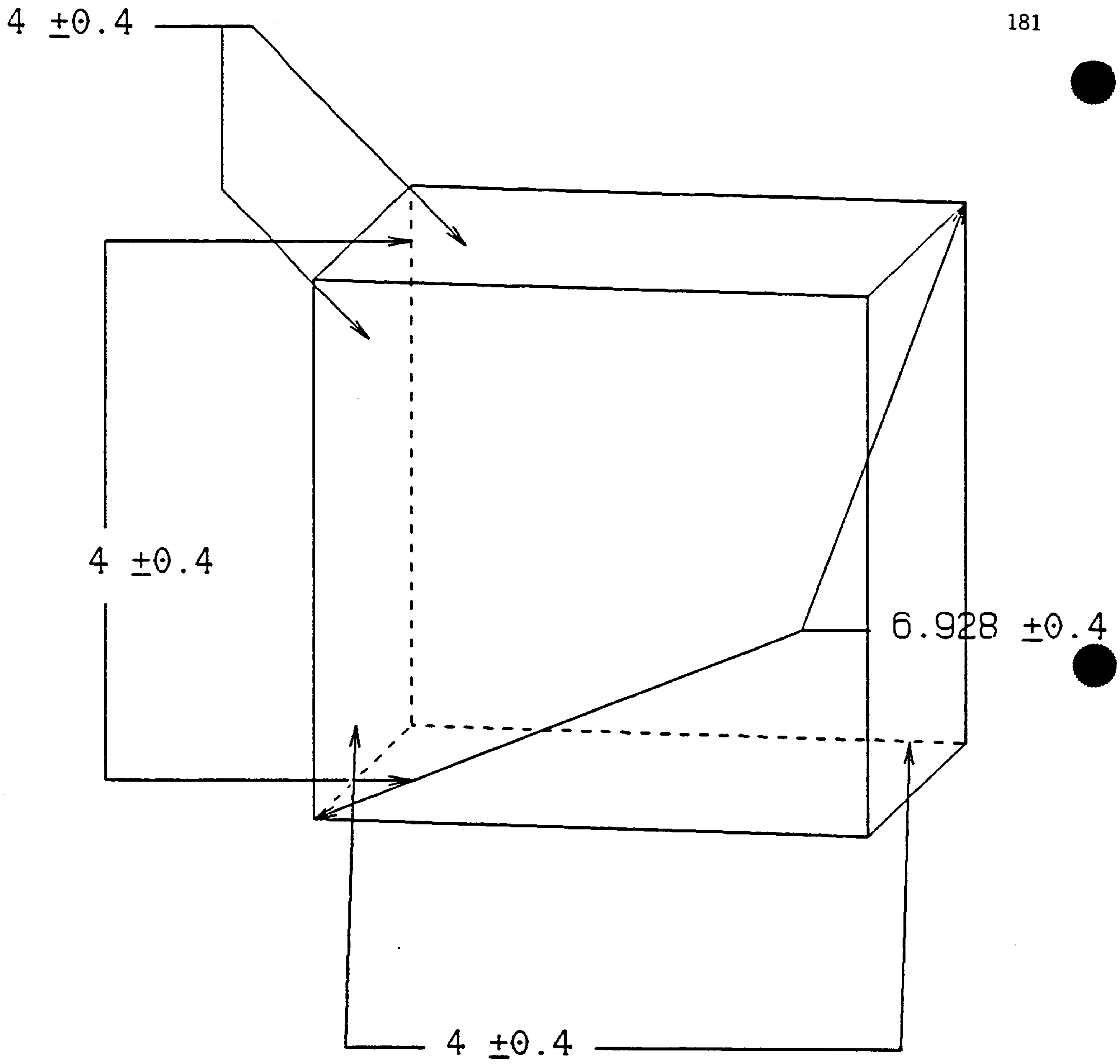
The tolerance anlyzer has been used to analyze a simple assembly with a one-dimensional tolerance stackup. a simple part exhibiting three-dimensional tolerance accumulation, and a complex assembly of parts from an actual product.

PROJECTED PLANS

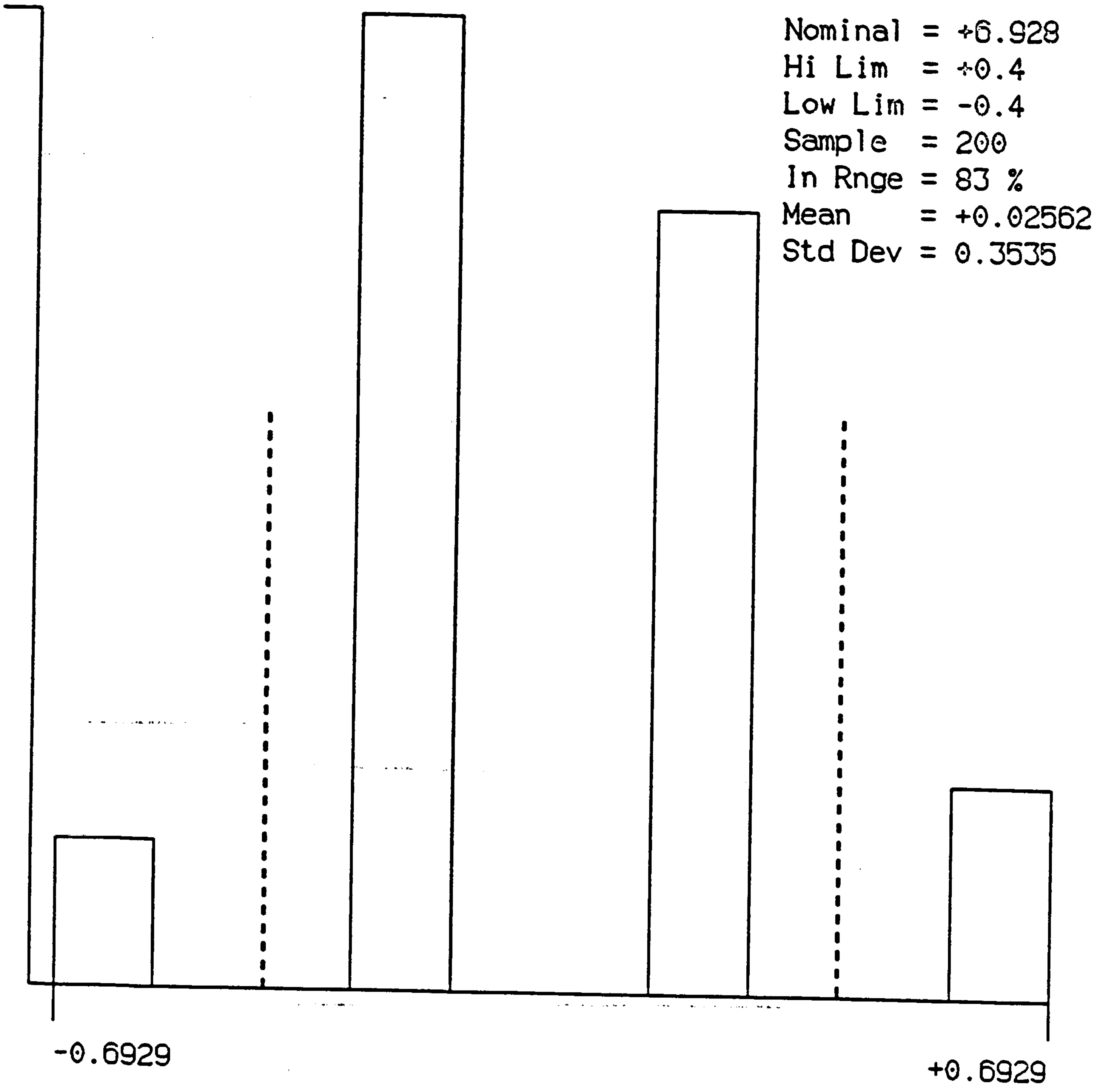
The tolerance analysis capability will be extended in the following directions:

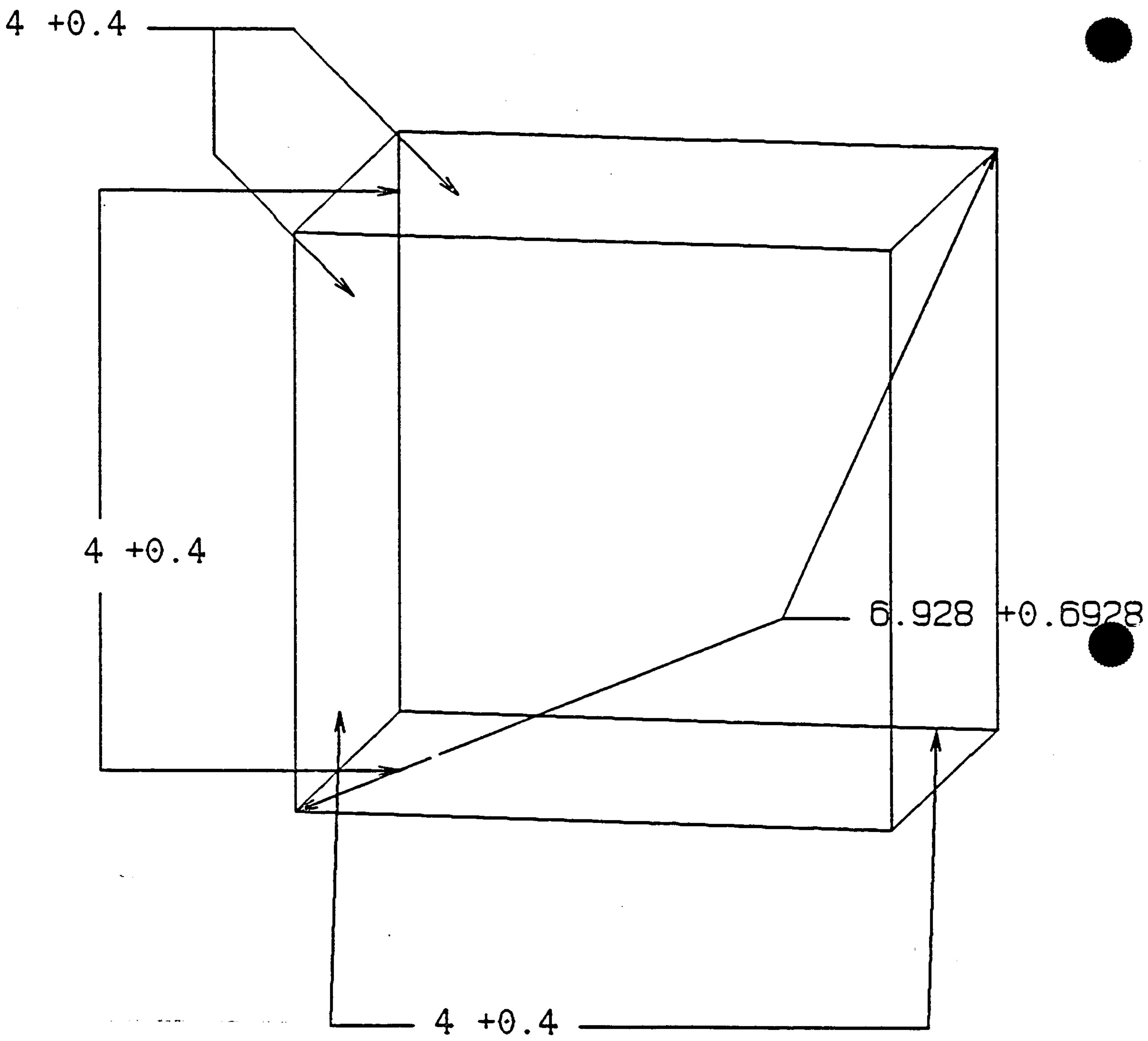
Variational Coverage: The tolerance analysis capabiltiy will be extended to permit the simulation of variation in feature position.

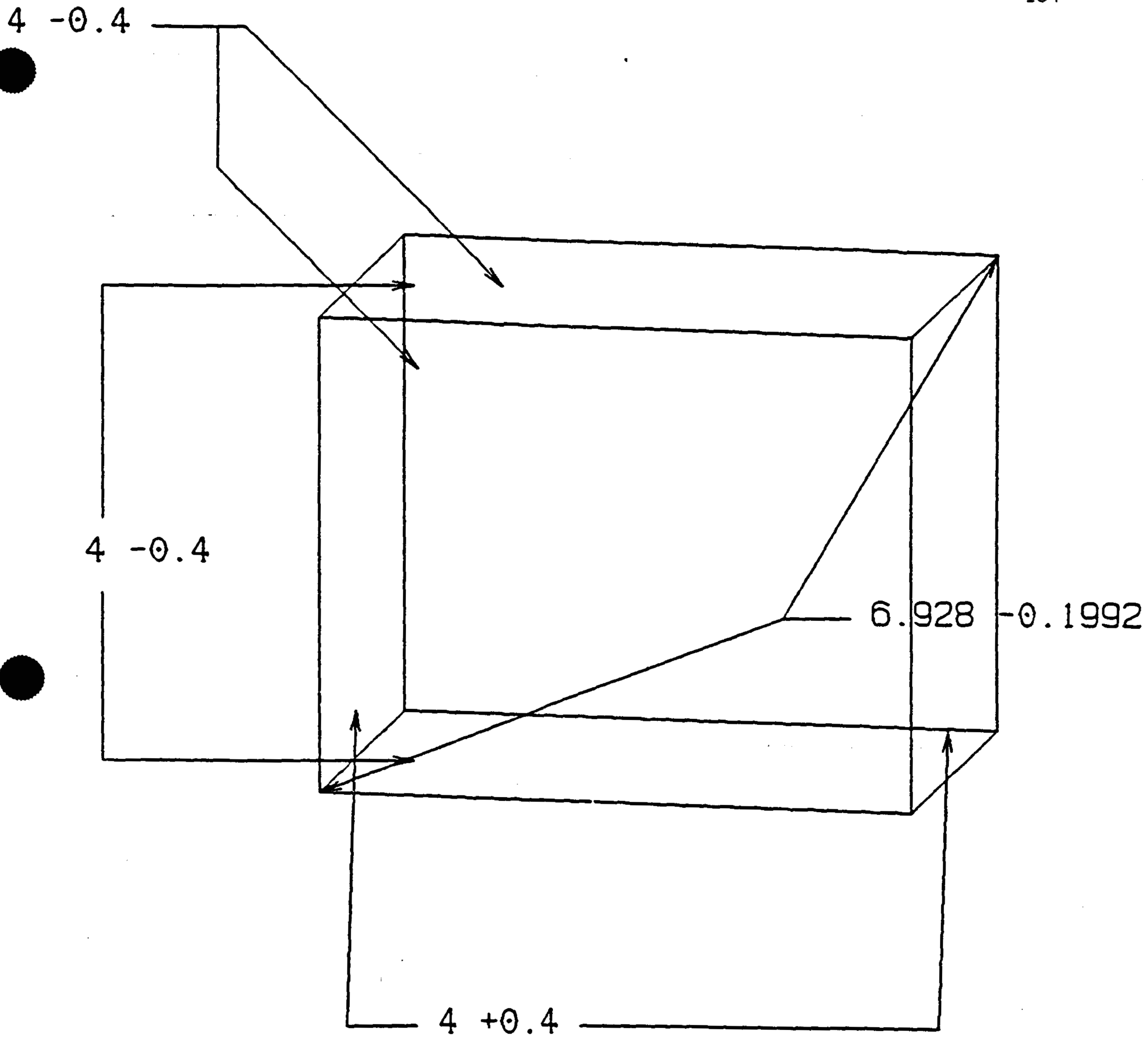
Analysis Algorithms: The Monte-Carlo algorithm will be supplemented by a linear programing algorithm. which will allow for a more rapid analysis of design criteria which are linear. or approximately linear in nature.

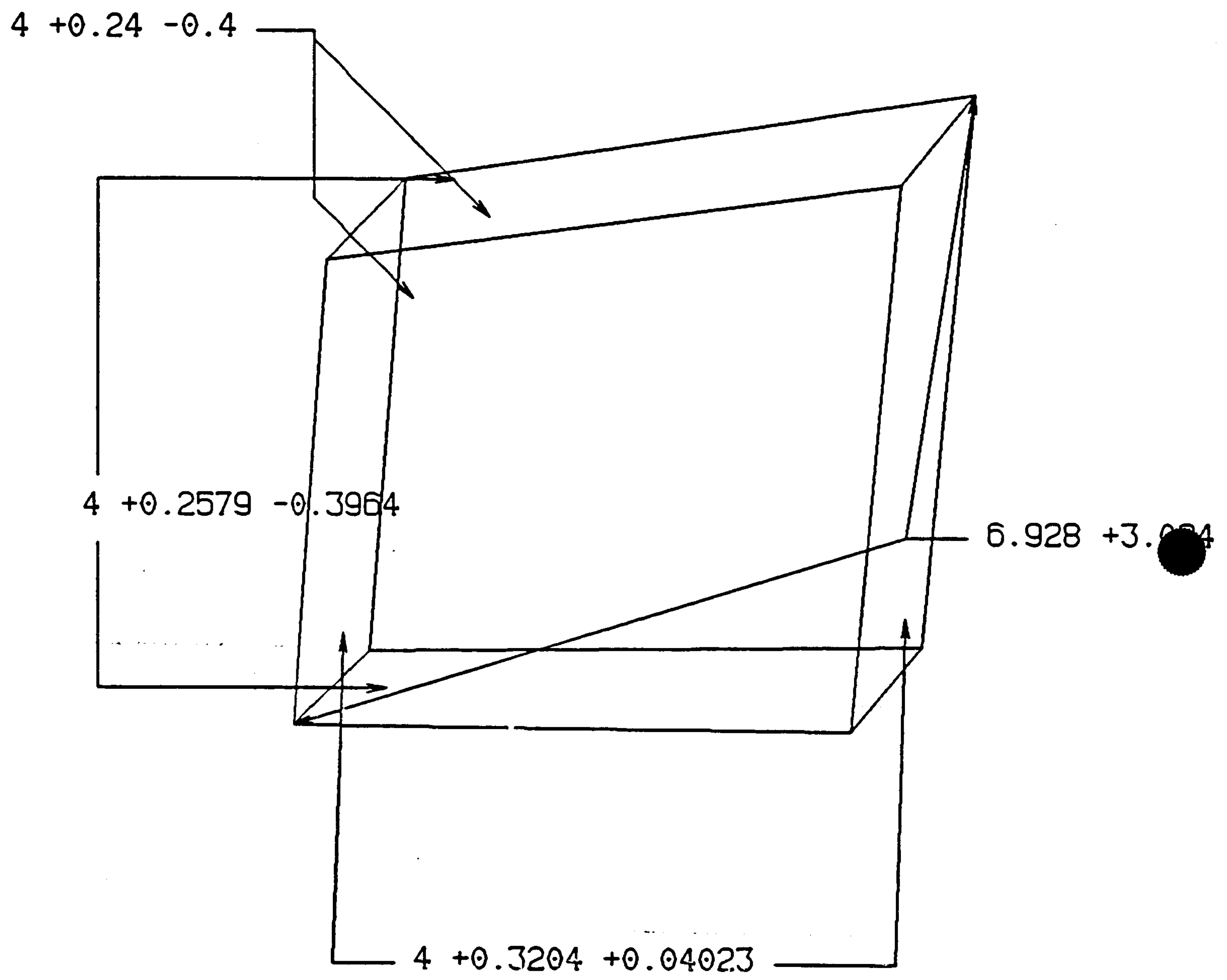

4 ±0.4
4 ±0.4
6.928 ±0.4
4 ±0.4

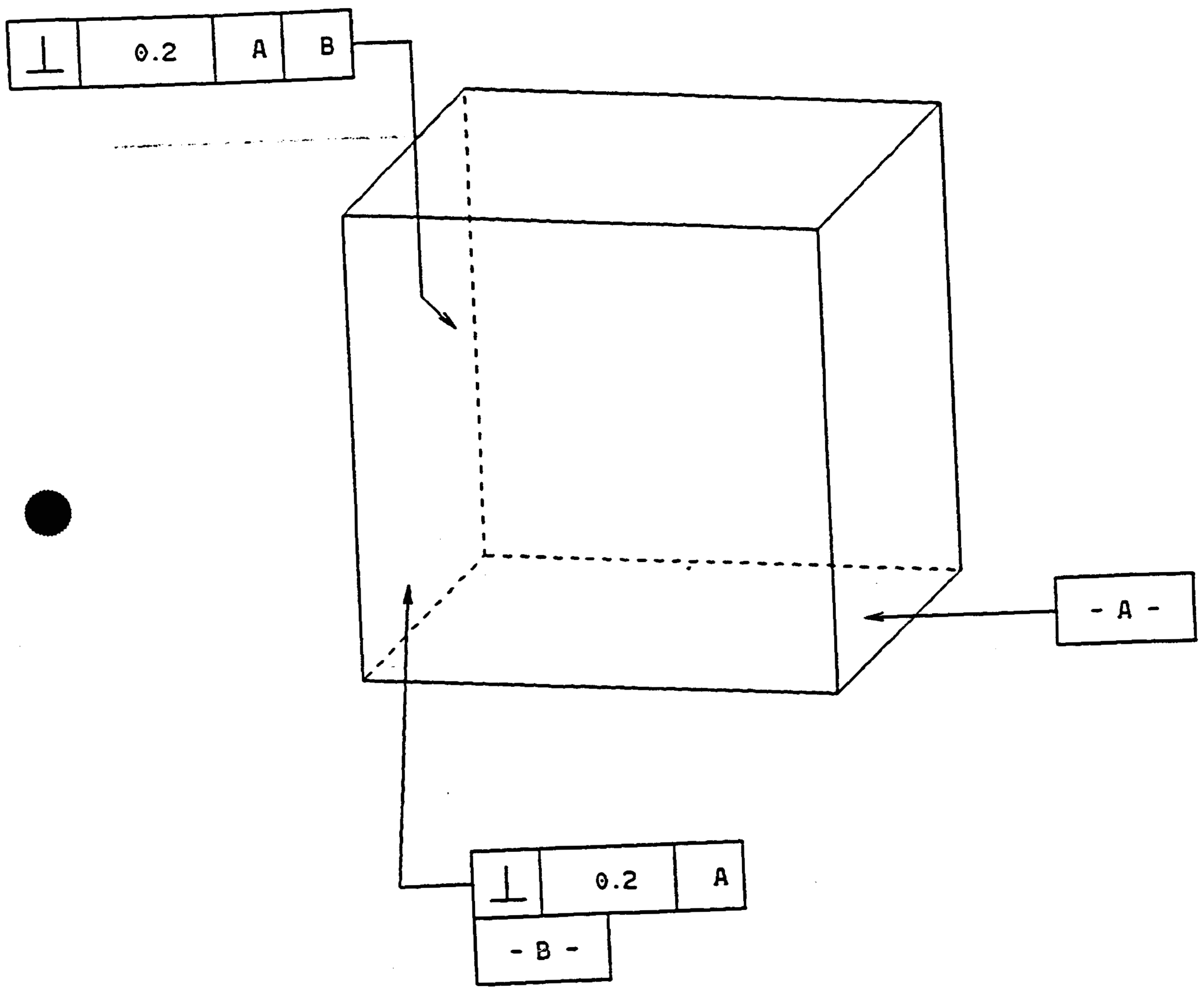
Analysis of CUBE with Worst-Case distribution.



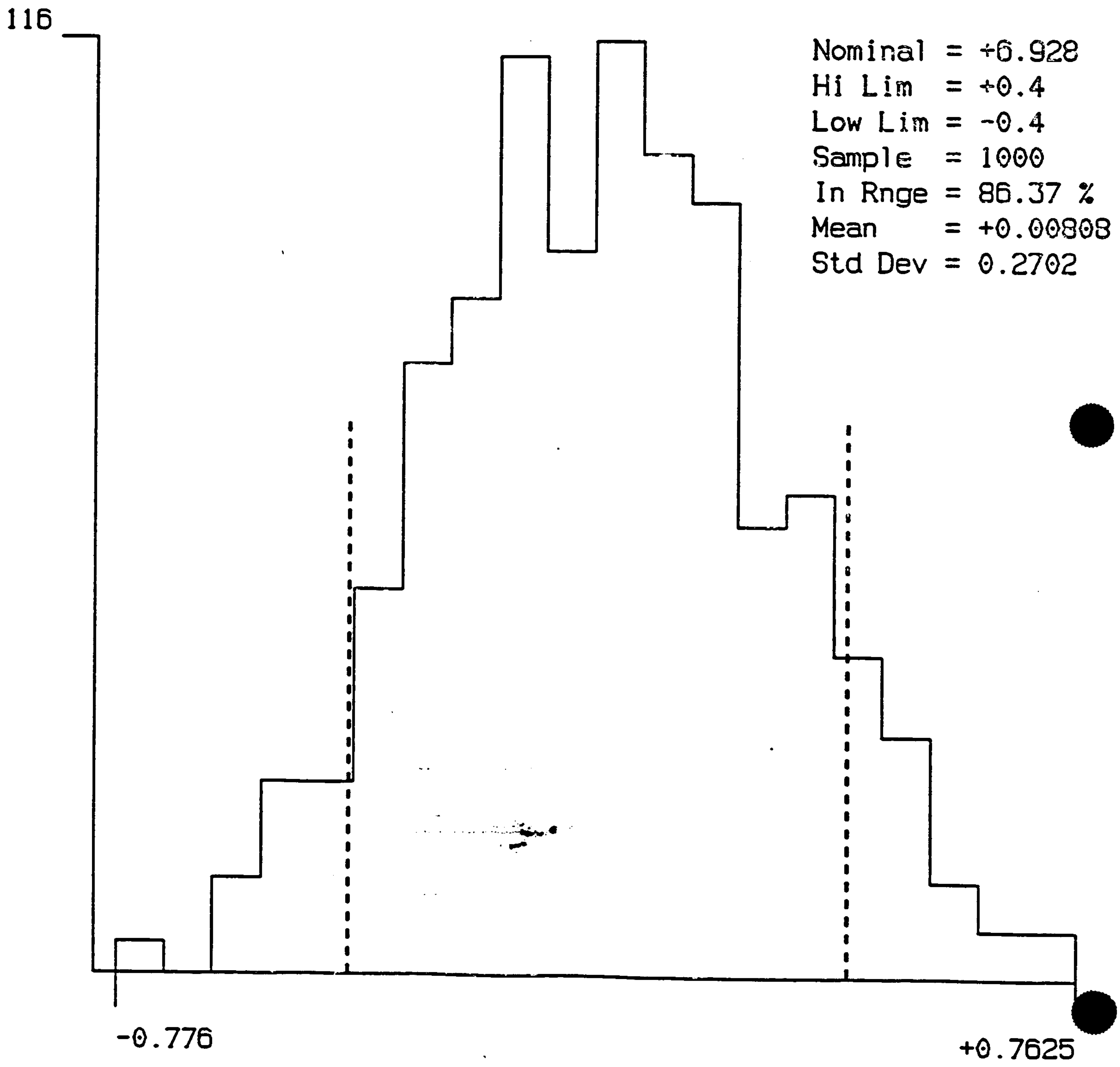

4 +0.4
4 +0.4
6.928 +0.6928
4 +0.4


4 -0.4
4 -0.4
6.928 -0.1992
4 +0.4

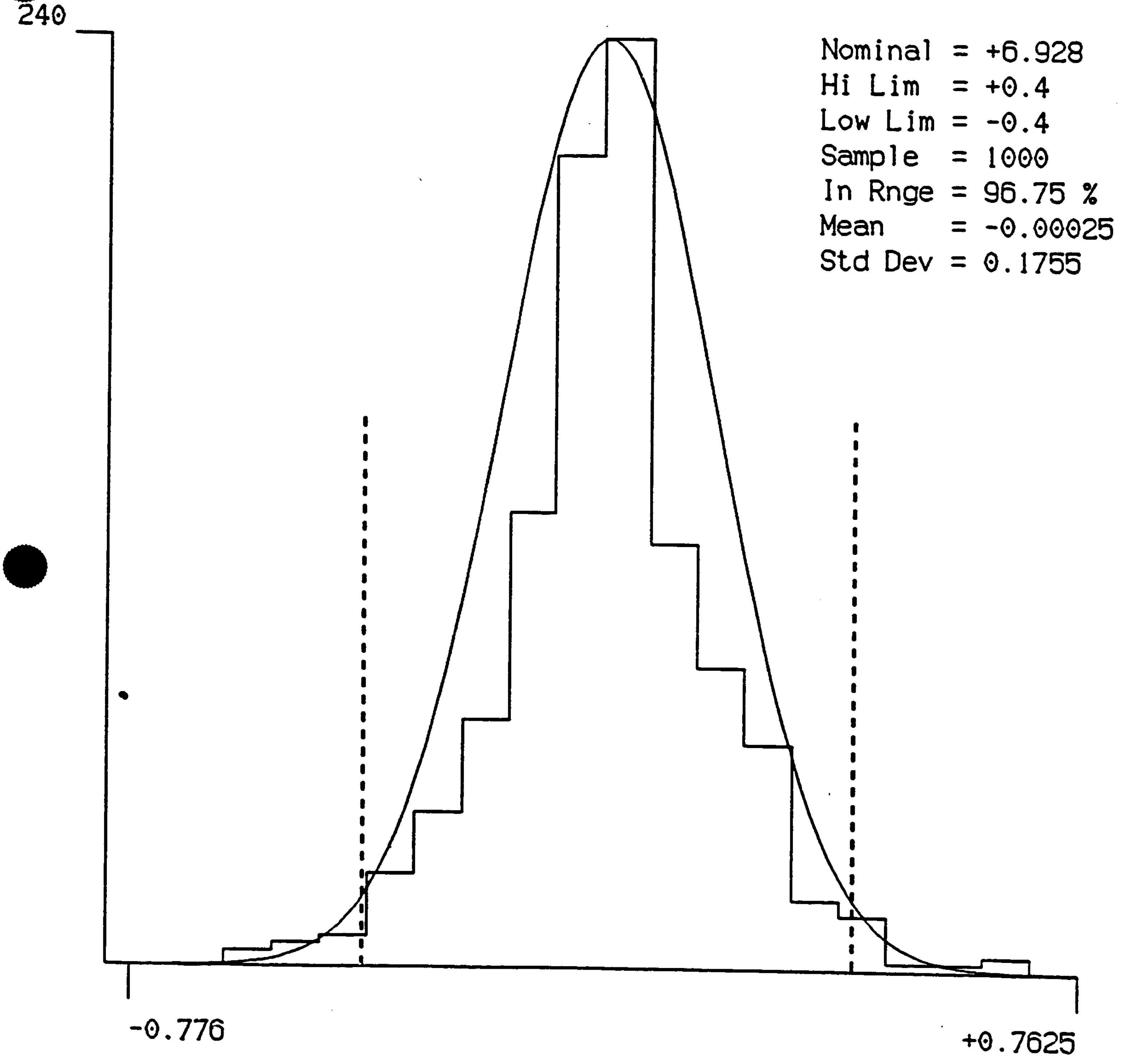

4 +0.24 -0.4
4 +0.2579 -0.3964
6.928 +3.[illegible]4
4 +0.3204 +0.04023


⊥ 0.2 A B
- A -
⊥ 0.2 A
- B -

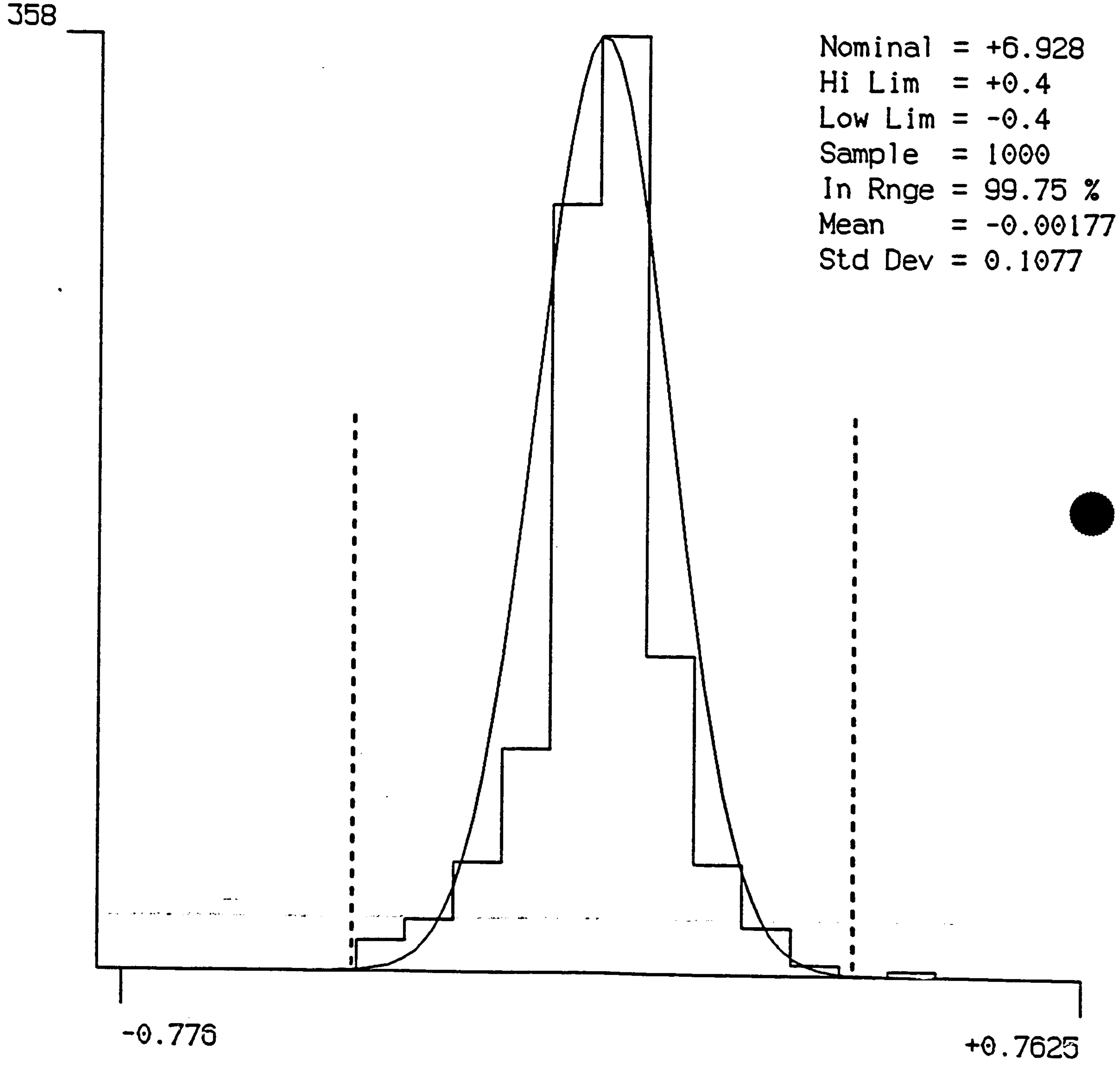
Analysis of CUBE3 with Worst-Case distribution.

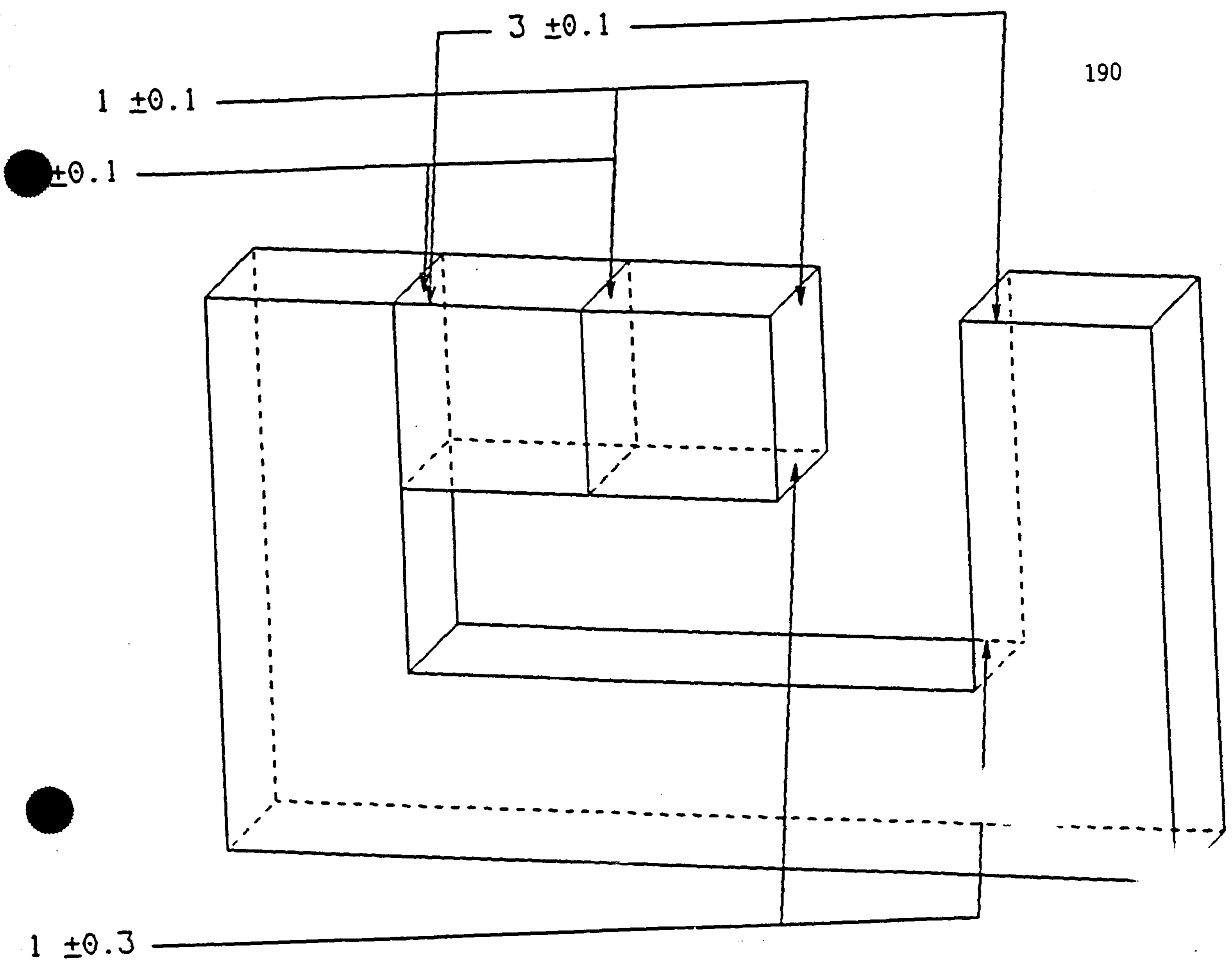


Analysis of CUBE3 with Uniform distribution.

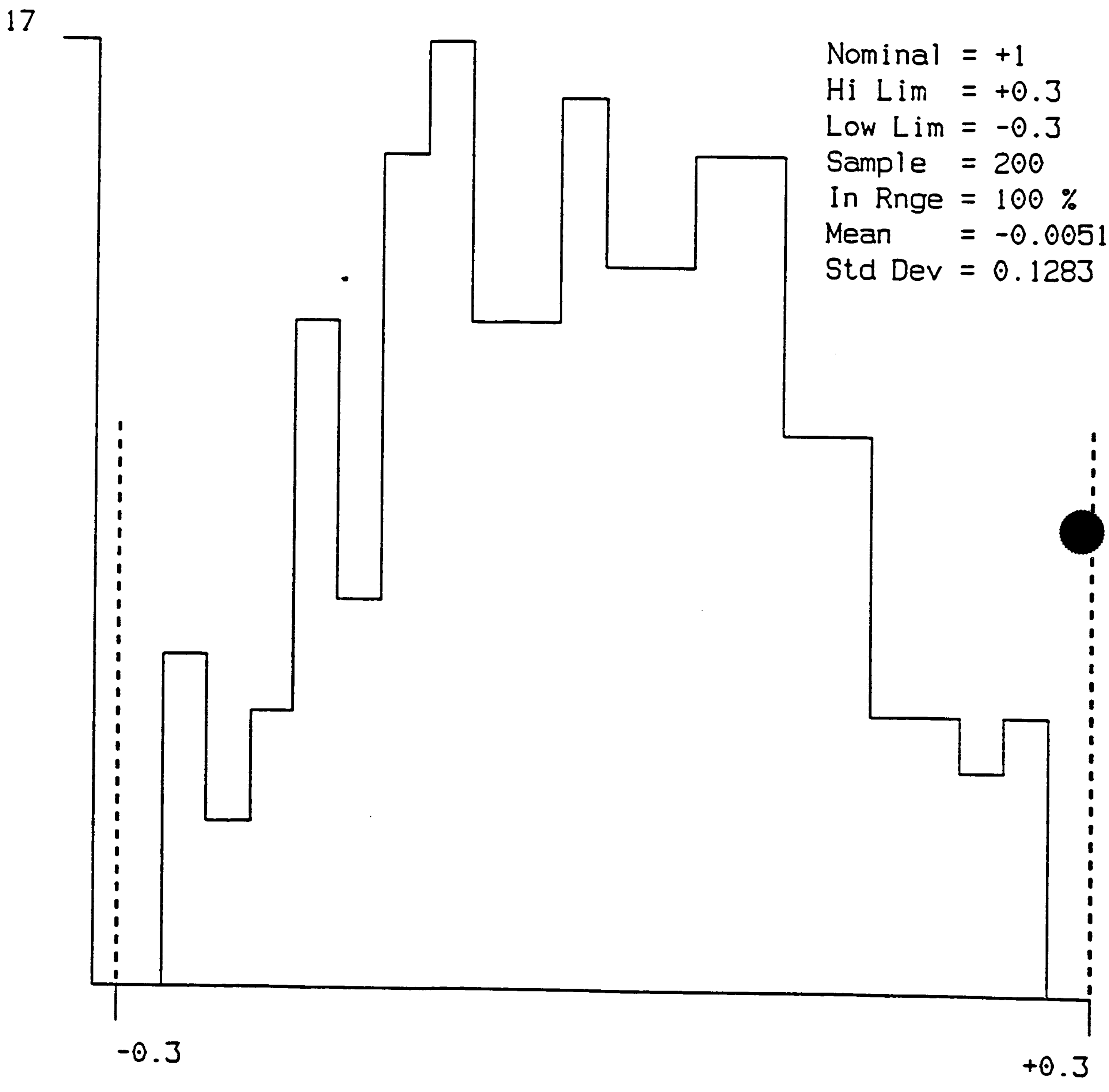


Analysis of CUBE3 with Normal distribution.

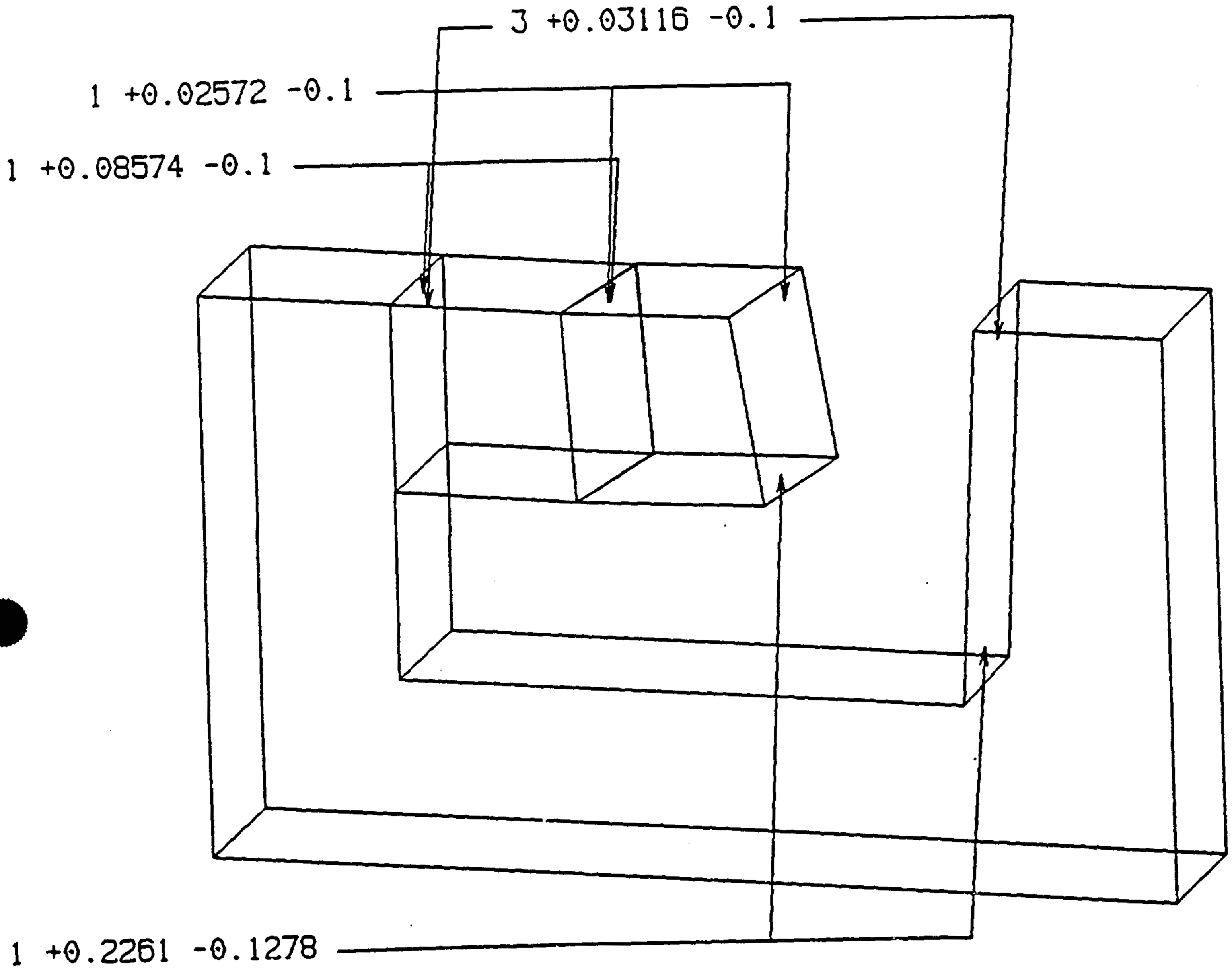



3 ±0.1
1 ±0.1
±0.1
1 ±0.3

Analysis of U5 with Worst-Case distribution.

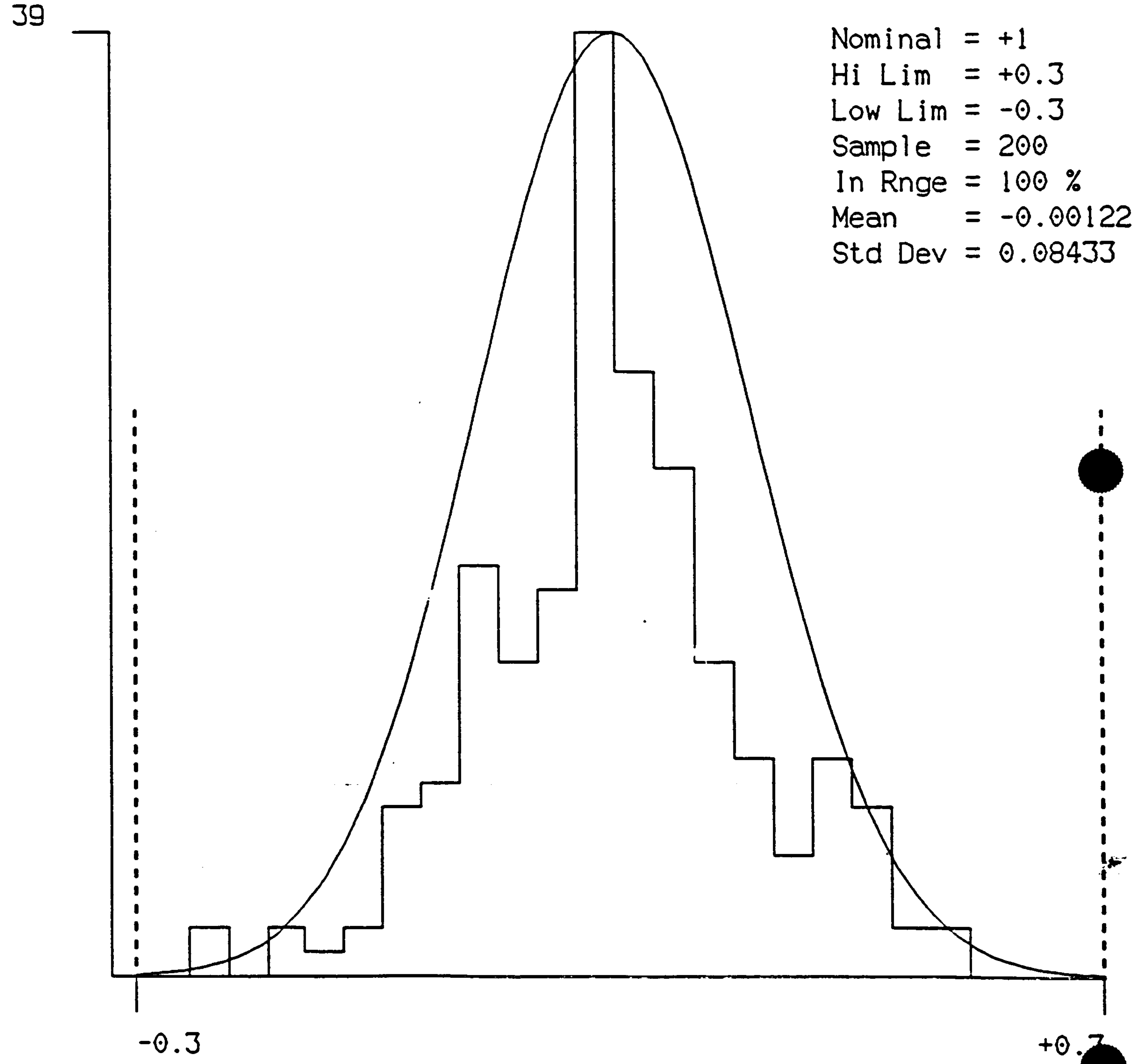


Nominal = +1
Hi Lim = +0.3
Low Lim = -0.3
Sample = 200
In Rnge = 100 %
Mean = -0.00514
Std Dev = 0.1283

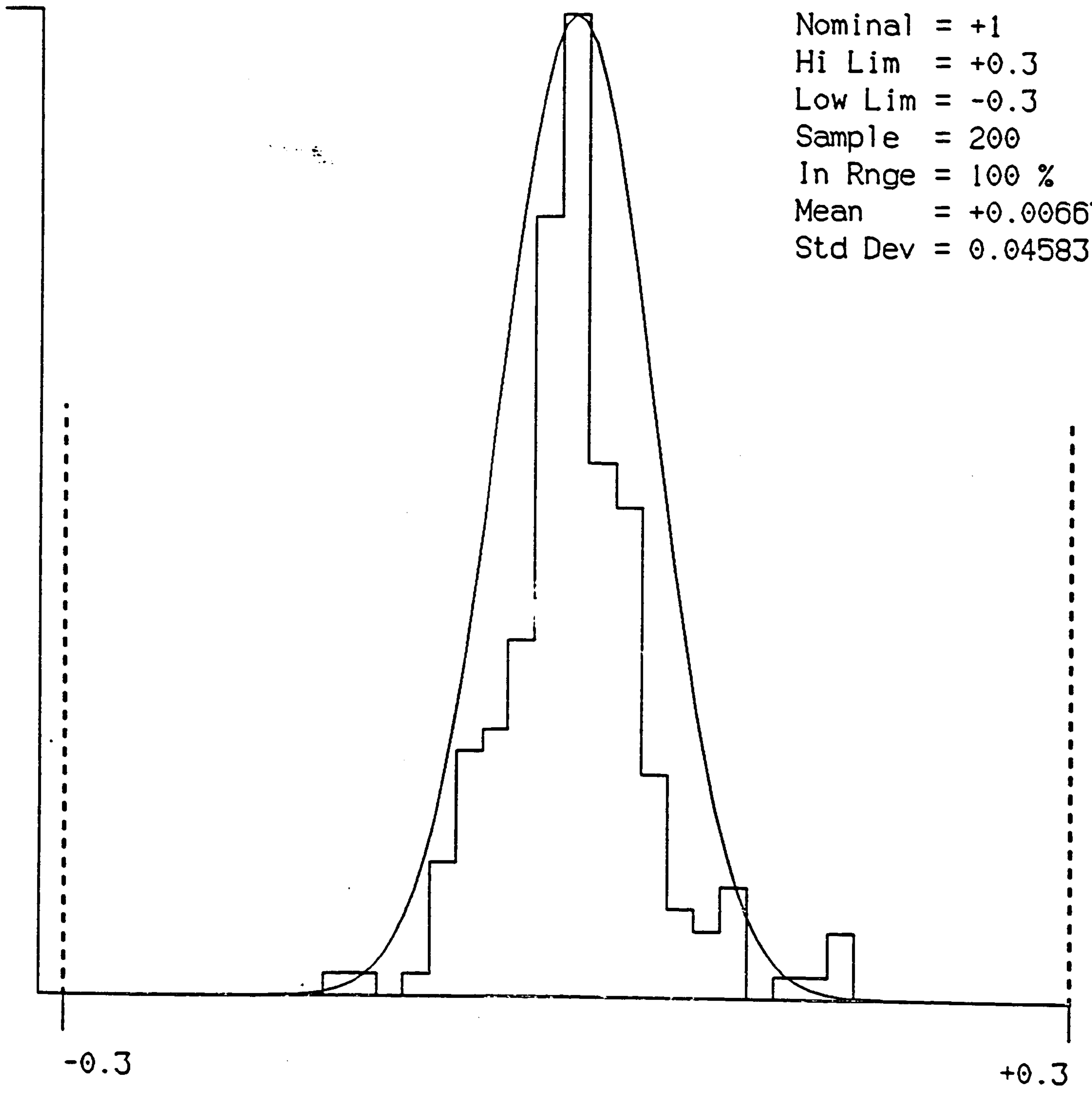


U5001 Instance 12.

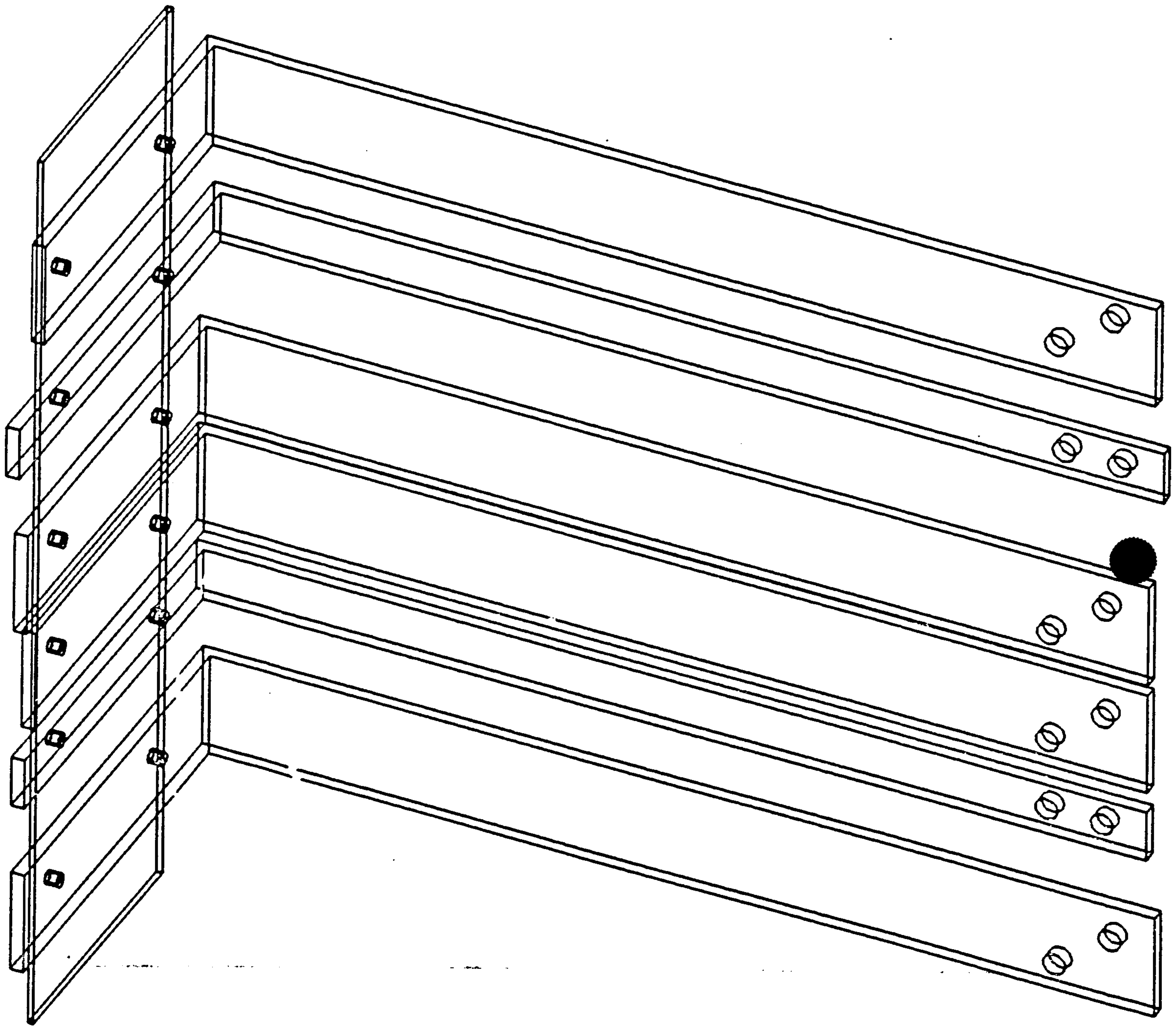
Analysis of U5 with Uniform distribution.

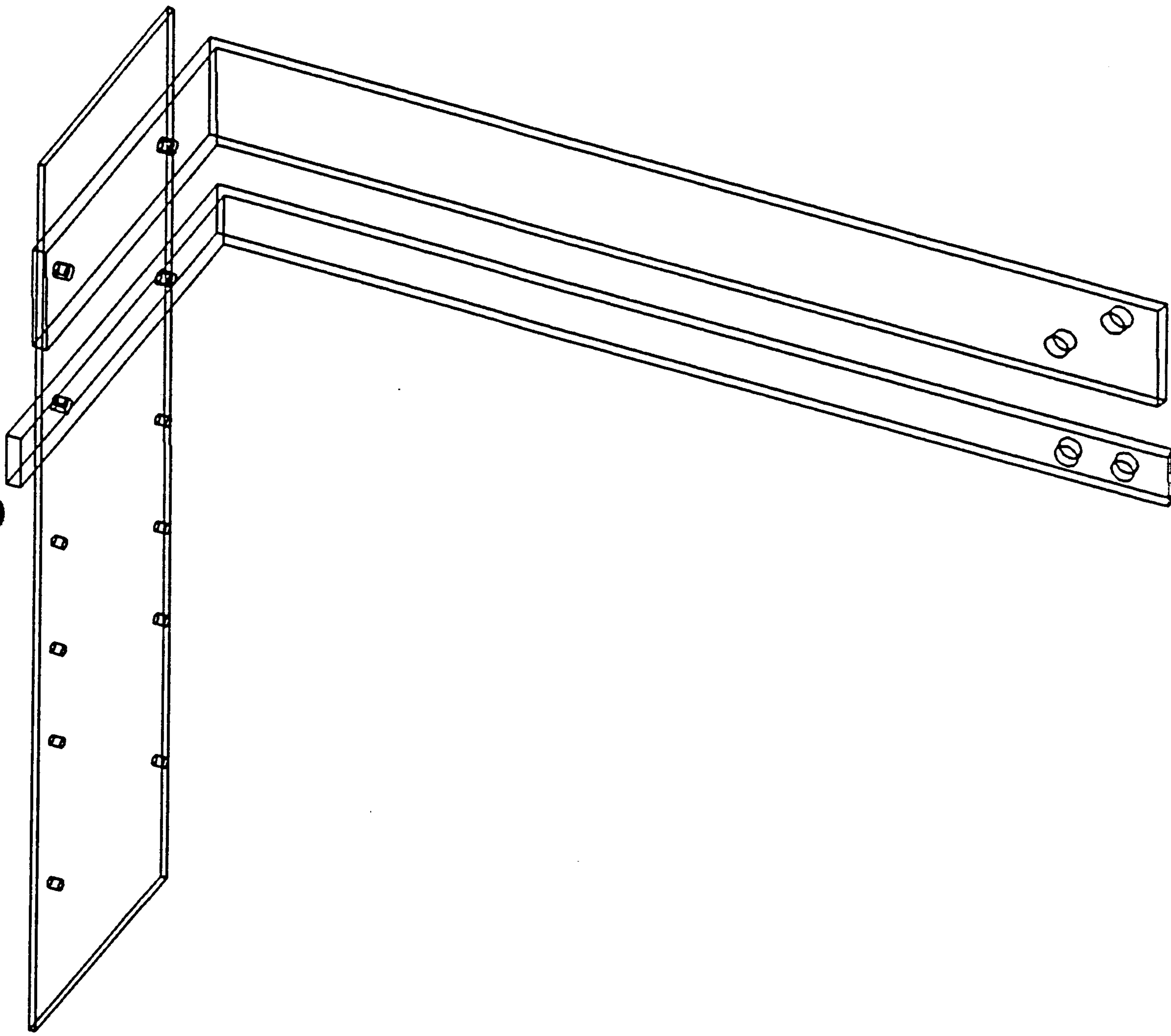


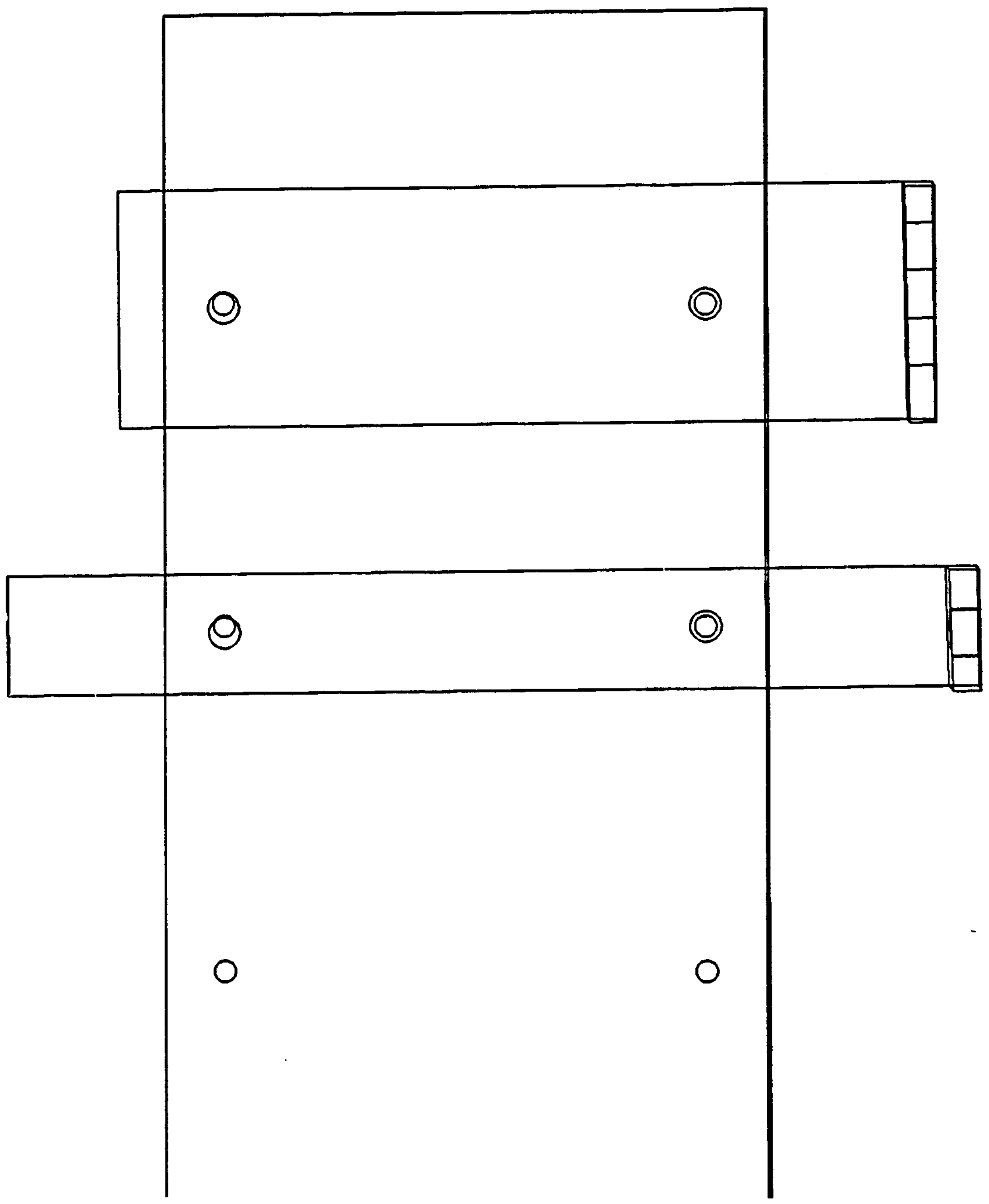
Analysis of U5 with Normal distribution.



```
Nominal  = +1
Hi Lim   = +0.3
Low Lim  = -0.3
Sample   = 200
In Rnge  = 100 %
Mean     = +0.00667
Std Dev  = 0.04583
```







TOLERANCES IN MECHANICAL DESIGN

Joshua U. Turner
David D. Hoh

SUMMARY

Design tolerances on the geometric properties of discrete mechanical parts establish limits on the permissible degree of variation that may be introduced by the parts's manufacturing processes.

These limits are essential to ensure that the parts will meet certain design criteria when they are brought together in an assembly. The actual values assigned to the tolerances have a significant influence over product quality and manufacturing cost. Prior to this project, very little has been done to take advantage of solid modeling technology to provide for the automatic assessment and synthesis of tolerance values.

The first goal of this project is to extend an existing solid modeling system, the geometric design processor (GDP), with capabilities for the representation of various tolerances and part relationships as intrinsic properties of the solid model.

The second goal is to use this information to automatically analyze the tolerance values for acceptability according to given design criteria, and to automatically synthesize optimum tolerance values for minimum manufacturing cost.

RESULTS

Efforts to date have focused on providing capabilities for the representation of various types of tolerances and part relationships, and on developing a capability for automated tolerance analysis.

Tolerance representation: both geometric and dimensional tolerances are supported. Geometric tolerancing capabilities provide for most of the common tolerances of form, orientation, and location. Dimensional tolerancing capabilities provide for tolerances on the size of any feature, and on the distance between any two features (faces, edges, vertices, and center lines of axial features).

Relative Positioning Capabilities:

- These capabilities allow various part relationships to be specified, and enforced as different properties of the model are varied.

At present, the following types of relative positioning capabilities are provided:

- Face-edge-vertex positioning: a face of each of two parts is identified. One part is positioned relative to the other such that the two faces are coplanar, and share a common edge and vertex.

- Datum plane positioning: three planar faces of a part are used to establish a datum reference system. The position of a part feature is maintained relative to this datum reference system.
- Coaxial positioning: one part is positioned relative to another so that a shared axis is maintained.
- Tolerance analysis: a capability has been implemented to generate variations in feature size.

PROJECTED PLANS

The tolerance analysis capability will be extended in the following directions:

Variational coverage: the tolerance analysis capability will be extended to incorporate facilities for generating variations in the distance between part features, in the orientation of part surfaces, and in the position of part features.

Analysis algorithms: the tolerance analysis capability will be provided with two complementary tolerance analysis algorithms:

- Linear programming algorithm: will allow for the rapid analysis of design criteria which are linear, or approximately linear in nature.
- Monte-Carlo algorithm: will apply random sampling methods for the analysis of design criteria which are non-linear or procedural in nature.