CADMADE - An Approach Towards a Device-Independent Standard for CAD/CAM

Software Development

by

Sankar Jayaram

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in
Mechanical Engineering

APPROVED:

Dr. A. Myklebust, Chairman

Dr. C. F. Reinholtz

Dr. M. J. Kurey

Dr. J. R. Mahan

Dr. M. P. Deisenroth

April 21, 1989
Blacksburg, Virginia
Every year thousands of specialized CAD/CAM applications programs are developed to meet the needs of industry, education and research. The international 3-D graphics standard, PHIGS, has proven to be very useful in the creation of custom CAD/CAM software. Although PHIGS+ promises to deliver some geometric modeling procedures, not nearly enough is being done to support the writing of CAD/CAM software. CAD/CAM applications programmers should have available a standardized high level applications programming environment which supports the creation of device-independent and portable design and manufacturing software.

In this dissertation, one approach towards the establishment of a CAD/CAM programming standard has been presented. This programming environment is called CADMADE - Computer-Aided Design and Manufacturing Applications Development Environment. CADMADE includes not only graphics programming support, but also high level procedures to support the creation of geometric modeling, mechanical design, manufacturing, expert systems and user interface software. The requirements of CADMADE have been created. CADMADE consists of five environments: the User Interface Environment (UIE), the Design and Modeling Environment (DME), the Virtual Manufacturing Environment (VME), the Expert Consultation Environment
(ECE) and the PHIGS+ Environment. The User Interface Environment has been designed in great detail. A prototype of the User Interface Environment has been created using PHIGS. Examples of applications programs which use the prototype User Interface Environment are presented. The Design and Modeling Environment has also been designed. A new set of logical input/output devices has been created for the Design and Modeling Environment. The requirements of the Expert Consultation Environment and some new concepts in expert system consultation are discussed.
Acknowledgements

I wish to express my sincere gratitude to my guide Dr. A. Myklebust for his excellent guidance and enthusiastic involvement in my dissertation research. I also appreciate the financial support from him during my tenure as a Ph.D. student. I thank Dr. M. J. Furey, Dr. J. R. Mahan, Dr. C. F. Reinholtz and Dr. M. P. Deisenroth for consenting to be on my advisory committee.

I want to express deep gratitude to my parents and my wife, , for their moral support. I also wish to thank my wonderful wife for helping me put this dissertation together. I wish to thank all other members of my family for their love and support.

I wish to express the debt I owe to my undergraduate school, the Indian Institute of Technology, Kharagpur, and to for encouraging me to take up a career in engineering.
# Table of Contents

1. Introduction .......................................................... 1

2. Literature Review ...................................................... 4
   2.1 User Interface Systems .................................................. 4
   2.2 Geometric Modeling and Mechanical Design in CAD ...................... 7
   2.3 Expert Systems in CAD ................................................ 10
   2.4 Standards .......................................................... 12

3. Problem Statement and Research Objectives ................................ 14

4. ACSYNT/VPI - A Case Study ........................................... 20
   4.1 ACSYNT/VPI ...................................................... 20
   4.2 ACSYNT/VPI User Interface ........................................... 21
      4.2.1 Menu System .................................................... 22
      4.2.2 Message Generation ............................................... 24
      4.2.3 Interactive Input Methods ........................................... 24
   4.3 Geometric Modeling and the Geometry Data Structure ......................... 25
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.4 Utility Functions</td>
<td>28</td>
</tr>
<tr>
<td>4.5 Conclusions</td>
<td>28</td>
</tr>
<tr>
<td>5. Overview of CADMADE</td>
<td>29</td>
</tr>
<tr>
<td>6. The User Interface Environment</td>
<td>35</td>
</tr>
<tr>
<td>6.1 The Menu and Icon Interface</td>
<td>37</td>
</tr>
<tr>
<td>6.1.1 Hierarchical Menu Data Organization</td>
<td>37</td>
</tr>
<tr>
<td>6.1.2 Menu Structure Elements</td>
<td>38</td>
</tr>
<tr>
<td>6.1.3 Menu Structure Data Display</td>
<td>45</td>
</tr>
<tr>
<td>6.1.4 Menu Structure Editing</td>
<td>50</td>
</tr>
<tr>
<td>6.1.5 Menu Types</td>
<td>51</td>
</tr>
<tr>
<td>6.1.6 The Menu Pick Logical Input Device</td>
<td>51</td>
</tr>
<tr>
<td>6.1.7 Applications Controlled Menu Structure Traversal</td>
<td>53</td>
</tr>
<tr>
<td>6.2 The User Communication Interface</td>
<td>57</td>
</tr>
<tr>
<td>6.2.1 Dialog Areas for Applications Program Input/Output</td>
<td>57</td>
</tr>
<tr>
<td>6.2.2 Dialog Areas for Menu/Icon Interface Input/Output</td>
<td>58</td>
</tr>
<tr>
<td>6.2.3 Dialog Areas and Host System Communication</td>
<td>61</td>
</tr>
<tr>
<td>6.2.4 Dialog Area Attributes</td>
<td>63</td>
</tr>
<tr>
<td>6.2.5 Other Features of Dialog Areas</td>
<td>64</td>
</tr>
<tr>
<td>6.3 The CAD View Manager</td>
<td>66</td>
</tr>
<tr>
<td>6.3.1 Predefined Views</td>
<td>66</td>
</tr>
<tr>
<td>6.3.2 Nonrectangular Windows</td>
<td>67</td>
</tr>
<tr>
<td>6.3.3 User Interface for Viewing Transformations</td>
<td>70</td>
</tr>
<tr>
<td>7. The Design and Modeling Environment</td>
<td>75</td>
</tr>
<tr>
<td>7.1 Modeling Entities</td>
<td>77</td>
</tr>
<tr>
<td>7.1.1 Wireframe Modeling</td>
<td>77</td>
</tr>
<tr>
<td>Appendix A. Suggested FORTRAN Bindings for the User Interface Environment</td>
<td>134</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>----</td>
</tr>
<tr>
<td>Control Functions</td>
<td>135</td>
</tr>
<tr>
<td>Output Primitive Functions</td>
<td>137</td>
</tr>
<tr>
<td>Menu/Icon Interface</td>
<td>137</td>
</tr>
<tr>
<td>User Communication Interface</td>
<td>140</td>
</tr>
<tr>
<td>Attribute Specification Functions</td>
<td>141</td>
</tr>
<tr>
<td>Menu/Icon Interface</td>
<td>141</td>
</tr>
<tr>
<td>User Communication Interface</td>
<td>156</td>
</tr>
<tr>
<td>CAD View Manager</td>
<td>164</td>
</tr>
<tr>
<td>Menu Structure Functions</td>
<td>167</td>
</tr>
<tr>
<td>Modified PHIGS Functions</td>
<td>173</td>
</tr>
<tr>
<td>CAD View Functions</td>
<td>175</td>
</tr>
<tr>
<td>Miscellaneous Functions</td>
<td>178</td>
</tr>
<tr>
<td>Appendix B. CADMADE Functions Supported by the Prototype UIE</td>
<td>181</td>
</tr>
<tr>
<td>Appendix C. Code Listing - Prototype UIE</td>
<td>184</td>
</tr>
<tr>
<td>Appendix D. Code Listing - Sample CAD Program</td>
<td>287</td>
</tr>
<tr>
<td>VITA</td>
<td>324</td>
</tr>
</tbody>
</table>
List of Illustrations

Figure 1. ACSYNT/VPI Screen Layout (Wampler, 1988) ..................... 23
Figure 2. Aircraft Model Displayed by ACSYNT/VPI ...................... 27
Figure 3. Overview of CADMADE ........................................ 31
Figure 4. Overview of the User Interface Environment .................... 36
Figure 5. Example of a Menu Tree ...................................... 39
Figure 6. Example of a Menu Tree Traversal ............................ 46
Figure 7. Menu Structure for Example of Traversal Time Binding ........ 48
Figure 8. Example Result of Traversal Time Binding ....................... 49
Figure 9. Example of a Sequence of Menu Picks .......................... 55
Figure 10. Applications Program Controlled Menu Traversal ............... 56
Figure 11. Example of a Menu Tree for String Menu Selection .......... 60
Figure 12. Example of UNIX File Hierarchy ................................ 62
Figure 13. Predefined Principal Orthographic Views ....................... 68
Figure 14. Example of Use of Nonrectangular Window .................... 69
Figure 15. Applications Program for View Manipulation Using PHIGS .... 71
Figure 16. Example of CADMADE View Transformation Maps ............. 73
Figure 17. Applications Program for View Manipulation Using CADMADE ... 74
Figure 18. Overview of the Design and Modeling Environment .......... 76
Figure 19. Example of the Alignment Operator ........................... 83

List of Illustrations

ix
List of Illustrations

Figure 20. The CAD Data Structure Hierarchy .................................. 90
Figure 21. Menu Tree Used to Test the Prototype UIE ......................... 101
Figure 22. Root Menu Screen of Test Program .................................. 103
Figure 23. Example of Text Attributes of Menus .................................. 104
Figure 24. Example of Background Attributes of Menus ....................... 105
Figure 25. Menu Location - Example 1 ........................................ 106
Figure 26. Menu Location - Example 2 ........................................ 107
Figure 27. "SHAPES" Menu Tree for Sample CAD Program .................... 109
Figure 28. "UTILITIES" Menu Tree for Sample CAD Program ............... 110
Figure 29. Menu Screen of Sample CAD Program - Example 1 ............. 111
Figure 30. Menu Screen of Sample CAD Program - Example 2 ............. 112
Figure 31. Model Created using the Sample CAD Program - Example 1 .... 114
Figure 32. Model Created using the Sample CAD Program - Example 2 .... 115
The following is a list of acronyms used in this dissertation.

**ACSYNT**  
AirCraft SYNThesis

**AI**  
Artificial Intelligence

**ANSI**  
American National Standards Institute

**CAD**  
Computer-Aided Design

**CADMADE**  
Computer-Aided Design and Manufacturing Applications Development Environment

**CAM**  
Computer-Aided Manufacturing

**DME**  
Design and Modeling Environment
ECE  Expert Consultation Environment
ET   Expert Technician
GKS  Graphical Kernel System
KBS  Knowledge-Based System
ISO  International Standards Organization
PHIGS Programmer's Hierarchical Interactive Graphics System
UIE  User Interface Environment
UIMS User Interface Management System
VME  Virtual Manufacturing Environment
1. Introduction

In recent years, methods of design have been greatly altered by computer-aided design and manufacturing (CAD/CAM). Sale of CAD/CAM hardware and software is a multi-billion dollar industry. However, as rapidly as software is being developed, we have only begun to scratch the surface on the integration of design software with CAD/CAM systems.

The CAD portion of CAD/CAM systems usually includes design-drafting, 3-D wireframe modeling, surface modeling, solid modeling, preprocessing for finite element methods or links to finite element solvers, geometric analyses and assorted specialized analysis techniques. Although a lot of money and time is being spent on integrating CAD with CAM, very little effort is being concentrated on integrating mechanical design with CAD. Integration of mechanical design with CAD systems is as important an issue as integration of CAD and CAM. It will not be solved by the development of databases and database standards alone.
The process of designing a part or assembly is very complex and often requires some decision making by an experienced engineer. Until a few years back, it was very difficult to program a computer to make such decisions based on subjective criteria. With the help of the recent advances made in the fields of Artificial Intelligence (AI) and Knowledge-Based Systems (KBS), it is now possible to program a set of rules into a computer and allow the computer to make decisions based on those rules. Moreover, the knowledge and experience of an engineer can also be transformed into facts and rules which the computer can use to make intelligent decisions. The time is thus ripe for CAD researchers to devote more time and effort to the integration of CAD systems with mechanical design processes.

Most custom CAD applications programs rely on computer graphics support systems for an efficient and effective use of CAD/CAM techniques. The emergence of the Graphical Kernel System (GKS) as the international 2-D graphics programming standard was a landmark event in the history of computer graphics. The concept of device-independent graphics and standard graphics routines was standardized for the first time. In 1988, Programmer's Hierarchical Interactive Graphics System (PHIGS) was accepted as the American National Standards Institute (ANSI) standard for 3-D graphics programming. PHIGS has greatly enhanced the capability of custom CAD/CAM systems by providing full 3-D functionality along with device independence.

Although custom CAD/CAM systems have benefitted greatly from the standardization of graphics, there is still not enough support for the easy creation of custom CAD software and for the integration of CAD with mechanical design.

In this dissertation, an approach towards the creation of a standardized CAD/CAM applications programming environment is presented. The overall system designed, called
CADMADE, is described. The elements of the programming environment directly related to computer-aided design are discussed in detail. A prototype of some portions of the device-independent environment is described.
2. Literature Review

For the purpose of the literature review, recent advances made in the field of computer-aided design and computer graphics applications in CAD can be broadly grouped into four categories:

1. User Interface Systems
2. Geometric Modeling and Mechanical Design in CAD
3. Expert Systems in CAD
4. Standards

2.1 User Interface Systems

A good user interface is essential for the efficient use of a CAD/CAM system by a designer. For example, a good user interface does not compel the user to follow a sequence of predetermined steps. Sata and Warman (1981) have compiled a number of
good publications addressing the topic of man-machine communication in CAD/CAM. A User Interface Management System (UIMS) assists applications programmers to provide good quality interfaces for their programs. Janni and Italiano (1986) have presented a general purpose user interface management system based on user driven menu handling and a hierarchical organization of graphics data structure. Myers and Buxton (1986) have designed a system that automatically creates the code for user interfaces while the designer demonstrates to the system how the interface should look and work. Gallop (1987) has discussed the importance of standards and standardization in developing user interface management systems. Kilgour (1987) has outlined the development of the user interface management system (UIMS) as a concept and has discussed some issues related to the design and implementation of such systems. Haeberli (1988) has discussed a user interface for dynamically building and modifying graphics applications.

Koivunen and Mantyla (1988) have described a user interface management system intended for applications in mechanical CAD. In this system, a three-layered internal architecture is used which clearly separates presentation, dialog control and applications processing. Olsen and Dance (1988) have described a macro facility that allows end users to extend interactive graphical applications as a part of a user interface management system. Both these systems were device dependent and were designed to be used on specific computer systems.

Wampler (1988) has described the design and implementation of a hierarchical menu system (based on PHIGS) as part of a user interface for an aircraft design code. Blevins (1988) has described a modular graphics menu manager which uses PHIGS. This menu manager is system dependent and creates command files and PHIGS archive files which
are used for the menu interface for applications programs. Although this menu manager
uses PHIGS as the graphics support system, it uses computer system dependent
command sequences to process menu selections and execute appropriate portions of the
applications program.

Some researchers have tried to improve man-machine interaction by designing new input
devices. Kankaanpaa (1986) has described an experimental display system that allows
the use of a pen-like device to draw directly on the display surface. Thatch and
Myklebust (1988) have described a six-degree-of-freedom input system using PHIGS for
rotation and translation of solid objects. Chen, et al. (1988) have described the design
and evaluation of four virtual controllers for use in rotating 3-D objects using a mouse.
They have studied user performance during simple and complex rotation tasks.

Ware and Jessome (1988) have described the configuration of a six-dimensional mouse
and the interfacing of the mouse with an Iris workstation. The six-dimensional mouse
allows an object to be translated in X, Y and Z coordinates and rotated about the three
axes. Chan and Hoeltzel (1988) have described a new 3-D input system aimed at
alleviating the labor-intensive task involved in specifying data input for the description
of 3-D objects. This input system is based on the use of an intelligent user interface and
a three-dimensional digitizer for the direct input of spatial coordinates.

Recently a number of window managers have emerged to the forefront of User Interface
Management Systems (UIMS). Myers (1988) has presented a taxonomy of window
manager user interfaces. He has evaluated the similarities and differences of various
window managers and has discussed some of the issues that need to be addressed by
designers of future window manager user interfaces. One of the window managers
discussed in the above-mentioned publication is the X window system (Scheifler and
The X window system has achieved fairly widespread popularity recently, particularly in the UNIX community.

2.2 Geometric Modeling and Mechanical Design in CAD

Geometric modeling provides an analytical, mathematical and abstract description of objects. Geometric modeling is an important part of CAD and CAM. Until recently, most CAD/CAM systems relied primarily on curves and/or surfaces for describing models. With the advent of fast CAD/CAM workstations, solid modeling has become the most popular geometric modeling method for CAD/CAM applications. Researchers are still trying to develop better methods of using solid modeling in CAD/CAM systems. For example, Saeed, et al. (1988) have discussed a class of operations called "offsetting" for solid models. However, recent interest in the field of integration of CAD and CAM has led to the development of "feature-based" modeling. A feature is defined as an entity which captures a characteristic of a local area of a part (e.g., hole, corner, shaft, etc.). Chen (1988) has discussed the concept of part feature and the possibility of using part feature to integrate CAD and CAM systems.

Shah and Rogers (1988) have briefly described the problems in applying solid modelers to engineering applications. They have emphasized the fact that solid modelers do not provide an environment conducive to creativity in design. They have described an expert system shell for form feature modeling. The expert system shell can also extract feature information from design data.
Woodwark (1988) has discussed the possible use of the set-theoretic (constructive solid geometry) solid model for feature recognition. Some of the problems of applying a feature recognition process to shape models have also been outlined in the same paper. The development of a graph-based method for the recognition of machined features from 3-D boundary representation solids is described by Joshi and Chang (1988). A model for topological and geometric reasoning about mechanical designs is described by Duffey and Dixon (1988). A computer program which uses the 2-D domain of extrusion cross-section as a case study is demonstrated as a model.

Deitze (1989) has discussed the importance of parametric programming and feature-based design. Gossard, et al. (1988) have addressed the topics of dimensions, tolerances and features in solid models. Luby, et al. (1986), Dixon and Simmons (1986), Drake and Sela (1989) and others have described research and applications in the area of feature-based modeling. Gandhi and Myklebust (1989) have discussed a natural language approach to feature-based modeling for the creation of geometric models of features.

Although the research in feature-based design was initiated by the need for a better and easier method for the creation of models of shapes and forms used frequently in mechanical design, very little research is being concentrated on integrating mechanical design with CAD. Serrano and Gossard (1986) described interactive software which integrates mathematical models (consisting of algebraic equations, etc.) with geometry models to aid in the preliminary (or conceptual) design of mechanical components. Cagan and Agogino (1987) developed a design methodology to perform innovative design of mechanical structures from first principle knowledge.
Various studies in the field of variational geometry have been presented by Lin, Gossard and Light (1981), Light and Gossard (1982), and Aldefeld (1988). "Variational Geometry" uses a single representation to describe the entire family of geometries that share a generic shape (Light and Gossard, 1983).

Gandhi and Myklebust (1986) have described methods for designing cams from design through manufacture using CADAM and GKS. Irwin and Myklebust (1986) have discussed a similar system for designing gears.

Every CAD/CAM applications program has to store and retrieve large amounts of data. This has led to research in the field of databases and data structures for CAD/CAM. Roy and Liu (1988) have proposed a semantic association model to organize explicit interface information between the major part components in an assembly environment.

Arora and Mukhopadhyay (1988) have described a generalized relational model to handle large matrices and tables that are encountered in numerous CAD applications. A database management system based on the model has also been described. Kaufman (1988) has discussed the recent explosion of interest in computerized material property databases and intelligent knowledge systems for flexible manufacturing processes. Colton (1988) has discussed the design and implementation of a relational materials property database.

Howard and Howard (1988) have described a conceptual approach for multimodal engineering database interface combining multipurpose graphics, an engineering query language and other interface methodologies in an engineering workstation environment.
2.3 Expert Systems in CAD

In recent years some leading researchers in the field of CAD/CAM have been concentrating on developing means of utilizing Artificial Intelligence (AI) to improve the techniques used in CAD/CAM. The following is a brief literature review of the various applications of AI and expert systems in CAD/CAM and mechanical computer-aided engineering (MCAE).

Hatvany (1984) presented a prediction of the future of CAD and identified knowledge-based techniques as the most important tool for CAD for the future. Calkins (1985) created an interactive graphics environment in conjunction with expert systems to create and display a three-dimensional representation of a sculptured surface object from three orthographic views. Davidson (1985) has drawn up a list of requirements for an expert system based CAD decision-making tool. Vaghul, et al. (1985) have described an expert system which evaluates designs of a sub-class of injection molded parts on simple manufacturability criteria. Etesami and Zarefār (1985) have discussed the design of a network of expert systems to emulate a design team. Kinoglu, et al. (1986) have described an expert system based implementation of a constraint based approach to design. Jayaram and Myklebust (1989) investigated the automatic generation of interfaces between design applications programs and CAD systems, with the aid of an expert system.

Rosenman, et al. (1986) have demonstrated the use of rule-based expert system shells in elementary design decision making. Ohsuga (1983) has discussed a new method for
describing and manipulating engineering models. This method makes use of a knowledge base and inference techniques and allows both model structure and model attributes to be represented in a declarative form. Whitney and Bond (1987) integrated a CAD system, CADAM, with an AI system using VM/PROLOG to produce an intelligent CAD/CAM system. This system allows the user to interact through one screen to (i) work graphically in the conventional CADAM system, (ii) write application programs in PROLOG, or a mixture of PROLOG and FORTRAN, (iii) interact with the applications program, set goals and monitor results in PROLOG and (iv) produce modified drawings in PROLOG and display them in CADAM.

Some researchers have tried to develop new computer languages to improve the current techniques used in CAD. A “Function-Description” language was developed by Lai (1987) and Lai and Wilson (1987) to describe functional relationships between elements in a mechanical design. This system applies a series of design rules to the design and generates a set of recommendations for improving the design. Takase and Nakajima (1984) developed a “Feature-Descriptor” language for describing and modifying assembled machines.

Chan and Paulson (1987) have discussed the representation of the dual nature of constraints to maintain design consistency as a design evolves. The dual nature of the constraints arises from the fact that constraints need not only check designs but can also be used to derive design solutions that satisfy the constraints. Nguyen and Rieu (1987) have defined a new methodology for semantic constraints management and control. This methodology provides a unified basis for modeling the dynamic nature of complex
objects. Aldefeld (1988) has described a method for processing generic geometric models on the basis of symbol manipulation and forward inferencing.

Ishii, et al. (1988) have developed a general framework for knowledge based computer tools that promote simultaneous engineering in mechanical design. The underlying concept for these knowledge-based systems focuses on the compatibility between the design specification and the proposed design, evaluates the design based on the knowledge of experts and suggests improvements. Ramchandran, et al. (1988) have presented the development of knowledge-based aids for the design of mechanical systems.

Other fields of mechanical engineering where AI has been applied include kinematics, material selection, process planning, control system design, design of structural parts, design optimization, heat fin design, valve selection and problem diagnosis.

2.4 Standards

Activities in proposing and accepting graphics standards have been prevalent for more than a decade. The Graphical Kernel System (GKS) became an ISO standard for computer graphics programming in 1985 (Hopgood, et al., 1986). PHIGS was accepted as a standard for 3-D graphics by ISO in 1988.

Chin (1988, 1) has described the scope of graphics standardization efforts. Carson (1988) has discussed the future of ISO graphics standards. Chin (1988, 2) has described
the new procedures for graphics standardization adopted by ISO and its working groups. Meads and Puk (1988) have reported the initiation of new study groups by the computer graphics standards.

Besides graphics, standards are available for other areas associated with CAD/CAM. The Initial Graphics Exchange Specification (IGES) is a standard for transferring model data between dissimilar CAD/CAM systems. Pasquill (1988) has discussed the reasons for wishing to transfer product definition data between dissimilar CAD/CAM systems and the practical use of IGES to perform this task. Brandli and Mittelstaedt (1989) have examined the standards for exchange of solid model data between different CAD systems. This is very important with respect to the field of Computer Integrated Manufacturing (CIM). X windows is becoming a de facto standard for window management.

The creation of standards for graphics has proven to be very useful in the creation of CAD software. Thatch and Myklebust (1988) have described a graphics preprocessor that interactively assists in the definition of spatial mechanism problems. PHIGS was used for graphics support in this preprocessor. Wampler, Myklebust, Jayaram and Gelhausen (1988), have described the development of a PHIGS-based aircraft design system. This CAD software is designed to be used on the new generation of high-speed imaging workstations. The use of PHIGS assures graphics device independence and thus the software can be used on a number of different workstations which support PHIGS.
3. Problem Statement and Research Objectives

Although research in the field of CAD/CAM integration has been prevalent for several years, the integration of CAD and mechanical design has only recently been given the importance it really deserves. Leading researchers in this field have been concentrating mainly on using artificial intelligence (AI) techniques and expert systems in CAD/CAM and Mechanical Computer-Aided Engineering (MCAE). Few investigators have directly addressed the topic of bringing together mechanical design and geometric modeling.

Many commercially available CAD/CAM systems provide 3-D wireframe modeling and some also allow the creation of solid models. But these large commercial CAD systems are usually unable to treat complex mechanical design problems. These CAD systems mainly allow the creation of geometric models and the corresponding drafting of the design and details. This is due to the fact that each design problem requires its own set of design rules and has to be tailored to comply with the functional requirements of the design. Thus no predefined CAD system, however versatile and varied it may be, can hope to be able to address all the mechanical design needs of each industry and
organization. As a result, thousands of applications programs are developed each year to meet the needs of industry, research and education.

The process of developing custom CAD/CAM software involves a series of repetitive and time-consuming preliminary processes. After a decision has been reached regarding the geometric modeling techniques that need to be used for the particular application, thousands of lines of computer code are typically written to realize these geometric models (surfaces, curves, solids, etc.) and store the models in some form of a data structure. The design and implementation of the data structure to serve this purpose can also prove to be a time-consuming and expensive process.

Sometimes the original choice of the geometric modeling techniques may need to be changed. For example, bicubic Ferguson surface patches may have been initially chosen for generating the geometric models. Bicubic Ferguson surface patches are only useful for tangent continuity between surface patches. If at any stage, a decision is taken to incorporate a higher order continuity, large amounts of new computer code will have to be written to allow the CAD/CAM system to implement a different surface model. Thus, changing the geometric modeling techniques in a custom CAD/CAM system to satisfy changing needs and objectives can become very time-consuming and expensive.

Geometric modeling and computer graphics are very closely related in CAD/CAM systems. Once the geometric model has been defined, it is usually necessary to display the model on a graphics device. In this respect, PHIGS has already proven to be very useful in the creation of custom CAD software. Although PHIGS+ promises to deliver some geometric modeling procedures, not nearly enough is being done to support the writing of CAD/CAM software. The inclusion of B-splines and Bezier curves as primitives in the proposed PHIGS+ extensions is an indication of the intentions of

3. Problem Statement and Research Objectives

15
PHIGS+ to better support CAD/CAM and geometric modeling in the future. However, it has just begun to scratch the surface of the requirements of a standard in geometric modeling. Besides curve and surface generation, the state of the art in geometric modeling has rapidly changed in recent years to include solid modeling, feature based modeling, parametric modeling, etc. Thus, geometric modeling has grown so much that it might not be possible to include all the important features of geometric modeling in PHIGS+ and similar graphics standards. Thus, for any custom CAD/CAM software to be able to display geometric models, it will be necessary to develop code to use existing graphics standards and generate the display structures.

Another common requirement of interactive CAD/CAM systems is a good quality human interface. A number of User Interface Management Systems are available in the market. The literature review presented in the previous chapter shows a growing interest in User Interface Systems. However, almost all these systems are device-dependent and cannot be used to provide user interfaces for device-independent CAD/CAM systems. More device-independent support needs to be provided in the area of views and viewing and menu/Icon creation without going to the extremes of X windows or NeWS. Some CAD/CAM systems have been developed and are being developed based on the PHIGS/PHIGS+ standards while some others are being converted to use PHIGS for graphics support. Although these standards provide interaction and viewing facilities required to develop user interfaces, each one of these CAD/CAM systems has had to have designed and implemented methods of generating menus, managing menus, defining menu trees, defining and modifying views, managing prompts/messages, etc.

The growing importance of expert systems in CAD/CAM systems integrated with mechanical design is evident from the explosion of research and publications in this field.

3. Problem Statement and Research Objectives
The process of designing a part or assembly often requires decisions to be made based on subjective criteria. With the help of AI and Knowledge-Based Systems (KBS), it is now possible to incorporate such decision making systems with computer-aided engineering design systems. However, these expert-system-based CAD/CAM systems have to be developed from basic principles using AI languages (e.g., LISP, PROLOG, etc.). Commercially available expert system shells are system dependent and do not provide the applications programmer with the flexibility to easily and fully integrate the expert systems developed in these shells into custom CAD/CAM applications.

Thus, custom CAD/CAM applications programmers have to develop their own knowledge base, write their expert system from basic principles, set up the interaction between the user and the expert system, and set up the relationships between the CAD/CAM database and the expert system.

In recent years, new high-speed imaging workstations have started replacing the ‘dumb’ graphics devices which need to be supported by mainframe computers. These workstations are excellent platforms for CAD/CAM applications since they provide very good graphics display and high speed computing facilities. Most of these workstations have special hardware/software to speed up modeling and graphics operations. However, these workstations are constantly changing and improving. Once every few months, a new and better engineering workstation is being announced by the various manufacturers of these workstations. In order to be able to continuously take advantage of the improving technology, custom CAD/CAM software should be developed with the help of a device-independent and portable system. This would ensure upward compatibility and prevent the software from being tied down to one workstation vendor.

3. Problem Statement and Research Objectives
The future of CAD/CAM depends on the easy creation of specialized CAD/CAM applications software rather than on a group of very large, general purpose CAD/CAM systems as we have now. CAD/CAM applications programmers should have available a standardized high-level language in the form of a set of procedures and data structures which support the creation of device-independent, portable CAD/CAM software. We have been fortunate in having standards for programming languages since the early days of computing. Although they were much needed, graphics standards (e.g., GKS, PHIGS) were much slower in coming. PHIGS has already proven to be very useful in the creation of custom CAD software. A wide variety of modeling procedures, including both surface and solid models, and a general modeling data structure need to be provided in a standard programming environment.

The standardized CAD/CAM language should allow the applications programmer to make high level calls and easily develop custom CAD software with the appropriate user interface without having to spend time on the basic and repetitive processes of developing code to support menu systems, geometric modeling, expert systems, etc. Artificial intelligence support should be available to the CAD/CAM software engineer in a consistent, device-independent fashion. The same is true for a general database and database manager. This system should also include the device-independent 3-D international graphics standard PHIGS as one of its features to allow easy conversion of existing device-independent code to include the new features of the device-independent CAD/CAM programming language.

The objectives of this research are to:

1. Investigate the concept of a machine-independent and device-independent CAD standard.
2. Determine functional requirements of the proposed CAD standard.

3. Design the standard programming environment.

4. Define the relationships between the various elements in the CAD standard.

5. Develop a prototype of a portion of the programming standard.
In order to examine the development of a typical custom CAD applications program, a case study was performed. The target of the case study was ACSYNT/VPI, a large computer-aided design system for conceptual design of aircraft. The development of this CAD system was closely studied and all the preliminary processes which were required for the development of the applications program were noted and are presented below. ACSYNT/VPI was chosen for the case study because of the author’s involvement in the development of this CAD system for two years as a graduate student and because of its influence on the ideas presented in this dissertation.

4.1 ACSYNT/VPI

ACSYNT (AirCraft SYNThesis) is a well-known aircraft conceptual design code which was developed in the early 1970’s at NASA’s Ames Research Center to fulfill the need
to study the effects of advanced technology on future aircraft (Gregory, 1973). Currently the code is used at NASA-Ames for the analysis and design of Advanced Supersonic Short Take-Off and Vertical Landing (ASTOVL) aircraft (Gelhausen and Wilson, 1986). It is also used at a number of aerospace companies, notably Northrop and Boeing. In 1987, the CAD/CAM laboratory at VPI&SU began developing a computer aided design (CAD) interface for ACSYNT (Wampler, et al., 1988). This interface permits the execution and control of the design process via interactive graphics menus and, by visual inspection of data and aircraft model shaded images, allows rapid evaluation of design configurations. This CAD interface was coded entirely with the new 3-D graphics standard, PHIGS. The CAD interface along with ACSYNT is called ACSYNT/VPI and is designed to be used on the new generation of high speed workstations. The use of PHIGS and ANSI standard FORTRAN 77 renders ACSYNT/VPI machine-independent and graphics device independent.

The methods and procedures used in developing the user interface, geometric modeling techniques, data storage, etc. are discussed below.

4.2 ACSYNT/VPI User Interface

The user interface for ACSYNT/VPI was designed to be user friendly. It allocates screen space and uses color to lead the user’s attention to what is most important. The menu layouts are kept simple and uncluttered to minimize hand and eye movement. The user
interface uses command menus, geometry views, message display areas and logical input devices. Figure 1 shows the ACSYNT/VPI screen layout.

4.2.1 Menu System

A menu-based design philosophy was used in ACSYNT/VPI in which all pickable items on the screen - menus as well as geometry - are treated as menus. A hierarchical command menu tree was designed to allow the user to select the various available functions and options. The command menu structure was implemented as a set of chained "menu modules", each representing one item in the menu structure. Menu modules either perform a function associated with a particular menu item, display a submenu, or both. Most menu modules consist of up to eight subroutines which are identical in function from one module to another. A second hierarchical menu tree called the "standard menu" is also available (shown on the bottom right of Fig. 1). The standard menu items are available to the user at all times, irrespective of the current command menu screen. A number of menu "utility" functions were designed to provide for the manipulation of menus. These utility routines amounted to approximately 1300 lines of documented FORTRAN code. They allow the creation and display of menus with a fixed "look and feel" designed specifically for ACSYNT/VPI. These menu utility functions, however, cannot display submenus automatically by traversing the menu tree. The applications programmer must take care of displaying submenus and removing them when the user wants to go back to the previous menu.
### ACSYNT/VPI - X1.0:

<table>
<thead>
<tr>
<th>MENU TITLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITEM 1</td>
</tr>
<tr>
<td>ITEM 2</td>
</tr>
<tr>
<td>ITEM 3</td>
</tr>
<tr>
<td>ITEM 4</td>
</tr>
<tr>
<td>ITEM 5</td>
</tr>
<tr>
<td>ITEM 6</td>
</tr>
<tr>
<td>ITEM 7</td>
</tr>
<tr>
<td>ITEM 8</td>
</tr>
<tr>
<td>ITEM 9</td>
</tr>
<tr>
<td>ITEM 10</td>
</tr>
<tr>
<td>ITEM 11</td>
</tr>
<tr>
<td>ITEM 12</td>
</tr>
<tr>
<td>ITEM 13</td>
</tr>
<tr>
<td>ITEM 14</td>
</tr>
<tr>
<td>ITEM 15</td>
</tr>
<tr>
<td>ITEM 16</td>
</tr>
<tr>
<td>ITEM 17</td>
</tr>
<tr>
<td>ITEM 18</td>
</tr>
</tbody>
</table>

**GEOMETRY DISPLAY AREA**

<table>
<thead>
<tr>
<th>MESSAGE LINE 1</th>
<th>COPY</th>
</tr>
</thead>
<tbody>
<tr>
<td>MESSAGE LINE 2</td>
<td>COLOR</td>
</tr>
<tr>
<td>MESSAGE LINE 3</td>
<td></td>
</tr>
<tr>
<td>MESSAGE LINE 4</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STRING INPUT AREA</th>
<th>HELP</th>
<th>EXIT</th>
</tr>
</thead>
</table>

**Figure 1.** ACSYNT/VPI Screen Layout (Wampler, 1988)
4.2.2 Message Generation

A number of "message utility" routines were designed to allow the display of messages on the ACSYNT/VPI screen. The message generator permits five message lines to be displayed at any time with the previous message lines being scrolled up when a new message line is inserted in the message area.

4.2.3 Interactive Input Methods

All of the interactive user input of ACSYNT/VPI is accomplished through pick, valuator and string input devices. All these input devices operate in event mode. The pick device is used to select menu items from any one of the ACSYNT/VPI menu types.

The PHIGS string device allows character string data (e.g., filename, component name, etc.) to be entered using a keyboard. The string input area for ACSYNT/VPI is shown in Fig. 1. The string input is also used for numeric input. Utility routines had to be designed to parse a character string input and extract numerical values from the input text.

Many portions of ACSYNT/VPI allow the use of valuator input. For example, valuators are used for view transformations and for entering scalar values for numerical input. Seven valuator devices allow the user to enter the following view transformation values:

1. Rotation about the X axis.
2. Rotation about the Y axis.
3. Rotation about the Z axis.
4. Translation about the X axis
5. Translation about the Y axis
6. Translation about the Z axis
7. Uniform Scaling

Viewing procedures had to be developed to get the input from the valuators, compute the view transformation matrix and apply the matrix to the 3-D PHIGS viewport causing the geometry on the screen to rotate, translate or scale.

4.3 Geometric Modeling and the Geometry Data Structure

Geometric modeling is used within ACSYNT/VPI to mathematically describe the geometry of an ACSYNT aircraft, display an image of the aircraft and provide ACSYNT with information required to perform some aerodynamic calculations. ACSYNT/VPI is designed to maintain the following models of aircraft:

- ACSYNT/VPI component model
- ACSYNT/VPI component cross-section model
- ACSYNT/VPI component surface model
• ACSYNT/VPI PHIGS display model

The ACSYNT/VPI component model defines each aircraft component parametrically. A hierarchical structure is defined for the aircraft components; i.e., each component has a "parent" component and can have a number of "children" components. The ACSYNT/VPI component cross-section model defines each component as a set of parametric cross-sections. The cross-section model is used to generate the surface model of the aircraft. The surface model of a component is generated by "lofting" parametric surface patches over the cross-sections defining the component. HERMITE bicubic patches are used in ACSYNT/VPI and are defined by four corner points, tangent vectors and twist vectors. PHIGS display structures are created from surface models by interrogating the surface patches with lines. Figure 2 shows an example of an aircraft model displayed by ACSYNT/VPI.

In order to be able to achieve the geometric modeling procedures described above, a data structure was designed and implemented to store and retrieve all the data for the various models described above. Utility routines were written to generate geometry coefficient matrices for surfaces from the surface data structure, generate points on the surfaces for drawing the lines to be displayed and create the PHIGS structure hierarchy from the component hierarchy.
Figure 2. Aircraft Model Displayed by ACSYNT/VPI
4.4 Utility Functions

To assist in the development of the user interface and geometric modeling in ACSYNT/VPI, a number of utility procedures (other than the ones mentioned above) were designed and implemented. These include mathematical procedures (vector operations, matrix manipulations, etc.), character string manipulation procedures, viewing procedures, numerical input procedures and curve generation and interpolation functions.

Other utilities are being developed to calculate cross-sectional areas and to display templates for data input.

4.5 Conclusions

The preliminary process of developing all the utility procedures mentioned above took approximately one man-year. Many of these procedures are developed by most CAD/CAM applications programmers as a prelude to the development of the actual CAD/CAM system. If most of these utilities had been available as higher-level, device-independent functions, much of the effort spent in the preliminary stages of software development could have been saved. This case study demonstrates the need for a higher-level, device-independent CAD/CAM programming environment which goes beyond the scope of graphics standards and allows easy and fast creation of custom CAD/CAM software.
5. Overview of CADMADE

CADMADE stands for Computer-Aided Design and Manufacturing Applications Development Environment. CADMADE is designed to be a programming environment or an Applications Programming Interface (API) which facilitates the development of custom CAD/CAM software. The implementation should include a library of procedures, a series of specialized data structures and other CAD/CAM programming aids which can be accessed by the programmer developing a CAD/CAM system or a design and manufacturing applications program. It may be viewed as a high-level language for design and manufacturing applications programmers. CADMADE consists of software which minimizes the need for rewriting basic algorithms in interactive computer graphics, user interfaces, data management, geometric modeling, design analysis, manufacturing and artificial intelligence.

CADMADE is designed to manage the organization and display of data in a centralized collection of specialized data structures. This implies that the data are available to all workstations at the same time. CADMADE will relieve the applications programmer of tasks not previously considered or fully treated by graphics standards. PHIGS
manages the storage and display of and the interaction with 3-D graphics data by creating and managing a hierarchical data structure. In addition to graphics data, CADMADE should create and manage data pertaining to CAD/CAM applications.

Figure 3 shows an overview of this applications programming interface. CADMADE consists of five programming environments with corresponding data structures and two database managers, all of which can be accessed by the applications program. The environments are:

1. The User Interface Environment (UIE)--User Interface data structure
2. PHIGS +--PHIGS + data structure
3. The Design and Modeling Environment (DME)--CAD data structure and Product Information data structure
4. The Virtual Manufacturing Environment (VME)--CAM data structure and Product Information data structure
5. The Expert Consultation Environment (ECE)

The database managers are:

1. Universal Database Manager (UDM)--Universal Database
2. Knowledge Base Manager (KBM)--Knowledge Base

The design, manufacturing, user interface and graphics data can be stored in a series of pre-defined, interlinked centralized data structures. These data structures can be accessed by the applications program via the programming environments.
Figure 3. Overview of CADMADE
The User Interface Environment

The User Interface environment provides the applications programmer with procedures to implement an interactive graphical interface for the CAD/CAM applications program. The UIE should contain support for displaying menus and Icons, messages, prompts, etc. It should allow the applications programmer to easily define and modify some of the most commonly used views: front view, top view, etc. This viewing system should be simpler to use than the viewing procedures supported currently by PHIGS. The data generated by the applications program through UIE should be stored in the User Interface data structure. This data structure should be editable. The UIE and the User Interface data structures are discussed in greater detail in the next chapter.

The Design and Modeling Environment

The Design and Modeling Environment (DME) facilitates the development of geometric modeling, design and analysis software. The DME should include procedures for generating both surface models and solid models. These models can be analyzed using pre-existing (implementation independent) analysis software or application program defined analysis software. The design and modeling data generated through the DME should be organized in a hierarchical data structure, the CAD data structure. The DME should also allow the applications program to generate product information (e.g., bill of materials, surface finish, etc.) which can be stored in the Product Information data structure. The DME and the CAD data structure are discussed in detail in a later section.
The Virtual Manufacturing Environment

The Virtual Manufacturing Environment (VME) is intended to assist the applications programmer in the creation of computer-aided manufacturing software. For example, the VME should allow the programmer to access the CAD data structure and the Product Information data structure and program machining functions to machine a part. The VME should also allow the applications program to access "virtual tools" and set up a "virtual shop" environment.

The Expert Consultation Environment

The Expert Consultation Environment (ECE) should allow the applications programmer to create, modify and utilize an expert system, the Expert Technician (ET), for the applications program and ultimately, the applications program user. The ET should have access to the CADMADE data structures and the Knowledge Base which is pre-defined during the creation of the CAD/CAM system and/or modified by the applications program user. The ET can be used by the applications program in a number of different operating modes which closely approximate the different ways in which a design/manufacturing engineer consults with an expert.

The Databases

The Universal database is a relational database which can be used by the applications program to store and retrieve any type of data. The applications programs can use the Universal database through the Universal database manager. The Knowledge Base contains information which is used by the Expert Technician (ET). The applications program can store information in and retrieve information from the Knowledge Base.
with the help of the Knowledge Base Manager. The Expert Technician should also have the ability to modify the Knowledge Base.
6. The User Interface Environment

The User Interface Environment (UIE) of CADMADE contains high-level procedures which assist the applications programmer in quickly setting up a customized user interface for a CAD/CAM applications program. PHIGS includes most of the 3-D graphics primitives and graphical input devices necessary for the development of an interactive graphical user interface. But engineers who develop CAD/CAM systems based on the PHIGS standard spend a great deal of time in implementing some of the common features of a user interface such as menus, Icons, message areas, etc. For example, thousands of lines of PHIGS-based support software may have to be written for each new application program to implement a system of hierarchical menus.

The UIE of the proposed CADMADE should include interfaces which relieve the programmer of such time consuming and intricate programming. Figure 4 shows an expansion of the UIE from Fig. 3. The UIE mainly consists of:

1. The Menu and Icon Interface
2. The User Communication Interface
Figure 4. Overview of the User Interface Environment
6.1 The Menu and Icon Interface

The Menu and Icon interface should allow the applications programmer to program sets of hierarchical menu structures as part of the user interface for the applications program. The UIE should manage the organization and display of menus by storing the menu data in a centralized data structure, the User Interface data structure.

6.1.1 Hierarchical Menu Data Organization

A menu in the hierarchical menu system is defined by a sequence of elements grouped into entities called “menu structures”. These elements include output primitives, attributes, etc. Since a menu system will be defined by one or more menu structures, these menu structures relate to each other in a hierarchical network. Menu structures can reference (execute) other menu structures. Recursive menu structure invocations should not be supported.

A hierarchical network of menu structures is called a “menu tree”. Each menu structure in a menu tree is referred to as a “node”. There is a special designated node called the “root menu”, from which all other nodes originate. A menu structure is a “parent” if it references other menu structures. It is a “child” if it is referenced by others. The “ancestors” of a menu structure are the menu structures from the root node to the.
current node. Figure 5 shows a fairly simple menu structure network (menu tree). In this tree, the root menu invokes submenus 1, 2, 3 and 4.

Submenu 2 invokes submenus 2.1 and 4. Submenu 4 invokes submenus 4.1 and 4.2. It should be noted that any node in the tree can invoke any other node as long as the child node being invoked is not an ancestor of the node invoking it. Any child node can have more than one parent. In the example presented above, submenu 4 has two parents in the tree, submenu 2 and the root menu.

6.1.2 Menu Structure Elements

Each menu structure consists of a series of elements, the smallest components of a structure. The main elements which can be included in a menu structure are:

1. Menu Primitives
2. Menu Attributes
3. Menu Location
4. Menu Invocations

Menu Primitives

A menu primitive is an element that generates the actual graphical output. Without menu primitives there would be no graphical data to display for the Menu/Icon system. There are four types of menu output primitives defined in the UIE:

1. Menu Title Text
Figure 5. Example of a Menu Tree
Each menu output primitive is associated with a unique menu identifier. When a menu primitive is selected by the applications program user, CADMADE should return the identifier of the primitive selected to the applications program.

The Menu Title Text and the Menu Title Icon will allow the applications program to define a title for a menu screen. This title is usually a graphical entity which is descriptive of the function of the menus on the screen. The most common method of defining a menu title is by using a character string. The UIE should allow the application program to define a character string as a menu title by using the Menu Title Text Primitive. The Menu Title Icon should allow any graphical entity or group of graphical entities to be displayed as the title Icon for the menu. In order to use the Menu Title Icon primitive, the applications program has to define the graphics image in a PHIGS display structure. This PHIGS structure should then be used by the UIE to display the graphics image as a menu title Icon.

The Menu Item Text and Menu Item Icon are graphical entities which can be selected by the applications program user to indicate his/her intention (or choice) to the applications program. The Menu Item Text should allow the display of a character string as a menu item. Other graphics entities can be displayed as menu item Icons using the Menu Item Icon primitives. The Menu Item Icon should use PHIGS display structures in a manner similar to the one described above for the Menu Title Icon primitive.
Every menu primitive can have a polygonal area as a background. This polygonal area is similar to the PHIGS Fill Area primitive and should be defined when the menu primitive is defined. This feature will allow the menu primitive to have rectangular, circular or other shapes in the background to improve visibility and distinguishability of the menu primitives from the other graphics images which are displayed on the screen.

**Menu Attributes**

Associated with each menu primitive should be a set of attributes that define its appearance. The attributes which can be used to affect the appearance of the menu primitives are listed below.

**Text Attributes:**

Text attributes affect the appearance of a character string on the display screen. These attributes apply to the Menu Title Text and the Menu Item Text.

*Text Font* - Selects a workstation character font.

*Text Precision* - Controls how accurately the characters represent the specified text attributes.

*Character Spacing* - Controls how far apart characters are spaced. The value is specified as a fraction of the font's nominal height.

*Character Expansion Factor* - Controls character width. It is the multiplicative factor applied to the nominal character width.

*Character Height* - Specifies the height of upper case characters. The value is specified in normalized device coordinates units.

*Text Color Index* - Selects a text color. The value is an index into the workstation color table.
**Character Up Vector** - Specifies the up direction of characters. This controls the angle of the text strings.

**Text Path** - Selects the writing direction of text strings: up, down, left or right.

**Text Alignment** - Positions the string at a starting location. It has two components, one for the horizontal alignment and one for the vertical alignment.

The text attributes described above are similar to the text attributes defined in PHIGS. At any given time during menu structure traversal, there should be two sets of menu text attributes which are active: one for the menu title text and one for the menu item text.

**Background Attributes:**

Each menu primitive can have a fill area as a background. The attributes of the background are listed below:

**Interior Style** - Specifies how the polygon is filled. The styles that may be selected are - solid, hatch, pattern, hollow and empty.

**Interior Style Index** - Selects an interior pattern or hatch style from a workstation pattern or hatch table.

**Interior Color Index** - Selects an interior color from the workstation color table.

**Outline Flag** - Specifies whether or not the background polygon's edges are to be displayed.

**Outline Type** - Selects a line pattern for the background outline (e.g., solid, dashed, dotted, etc.).

**Outline Width Scale Factor** - Determines the line thickness for the background outline. It is specified as a multiple of the workstation's nominal line width.

The background attributes defined above are similar to the fill area attributes defined in PHIGS. At any given time during menu structure traversal, there should be two sets
of menu background attributes which are active: one for the menu title and one for the menu item.

Other Attributes:

Menu Title Icon Transformation Matrix
This attribute specifies a transformation matrix to be applied to the PHIGS structure which is to be used as a Menu Title Icon. This matrix will allow the PHIGS image to be rotated, scaled and translated before being displayed as an Icon. Using this feature will allow applications programs to also have animation sequences for Icons.

Menu Item Icon Transformation Matrix
This attribute has the same effect as the Menu Title Icon Transformation Matrix except that this matrix defines the transformation for the Menu Item Icon and not the Menu Title Icon.

Menu Prompt
Most menu-driven applications programs prompt the user by sending a character string to the display screen. Usually, this prompt is unvarying for a particular menu screen. Thus, a menu prompt can be included as an attribute for a menu structure. Whenever a menu screen is displayed, the appropriate menu prompt should be automatically displayed using dialog areas (discussed later in this chapter).

Menu Prompt Dialog Identifier
This attribute attaches a dialog area to the menu structure for the display of prompts.
Menu Locations

The menu locations specify the location, orientation and extent of the menu items defined in a menu structure. The menu locations can be defined in two ways:

1. Individual
2. Bundled

In the "individual" mode, the applications program has to define the location of each menu item individually. The menu item primitive invocation also has to define a background polygon for each menu item.

In the "bundled" mode, the applications program needs to define only one rectangular area for the menu items. A "bundled" locations path can also be defined by the applications programmer (up, down, left or right). CADMADE should automatically space the menu items in the rectangular area following the predefined bundle location path.

The menu locations should be defined by the applications program using Normalized Device Coordinates (NDC) system.

Menu Invocations

Menu structure invocations are the elements used for building hierarchical menu trees. For example, one menu structure "executes" another which "executes" another, and so on, once for each level of hierarchy. The "Execute Menu Structure" element causes traversal of a specified menu structure. CADMADE should automatically save the current system state before transferring control, and restore the current system state.
when control returns to the parent structure. This transfer of control is explained in greater detail in the next section, "Menu Structure Data Display".

**Menu Structure Labels**

A label is a menu structure element which has no effect on the appearance or the hierarchy of the menus. It serves as a marker to facilitate structure editing.

### 6.1.3 Menu Structure Data Display

**Traversal**

In order to display a menu structure network, the "traverser" starts from the root node and processes each element in sequence. Each menu primitive is rendered with the current attributes. For example, the menu tree shown in Fig. 6 will be traversed in the following order for setting the menu attributes:

1-2-1-3-5-3-6-3-1-4-1

Thus, the traversal of a menu structure is temporarily suspended when an "execute menu" element is encountered. The traversal of the parent is resumed when the traversal of the child is complete. The actual time of traversal, however, should be controlled by the applications program as described in Section 6.1.7. In order to display a menu from a menu tree, the root menu has to be "posted" to the workstation. The menu display entities have a higher display priority than any other graphics entity on the display surface. The "Unpost Menu Root" function removes a menu tree from the workstation.
Figure 6. Example of a Menu Tree Traversal
The workstation state list should include a "current menu list" which consists of a current menu for each menu tree posted to the workstation. Only the current menus should be available to the applications program user for menu selection.

**Traversal Time Binding**

The menu primitive attributes and locations should be assigned to primitives during traversal of the structure and not at the data definition time. In the example of a menu structure shown in Fig. 7, the menu item "One" is displayed at (0.3, 0.3) in red with a hollow background and the menu item "Two" is displayed at (0.6, 0.6) in green with a hatched background (Fig. 8).

**Inheritance**

Attributes defined within a menu structure should subsequently affect the menu primitives in children structures. Thus, a child menu structure "inherits" attributes of its parent. If a child menu structure is instanced more than once, each instance should inherit the attributes of its immediate parent. The root menu structure should inherit its attributes from the CADMADE description table which should contain the system default values. The attributes of a child structure are different from those of the parent only if the attributes are defined within the child itself. Thus, the same menu structure can be invoked from several parent menus with different attributes being associated each time (when the attributes are inherited) or with the same attributes each time (when the attributes are defined within the child menu structure).

The menu location elements of menu structure should also be inherited from parent menus unless they are redefined in the children menus. Thus, the applications programmer has the flexibility to easily bring up submenus in the same menu area or have submenus come up at different locations on the screen.
Figure 7. Menu Structure for Example of Traversal Time Binding
Figure 8. Example Result of Traversal Time Binding
The useful consequences of hierarchy are:

1. Compactness of data storage.
2. Guaranteed consistency in case of multiple invocations of the same menu structure.
3. Ability to control the appearance of the menus at traversal time.

6.1.4 Menu Structure Editing

Menu data in the Menu data structure can be modified by menu structure editing. An element pointer should point to a location in a menu structure, and modifications made to the menu structure should be dependent on the location of the element pointer. The "Open Menu Structure" command should open a menu structure in the menu data structure for editing. If the menu structure does not exist, a new menu structure should be created. The "Close Menu Structure" should end the editing of the menu structure. The editing capabilities of the menu structure editor should include:

1. Move element pointer
2. Insert element
3. Delete element or range of elements
4. Copy contents of a structure
5. Inquire elements
6. Locate element pointer at label
7. Empty menu structure
8. Delete menu structure
6.1.5 Menu Types

Two types of menus have been defined for CADMADE:

1. Global menus
2. Local menus

A global menu is a menu tree which should be always available to the applications program user unless it is specifically unposted by the applications program. A local menu should be available to the applications program user only if it is the "current" menu in the posted menu tree. A global menu can invoke a local menu but a local menu cannot invoke a global menu. Only one global menu tree can be defined in CADMADE at any given time. A global menu tree can be defined by setting up a regular menu tree and assigning the root of the menu tree to the global menu root.

6.1.6 The Menu Pick Logical Input Device

An input device is a device which an applications program user uses to generate graphical input to the applications program. A logical input device is an abstraction of physical input devices with similar characteristics and input behaviors.

Physical input devices include keyboard, thumbwheels, light pens, tablets, mouse, dials, etc. The applications program has to map each logical input device to a physical input device which the program user can manipulate. PHIGS supports six classes of logical input devices:
For the UIE of CADMADE a seventh class of logical input device has been defined, the "Menu Pick" device. In order to allow the application program to pick the displayed menu primitive, each menu primitive should have a menu pick identifier associated with it which is returned when the primitive is selected. The menu pick should return a menu pick path which identifies the menu primitive picked. This path should also return the menu structure in which the primitive is defined and the menu structure path from the root menu to the picked menu. All physical devices which are capable of PHIGS pick input should also be capable of menu pick. In certain cases, the applications program user should be able to key in a menu selection using a string device. This feature is further discussed in section 6.2.2.

PHIGS supports three operating modes for the logical input devices:

1. Request
2. Sample
3. Event

The menu pick input should be capable of operating in all the three above-mentioned modes for PHIGS compatibility.
6.1.7 Applications Controlled Menu Structure Traversal

The menu structure traversal in CADMADE should be partially controlled by the applications program. When a menu structure is invoked within another menu structure the following actions should occur:

1. The parent menu structure is made unpickable.
2. The parent menu structure may or may not be set invisible (depending on the applications program).
3. The attribute values of the parent are passed on to the child.
4. The child menu is made visible and pickable.
5. Control is passed back to the applications program.

The invocation and display of the submenu structures should be controlled by the applications program. When a root menu is posted, it should be added to the current menu list on the workstation and all the menu output primitives in the menu structure should be displayed. Control should then be returned to the applications program. After a "menu pick", the applications program should be able to get the hierarchical menu pick path from CADMADE. The applications program can then invoke a procedure to continue the menu structure traversal and automatically display the next menu in the hierarchy depending on the menu item selected. The applications program has to pass the menu pick path of the menu pick back to CADMADE. The new menu should be added to the list of current menus on the workstation state list and the previous menu removed from this list. Thus, the traversal time of a menu structure will be controlled by the applications program. This feature will allow the applications program to perform other necessary operations before a new menu is displayed. If the
lowest element in the menu pick path indicates a return, the current menu should be removed from the workstation current menu list and the parent of the current menu added to the list of current menus on the workstation. A predefined menu pick identifier should display the global menu and add the global menu to the workstation current menu list (if it is not already there). This pick identifier can be attached to any menu item in a menu tree.

Figure 9 shows an example of a menu tree with a sample sequence of menu picks. In this example, the applications program first posts menu structure 1. The traverser displays menu structure 1 and returns control to the applications program. At this time, the applications program can perform computations called for by the selected menu item. For example, if the applications program user selects a menu “DISPLAY”, the applications program has to do the necessary computations and display a model on the graphics screen before allowing the user to select another menu item. After the application program user selects menu 2, the applications program calls “next menu” which automatically traverses menu 2 and displays it. Similarly, the menu structure 5 is displayed when the user selects menu 5 and the application program invokes “next menu”. When the user selects a return from menu 5, the previous menu (2 in this case) is displayed (when the application program invokes the next menu function). This sequence of events is shown in Fig. 10. Similarly, another return selection displays the root menu structure once again.
Figure 9. Example of a Sequence of Menu Picks

MENU PICK SEQUENCE:
2 - 5 - RETURN - RETURN

CONTROL RETURNED TO THE APPLICATIONS PROGRAM
Figure 10. Applications Program Controlled Menu Traversal

6. The User Interface Environment
6.2 The User Communication Interface

The User Communication Interface of the UIE should allow the applications program to communicate with the applications program user through rectangular windows. These windows are called "Dialog Areas". These dialog areas are meant only for text (character) input/output. The dialog areas should be simple to define and use. They should not be as complicated to use as X windows or NeWS. The user communication interface is not a full-fledged window manager, but it should allow easy creation of rectangular windows for user input/output. Using the user communication interface, the applications program can set up multiple dialog areas on the display surface.

6.2.1 Dialog Areas for Applications Program Input/Output

PHIGS implementations support various logical input devices which can be mapped to the alphanumeric keyboard as the physical device. For example, the "valuator" can get numeric input from the user through the keyboard and the "string" can get text input from the user through the keyboard. In order to use any of these input devices, the applications programmer has to go through a series of procedures to initialize the logical input devices and associate them with the physical devices.

The user communication interface will greatly simplify the use of the keyboard for string and valuator input. Any keyboard input to a dialog area should be considered as string input. A utility procedure supported by the UIE should allow the applications program to extract number input from the string input to the dialog area. Thus the same physical
device (the keyboard) can be used for both string and valuator inputs without having to
go through the process of initializing the devices and then getting the input from the
devices.

The dialog areas should also allow the applications program to display messages. In
PHIGS, in order to display a message text, the text string has to be inserted in a PHIGS
structure along with its attributes and then posted to a workstation and a view. In order
to change the message, the PHIGS structure has to be edited and the image regenerated.
Having a message area with multiple lines of message and scrolling messages is much
more complex to program. PHIGS does support a "message" function but the
applications program has no control over the location, attributes, etc. of the output text
and cannot have scrolling messages or multiple lines of messages.

The user communication interface will greatly simplify the process of sending messages
to the screen. The dialog areas can display applications program defined message
strings. The attributes of these text strings are controlled by the attributes of the dialog
area (discussed below). Thus the applications program has to define the dialog area
once and then can display messages at any time by invoking only one message function.

6.2.2 Dialog Areas for Menu/Icon Interface Input/Output

The dialog areas can also be used for input/output related to the Menu/Icon interface.
Inside a menu structure, a prompt string and a dialog area identifier can be inserted as
attributes of the menu structure. The UIE should automatically send the prompt string
as a message to the specified dialog area whenever the menu is added to the list of
current menus on the workstation state list. This feature blends well with the feature

6. The User Interface Environment
of automatically bringing up selected submenus, since the applications programmer does not have to keep track of the appropriate prompts for the various menu levels.

The dialog areas can also be used for menu input from the applications program user. Each dialog area can be associated with one root menu structure. A keyboard string input, with a predefined character prefix, to the dialog area should be associated with menu input for the menu tree with which the dialog area is associated. The character prefix which indicates a Menu/Icon input can be defined by the applications program. When CADMADE receives a string input with the special prefix, it should search the current menu item text strings to locate the most probable menu item that can be selected by the user and generate a hierarchical menu pick event which can then be treated as a regular hierarchical menu pick. In case of any ambiguity in the string menu input, the first menu item text that matches the input string should be taken as the menu input item.

An example of a menu tree which can cause ambiguity in string menu selection is presented in Fig. 11. In this example, when "MAIN" is the current menu, a menu string input of "SUB" could be ambiguous, but CADMADE returns a menu pick identifier of "2" corresponding to the first menu item text which has "SUB" as a sub-string. A menu input of "SUB 3", however, is not ambiguous and the menu pick identifier returned is "4". This string menu selection feature can allow the applications programmer to go to any node in the tree just by keying in a set of menu strings. This is similar to the type-ahead feature found in many menu systems.

An analogy can be drawn between the menu string input and the "change directory" command in UNIX and other operating systems. In UNIX, the sub directories are arranged in a hierarchical tree. For example, UNIX command "cd /usr/menu" would
Figure 11. Example of a Menu Tree for String Menu Selection
take the user to the directory "menu" shown in Fig. 12. The first "/" indicates that the change directory command starts from the root directory. A similar feature should allow the applications program user to change menus and traverse the menu tree by keying in "change menu" commands. For example, if "cm " is defined as the prefix for menu input, the string input of "cm SUB 1" to the appropriate dialog area should generate a menu pick identifier of 2 and a pick hierarchy of 1-2 (when "MAIN" is the current menu, Fig. 11). A string input of "cm /SUB 1/SUB 12/SUB 121" should generate a menu pick event with a menu pick identifier of "7" and a menu pick hierarchy of 1-2-6-7.

6.2.3 Dialog Areas and Host System Communication

The U1E dialog areas can also be used to allow the applications program user to communicate with the host computer system during applications program execution. This concurrent processing of system commands during applications program execution can be made possible with the use of multitasking operating systems like UNIX. Although a number of window managers (X-windows, NeWS, etc.) allow the user to define multiple windows for host system communication, the CADMADE dialog area should allow the system commands to go through the applications program. Using the user communication interface, the applications program can issue host system commands directly or allow the applications program user to issue system commands.

In a manner similar to the one described above for string input for menu selection, the applications program can define a character prefix which signifies a host system command. A dialog area can be attached to logical input/output units by defining a logical unit number for the dialog area. When CADMADE receives a string input with
Figure 12. Example of UNIX File Hierarchy
the special prefix, it should send the character string (after stripping the prefix) to the host computer as a system input from the logical unit to which the dialog area is attached.

An example of the use of such facilities is allowing the applications program user to view a list of files and modify data files during applications program execution.

6.2.4 Dialog Area Attributes

Each dialog area should have its own set of attributes which control the appearance of the dialog area. The attributes of the dialog areas should be stored in the workstation state list. Thus, any changes made to the attributes of the dialog areas will be instantly applied to the dialog areas being displayed. The attributes of dialog areas are:

Number of Lines: Specifies the number of lines of text (including the string input line) to be displayed in the dialog area.

Number of Characters per Line: Specifies the maximum number of characters to be displayed in each line in the dialog area.

Dialog Echo Mode: Specifies whether inputs to the dialog area should be echoed back to the dialog area as a message.

Dialog Scroll Mode: Specifies whether the dialog area lines are to be in scroll mode or overwrite mode.

Dialog Area Location: Specifies the corner points of the rectangular dialog area in Normalized Device Coordinates (NDC).
The rest of the dialog area attributes listed below perform functions similar to those of the attributes described for the menu text and the menu background area.

- **Dialog Area Character Height**
- **Dialog Area Character Spacing**
- **Dialog Area Character Expansion Factor**
- **Dialog Area Text Color Index**
- **Dialog Area Text Font**
- **Dialog Area Text Precision**
- **Dialog Area Background Flag**
- **Dialog Area Outline Flag**
- **Dialog Area Background Color Index**
- **Dialog Area Outline Color Index**
- **Dialog Area Background Interior Style**
- **Dialog Area Background Interior Style Index**
- **Dialog Area Outline Type**
- **Dialog Area Outline Width Factor**

### 6.2.5 Other Features of Dialog Areas

Some attributes of dialog areas should not be stored in the workstation state list. These attributes affect the dialog areas only when the workstation is updated. The following is a list of such attributes:

**Dialog Area Buffer Size:** The buffer size of a dialog area specifies the number of lines of input/output to be retained in the buffer.
**Dialog Area Visibility:** A dialog area can be rendered visible/invisible by setting the dialog area visibility.

**Dialog Area Priority:** Each dialog area should have a priority similar to the view priority supported by PHIGS. The priorities of the dialog areas should decide how overlapping dialog areas appear on the screen.

**Dialog Area Scroll:** A function in the UIE should allow the applications program to scroll up or down in the dialog area within limits of the dialog buffer. In order to be able to use this function, the dialog area must be in scroll mode.
6.3 The CAD View Manager

PHIGS supports viewing transformations which map world coordinates into viewing coordinates. The viewing transformations define the orientation, location and display of a PHIGS graphics structure on the display surface. Viewing transformations are defined by view matrices stored in the workstation state list view table. These viewing facilities give the applications programmer enough flexibility to define and transform displays of structures in any desired manner. However, certain operations involving these viewing transformations are important to CAD/CAM applications and involve large amounts of repetitive code which are usually duplicated from applications program to applications program. The CAD View Manager is designed to make these sets of operations available to the applications programmer as higher level procedures, thus eliminating the time and effort spent in rewriting large portions of computer code.

6.3.1 Predefined Views

Most CAD/CAM applications/systems set up principal orthographic views and isometric views for the display of CAD models. These views are defined using PHIGS viewing facilities or other device-dependent viewing functions. In order to define these commonly used views, the applications program has to calculate the transformation matrices from the transformation values (translations, rotations, etc.) and apply these matrices to the views.
The UIE should allow the applications program to invoke higher level procedures and display CAD objects in such predefined views in a simple and easy manner. The applications program need only identify the view index and a view transformation index to display a predefined view. The seven principal views which should be supported by the UIE are shown in Fig. 13.

Other than the default predefined views, the UIE should also allow the applications program to store viewing transformations and create "applications-defined predefined views". The applications program can store the current viewing transformation of any view by specifying a "predefined view transformation index". This predefined view transformation index must be different from the indices used by the seven default predefined views. This applications defined viewing transformation can then be applied to any view to easily display models in commonly used views (other than the principal orthographic views).

6.3.2 Nonrectangular Windows

PHIGS view facilities only allow the creation of rectangular windows and viewports. It is not uncommon for designers to use nonrectangular windows for special cases. One example of the use of a nonrectangular window is demonstrated in Fig. 14. In this case, a circular window is used to display a magnified view of a part of model to show the details of the part. Device-independent viewing standards should allow the applications program to define such nonrectangular views.
Figure 13. Predefined Principal Orthographic Views
Figure 14. Example of Use of Nonrectangular Window
The CAD view manager should allow the creation of nonrectangular windows by allowing the applications program to define any polygonal shape for the window and viewport. The only limitation should be that the number of sides of the polygons for the window and viewport must be the same in order to avoid distortion of the image during mapping from the world coordinate system to the viewing coordinate system. The polygon defining the window should have available to it all the attributes available to PHIGS fill area primitives. These attributes should define the background and border characteristics of the view.

6.3.3 User Interface for Viewing Transformations

A common feature found in many CAD/CAM applications/systems is the use of graphics input devices to allow the applications program user to apply viewing transformations to different views. Figure 15 shows a typical sequence of events followed by an applications program in order to allow the user to apply viewing transformations using valuator input devices and PHIGS.

The process of getting the valuator input, deciding on the transformation matrix and applying the matrix to a view are very commonly used and should be available to the applications programmer as one higher level procedure. This higher level function of the UIE should allow the applications program to attach logical input devices to PHIGS views for the automatic application of viewing transformation to views. CADMADE can accomplish this by maintaining two viewing transformation maps in the workstation view table. The seven commonly used viewing transformations which should be supported are the X, Y, Z translations and the X, Y, Z rotations and uniform scaling.

6. The User Interface Environment
Figure 15. Applications Program for View Manipulation Using PHIGS
The first map (the logical-input-device-to-transformation map) maps the logical input devices to the above mentioned transformations. The second map (transformation-to-view map) maps the various transformations to view indices in the workstation's view table. Figure 16 shows an example of the CADMADE view transformation maps. Using this facility, the applications programmer need only initialize the logical input devices and define the above mentioned maps. When an input is generated by one of the input devices defined for viewing transformations, CADMADE should automatically go through the process of getting the input value and applying the transformation to the appropriate views. Figure 17 shows a typical sequence of processes followed by an applications program in order to use the automatic viewing transformation feature of CADMADE.

The automatic application of view transformations from input devices can make good use of the intelligent graphics devices which have been available in the market in recent years. For example, in CADAM-ISD, the dials attached to IBM 5080 graphics devices are set up for rotations, translations and scaling. However, these transformations are performed locally by an intelligent processor on the graphics device. Thus, these transformations do not use any processing time of the host computer system. Such local graphics device intelligence cannot be used if the applications program has to get the input and calculate the transformations. The use of CADMADE view transformation maps would allow device drivers to utilize the local intelligence of the graphics device and hence improve the performance of the display device. Thus, the applications program would use device-independent and machine-independent functions to improve the performance of viewing transformation processes on intelligent graphics display devices.

6. The User Interface Environment
Figure 16. Example of CADMADE View Transformation Maps
Figure 17. Applications Program for View Manipulation Using CADMADE
7. The Design and Modeling Environment

Geometric modeling is an important part of computer-aided design and manufacturing. The Design and Modeling Environment (DME) should provide the applications programmer with high-level geometric modeling procedures and other aids which facilitate the development of geometric modeling, design and analysis software. The DME should eliminate the time spent by CAD engineers in developing specialized procedures to describe and store shapes of objects using geometric modeling procedures. The applications program should be able to easily generate geometric data and store the data in the CAD data structure. Figure 18 shows an expansion of the DME from Fig. 3.

The DME consists of:

1. Modeling Entities
2. Modeling Operators
3. Analysis and Query
4. Modeling Utility Procedures
Figure 18. Overview of the Design and Modeling Environment

7. The Design and Modeling Environment
7.1 Modeling Entities

A geometric model can be described for CAD applications in a number of different ways. The most commonly used method for geometric modeling is "wireframe modeling" which uses curves and surfaces to represent a geometric shape. However, there are certain limitations of the wireframe modeling technique when three-dimensional objects are to be represented and analysed. "Solid Modeling" is intended to overcome these limitations and to allow the creation of unambiguous and complete geometric representations of objects. "Feature-Based Modeling" is a very new technique in geometric modeling in which similar shapes are grouped into a generic "feature" to allow designers to easily create commonly used complex shapes. The DME of CADMADE should support the creation of all three types of modeling: wireframe, solid and feature.

7.1.1 Wireframe Modeling

A wireframe model is composed of lines, curves and surfaces which define the edges and faces of an object. Each curve or surface is constructed separately and independently. The types of curves discussed below should be included in the DME.
Curves

Parametric Polynomial Curves: These curves include several forms of piecewise parametric three-dimensional curves. These forms are commonly described by an ordered set of three-dimensional points, a polynomial order and a "curve basis" which specifies the relationship of the resulting curve to the defining data. Parametric polynomial curves include Ferguson curves, Bezier curves, uniform B-splines, rational curves, etc.

The DME should support the creation of parametric curves in a number of different ways. For example, in generating a parametric cubic curve, the applications program can use the four-interpolating-point form, the Bezier form, or the two end point and two tangent form, or the specification of control points and weights for a rational cubic curve.

Nonuniform B-Splines have recently become very popular because of their localized curve control characteristics. The DME should support the creation of both rational and nonrational nonuniform B-Splines.

Spline curves are very important in the aircraft and ship building industries. Spline curves should be supported by the DME of CADMADE. Spline curves behave structurally like a beam, with bending deflections forming them into smooth curves.

Surfaces

The surfaces patches which should be included in the DME are:
1. Coons patch
2. Tensor product patch
3. Bezier patch
4. B-spline patch
5. Rational patches

The Coons patch is constructed solely in terms of information given on its boundary and certain auxiliary scalar functions. The blending functions used in Coons patches are linear and uniform in nature.

The Tensor product patch is defined by four corner points, tangents at the corner points in two orthogonal directions (derivatives with respect to the two parameters u and w) and twist vectors at the four corners. It is only necessary to match these vector quantities at the corner points to get zero-order and first-order continuity across surface patches.

The Bezier patch is defined with the help of a characteristic polyhedron. The vertices of this polyhedron define the tangent vectors and twist vectors at the patch boundaries. These control points can be easily manipulated to change the shape of the surface patch.

The B-spline patch is also defined with the help of a characteristic polyhedron. The control points normally do not lie on the patch. Curvature continuity can be obtained with B-spline patches. The DME should support the creation and manipulation of uniform and nonuniform B-spline surface patches.

The DME should support rational forms of the Tensor product patch, the Bezier patch and the B-spline patch. In the rational form, the surface polynomials are defined as a
ratio of two polynomial functions. The control points for the patches have to be defined in a homogeneous coordinate system with the fourth coordinate being a weight factor associated with each control point. These weights allow greater control over the shape of the surface patch.

7.1.2 Solid Modeling

The DME should support three methods of representing solid models:

1. Constructive Solid Geometry (CSG)
2. Boundary Representation (B-Rep)
3. Sweep Representation

Constructive Solid Geometry

In the Constructive Solid Geometry (CSG) method, complex solids are defined as compositions of simpler solids. Boolean operators are used to execute the composition. CSG uses all the regularized boolean operators: union, intersection and difference.

Boundary Representation

In the Boundary Representation (B-Rep) method, solids are defined by the faces bounding the solid volume. This method allows the conversion of surface models to solid models.

Sweep Representation

Sweep representation of solids are generated by surfaces moving along some path.
7.1.3 Feature-Based Modeling

Features are denoted by descriptive words commonly used by design engineers to describe the geometry of a part (e.g., hole, boss, keyway). A set of standard features should form a part of the DME. Gandhi (1989) has compiled an exhaustive list of features used by designers. These features have been grouped into various categories based on form and topology. Parametric equations have been formulated for defining these features. CADMADE should include this extensive "dictionary" of features as modeling entities that can be used by the applications program. These features should be available as both wireframe and solid entities that can be sent to the CAD data structure.

The DME of CADMADE should also allow the creation of "super features". Super features can be compared with the "Generalized Drawing Primitive (GDP)" supported by graphics standards. The applications program should be able to define a "super feature" as a collection of features, set up the parameters for defining the super feature and then use the super feature as a regular feature. For example, a keyed shaft can be defined as a super feature made of two features, shaft and keyway. The dimensions and location of the keyway can be defined as functions of the dimensions of the shaft. Thus, defining the radius and length of the shaft would automatically size and locate the keyway. An implementation of the DME could contain a library of some commonly used super features; e.g., springs, fasteners, etc.
7.2 Modeling Operators

Modeling operators should allow certain operations to be performed on geometric entities in the CAD data structure in order to modify or reshape the entities.

**Boolean Operators**

Boolean operators allow the modification of solids in the CAD data structure. All the regular Boolean operations should be supported by the DME.

**Alignment Operators**

The alignment operator will allow the applications program to easily align various parts in the CAD data structure prior to assembly or other operations. The alignment operator requires a knowledge base of alignment parameters for the features in the DME library. For example, invoking a function to align the axis of a shaft with axis of a hole should align the shaft with the hole, as shown in Fig. 19. The alignment operator will greatly reduce the calculations that would normally need to be performed by the applications program.

**Attachment Operators**

The attachment operator will allow the applications program to attach various parts of a geometric model to each other. Two different types of attachment operators have been defined for CADMADE:

1. Assembly
2. Integral
Figure 19. Example of the Alignment Operator
The concept of the assembly operator is well developed and is supported by a number of commercial CAD/CAM systems. The integral attachment operator simulates physical attachment which is often encountered; e.g., welding, glueing, collection of primitives machined from the same piece of material, etc. In an assembly attachment, the two parts being assembled retain their individual identities and a change made to one part (e.g., location change) does not effect the other. In Boolean operations, the solids being joined together lose their individual identities and become one solid. The integral attachment operator should allow the attached parts to retain their individual identities while behaving as one part. The attachment operator should also support tolerancing specifications for the objects being attached.

**Reshaping Operators**

The reshaping operator should allow an object to be reshaped using deformation equations. Examples of reshaping operators are bending, stretching, twisting, etc. This operator is meant to allow design engineers to deform various parts to suit specific needs of the design as is often done in the workshop (e.g., coiling a wire to form a spring).

### 7.3 Analysis and Query

The Analysis and Query system of the DME should allow the applications program to analyze objects/parts for various properties of the model and extract information about the model from the CAD data structure. The various analysis and query capabilities of an applications program using DME should be:
1. **Property Analysis**: The applications program should be able to analyze a solid object in the CAD data structure for mass properties and material properties.

2. **Interference Analysis**: Interference analysis is often very important in CAD systems for final design of assemblies. The applications program should be able to determine the interferences in an assembly in the CAD data structure. The tolerancing in the assembly should also be obtainable from the CAD data structure.

3. **Queries on Surfaces and Curves**: The applications program should be able to make queries on surfaces and curves and obtain information pertaining to the entities (e.g., curvature, tangent, entity type, etc.).

4. **Queries on Solids**: Applications programs should be able to perform queries on the solids in the CAD data structure. Examples of such queries are: intersections between solids, union of solids, etc.

5. **Queries on Features**: A number of queries should be available for the features supported by the feature-based modeler. The applications program should be able to query a feature to determine the parameters that are required to define the feature, names (character strings) which describe the parameters and the system defaults for the parameter ratios. For example, a query on "shaft" would return the two parameters "radius" and "length" and the default value for the radius to length ratio. This information could be used by the applications programmer in setting up a template for the user interface for designing with features. The applications programmer does not need to have knowledge of all the features she/he uses from the DME library.

Other inquiries on features would enable the applications programmer to fabricate the feature, other features with a similar topology and other features with similar functions.
7.4 Logical Modeling Input/Output Devices

The concept of logical input devices is well developed and is supported by graphics standards (GKS and PHIGS). A similar concept has been developed for the DME to facilitate user interactions with geometric models created by CAD/CAM applications programs during a modeling/design cycle. Logical modeling input/output devices will provide the applications program user with a very versatile and realistic environment for model building.

The Six-Degree-of-Freedom Input Device

The location and orientation of any three-dimensional object requires the specification of six values in spatial coordinates. A logical six-degree-of-freedom input device in the DME will allow the applications program to use the device on various workstations which support different physical input devices without having to modify the applications program. For example, one workstation might use six dials for the six-degree-of-freedom input device whereas another might use a six-degree-of-freedom mouse similar to the one described by Ware and Jessome (1988). This logical input device could be used for orienting and locating objects in the three-dimensional model.

The Logical Gripper

The logical gripper is a logical input device which can be used to grip objects and move them in three-dimensional space. For example, a logical gripper could be used in conjunction with a six-degree-of-freedom input device to move in three-dimensional...
space, grip an object and manipulate it. This kind of input would provide a much better user interface than would the use of valuators (or dials) to orient an object.

**The Logical Feel Device**

CAD/CAM systems have always used visual devices as the main output devices. However, when a designer/engineer builds a prototype in real life, feel as well as sight contribute to the design evaluation process. The concept of "Data-Glove" has demonstrated the feasibility of devices which allow human beings to feel objects which exist only in computer memory. A logical feel device has been included for this purpose in the DME. The logical feel device should greatly enhance the design feedback to the applications program user, especially if it is coupled with the grip and six-degree-of-freedom input devices to allow the user to grip and feel the object while manipulating it.

### 7.5 Other Features of the DME

**Modeling Utility Procedures**

The development of geometric modeling applications usually requires the support of a number of commonly used procedures which should be included as "utilities" in the DME. For example, most CAD/CAM applications programmers require the use of a number of vector and matrix functions (e.g., dot product, cross product, etc.), some
interpolation procedures and other mathematical operations. The utilities in the DME should include:

1. Vector Manipulation Functions
2. Matrix Manipulation Functions
3. Interpolation Procedures
4. Curve and Surface Utilities
5. Coordinate Transformations and other Transformation Utilities

The Equation Solver - Parametric Geometry

Equation solvers and algebraic processors are sometimes of great assistance to CAD applications programmers. This facility in the DME should allow applications programs to attach equations to models for the automatic resizing of the various interrelated dimensions of the parts when some of the dimensions are changed. This should greatly enhance the creation of CAD systems which incorporate parametric geometry techniques.

Design Evaluation

An effective CAD/CAM system offers design evaluation features to its end users. The DME should support the creation of design evaluation software by maintaining a library of commonly used design equations and rules of thumb. Some of the basic design/strength analysis equations (e.g., beam bending, spring design, shaft twisting, etc.) should be included in the DME. The DME should provide interfaces for finite element analysis systems, optimization procedures and other evaluation systems.
7.6 The CAD Data Structure

Most CAD applications use a hierarchical data structure to store and organize models. A data structure that would facilitate the development of mechanical design applications has been designed for CADMADE. The modeling entities are stored in a hierarchical manner in the data structure. The model of an assembly can be broken down into a number of lower level entities which form the different levels of the hierarchy. Figure 20 shows the hierarchical arrangement of entities in the data structure. The entities are:

1. Primitives (curves, surfaces and solids)
2. Features and Super Features
3. Objects
4. Parts
5. Assembly
6. Instance

A feature is modeled with the help of surface and solid primitives which form the lowest level of the hierarchy. The concept of super features has been discussed earlier in this chapter. A number of features can be grouped to form an object. Features cannot be "assembled" to form an object; they have to be "integrally attached" or Boolean operations must be performed on the features being grouped to form an object. Different objects can be grouped (assembled or integrally attached) to form parts which can be assembled (not integrally attached) to form an assembly.
Figure 20. The CAD Data Structure Hierarchy
The concept of "instancing" can be used in the CAD data structure. An object, part or assembly can be used as an instance and used in other parts or assemblies. Recursive instancing should not be supported.

The locations of assembled objects and parts can be stored in a parametric form in the "parent". The locations can be specified as functions of various dimensions to allow automatic relocation during redesign. The CAD data structure should also allow the applications program to define attributes for the model.

Every object, part and assembly should have a name and a number associated with it. These identifiers can be defined by the applications program during definition of the model and can be used at a later stage for analysis, inquiry or modification procedures. Constraints can be attached to each entity in the data structure. An example of a constraint is the maximum number of times an object can be instanced. This limits the end user to work within certain limitations imposed on her/him by the applications program.

Entities can be grouped by their names and/or instance numbers and constraints and other attributes can be applied to these groups.

7.6.1 Data Structure Operations

A number of operations can be performed on the CAD data structure by the applications program.

1. DEFINE - The "define" function should allow the program to begin defining an object, part or assembly.
2. ADD TO - This function should allow the applications program to add entities to an existing object, part or assembly.

3. INSTANCE - The "instance" function should allow the program to use instances of an object, part or assembly and add it to another part or assembly.

4. DELETE - This function should allow the program to delete an object, part or assembly.

5. DELETE ALL - This function should allow the program to delete all instances of an object, part or assembly.

6. COPY - This function should allow the program to copy an object, part or assembly with a new name and number.

7. MODIFY - This function should allow selective deletion of features in an object, objects in a part or parts in an assembly.

8. UNDO - This function should allow the last entry in the data structure to be deleted.

7.6.2 Difference Between Primitives in PHIGS+ and the DME

PHIGS+ allows the creation of a number of curves and surfaces. Parametric polynomial and nonuniform B-spline curves and surfaces can be inserted into PHIGS+ data structures as primitives. However, the curves and surfaces in PHIGS+ are stored in the data structure only for display purposes.

The curves and surfaces in the DME CAD data structure will be stored as complete geometric models. These models can be analyzed and can be modified by the applications program. These entities can also be used to create higher level entities (e.g., solids and features). Moreover, PHIGS+ only supports curves and surfaces, whereas CADMADE will include solids and features as entities in the data structure.
8. The Expert Consultation Environment and the Knowledge Database

The Expert Consultation Environment (ECE) should allow the applications programmers to develop custom expert systems to support their CAD/CAM applications program. These expert systems can be created with the help of high-level procedures and functions. Using the ECE, the applications programmer should not have to spend time developing some of the basic features of expert systems; e.g., inference engines, forward and backward chaining techniques, etc. Instead, the applications program can define the objects or parameters which are to be analyzed by the expert system, define rules in an English-like manner and invoke the expert system, called the "Expert Technician".
8.1 The Knowledge Base and the Inference Engine

The Expert Technician (ET) consists of two parts: the Knowledge Base and the inference engine. The inference engine is the reasoning and problem-solving method manager. The inference engine should contain strategies used to solve problems and acquire knowledge. The ECE should support two types of inference engines:

1. Backward chaining
2. Forward chaining

In backward chaining the inference engine is goal driven and works backward from the goal to determine supporting data. In forward chaining, the solution is constructed from the knowledge in the Knowledge Base.

The Knowledge Base is a repository of information made up of factual knowledge (e.g., of material properties and manufacturing processes) and heuristic knowledge (e.g., rules of thumb). This knowledge could consist of facts about the subject, definitions, formal rules, rules of thumb, or rules of good guessing, and truly represents the expertise of the expert.

The Knowledge Base of CADMADE should be constructed from two types of objects:

1. Parameters
2. Rules
The applications programmer can specify the parameters for the Knowledge Base using the Knowledge Base Manager. Parameters can have constraints and an indication of whether the value of the parameter is to be obtained from the applications program user, the default value in the Knowledge Base or the processing of rules. Rules establish relationships between parameters. Rules can be sent to the Knowledge Base using an English-like syntax.

8.2 Modes of Interaction with the Expert Technician

The ECE has been designed to closely approximate the way an expert is consulted by a designer/engineer. The applications program has the flexibility to decide on the applications program--ET interaction and the applications program user--ET interaction methods. The applications program can use the User Interface Environment of CADMADE to allow the end user to interact with the ET. A dialog area can be attached to each Expert Technician and the user can interact with one ET at a time or with all of them through one dialog area.

Five different modes of interaction have been identified for inclusion in the ECE:

1. Transaction Mode
2. Dialog Mode
3. Observer Mode
4. Teach Mode
5. Learn Mode
Transaction Mode

In the *Transaction* mode the ET should reply once to every query from the applications program. Thus, it is a one query, one reply mode. The query could come from the applications program or directly from the applications program user through the dialog areas supported by the User Interface Environment. In the transaction mode, there will be one output from the applications program to the ET and one input from the ET to the program.

Dialog Mode

The *Dialog* mode is a multiple query, multiple reply mode in which the ET may answer one query with multiple replies, multiple queries with one reply or multiple queries with multiple replies.

Observer Mode

In the *Observer* mode, the ET should act as a person observing the design process and interrupt the designer with suggestions and warnings whenever necessary. Thus, when the ET is in observer mode, there will be continuous output of information from the applications program to the expert system and occasional input to the applications program from the ET.

This mode of interaction can be compared with the "event mode" input supported by GKS and PHIGS in which the user triggers an interrupt. In the observer mode, the ET triggers an interrupt based on rules in the Knowledge Base.
Teach Mode

The teach mode can be used by the applications program to "teach" the ET. The applications program user acts as a teacher and sends information/knowledge to the ET and the Knowledge Base.

Learn Mode

The learn mode differs from the teach mode in the sense that the ET selectively adds knowledge to the Knowledge Base. In the learn mode, the ET should continuously observe the output to the expert system from the applications program and selectively add important information to the Knowledge Base.

The modes of interaction described above can be compared with the modes of interaction allowed for the input devices supported by PHIGS. A major difference between the two is that the ET should be able to operate in more than one mode at any given time.
A prototype of the User Interface Environment was created on an IBM VM/CMS system. The routines were coded in FORTRAN. PHIGS was used for graphics support.

9.1 The Menu/Icon System

The routines to support the creation of menu structures were written. The menu structures are stored in a file which acts as a part of the UIE data structure. Only the Menu Text Primitives (both Title and Item) are supported in this prototype. The support routines for Menu Title Icons and Menu Item Icons only store the Icon data in the data structure. The display of Icons is not supported. All attributes of the Menu primitives (except Icons) are supported in this prototype implementation. Attribute hierarchy is achieved by pushing attributes on to a stack when an "execute submenu"
function is encountered. Only individual specifications of menu locations is possible in the prototype UIE. Menu structure editing and global menus are not supported.

Use of PHIGS for the Menu Interface

The menu data are stored in a file on the VM/CMS system. When the "CRMNRT" (CReate MeNu RooT) function is invoked, the PHIGS structures used to display the menus are created. PHIGS structures 10000 to 90000 are used for root structures posted to the workstation and structures 1000 to 9999 are used for displaying menus. The name set 126 is used for invisible menus and 127 is used for visible menus. The menus are displayed using view 0 and the Normalized Device Coordinates used range from -1.0 to 1.0 in the X and Y directions.

The PHIGS pick input device is used for menu pick. The pick path is checked to find the structure containing the picked element. If it is a menu structure, a menu pick is returned along with the menu pick path. A menu identifier of -1 indicates a return and a menu identifier 0 indicates a root menu. Defaults are defined for menu attributes and locations. A new input device class has been defined for menu pick. In the event mode of input, a menu pick event has a class value of 7.

9.2 The User Communication Interface

The prototype of the UIE includes support software for defining and using one dialog area. Most of the attributes related to dialog areas can be used by applications
programs using the prototype UIE. Messages can be sent to this dialog area using a CADMADE message utility. A string input device (keyboard) is attached to the dialog area. The character "$" signifies a system command for the string input. Any string input starting with a "$" is treated as a CMS command and executed using the VM/CMS "SYSCAL" function. The string input for menu selection, dialog buffer and dialog overwrite mode are not supported in the prototype UIE.

9.3 The CAD View Manager

One function in the prototype UIE allows the applications program to automatically apply viewing transformations to a view by using valuators. The seven view transformations (X, Y, Z rotations, X, Y, Z translations and scaling) are mapped to seven valuator input devices.

9.4 Testing the Prototype UIE

An applications program was written to test the various features of the menu system and the user communication interface of the prototype UIE. The menu hierarchy shown in Fig. 21 was used to create the menu structures in the UIE. This hierarchy also tested the capability of the UIE to allow the same menu structure to be invoked in different
Figure 21. Menu Tree Used to Test the Prototype UIE
Figure 22 shows the root menu with default UIE attributes and locations. In the menu structure "MAIN", attributes were not set for the menu primitives. The system default attributes and the default location parameters were used by the UIE to display this menu screen. In the menu structure "CHARACTER" different colors and sizes have been used for the menu title and the menu items. Figure 23 shows these text attributes. An example of different background attributes is shown in Fig. 24. In this menu structure, the locations of the menu items were modified and the background fill area vertices, fill area styles and outline line types were also changed. Figures 25 and 26 also demonstrate the capability of the applications program to modify menu locations. In all these figures, the message in the dialog area is generated automatically by the UIE from the "Menu Prompt" attribute of the menu structure.

9.5 A Sample CAD Applications Program Using CADMADE

A sample CAD applications program was created using the prototype UIE and the prototype feature-based modeler created by Gandhi (1989). This applications program allows the user to step through a hierarchical menu tree and design parts using some basic features supported by the feature based modelers.
Figure 22. Root Menu Screen of Test Program
Figure 23. Example of Text Attributes of Menus
Figure 24. Example of Background Attributes of Menus
Figure 25. Menu Location - Example 1
Figure 26. Menu Location - Example 2
Two menu trees were used for this CAD applications program. The first tree, "SHAPEs", was created to allow the user to define the generic shape of the feature and then create a specific feature by selecting the feature from the menu (Fig. 27). The second menu tree, "UTILITIES" allows the user to access a deleting facility provided by the program to delete the entire object or just a feature of the object being designed (Fig. 28). Figures 29 and 30 show examples of the menu screens for the CAD applications program.

The sequence of processes invoked by the applications program to implement the menu system and the feature-based modeler is:

1. Open PHIGS
2. Open Workstation
3. Initialize Workstation (colors, input devices, etc.)
4. Open the UIE
5. Create Menu Structures
6. Open the DME
7. Initialize the Feature-Based Modeler
8. Create Menu Root for SHAPEs menu tree
9. Create Menu Root for UTILITIES menu tree
10. Post SHAPEs menu tree
11. Post UTILITIES menu tree
12. CADMADE Await Event
13. If a menu pick is encountered, process the menu pick: either call "Next Menu" to display the next menu screen or create or delete features after further user input.
Figure 27. "SHAPES" Menu Tree for Sample CAD Program
Figure 28. "UTILITIES" Menu Tree for Sample CAD Program

9. A Prototype of the UIE
Figure 29. Menu Screen of Sample CAD Program - Example 1
Figure 30. Menu Screen of Sample CAD Program - Example 2
The last step in the process listed above would have been the most complicated programming task in the absence of the UIE and the feature-based modeler. The UIE allows the applications program to automatically display the correct menu with the menu prompt when a menu item is picked with just one FORTRAN subroutine invocation. If the menu picked is at the lowest level of the "SHAPES" tree, the applications program has to invoke only one FORTRAN subroutine of the feature-based modeler to display an input template for the correct feature and create the feature. Seven valuators are initialized by the CAD program and the prototype UIE automatically applies viewing transformations to the geometry view when any valuator is used.

Figures 31 and 32 show models which were created using this sample CAD applications program.

9.6 Conclusions

The implementation of the prototype UIE consisted of approximately 10 000 lines of documented FORTRAN code. The feature-based modeler consisted of 12 000 lines of FORTRAN code. The sample CAD application was coded using 2300 lines of documented FORTRAN code. Thus, the CAD applications program had 22 000 lines of program supporting it.

Setting up menus for this application was an extremely simple programming task. In the absence of the UIE, a method for storing and displaying menus would have had to be
Figure 31. Model Created using the Sample CAD Program - Example 1
Figure 32. Model Created using the Sample CAD Program - Example 2
designed. FORTRAN code would have had to be written for displaying the menus, setting pick identifiers for the menus, making previous menus invisible and current menus visible, displaying prompts, setting up a dialog area, scrolling messages, displaying new messages, and various other similar functions. All these functions would have had to be programmed using PHIGS.

Because of the high-level functions provided by the UIE, this sample CAD program only had to define the character strings for the menu items in the menu structures, invoke submenus from within the menu structures, and post the root menus. The dialog area was automatically created by the prototype UIE and prompts for menus were automatically sent to the dialog area. The CAD program did not have to scroll the messages every time a message was sent to the dialog area. The use of valuators for viewing transformations was also greatly simplified by the prototype UIE.

The feature-based modeler was used as a prototype of a portion of the DME. The feature-based modeler was used without having to learn about the various features involved. A default template supplied by the feature-based modeler was used for user input. In the absence of the feature-based modeler, the CAD programmer would have had to create a vast database of the various features, the parameters required to create the features and default values of the parameter ratios. Code would have had to be written to find the appropriate parameters for the feature selected by the user, use PHIGS to create a template, associate the PHIGS template structure with a view, account for the user interactions with the template, create surface patches to model the feature, generate PHIGS structures to display the model, draw lines in the PHIGS structures to display the patches, associate the structures with a view and post the PHIGS structure to the workstation. This sequence of functions was achieved by the
sample CAD program by invoking only one FORTRAN subroutine in the prototype feature-based modeler.

Thus, the creation of this sample CAD applications program was achieved with considerable ease because of the availability of high-level functions for creating the user interface and generating geometric models.
10. Summary and Recommendations

10.1 Summary

In this dissertation, an approach towards the establishment of a CAD/CAM programming standard has been presented. This programming environment is called CADMADE - Computer-Aided Design and Manufacturing Applications Development Environment. CADMADE includes not only graphics programming support, but also high-level procedures to support the creation of geometric modeling, mechanical design, manufacturing, expert systems and user interface software. CADMADE consists of five programming environments and two database managers. The programming environments are:

1. The User Interface Environment (UIE)--User Interface data structure
2. PHIGS + --PHIGS + data structure
3. The Design and Modeling Environment (DME)--CAD data structure and Product Information data structure
4. The Virtual Manufacturing Environment (VME)--CAM data structure and Product Information data structure

5. The Expert Consultation Environment (ECE)

The database managers are:

1. Universal Database Manager (UDM)--Universal Database
2. Knowledge Base Manager (KBM)--Knowledge Base

The User Interface Environment (UIE) was designed in great detail. Various aspects of user--applications program interaction were addressed. A data structure for storing the UIE data was created. Higher level procedures to facilitate viewing operations for CAD/CAM applications were designed. Suggested FORTRAN bindings were created for the functions supported by the UIE and for some control functions of CADMADE.

The Design and Modeling Environment (DME) was designed. The requirements of the CAD data structure which supports the DME were created. A new concept in logical input/output devices for the DME was introduced. These logical modeling input/output devices facilitate user interactions with geometric models created by CAD/CAM applications. The functional requirements of some of these logical modeling input/output devices were created.

The requirements for the Expert Consultation Environment (ECE) and the Knowledge Base were created. A new concept in the modes of interaction between expert systems and the applications program and applications program user was created. These modes were modeled after the input device interaction modes supported by graphics standards.

10. Summary and Recommendations
A prototype of the User Interface Environment was created based on PHIGS. This prototype supports most of the functions specified in the UIE. Example results of output from this prototype UIE were presented. A sample CAD applications program was created using the prototype UIE and a prototype feature-based modeler created by Gandhi (1989). The advantages of using higher level programming environments were discussed based on the sample CAD applications program.

10.2 Recommendations

The following is a list of suggested areas of research to further the concept of machine-independent and device-independent CAD/CAM programming interfaces similar to CADMADE.

1. The Virtual Manufacturing Environment

The Virtual Manufacturing Environment (VME), the CAM data structure and the Product Information data structure are some very important features of CADMADE which were not designed in this dissertation. These features have to be designed with great care and implemented in a device-independent manner to allow an implementation of CADMADE to be complete.
2. The Expert Consultation Environment

The Expert Consultation Environment and the Knowledge Base need to be designed in detail before bindings can be established to allow the ECE to be supported in implementations of CADMADE. Some other features of expert systems and artificial intelligence which need to be included in the ECE are natural language processing and uncertainty reasoning.

3. Logical Modeling Input/Output Devices

The concept of logical modeling input/output devices has been introduced for the first time in CADMADE. Further research is necessary in this field to include devices other than the ones described in this dissertation. An example is a "deformation" device which would allow the user to grip a model and deform it with her/his fingers.

4. Feature Recognition

CADMADE has been designed to allow feature-based design. Feature recognition is a useful implement which often facilitates the integration of CAD and CAM. Feature recognition should be included in the DME.

5. The Design and Modeling Environment Bindings

Language bindings have to be designed in order to allow the DME to be included in an implementation of CADMADE.
6. The Universal Database

The database to be provided for the use of the applications program should be carefully designed and included in CADMADE.

7. Error Handling

Error handling is an important feature of graphics standards and error handling in CADMADE must be designed carefully for efficiency and consistency.

The use of applications programming environments similar to CADMADE will free applications programmers from the task of writing support software (e.g., graphics, geometric modeling, etc.) for their CAD/CAM applications and they can concentrate directly on developing the overall system and the technology involved. Such a collection of high-level CAD/CAM programming procedures would provide an environment for easier, faster and more correct development of CAD/CAM applications. If, in the future, such a system can be agreed upon and accepted as a standard by ANSI and ISO, CAD/CAM programmers will be able to write portable software much more rapidly than current graphics standards allow.
References


References 132
Appendix A. Suggested FORTRAN Bindings for the User Interface Environment

In this appendix FORTRAN language bindings are suggested for an implementation of the UIE. Bindings for some CADMADE control functions have also been created and are included in this section.
Control Functions

Open User Interface Environment

SUBROUTINE CMOPUI (WKID, HPDNR, ERFIL, FNAME)

Input Parameters:
INTEGER WKID workstation identifier
INTEGER HPDNR hierarchical pick device no.
INTEGER ERFIL error file logical unit number
CHARACTER FNAME error file name

Description:
Open the User Interface Environment

Initialize Feature Based Modeling

SUBROUTINE CMIFBM (WKID, ASSNUM, PRTNUM, OBJNUM,
ASNAME, PRTNAM, OBNAME, UNUM, WNUM,
VIEWID, PRIORT)

Input Parameters:
INTEGER WKID workstation identifier
INTEGER ASSNUM assembly structure identifier
INTEGER PRTNUM part structure identifier
INTEGER OBJNUM object structure identifier
INTEGER UNUM no. of u lines/surface patch
INTEGER WNUM  no. of w lines/surface patch
INTEGER VIEWID  view identifier
CHARACTER ASNAME  assembly name
CHARACTER PRTNAM  part name
CHARACTER OBNAME  object name
REAL PRIORT  model PHIGS structure priority

Description:
Initialize the Feature Based Modeler

Open Design and Modeling Environment

SUBROUTINE CMOPDM (WKID, ERFIL, FNAME)

Input Parameters:
INTEGER WKID  workstation identifier
INTEGER ERFIL  error file logical unit number
CHARACTER FNAME  name of error file

Description:
Open the Design and Modeling Environment
Output Primitive Functions

Menu/Icon Interface

Menu Title Text

SUBROUTINE CMMTTX (LENGTH, NPTBFA, XBFA, YBFA, TITLE)

Input Parameters:
INTEGER LENGTH     no. of characters in title text
INTEGER NPTBFA     no. of vertices of the background fill area
REAL XBFA(*)      x coordinate array
REAL YBFA(*)      y coordinate array
CHARACTER TITLE    text for menu title

Description:
Draw a menu title text

Menu Title Icon

SUBROUTINE CMMTIC (STRID, NPTBFA, XBFA, YBFA)

Input Parameters:
INTEGER STRID     PHIGS structure to be used as the title Icon
INTEGER NPTBFA(*) no. of vertices for the background fill areas
REAL XBFA(*) X coordinates array for background fill area
REAL YBFA(*) Y coordinate array for background fill area

Description:
Draw a title Icon

Menu Item Text

SUBROUTINE CMMITX (NITEMS, LENGTH, NPBFA, XBFA, YBFA, ITEMS)

Input Parameters:
INTEGER NITEMS no. of items
INTEGER LENGTH(*) array having no. of characters in each item
INTEGER NPBFA(*) no. of vertices for the background fill areas
REAL XBFA(*) X coordinates array for background fill area
REAL YBFA(*) Y coordinates array for background fill area
CHARACTER ITEMS(*) array containing texts for menu items

Description:
Draw menu text items

Menu Item Icon

SUBROUTINE CMMIIC (NITEMS, STRID, NPBFA, MENUID, XBFA, YBFA)

Input Parameters:
INTEGER NITEMS number of menu items
INTEGER STRID(*) PHIGS structures to be used as the
item Icons

INTEGER NPBFA(*)  
no. of vertices for the background fill areas

INTEGER MENUID(*)  
array of menu pick identifiers

REAL XBFA(*)  
X coordinate array for background fill areas

REAL YBFA(*)  
Y coordinate array for background fill areas

*Description:*  
Draw item Icons
User Communication Interface

Send Message to Dialog Area

SUBROUTINE CMMSG (WKID, DGID, INSTR)

Input Parameters:

INTEGER WKID          workstation identifier
INTEGER DGID          dialog area identifier
CHARACTER INSTR       message string

Description:

Display a character string as a message in a specified dialog area.
Attribute Specification Functions

Menu/Icon Interface

Set Menu Title Text Path

SUBROUTINE CMMTCP (PATH)

*Input Parameters:*

INTEGER PATH \hspace{1cm} \text{menu title text path}

(RIGHT, LEFT, UP, DOWN)

*Description:*

Set the menu title text path

Set Menu Title Character Height

SUBROUTINE CMMTCH (HEIGHT)

*Input Parameters:*

REAL HEIGHT \hspace{1cm} \text{menu title character height}

*Description:*

Set the menu title character height

Set Menu Title Character Spacing

SUBROUTINE CMMTCS (SPACE)
**Input Parameters:**

REAL SPACE  menu title character spacing

*Description:*

Set the menu title character spacing

---

**Set Menu Title Character Expansion Factor**

SUBROUTINE CMMTXP (EXPANS)

*Input Parameters:*

REAL EXPANS  menu title character expansion factor

*Description:*

Set the menu title character expansion factor

---

**Set Menu Title Text Alignment**

SUBROUTINE CMMTAL (HORIZ, VERT)

*Input Parameters:*

REAL HORIZ  menu title text alignment horizontal
            (NORMAL, LEFT, CENTER, RIGHT)

REAL VERT  menu title text alignment vertical
            (NORMAL, TOP, CAP, HALF)

*Description:*

Set the menu title text alignment

---

**Set Menu Title Text Color Index**

SUBROUTINE CMMTCI (COLOR)
Input Parameters:
INTEGER COLOR menu title text color index

Description:
Set the menu title color index

Set Menu Title Text Font

SUBROUTINE CMMTFO (FONT)

Input Parameters:
INTEGER FONT menu title text font

Description:
Set the menu title text font

Set Menu Title Text Index

SUBROUTINE CMMTTI (INDEX)

Input Parameters:
INTEGER INDEX menu title bundled text attribute index

Description:
Set the menu title bundled text attribute index

Set Menu Title Text Precision

SUBROUTINE CMMTPR (PREC)

Input Parameters:
INTEGER PREC menu title text precision
Description:
Set the menu title text precision

Set Menu Title Character Up Vector

SUBROUTINE CMMTUP (CHUX, CHUY)

Input Parameters:
REAL CHUX, CHUY menu title character up vector

Description:
Set the menu title character up vector

Set Menu Title Area Background Flag

SUBROUTINE CMMTBF (FLAG)

Input Parameters:
INTEGER FLAG menu title area background flag
(ON, OFF)

Description:
Specifies whether background fill area is to be displayed

Set Menu Title Area Outline Flag

SUBROUTINE CMMTOF (FLAG)

Input Parameters:
INTEGER FLAG menu title area outline flag
Description:
Specifies whether outline of menu title background is to be displayed

Set Menu Title Area Background Color Index

SUBROUTINE CMMTBC (COLOR)

Input Parameters:
INTEGER COLOR menu title area background color index

Description:
Set the menu title background fill area color index

Set Menu Title Area Outline Color Index

SUBROUTINE CMMTOC (COLOR)

Input Parameters:
INTEGER COLOR menu title area outline color index

Description:
Set the menu title background area outline color index

Set Menu Title Area Background Interior Style

SUBROUTINE CMMTBS (STYLE)

Input Parameters:
INTEGER STYLE

menu title area background interior style

Description:
Set the menu title background interior style

Set Menu Title Area Background Interior Style Index

SUBROUTINE CMMTBI (INDEX)

Input Parameters:
INTEGER INDEX menu title area background interior style index

Description:
Set the menu title background interior style index

Set Menu Title Area Outline Line Type

SUBROUTINE CMMTOT (TYPE)

Input Parameters:
INTEGER TYPE menu title area outline line type

Description:
Specifies the line type for menu title area background outline

Set Menu Title Area Outline Width Factor

SUBROUTINE CMMTOW (WIDTH)

Input Parameters:
REAL WIDTH menu title area outline width factor

Description:
Set the menu title area outline width factor

Set Menu Item Text Path

SUBROUTINE CMMICP (PATH)

Input Parameters:
INTEGER PATH menu item text path
(RIGHT, LEFT, UP, DOWN)

Description:
Set the menu item text path

Set Menu Item Character Height

SUBROUTINE CMMICH (HEIGHT)

Input Parameters:
REAL HEIGHT menu item character height

Description:
Set the menu item character height

Set Menu Item Character Spacing

SUBROUTINE CMMICS (SPACE)

Input Parameters:
REAL SPACE menu item character spacing
Description:
Set the menu item character spacing

Set Menu Item Character Expansion Factor

SUBROUTINE CMMIXP (EXPANS)

Input Parameters:
REAL EXPANS menu item character expansion factor

Description:
Set the menu item character expansion factor

Set Menu Item Text Alignment

SUBROUTINE CMMIAL (HORIZ, VERT)

Input Parameters:
REAL HORIZ menu item text alignment horizontal
(NORMAL, LEFT, CENTER, RIGHT)
REAL VERT menu item text alignment vertical
(NORMAL, TOP, CAP, HALF)

Description:
Set the menu item text alignment

Set Menu Item Text Color Index

SUBROUTINE CMMICI (COLOR)

Input Parameters:
INTEGER COLOR menu item text color index

Description:
Set the menu item text color index

Set Menu Item Text Font

SUBROUTINE CMMIFO (FONT)

Input Parameters:
INTEGER FONT menu item text font

Description:
Set the menu item text font

Set Menu Item Text Index

SUBROUTINE CMMITI (INDEX)

Input Parameters:
INTEGER INDEX menu item text index

Description:
Set the menu item bundled text attribute index

Set Menu Item Text Precision

SUBROUTINE CMMIPR (PREC)

Input Parameters:
INTEGER PREC menu item text precision

Description:
Set Menu Item Character Up Vector

SUBROUTINE CMMIUP (CHUX, CHUY)

*Input Parameters:*

REAL CHUX, CHUY menu item character up vector

*Description:*

Set the menu item character up vector

Set Menu Item Area Background Flag

SUBROUTINE CMMIBF (FLAG)

*Input Parameters:*

INTEGER FLAG menu item area background flag

(ON, OFF)

*Description:*

Specifies whether background fill area is to be displayed

Set Menu Item Area Outline Flag

SUBROUTINE CMMIOF (FLAG)

*Input Parameters:*

INTEGER FLAG menu item area outline flag

(ON, OFF)

*Description:*


Specifies whether outline of menu title background fill area is to be displayed

**Set Menu Item Area Background Color Index**

SUBROUTINE CMMIBC (COLOR)

*Input Parameters:*

INTEGER COLOR  
menu item background color index

*Description:*

Set the menu item background fill area color index

**Set Menu Item Area Outline Color Index**

SUBROUTINE CMMIOC (COLOR)

*Input Parameters:*

INTEGER COLOR  
menu item area outline color index

*Description:*

Set the menu item outline color index

**Set Menu Item Area Background Interior Style**

SUBROUTINE CMMIBS (STYLE)

*Input Parameters:*

INTEGER STYLE  
menu item area background interior style

*Description:*

Set the menu item area background interior style
Set Menu Item Area Background Interior Style Index

SUBROUTINE CMMIBI (INDEX)

Input Parameters:
INTEGER INDEX menu item area background interior style index

Description:
Set the menu item area background interior style index

Set Menu Item Area Outline Line Type

SUBROUTINE CMMIOT (TYPE)

Input Parameters:
INTEGER TYPE menu item area outline line type

Description:
Set the menu item area outline line type

Set Menu Item Area Outline Width Factor

SUBROUTINE CMMIOW (WIDTH)

Input Parameters:
REAL WIDTH menu item area outline width factor

Description:
Set the menu item area outline width factor
Set Menu Prompt Dialogue Identifier

SUBROUTINE CMMPDA (DLGID)

*Input Parameters:*

INTEGER DLGID  menu prompt dialogue identifier

*Description:*

Specifies dialogue area for automatic prompt display for the menu structure

Set Menu Prompt String

SUBROUTINE CMMPST (LENGTH, STRING)

*Input Parameters:*

INTEGER LENGTH  length of prompt string

CHARACTER STRING  menu prompt string

*Description:*

Specifies prompt string for the menu structure

Set Menu Title Location

SUBROUTINE CMMTLC (XLOC, YLOC)

*Input Parameters:*

REAL XLOC  X coordinate of menu title

REAL YLOC  Y coordinate of menu title

*Description:*

Set the menu title location
Set Menu Item Location

SUBROUTINE CMMILC (NITEMS, XLOC, YLOC)

*Input Parameters:*

INTEGER NITEMS  
no. of menu items

REAL XLOC(*)  
X coordinates of menu items

REAL YLOC(*)  
Y coordinates of menu items

*Description:*

Set the menu item location

Set Title Icon Transformation Matrix

SUBROUTINE CMMTXF (XFORM)

*Input Parameters:*

REAL XFORM(4, 4)  
Transformation matrix for menu title icon

*Description:*

Set the title Icon transformation matrix

Set Items Icon Transformation Matrices

SUBROUTINE CMMIXF (NITEMS, XFORM)

*Input Parameters:*

INTEGER NITEMS  
no. of menu items

REAL XFORM(4, 4, *)  
Transformation matrices for menu item icons

*Description:*

Set the item Icon transformation matrices

Set Menu Location Mode

SUBROUTINE CMMLCM (FLAG)

Input Parameters:
INTEGER FLAG menu location mode
(INDIVIDUAL, BUNDLED)

Description:
Set the menu location mode

Set Menu Bundled Location Path

SUBROUTINE CMMBLP (PATH)

Input Parameters:
INTEGER PATH path for spacing menu items for
bundled location mode
(LEFT, RIGHT, UP, DOWN)

Description:
Set the path for locating menu items in the bundled location mode
User Communication Interface

Set Dialog Area Character Height

SUBROUTINE CMDGCH (DGID, HEIGHT)

Input Parameters:
- INTEGER DGID dialog area identifier
- REAL HEIGHT dialog area character height

Description:
Set the dialog area character height

Set Dialog area Character Spacing

SUBROUTINE CMDGCS (DGID, SPACE)

Input Parameters:
- INTEGER DGID dialog area identifier
- REAL SPACE dialog area character spacing

Description:
Set the dialog area character spacing

Set Dialog Area Character Expansion Factor

SUBROUTINE CMDGXP (DGID, EXPANS)

Input Parameters:
- INTEGER DGID dialog area identifier

Appendix A. Suggested FORTRAN Bindings for the User Interface Environment
REAL EXPANS dialog area character expansion factor

Description:
Set the dialog area character expansion factor

Set Dialog Area Text Color Index

SUBROUTINE CMDGCI (DGID, COLOR)

Input Parameters:
INTEGER DGID dialog area identifier
INTEGER COLOR dialog area text color index

Description:
Set the dialog area text color index

Set Dialog Area Text Font

SUBROUTINE CMDGFO (DGID, FONT)

Input Parameters:
INTEGER DGID dialog area identifier
INTEGER FONT dialog area text font

Description:
Set the dialog area text font

Set Dialog Area Text Index

SUBROUTINE CMDGTI (DGID, INDEX)

Input Parameters:
INTEGER DGID dialog area identifier
INTEGER INDEX dialog area text index

Description:
Set the dialog area bundled text attribute index

Set Dialog Area Text Precision
SUBROUTINE CMDGPR (DGID, PREC)

Input Parameters:
INTEGER DGID dialog area identifier
INTEGER PREC dialog area text precision

Description:
Set the dialog area text precision

Set Dialog Area Background Flag
SUBROUTINE CMDGBF (DGID, FLAG)

Input Parameters:
INTEGER DGID dialog area identifier
INTEGER FLAG dialog area background flag
(ON, OFF)

Description:
Specifies whether the dialog background fill area is to be displayed

Set Dialog Area Outline Flag
SUBROUTINE CMDGOF (DGID, FLAG)
Input Parameters:
INTEGER DGID         dialog area identifier
INTEGER FLAG        dialog area outline flag

Description:
Specifies whether the dialog area outline is to be displayed

Set Dialog Area Background Color Index
SUBROUTINE CMDGBC (DGID, COLOR)

Input Parameters:
INTEGER DGID         dialog area identifier
INTEGER COLOR       dialog area background color

Description:
Set the dialog area background color index

Set Dialog Area Outline Color Index
SUBROUTINE CMDGOC (DGID, COLOR)

Input Parameters:
INTEGER DGID         dialog area identifier
INTEGER COLOR       dialog area outline color index

Description:
Set the dialog area outline color index

Set Dialog Area Background Interior Style
SUBROUTINE CMDGBS (DGID, STYLE)
Input Parameters:

INTEGER DGID         dialog area identifier
INTEGER STYLE       dialog area background interior style

Description:
Set the dialog area background interior style

Set Dialog Area Background Interior Style Index

SUBROUTINE CMDGBI (DGID, INDEX)

Input Parameters:

INTEGER DGID         dialog area identifier
INTEGER INDEX       dialog area background interior style index

Description:
Set the dialog area background interior style index

Set Dialog Area Outline Line Type

SUBROUTINE CMDGOT (DGID, TYPE)

Input Parameters:

INTEGER DGID         dialog area identifier
INTEGER TYPE        dialog area outline line type

Description:
Set the dialog area outline line type

Set Dialog Area Outline Width Factor

SUBROUTINE CMDGOW (DGID, WIDTH)
Input Parameters:

INTEGER DGID  
dialog area identifier
REAL WIDTH  
dialog area outline width factor

Description:

Set the dialog area outline width factor

Set Dialog Area Buffer Size

SUBROUTINE CMDGBU (DGID, NBUF)

Input Parameters:

INTEGER DGID  
dialog area identifier
INTEGER NBUF  
dialog area buffer size

Description:

Set the no. of dialog area lines to be saved in the buffer

Set Dialog Area No. of Displayed Lines

SUBROUTINE CMDGNL (DGID, NLINE)

Input Parameters:

INTEGER DGID  
dialog area identifier
INTEGER NLINE  
dialog area no. of displayed lines

Description:

Set the no. of dialog area lines to be displayed
(including the input line)

Set Dialog Area No. of Characters Per Line
SUBROUTINE CMDGNC (DGID, NCHAR)

Input Parameters:
INTEGER DGID  dialog area identifier
INTEGER NCHAR  dialog area no. of characters per line

Description:
Set the dialog area no. of characters per line

Set Dialog Area Scroll Mode

SUBROUTINE CMDGMD (DGID, MODE)

Input Parameters:
INTEGER DGID  dialog area identifier
INTEGER MODE  dialog area scroll mode
(no-scroll/scroll)

Description:
Set the dialog area scroll mode

Set Dialog Area Location

SUBROUTINE CMDGLC (DGID, X1, X2, Y1, Y2)

Input Parameters:
INTEGER DGID  dialog area identifier
REAL X1, X2, Y1, Y2  corner points of the rectangle

Description:
Define the corner points of the rectangular dialog area
Set Dialog Area String Prefixes

SUBROUTINE CMSDPF (DGID, MENU, SYSTEM)

**Input Parameters:**
- INTEGER DGID  
  dialog area identifier
- CHARACTER*1 MENU  
  prefix for menu input
- CHARACTER*1 SYSTEM  
  prefix for host system command

**Description:**
Set the special character prefixes for dialog area string input

Set Dialog Area Echo Mode

SUBROUTINE CMDGEM (DGID, ECHO)

**Input Parameters:**
- INTEGER DGID  
  dialog area identifier
- INTEGER ECHO  
  dialog area echo mode
  (ON, OFF)

**Description:**
Set the dialog area echo mode
CAD View Manager

Set View Area Background Flag

SUBROUTINE CMVWBF (VWID, FLAG)

Input Parameters:
INTEGER VWID view identifier
INTEGER FLAG view area background flag
(ON, OFF)

Description:
Specifies whether the view area background is to be displayed

Set View Area Outline Flag

SUBROUTINE CMVWOF (VWID, FLAG)

Input Parameters:
INTEGER VWID view identifier
INTEGER FLAG view area outline flag

Description:
Specifies whether the view area outline is to be displayed

Set View Area Background Color Index

SUBROUTINE CMVWBC (VWID, COLOR)

Input Parameters:
INTEGER VWID  
view identifier

INTEGER COLOR  
view area background color index

Description:
Set the view area background color index

Set View Area Outline Color Index

SUBROUTINE CMVWOC (VWID, COLOR)

Input Parameters:
INTEGER VWID  
view identifier

INTEGER COLOR  
view area outline color index

Description:
Set the view area outline color index

Set View Area Background Interior Style

SUBROUTINE CMVWBS (VWID, STYLE)

Input Parameters:
INTEGER VWID  
view identifier

INTEGER STYLE  
view area background interior style

Description:
Set the view area background interior style

Set View Area Background Interior Style Index

SUBROUTINE CMVWBI (VWID, INDEX)
Input Parameters:

INTEGER VWID  
view identifier

INTEGER INDEX  
view area background interior style index

Description:
Set the view area background interior style index

Set View Area Outline Line Type

SUBROUTINE CMVWOT (VWID, TYPE)

Input Parameters:

INTEGER VWID  
view identifier

INTEGER TYPE  
view area outline line type

Description:
Set the view area background outline line type

Set View Area Outline Width Factor

SUBROUTINE CMVWOW (VWID, WIDTH)

Input Parameters:

INTEGER VWID  
view identifier

REAL WIDTH  
view area outline width factor

Description:
Set the view area outline width factor
Menu Structure Functions

Create Menu Root

SUBROUTINE CMCMRT (MENUID)

*Input Parameters:*

INTEGER MENUID menu structure identifier

*Description:*

Creates a menu tree starting from a specified menu structure (root menu)

Unpost Menu Root

SUBROUTINE CMUPMN (WKID, MENUID)

*Input Parameters:*

INTEGER WKID workstation identifier
 INTEGER MENUID menu structure identifier

*Description:*

Unposts a menu root from a workstation.

Execute Menu

SUBROUTINE CMEXMN (MENUID)

*Input Parameters:*

INTEGER MENUID menu structure identifier
Description:
Executes a menu sub structure

Open Menu Structure

SUBROUTINE CMOMST (MSTRID)

Input Parameters:
INTEGER MSTRID menu structure identifier

Description:
Opens a menu sub structure

Close Menu Structure

SUBROUTINE CMCMST

Input Parameters:
none

Description:
Closes a menu sub structure

Set Menu Structure Label

SUBROUTINE CMMSLB (LABEL)

Input Parameters:
INTEGER LABEL label

Description:
Sets a label in a menu structure
Set Element Pointer in Menu Structure

SUBROUTINE CMMSEP (EP)

*Input Parameters:
INTEGER EP    element pointer

*Description:
Sets the current element pointer in the menu structure

Offset Element Pointer in Menu Structure

SUBROUTINE CMMOEP (OFFSET)

*Input Parameters:
INTEGER OFFSET no. of elements by which to offset the pointer

*Description:
Offsets the current element pointer in the menu structure

Set Element Pointer at Label in Menu Structure

SUBROUTINE CMMEPL (LABEL)

*Input Parameters:
INTEGER LABEL label

*Description:
Sets the current element pointer at a label in a menu structure

Delete Menu Structure Element
SUBROUTINE CMMDEL

Input Parameters:
none

Description:
Deletes the current element in a menu structure

Delete Menu Structure Elements Between Range

SUBROUTINE CMMDER (EP1, EP2)

Input Parameters:
INTEGER EP1 start element number
INTEGER EP2 end element number

Description:
Deletes all the elements in a given range in a menu structure

Delete Menu Structure Elements Between Labels

SUBROUTINE CMMDLL (LABEL1, LABEL2)

Input Parameters:
INTEGER LABEL1 label from which elements are to be deleted
INTEGER LABEL2 label till which elements are to be deleted

Description:
Deletes all the elements between two labels in a menu structure

Empty Menu Structure
SUBROUTINE CMEMST (MSTRID)

Input Parameters:
INTEGER MSTRID menu structure identifier

Description:
Empties a menu structure and removes all the elements in the menu structure

Delete Menu Structure

SUBROUTINE CMDMST (MSTRID)

Input Parameters:
INTEGER MSTRID menu structure identifier

Description:
Deletes a menu structure

Delete All Menu Structures

SUBROUTINE CMDAMS

Input Parameters:
none

Description:
Deletes all menu structures

Unpost All Menu Structures

SUBROUTINE CMUAMS (WKID)
Input Parameters:

INTEGER WKID    workstation identifier

Description:

Unposts all menu structures from a workstation
**Modified PHIGS Functions**

**Await Event**

SUBROUTINE CMWAIT (TIME, WKID, ICL, IDNR)

*Input Parameters:*

REAL TIME \[\text{time out (seconds)}\]

INTEGER WKID \[\text{workstation identifier}\]

*Output Parameters:*

INTEGER ICL \[\text{class of input device}\]

INTEGER IDNR \[\text{logical input device number}\]

*Description:*

Await event

**Get Hierarchical Pick**

SUBROUTINE CMGTHP (NDEPTH, ISTAT, PPD, PP)

*Input Parameters:*

INTEGER NDEPTH \[\text{depth of pick path to be returned}\]

*Output Parameters:*

INTEGER ISTAT \[\text{pick status}\]

INTEGER PPD \[\text{depth of pick path returned}\]

INTEGER PP(3,*) \[\text{pick path}\]

*Description:*

Appendix A. Suggested FORTRAN Bindings for the User Interface Environment
Get hierarchical pick path

Get String

SUBROUTINE CMGTST (LOST, STR)

Input Parameters:
none

Output Parameters:
INTEGER LOST length of input string
CHARACTER STR*(*) input string

Description:
Get string input pick path

Initialize String Input Device

SUBROUTINE CMINST (WKID, DGID, STDNR, LENT, TEXT)

Input Parameters:
INTEGER WKID workstation identifier
INTEGER DGID dialog area identifier
INTEGER STDNR string input device number
INTEGER LENT length of initial string
CHARACTER TEXT*(*) initial string

Description:
Initialize string input device for dialog area
CAD View Functions

Define Transformation for Predefined Viewing Facility

SUBROUTINE CMDFVW (VIEWID, PRDFID)

Input Parameters:

INTEGER VIEWID view index of view whose current transformation is to be stored
INTEGER PRDFID predefined view transformation index

Description:
Stores viewing transformation with a view transformation index

Set Predefined Viewing Transformation

SUBROUTINE CMSPVW (VIEWID, PRDFID)

Input Parameters:

INTEGER VIEWID view index to be used for predefined viewing transformation
INTEGER PRDFID predefined view transformation index

Description:
Transforms a view with a predefined view transformation index
Define View Transformation Index for a Predefined Transformation Table

SUBROUTINE CMDVTI (N, INDEX, XFORM)

Input Parameters:

INTEGER N  no. of indices to be defined
INTEGER INDEX(*)  array of transformation indices
INTEGER XFORM(*)  array of predefined transformations
(XROT, YROT, ZROT, XTRAN, YTRAN, ZTRAN, SCALE)

Description:

Defines view transformation indices for predefined transformations

Define Input Device - View Transformation Map

SUBROUTINE CMIDTM (IDNR, NXFORM, XFORM)

Input Parameters:

INTEGER IDNR  input device number (valuator)
INTEGER NXFORM  no. of view transformation indices to be mapped to the input device
INTEGER XFORM(*)  array of view transformation indices

Description:

Maps an input device (valuator) to a no. of view transformation indices

Define View Transformation Index - View Index Map

SUBROUTINE CMVTVM (VTINDX, NVIEWS, VIEWS)
Input Parameters:

INTEGER VTINDX  predefined view transformation index
INTEGER NVIEWS no. of views to be mapped to the view transformation index
INTEGER VIEWS(*) array of view indices

Description:
Maps a view transformation index to a number of PHIGS views
Miscellaneous Functions

Set Dialog Area Visibility

SUBROUTINE CMDGVS (DGID, VIS)

*Input Parameters:*

INTEGER DGID  dialog area identifier
INTEGER VIS   dialog area visibility
               (ON, OFF)

*Description:*

Sets the dialog area visibility flag

Define Global Menu

SUBROUTINE CMDFGM (ROOT)

*Input Parameters:*

INTEGER ROOT  root of the global menu tree

*Description:*

Sets a menu tree to be the global menu tree

Extract Values From String

SUBROUTINE CMEXNM (STRING, IER, NVALS, VALS)
Input Parameters:

CHARACTER*(*) STRING character string containing numerical values

Output Parameters:

INTEGER IER error indicator
INTEGER NVALS number of values
REAL VALS(*) numerical values in the string

Description:

Parses a character string and extracts numerical values

Associate Dialog Area With A Menu Tree

SUBROUTINE CMADGM (DGID, ROOT)

Input Parameters:

INTEGER DGID dialog area identifier
INTEGER ROOT root menu structure of menu tree

Description:

Associates a dialog area with a menu tree for string menu selection

Set Dialog Area Priority

SUBROUTINE CMDGPR (DGID, REFDG, RPRIOR)

Input Parameters:

INTEGER DGID dialog area identifier
INTEGER REFDG reference dialog area identifier
INTEGER RPRIOR relative priority (HIGHER, LOWER)

Description:
Sets the priority of a dialog area with respect to another dialog area

**Scroll Dialog Area**

SUBROUTINE CMSCDG (DGID, NLINES)

*Input Parameters:*
- INTEGER DGID dialog area identifier
- INTEGER NLINES no. of lines to be scrolled
  (positive for scroll up, negative for scroll down)

*Description:*
Srolls a dialog area

**Reset Scrolled Dialog Area**

SUBROUTINE CMRSDG (DGID)

*Input Parameters:*
- INTEGER DGID dialog area identifier

*Description:*
Resets a scrolled dialog area
Appendix B. CADMADE Functions Supported by the Prototype UIE

This section contains a list of CADMADE functions which are supported by the prototype implementation of the UIE discussed in this dissertation. The names of the subroutines are arranged in alphabetical order. A * after the subroutine name indicates that the subroutine in the implementation differs from the binding suggested in appendix A. A ** after the subroutine name indicates that the function has not been fully implemented.
CMCMRT          CMCMST          CMDGBC
CMDGBF          CMDGBI          CMDGBS
CMDGBU**        CMDGCH          CMDGCI
CMDGCS          CMDGFO          CMDGLC
CMDGMD**        CMDGNC          CMDGNL
CMDGOC          CMDGOF          CMDGOT
CMDGOW          CMDGPR          CMDGTI
CMDGVS          CMDGX0          CMEXMN
CMGTHP          CMGTST          CMIFBM
CMINST*         CMMDEL**        CMMDER**
CMMDLL**        CMMEPL**        CMMIAL
CMMIBC          CMMIBF          CMMIBI
CMMIBS          CMMICH          CMMICI
CMMICP          CMMICS          CMMIFO
CMMIIC**        CMMILC          CMMIOC
CMMIOF          CMMIOT          CMMIOW
CMMIPR          CMMIST          CMMITI
CMMITX          CMMIUP          CMMIXP
CMMOEP**        CMMPDA          CMMPST
CMMSSEP**       CMMSG           CMMSLB
CMMTAL          CMMTBC          CMMTBF
CMMTBI          CMMTBS          CMMTCH
CMMTCI          CMMTCP          CMMTCS
CMMTFO          CMMTIC**        CMMTLC
CMMTOC          CMMTOF          CMMTOT
CMMTOW          CMMTPR          CMMTST

Appendix B. CADMADE Functions Supported by the Prototype UIE
<table>
<thead>
<tr>
<th>CMMTTI</th>
<th>CMMTTX</th>
<th>CMMTUP</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMMTXP</td>
<td>CMNXMN</td>
<td>CMOMST</td>
</tr>
<tr>
<td>CMOPDM**</td>
<td>CMOPUI</td>
<td>CMPOMN</td>
</tr>
<tr>
<td>CMUPMN</td>
<td>CMWAIT</td>
<td></td>
</tr>
</tbody>
</table>
Appendix C. Code Listing - Prototype UIE

This section contains a listing of the code written to implement the prototype User Interface Environment. All modules starting with "CM" are CADMADE functions which can be accessed by the applications programmer.
SUBROUTINE CMOPUI (MKID, HPDNR, ERFIL, FNAME)

INTEGER MKID, HPDNR, ERFIL
CHARACTER(*) FNAME

INTEGER DATLLN
COMMON /DATSTR/ DATLLN

C SET THE LOGICAL UNIT NUMBER FOR THE DATA STRUCTURE FILE
DATLLN = 99

C OPEN THE USER INTERFACE ERROR FILE
OPEN (UNIT=ERFIL, FILE = FNAME)

C SET THE MENU ATTRIBUTE DEFAULTS
CALL CLMNAT

C SET THE DIALOG AREA ATTRIBUTE DEFAULTS
CALL CLDSAT

C SET THE NAME 126 TO BE NOT PICKABLE
CALL PSHPFT (MKID, HPDNR, 1, 127, 1, 126)

C SET THE NAME 126 TO BE INVISIBLE
CALL PSIVFT (MKID, 1, 126, 1, 127)

RETURN
END
C INPUT PARAMETERS:
C WKID - INTEGER - WORKSTATION IDENTIFIER
C ASSNUM - INTEGER - ASSEMBLY IDENTIFIER
C OBJNUM - INTEGER - OBJECT IDENTIFIER
C ASNAME - CHARACTER(*) - ASSEMBLY NAME
C OBJNAME - CHARACTER(*) - OBJECT NAME
C UNNUM - INTEGER - NUMBER OF INTERPOLATION LINES IN THE U
DIRECTION FOR MIREFRAME SURFACE PATCHES
C WNNUM - INTEGER - NUMBER OF INTERPOLATION LINES IN THE W
DIRECTION FOR MIREFRAME SURFACE PATCHES
C VIEWID - INTEGER - VIEW INDEX FOR DISPLAYING THE FEATURE MODELS
C PRIORT - INTEGER - PHIGS STRUCTURE PRIORITY FOR THE MODELS
C
C OUTPUT PARAMETERS:
C
C COMMON INPUTS:
C
C COMMON OUTPUTS:
C
C LOCAL VARIABLES:
C
C LOCAL VARIABLES:
C
C FUNCTIONAL DESCRIPTION:
C CADMAKE - INITIALIZE FEATURE BASED MODELING
C
C MODLES CALLED:
C
C CODED BY: SANKAR JAYARAM
C DATE: 03/05/89 (18:19:54)

SUBROUTINE CADFBM (WKID, ASSNUM, OBJNUM, INNUM, ASNAME, OBNAME,
; UNNUM, WNNUM, VIEWID, PRIORT)
INTEGER WKID, ASSNUM, OBJNUM, INNUM, ASNAME, OBNAME,
RESREAL PRIORT
CHARACTER*50 ASNAME, OBNAME
INTEGER A$NN, ®NN, INN
CHARACTER*C(900) ASSEAN900), ®ject(900)
COMMON /IDS/ ASNUM, OBNUM, INNUM
COMMON /COMP/ ASSEM, OBJECT
COMMON /PRTOF/ CNUM
COMMON /PATCH/ NU, NM
REAL VN(4), VT(4)
DATA VN/ 0.1, 0.1, 0.1, 0.1 /
DATA VT/ 0.1, 0.95, 0.005, 0.145 /

C INITIALIZE THE COMMON BLOCKS FOR THE FEATURE BASED MODELER
ASNUM = ASSNUM
OBNUM = OBJNUM
INNUM = INNUM
CNUM = OBNUM
ASSEM ASSNUM = ASNAME
OBJECT(OBJNUM) = OBNAME
NU = UNNUM
NM = WNNUM

C INITIALIZE THE VIEW FOR THE TEMPLATES OF THE FEATURE BASED MODELER
CALL GPMHP2(WKID,5,MIN, VPT)
CALL GPVCH (WKID,5,1,1,1,2,0,2,1,2)

C INITIALIZE THE OBJECT PHIGS STRUCTURE
CALL POPST (OBJNUM)
CALL PSVWIVIEWID
CALL PAD5(1,127)
CALL PSHPID(OBJNUM)
CALL PCLST
POST THE OBJECT PHIGS STRUCTURE
CALL PPORT (MKID, OBJNUM, PRIORITY)
RETURN
END

C=================================================================================================
C PROJECT: CADMADE
C VERSION: X1.0.0
C PART: UIE-
C MODULE : 
C=================================================================================================
C CALL MOPDM (MKID, ERFIL, FNAME)
C=================================================================================================
C INPUT PARAMETERS:
C MKID - INTEGER - WORKSTATION IDENTIFIER
C ERFIL - INTEGER - ERROR FILE LOGICAL UNIT NUMBER
C FNAME - CHARACTER(*) - ERROR FILE NAME
C=================================================================================================
C OUTPUT PARAMETERS:
C
C=================================================================================================
C COMMON INPUTS:
C NONE
C=================================================================================================
C COMMON OUTPUTS:
C NONE
C=================================================================================================
C LOCAL VARIABLES:
C
C FUNCTIONAL DESCRIPTION:
C CADMADE - OPEN DESIGN AND MODELING ENVIRONMENT
C=================================================================================================
C MODULES CALLED:
C
C=================================================================================================
C Coded By: Sankar Jayaram
C Date: 03/05/89 (16:19:54)
C=================================================================================================

SUBROUTINE CHOPDM (MKID, ERFIL, FNAME)

***THIS SUBROUTINE IS NOT FUNCTIONAL YET
INTEGER MKID, ERFIL
CHARACTER(*) FNAME

C OPEN THE CAD DATA STRUCTURE FILE
C OPEN VARIOUS OTHER FILES INCLUDING THE ERROR FILES
C SET THE DEFAULTS

RETURN
END

C=================================================================================================
C PROJECT: CADMADE
C VERSION: X1.0.0
C PART: UIE-
C MODULE : 
C=================================================================================================
C CALL CMITTX (LENGTH, NPTBFA, XBFA, YBFA, TITLE)
C=================================================================================================
C INPUT PARAMETERS:
C LENGTH - INTEGER - LENGTH OF THE TITLE CHARACTER STRING
C NPTBFA - INTEGER - NO. OF POINTS IN THE BACKGROUND FILL AREA
C XBFA - REAL(*) - ARRAY OF X COORDINATES OF THE FILL AREA

Appendix C. Code Listing - Prototype UIE
SUBROUTINE CMFTTX (LENGTH, NPTBFA, XBFA, YBFA, TITLE)
INTEGER LENGTH, NPTBFA
REAL XBFA(*), YBFA(*)
CHARACTER TITLE
INTEGER ELEMID, DATLIN
COMMON /DATSTR/ DATLIN
PARAMETER (ELEMID = 1010)
WRITE(DATLIN, *) ELEMID
WRITE THE INTEGER PARAMETERS
WRITE(DATLIN, *) LENGTH, NPTBFA
WRITE THE REAL PARAMETERS
WRITE(DATLIN, 1001) (XBFA(I), I=1,NPTBFA)
WRITE(DATLIN, 1001) (YBFA(I), I=1,NPTBFA)
WRITE THE CHARACTER PARAMETER
WRITE(DATLIN, '(A)') TITLE
1001 FORMAT (6E12.5)
RETURN
ENDC

Appendix C. Code Listing - Prototype UIE
FUNCTIONAL DESCRIPTION:
1012 - CADMADE MENU TITLE ICON

MODULES CALLED:

CODED BY: SANKAR JAYARAM
DATE: 02/19/89 (10:17:59)

Subroutine CMITX (STRID, NPTBFA, XBFA, YBFA)

INTEGER ELEMID, DATLUN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1012)

WRITE(DATLUN, *) ELEMID

RETURN
END
WRITE(DATLUN, #) ELEMID
C WRITE THE INTEGER PARAMETERS
WRITE(DATLUN, #) NITEMS
WRITE(DATLUN, 1002) (LENGTH(I), I=1,NITEMS)
WRITE(DATLUN, 1002) (NPBFA(I), I=1,NITEMS)
WRITE(DATLUN, 1002) (MENUID(I), I=1,NITEMS)
C CALCULATE THE TOTAL NUMBER OF POINTS REQUIRED FOR THE BACKGROUND
NTOT = 0
DO 100 I = 1, NITEMS
   NTOT = NTOT + NPBFA(I)
100 CONTINUE
C WRITE THE REAL PARAMETERS
WRITE(DATLUN, 1001) (XBFA(I), I=1,NTOT)
WRITE(DATLUN, 1001) (YBFA(I), I=1,NTOT)
C WRITE THE CHARACTER PARAMETER
DO 200 I = 1, NITEMS
   WRITE(DATLUN, '(A)') ITEMS(I)
200 CONTINUE
1001 FORMAT (6E12.5)
1002 FORMAT (6I12)
RETURN
END
C==============================================
C CADMADE MODULE CHMIIC
C==============================================
C PROJECT: CADMADE
C VERSION: X1.0.0
C PART : UIE---
C MODULE : 
C==============================================
C COMPILER: FORTVS2 (FORTRAN 77)
C CALLING SEQUENCE: 
C CALL CHMIIC (NITEMS, STRID, NPBFA, MENUID, XBFA, YBFA)
C==============================================
C INPUT PARAMETERS:
C NITEMS - INTEGER - NUMBER OF ITEMS
C STRID - INTEGER(*) - PHIGS STRUCTURES TO BE USED AS THE ICONS
C NPBFA - INTEGER(*) - NO. OF POINTS IN EACH BACKGROUND FILL AREA
C MENUID - INTEGER(*) - ARRAY OF MENU PICK IDENTIFIERS
C XBFA - REAL(*) - ARRAY OF X COORDINATES OF THE FILL AREAS
C YBFA - REAL(*) - ARRAY OF Y COORDINATES OF THE FILL AREAS
C==============================================
C OUTPUT PARAMETERS:
C==============================================
C COMMON INPUTS:
C DATSTR 
C==============================================
C COMMON OUTPUTS:
C NONE
C==============================================
C LOCAL VARIABLES:
C==============================================
C FUNCTIONAL DESCRIPTION:
C 1016 - CADMADE MENU ITEM ICON
C==============================================
C MODULES CALLED:
C==============================================
C CODED BY: SANKAR JAYARAJ
C DATE: 02/19/89 (10:19:10)
C==============================================

SUBROUTINE CHMIIC (NITEMS, STRID, NPBFA, MENUID, XBFA, YBFA)
INTEGER ELEMID, DATLUN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1016)
WRITE(DATLUN, #) ELEMID

Appendix C. Code Listing - Prototype UIE
C CALL CNNTLC (XLOC, YLOC)
C
C COMMON INPUTS:
C DATSTR
C COMMON OUTPUTS:
C NONE
C LOCAL VARIABLES:
C
C FUNCTIONAL DESCRIPTION:
C 1020 - CADNADE MENJ TITLE LOCATION
C MODULES CALLED:
C
C CODED BY: SANKAR JAYARAM
C DATE: 02/19/89 (10:25:15)
C
SUBROUTINE CMILC (XLOC, YLOC)
REAL XLOC, YLOC
INTEGER ELEMID, DATLUN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1020)
WRITE(DATLUN, *) ELEMID
WRITE THE MENU TITLE LOCATION INTO THE DATA STRUCTURE
WRITE(DATLUN, *) XLOC, YLOC
RETURN
END

 appending
SUBROUTINE CMHEL (NITEMS, XLOC, YLOC)

INTEGER NITEMS
REAL XLOC(*), YLOC(*)

INTEGER ELEMID, DATLUN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1030)

INTEGER MAXITM, NMAX
PARAMETER (MAXITM = 20)

WRITE(DATLUN, *) ELEMID

C CHECK TO FIND THE MAXIMUM NUMBER OF MENU ITEMS
NMAX = MIN (NITEMS, MAXITM)

C WRITE THE NUMBER OF ITEM LOCATIONS SPECIFIED
WRITE(DATLUN, *) NMAX

C WRITE THE MENU TITLE LOCATION INTO THE DATA STRUCTURE
WRITE(DATLUN, 1001) (XLOC(i), i=1, NMAX)
WRITE(DATLUN, 1001) (YLOC(i), i=1, NMAX)

1001 FORMAT (6E12.5)

RETURN
END

Appendix C. Code Listing - Prototype UIE
```
C DATE: 03/05/89 (09:17:43)

SUBROUTINE CHEXIDN (MENUID)
INTEGER MENUID
INTEGER ELEMID, DATLUN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1001)
WRITE(DATLUN, *) ELEMID
WRITE(DATLUN, *) MENUID
RETURN
END

END
```

```
C CADMADU MODULE CMHTCP

C PROJECT: CADMADU
C VERSION: X1.0.0
C PART: UIE-__
C MODULE: __
C CALLING SEQUENCE:
C CALL CMHTCP (PATH)
C INPUT PARAMETERS:
C PATH - INTEGER - TEXT PATH
C COMMON INPUTS:
C DATSTR
C COMMON OUTPUTS:
C NONE
C LOCAL VARIABLES:
C
C FUNCTIONAL DESCRIPTION:
C 1050 - CADMADU SET MENU TITLE TEXT PATH
C MODULES CALLED:
C Coded by: SANKAR JAYARAM
C DATE: 02/18/89 (20:01:38)

SUBROUTINE CMHTCP (PATH)
INTEGER PATH
INTEGER ELEMID, DATLUN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1050)
WRITE(DATLUN, *) ELEMID
WRITE(DATLUN, *) PATH
RETURN
END
```
C MODULE :
C=============================================================================
C COMPILER: FORTSZ2 (FORTRAN 77)
C=============================================================================
C CALLING SEQUENCE:
C CALL CMHTCS (HEIGHT)
C=============================================================================
C INPUT PARAMETERS:
C HEIGHT - REAL - CHARACTER HEIGHT
C=============================================================================
C OUTPUT PARAMETERS:
C=============================================================================
C COMMON INPUTS:
C DATSTR
C=============================================================================
C COMMON OUTPUTS:
C NONE
C=============================================================================
C LOCAL VARIABLES:
C=============================================================================
C FUNCTIONAL DESCRIPTION:
C 1051 - CADMADE SET MENU TITLE CHARACTER HEIGHT
C=============================================================================
C MODULES CALLED:
C=============================================================================
C CODED BY: SANKAR JAYARAH
C DATE: 02/18/89 (20:00:32)
C=============================================================================

SUBROUTINE CMHTCS (HEIGHT)

REAL HEIGHT
INTEGER ELEMID, DATLUN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1051)

WRITE(DATLUN, *) ELEMID
WRITE(DATLUN, *) HEIGHT

RETURN
END

C CADMADE MODULE CMHTCS
C=============================================================================
C PROJECT: CADMADE
C VERSION: X1.0.0
C PART : UIE___
C MODULE :
C=============================================================================
C COMPILER: FORTSZ2 (FORTRAN 77)
C=============================================================================
C CALLING SEQUENCE:
C CALL CMHTCS (SPACE)
C=============================================================================
C INPUT PARAMETERS:
C SPACE - REAL - CHARACTER SPACING
C=============================================================================
C OUTPUT PARAMETERS:
C=============================================================================
C COMMON INPUTS:
C DATSTR
C=============================================================================
C COMMON OUTPUTS:
C NONE
C=============================================================================
C LOCAL VARIABLES:
C=============================================================================
C FUNCTIONAL DESCRIPTION:
C 1052 - CADMADE SET MENU TITLE CHARACTER SPACING
C=============================================================================
C MODULES CALLED:
C=============================================================================
C CODED BY: SANKAR JAYARAH

Appendix C. Code Listing - Prototype UIE
SUBROUTINE CHMTC5 (SPACE)

REAL SPACE

INTEGER ELEMID, DATLLN
COMMON /DATSTR/ DATLLN
PARAMETER (ELEMID = 1052)
WRITE(DATLLN, *) ELEMID

WRITE THE SPACING DATA INTO THE DATA STRUCTURE
WRITE(DATLLN, *) SPACE
RETURN
END

C CADHADE MODULE CHMTXP

C PROJECT: CADHADE
C VERSION: X1.0.D
C PART : UIE-
C MODULE :
C COMPILER: FORTVSZ (FORTRAN 77)
C CALLING SEMIEMIE:
C CALL CHMTXP (EXPANS)
C INPUT PARAMETERS:
C EXPANS - REAL - CHARACTER EXPANSION FACTOR
C COMMON INPUTS:
C DATSTR
C COMMON OUTPUTS:
C NONE
C LOCAL VARIABLES:
C FUNCTIONAL DESCRIPTION:
C 1053 - CADHADE SET MENU TITLE CHARACTER EXPANSION FACTOR
C MODULES CALLED:
C Coded BY: SANKAR JAYARAM
C DATE: 02/18/89 (19:56:04)

SUBROUTINE CHMTXP (EXPANS)

REAL EXPANS

INTEGER ELEMID, DATLLN
COMMON /DATSTR/ DATLLN
PARAMETER (ELEMID = 1053)
WRITE(DATLLN, *) ELEMID

WRITE THE EXPANSION FACTOR INTO THE DATA STRUCTURE
WRITE(DATLLN, *) EXPANS
RETURN
END

C CADHADE MODULE CHMTAL

C PROJECT: CADHADE
C VERSION: X1.0.0
C PART : UIE-
C MODULE :
C COMPILER: FORTVSZ (FORTRAN 77)

Appendix C. Code Listing - Prototype UIE
CALLING SEQUENCE:
CALL CHMTAL (HORIZ, VERT)

INPUT PARAMETERS:
HORIZ - INTEGER - HORIZONTAL ALIGNMENT
VERT - INTEGER - VERTICAL ALIGNMENT

OUTPUT PARAMETERS:

COMMON INPUTS:

DATSTR

COMMON OUTPUTS:
NONE

LOCAL VARIABLES:

FUNCTIONAL DESCRIPTION:

1054 - CADMADE SET MENU TITLE TEXT ALIGNMENT

MODULES CALLED:

Coded by: Sankar Jayaram
Date: 02/18/89 (19:54:17)

SUBROUTINE CHMTAL (HORIZ, VERT)

INTEGER HORIZ, VERT

INTEGER ELEMID, DATLUN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1054)

WRITE(DATLUN, *) ELEMID

WRITE THE ALIGNMENT INDICATORS INTO THE DATA STRUCTURE
WRITE(DATLUN, *) HORIZ, VERT

RETURN
END

CADMADE MODULE CHMTAL

PROJECT: CADMADE
VERSION: X1.0.0
PART: ___
MODULE:

COMPILED: FORTVS2 (FORTRAN 77)

CALLING SEQUENCE:
CALL CHMTAL (COLOR)

INPUT PARAMETERS:
COLOR - INTEGER - COLOR INDEX

OUTPUT PARAMETERS:

COMMON INPUTS:

DATSTR

COMMON OUTPUTS:
NONE

LOCAL VARIABLES:

FUNCTIONAL DESCRIPTION:

1055 - CADMADE SET MENU TITLE TEXT COLOR INDEX

MODULES CALLED:

Coded by: Sankar Jayaram
Date: 02/18/89 (19:54:17)
SUBROUTINE CMITCI (COLOR)
INTEGER COLOR

INTEGER ELEMID, DATLIN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1055)

WRITE(DATLUN, *) ELEMID

WRITE THE COLOR INDEX INTO THE DATA STRUCTURE
WRITE(DATLUN, *) COLOR

RETURN
END

SUBROUTINE CMITFO (FONT)
INTEGER FONT

INTEGER ELEMID, DATLIN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1057)

WRITE(DATLUN, *) ELEMID

WRITE THE FONT ID INTO THE DATA STRUCTURE
WRITE(DATLUN, *) FONT

RETURN
END
FUNCTIONAL DESCRIPTION:
1059 - CADMADE SET MENU TITLE TEXT INDEX

LOCAL VARIABLES:

SUBROUTINE CHMTTI (INDEX)

INTEGER INDEX

INTEGER ELEMID, DATLUN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1058)

WRITE(DATLUN, *) ELEMID
WRITE(DATLUN, *) INDEX

RETURN
END

MODULES CALLED:

CODED BY: SANKAR JAYARAM
DATE: 02/18/89 (19:51:03)

SUBROUTINE CHMTTI (INDEX)

INTEGER INDEX

INTEGER ELEMID, DATLUN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1058)

WRITE(DATLUN, *) ELEMID
WRITE(DATLUN, *) INDEX

RETURN
END

MODULES CALLED:

CODED BY: SANKAR JAYARAM
DATE: 02/18/89 (19:49:58)
SUBROUTINE CHMTPR (PREC)

INTEGER PREC

INTEGER ELEMID, DATLUN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1059)

WRITE(DATLUN, *) ELEMID
C
WRITE THE PRECISION INTO THE DATA STRUCTURE
WRITE(DATLUN, *) PREC
RETURN
END

C=

C== CADMADE MODULE CHMTPR ==
C PROJECT: CADMADE
C VERSION: X1.0.0
C PART : UIE-
C MODULE :
C===
C CALLING SEQUENCE:
C CALL CHMTPR (CHUX, CHUY)
C===
C INPUT PARAMETERS:
C CHUX - REAL - CHARACTER UP VECTOR X COMPONENT
C CHUY - REAL - CHARACTER UP VECTOR Y COMPONENT
C OUTPUT PARAMETERS:
C COMMON INPUTS:
C DATSTR
C COMMON OUTPUTS:
C NONE
C LOCAL VARIABLES:
C FUNCTIONAL DESCRIPTION:
C 1060 - CADMADE SET MENU TITLE CHARACTER UP VECTOR
C MODULES CALLED:
C CODED BY: SANJAR JAYARAM
C DATE: 02/18/89 (19:49:09)
C===
SUBROUTINE CHMTPR (CHUX, CHUY)

REAL CHUX, CHUY

INTEGER ELEMID, DATLUN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1060)

WRITE(DATLUN, *) ELEMID
C
WRITE THE CHARACTER UP VECTOR INTO THE DATA STRUCTURE
WRITE(DATLUN, *) CHUX, CHUY
RETURN
END

C== CADMADE MODULE CHMTPR ==
C PROJECT: CADMADE
C VERSION: X1.0.0
C PART : UIE-
C MODULE :
C===
C CALLING SEQUENCE:
C CALL CHMTPR (FLAG)
SUBROUTINE CMHTOF (FLAG)

INTEGER FLAG

INTEGER ELEMID, DATLUN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1061)

WRITE(DATLUN, *) ELEMID
WRITE(DATLUN, *) FLAG

RETURN
END

SUBROUTINE CMHTOF (FLAG)
INTEGER FLAG

INTEGER ELEM0, DATL0N
COMMON /DATSTR/ DATL0N
PARAMETER (ELEM0 = 1062)
WRITE(DATL0N, *) ELEM0

WRITE AREA OUTLINE FLAG INTO THE DATA STRUCTURE
WRITE(DATL0N, *) FLAG

RETURN
END

C=======================================================================
C CADMADE MODULE CMHTBC
C=======================================================================
C PROJECT: CADMADE
C VERSION: X1.0.0
C PART : UIE-___
C MODULE :
C=======================================================================
C COMPILER: FORTV52 (FORTRAN 77)
C=======================================================================
C CALLING SEQUENCE:
C CALL CMHTBC (COLOR)
C=======================================================================
C INPUT PARAMETERS:
C COLOR - INTEGER - AREA BACKGROUND COLOR INDEX
C=======================================================================
C OUTPUT PARAMETERS:
C=======================================================================
C COMMON INPUTS:
C DATSTR
C=======================================================================
C COMMON OUTPUTS:
C NONE
C=======================================================================
C LOCAL VARIABLES:
C=======================================================================
C FUNCTIONAL DESCRIPTION:
1063 - CADMADE SET MENU TITLE AREA BACKGROUND COLOR INDEX
C=======================================================================
C MODULES CALLED:
C=======================================================================
C CODED BY: SANKAR JAYARAM
C DATE: 02/18/89 (19:45:30)
C=======================================================================

SUBROUTINE CMHTBC (COLOR)

INTEGER COLOR

INTEGER ELEM0, DATL0N
COMMON /DATSTR/ DATL0N
PARAMETER (ELEM0 = 1063)
WRITE(DATL0N, *) ELEM0

WRITE THE AREA BACKGROUND COLOR INDEX INTO THE DATA STRUCTURE
WRITE(DATL0N, *) COLOR

RETURN
END

C=======================================================================
C CADMADE MODULE CMHTOC
C=======================================================================
C PROJECT: CADMADE
C VERSION: X1.0.0
C PART : UIE-___
C MODULE :
C=======================================================================
C COMPILER: FORTV52 (FORTRAN 77)
C=======================================================================
C CALLING SEQUENCE:
C CALL CMHTOC (COLOR)
C=======================================================================
C INPUT PARAMETERS:
C COLOR - INTEGER - AREA OUTLINE COLOR INDEX
SUBROUTINE CMHOC (COLOR)

INTEGER COLOR

INTEGER ELEMID, DATLUN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1064)

WRITE(DATLUN, *) ELEMID

WRITE AREA OUTLINE COLOR INDEX INTO THE DATA STRUCTURE
WRITE(DATLUN, *) COLOR

RETURN
END

SUBROUTINE CMHOC (STYLE)

INTEGER STYLE

INTEGER ELEMID, DATLUN

Appendix C. Code Listing - Prototype UIE
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1065)

WRITE(DATLUN, *) ELEMID

C WRITE AREA BACKGROUND INTERIOR STYLE INTO THE DATA STRUCTURE
WRITE(DATLUN, *) STYLE
RETURN
END

C=====================================================================
C CADMADE MODULE CMUJIBI
C=====================================================================
C PROJECT: CADMADE
C VERSION: X1.0.0
C PART : UIE--
C MODULE :
C=====================================================================
C COMPILER: FORTVS2 (FORTRAN 77)
C=====================================================================
C CALLING SEQUENCE:
C CALL CMUJIBI (INDEX)
C=====================================================================
C INPUT PARAMETERS:
C INDEX - INTEGER - AREA BACKGROUND INTERIOR STYLE INDEX
C=====================================================================
C OUTPUT PARAMETERS:
C=====================================================================
C COMMON INPUTS:
C DATSTR
C=====================================================================
C COMMON OUTPUTS:
C NONE
C=====================================================================
C LOCAL VARIABLES:
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C 1066 - CADMADE SET MENU TITLE AREA BACKGROUND INTERIOR STYLE INDEX
C=====================================================================
C MODULES CALLED:
C=====================================================================
C CODED BY: SANKAR JAYARAM
C DATE: 02/18/89 (18:58:43)
C=====================================================================

SUBROUTINE CMUJIBI (INDEX)

INTEGER INDEX

INTEGER ELEMID, DATLUN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1066)

WRITE(DATLUN, *) ELEMID

C WRITE AREA BACKGROUND INTERIOR STYLE INDEX TO THE DATA
WRITE(DATLUN, *) INDEX
RETURN
END

C=====================================================================
C CADMADE MODULE CMUJTOT
C=====================================================================
C PROJECT: CADMADE
C VERSION: X1.0.0
C PART : UIE--
C MODULE :
C=====================================================================
C COMPILER: FORTVS2 (FORTRAN 77)
C=====================================================================
C CALLING SEQUENCE:
C CALL CMUJTOT (TYPE)
C=====================================================================
C INPUT PARAMETERS:
C TYPE - INTEGER - AREA OUTLINE LINE TYPE
C=====================================================================
C OUTPUT PARAMETERS:
SUBROUTINE CMMT0T (TYPE)

INTEGER TYPE

INTEGER ELEMID, DATLUN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1067)

WRITE(DATLUN, *) ELEMID

WRITE AREA OUTLINE LINE TYPE TO THE DATA STRUCTURE
WRITE(DATLUN, *) TYPE

RETURN
END

SUBROUTINE CMMT0M (WIDTH)

REAL WIDTH

INTEGER ELEMID, DATLUN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1068)

RETURN
END
SUBROUTINE CMICH (PATH)
INTEGER PATH
INTEGER ELEMID, DATLN
COMMON /DATSTR/ DATLN
PARAMETER (ELEMID = 1070)
WRITE (DATLN, *) ELEMID

WRITE AREA OUTLINE WIDTH FACTOR TO THE DATA STRUCTURE
WRITE (DATLN, *) WIDTH
RETURN
END

SUBROUTINE CMICH (PATH)
INTEGER PATH
INTEGER ELEMID, DATLN
COMMON /DATSTR/ DATLN
PARAMETER (ELEMID = 1070)
WRITE (DATLN, *) ELEMID

WRITE AREA OUTLINE WIDTH FACTOR TO THE DATA STRUCTURE
WRITE (DATLN, *) WIDTH
RETURN
END
SUBROUTINE CMICS (HEIGHT)

REAL HEIGHT

INTEGER ELEMID, DATLUN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1071)

WRITE (DATLUN, *) ELEMID

WRITE (DATLUN, *) HEIGHT

RETURN
ENDE

Appendix C. Code Listing - Prototype UIE
WRITE(DATLUN, *) SPACE
RETURN
END

C==============================================================================
C CADMADE MODULE CMMIXXP
C==============================================================================
C PROJECT: CADMADE
C VERSION: X1.0.0
C PART : UIE____
C MODULE : ______
C==============================================================================
C COMPILER: FORTYSZ (FORTRAN 77)
C==============================================================================
C CALLING SEQUENCE: C
C CALL CMMIXXP (EXPANS)
C==============================================================================
C INPUT PARAMETERS:
C EXPANS - REAL - CHARACTER EXPANSION FACTOR
C==============================================================================
C OUTPUT PARAMETERS:
C==============================================================================
C COMMON INPUTS:
C DATSTR
C==============================================================================
C COMMON OUTPUTS:
C==============================================================================
C LOCAL VARIABLES:
C==============================================================================
C FUNCTIONAL DESCRIPTION:
C 1073 - CADMADE SET MENU ITEM CHARACTER EXPANSION FACTOR
C==============================================================================
C MODULES CALLED:
C==============================================================================
C CODED BY: SANKAR JAYARAM
C DATE: 02/18/89 (19:56:04)
C==============================================================================

SUBROUTINE CMMIXXP (EXPANS)
REAL EXPANS
INTEGER ELEHID• DATLLN
COMMON /DATSTR/ DATLLN
PARAMETER (ELEHID = 1073)
WRITE(DATLUN, *) ELEHID
C WRITE THE EXPANSION FACTOR INTO THE DATA STRUCTURE
WRITE(DATLUN, *) EXPANS
RETURN
END

C==============================================================================
C CADMADE MODULE CMI AL
C==============================================================================
C PROJECT: CADMADE
C VERSION: X1.0.0
C PART : UIE____
C MODULE : ______
C==============================================================================
C COMPILER: FORTYSZ (FORTRAN 77)
C==============================================================================
C CALLING SEQUENCE: C
C CALL CMIAL (HORIZ, VERT)
C==============================================================================
C INPUT PARAMETERS:
C HORIZ - INTEGER - HORIZONTAL ALIGNMENT
C VERT - INTEGER - VERTICAL ALIGNMENT
C==============================================================================
C OUTPUT PARAMETERS:
C==============================================================================
C COMMON INPUTS:
C DATSTR
C==============================================================================
C COMMON OUTPUTS:
SUBROUTINE CMHIAL (HORIZ, VERT)
REAL HORIZ, VERT
INTEGER ELEMID, DATLUN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1074)
WRITE(DATLUN, *) ELEMID
C WRITE THE ALIGNMENT INDICATORS INTO THE MENU STRUCTURE
WRITE(DATLUN, *) HORIZ, VERT
RETURN
END

SUBROUTINE CMMICI (COLOR)
INTEGER COLOR
INTEGER ELEMID, DATLUN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1075)
WRITE(DATLUN, *) ELEMID
C WRITE THE COLOR INDEX INTO THE DATA STRUCTURE
WRITE(DATLUN, *) COLOR
SUBROUTINE CMIF0 (FONT)

INTEGER FONT

INTEGER ELEMID, DATLUN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1077)

WRITE(DATLUN, *) ELEMID
C
WRITE THE FONT ID INTO THE DATA STRUCTURE
WRITE(DATLUN, *) FONT
C
RETURN
END
SUBROUTINE CMHI  (INDEX)
INTEGER INDEX
INTEGER ELEMID, DATLUN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1078)
WRITE(DATLUN, *) ELEMID
C WRITE NEXT INDEX ID TO DATA STRUCTURE
WRITE(DATLUN, *) INDEX
RETURN
END

SUBROUTINE CMHIPR (PREC)
INTEGER PREC
INTEGER ELEMID, DATLUN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1079)
WRITE(DATLUN, *) ELEMID
C WRITE TEXT PRECISION INTO THE DATA STRUCTURE
WRITE(DATLUN, *) PREC
RETURN
END
Appendix C. Code Listing - Prototype UIE

```c
SUBROUTINE CHIUP (CHUX, CHUY)

REAL CHUX, CHUY
INTEGER ELEMID, DATLUN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1080)
WRITE(DATLUN, *) ELEMID
WRITE(DATLUN, *) CHUX, CHUY
RETURN
END
```

```c
SUBROUTINE CHIUF (FLAG)

REAL CHUX, CHUY
INTEGER ELEMID, DATLUN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1080)
WRITE(DATLUN, *) ELEMID
WRITE(DATLUN, *) CHUX, CHUY
RETURN
END
```
SUBROUTINE CHMIBF (FLAG)

INTEGER FLAG

INTEGER ELEMID, DATLUN

COMMON /DATSTR/ DATLUN

PARAMETER (ELEMID = 1081)

WRITE(DATLUN, *) ELEMID

C WRITE AREA BACKGROUND FLAG TO THE DATA STRUCTURE

WRITE(DATLUN, *) FLAG

RETURN

END

C CADHADE MODULE CHMIOF

C PROJECT: CADHADE

C VERSION: X1.0.0

C PART: UIE

CIIIDULEC

COMPILER: FORTVSZ (FORTRAN 77)

C CALLING SEQUENCE:

C CALL CHMIOF (FLAG)

C INPUT PARAMETERS:

C INTEGER FLAG

C COMMON INPUTS:

C DATSTR

C COMMON OUTPUTS:

C NONE

C LOCAL VARIABLES:

C FUNCTIONAL DESCRIPTION:

C 1082 - CADHADE SET MENU ITEM AREA OUTLINE FLAG

C MODULES CALLED:

C CODED BY: SANKAR JAYARAM

C DATE: 02/18/89 (19:46:27)

C******************************************************************************

SUBROUTINE CHMIOF (FLAG)

INTEGER FLAG

INTEGER ELEMID, DATLUN

COMMON /DATSTR/ DATLUN

PARAMETER (ELEMID = 1082)

WRITE(DATLUN, *) ELEMID

C WRITE BACKGROUND AREA FLAG INTO THE DATA STRUCTURE

WRITE(DATLUN, *) FLAG

RETURN

END

C CADHADE MODULE CHMIBC

Appendix C. Code Listing - Prototype UIE
SUBROUTINE CMMIBC (COLOR)
INTEGER COLOR
INTEGER ELEMID, DATLUN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1083)
WRITE(DATLUN, *) ELEMID
WRITE(DATLUN, *) COLOR
RETURN
END
SUBROUTINE CMHIBS (COLOR)

INTEGER COLOR

INTEGER ELEMID, DATLUN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1084)

WRITE(DATLUN, *) ELEMID

WRITE AREA OUTLINE COLOR INDEX INTO THE DATA STRUCTURE
WRITE(DATLUN, *) COLOR

RETURN
END

SUBROUTINE CMHIBS (STYLE)

INTEGER STYLE

INTEGER ELEMID, DATLUN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1085)

WRITE(DATLUN, *) ELEMID

WRITE AREA BACKGROUND INTERIOR STYLE INTO THE DATA STRUCTURE
WRITE(DATLUN, *) STYLE

RETURN
END
SUBROUTINE CHMIBI (INDEX)
  INTEGER TYPE
  INTEGER ELEMID, DATLUN
  COMMON /DATSTR/ DATLUN
  PARAMETER (ELEMID = 1086)
  WRITE(DATLUN, *) ELEMID
  WRITE BACKGROUND INTERIOR STYLE INTO THE DATA STRUCTURE
  WRITE(DATLUN, *) INDEX
RETURN
END
SUBROUTINE CHIOT (TYPE)
INTEGER TYPE
INTEGER ELEMID, DATLUN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1087)
WRITE(DATLUN, *) ELEMID
C WRITE AREA OUTLINE LINE TYPE TO THE DATA STRUCTURE
WRITE(DATLUN, *) TYPE
RETURN
END

C CADDWRITEODULECHHIDN
C PROJECT: CADMADE
C VERSION: X1.0.0
C PART : UIE__
C COMPILER: FORTVS2 (FORTRAN 77)
C CALLING SEQUENCE:
C CALL CHMION (WIDTH)
C INPUT PARAMETERS:
C WIDTH — REAL - AREA OUTLINE WIDTH FACTOR
C OUTPUT PARAMETERS:
C COMMON INPUTS:
C NONE
C COMMON OUTPUTS:
C NONE
C LOCAL VARIABLES:
C COMMON DESCRIPTION:
C 1087 - CADMADE SET MENU ITEM AREA OUTLINE WIDTH FACTOR
C MODULES CALLED:
C CODED BY: SANKAR JAYARAM
C DATE: 02/18/89 (18:56:08)
C
SUBROUTINE CHMION (WIDTH)
REAL WIDTH
INTEGER ELEMID, DATLUN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1088)
WRITE(DATLUN, *) ELEMID
C WRITE AREA OUTLINE WIDTH FACTOR INTO THE DATA STRUCTURE
WRITE(DATLUN, *) WIDTH
RETURN
END

C C CADDWRITEODULECHHIDN
C PROJECT: CADMADE
C VERSION: X1.0.0
C PART : UIE__
C COMPILER: FORTVS2 (FORTRAN 77)
C CALLING SEQUENCE:
C CALL CHMION (WIDTH)
C INPUT PARAMETERS:
C WIDTH — REAL - AREA OUTLINE WIDTH FACTOR
C OUTPUT PARAMETERS:
C COMMON INPUTS:
C NONE
C COMMON OUTPUTS:
C NONE
C LOCAL VARIABLES:
C COMMON DESCRIPTION:
C 1088 - CADMADE SET MENU ITEM AREA OUTLINE WIDTH FACTOR
C MODULES CALLED:
C CODED BY: SANKAR JAYARAM
C DATE: 02/18/89 (18:56:08)
C
SUBROUTINE CHMION (WIDTH)
REAL WIDTH
INTEGER ELEMID, DATLUN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1088)
WRITE(DATLUN, *) ELEMID
C WRITE AREA OUTLINE WIDTH FACTOR INTO THE DATA STRUCTURE
WRITE(DATLUN, *) WIDTH
RETURN
END

Appendix C. Code Listing - Prototype UIE
SUBROUTINE CMPDA (DLGID)

INTEGER DLGID

INTEGER ELEMID, DATLUN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 1091)

WRITE(DATLUN, *) ELEMID

WRITE MENU PROMPT DIALOGUE IDENTIFIER TO THE DATA STRUCTURE
WRITE(DATLUN, *) DLGID

RETURN
END
C DATE: 02/18/89 (18:16:51)
C===============================================================================
C SUBROUTINE CMPST (LENGTH, STRING)
 INTEGER LENGTH
 CHARACTER STRING(*)
 INTEGER ELEMID, DATLUN
 COMMON /DATSTR/ DATLUN
 PARAMETER (ELEMID = 1092)
 WRITE(DATLUN, *) ELEMID
 C WRITE MENU PROMPT STRING INTO THE DATA STRUCTURE
 WRITE(DATLUN, "(A)") STRING
 RETURN
 END

C===============================================================================
C MODULE ERRMSG
C PROJECT: CADHADE
C VERSION: X1.0.0
C PART : UIE-
C===============================================================================
C COMPILER: FORTVS2 (FORTRAN 77)
C CALLING SEQUENCE:
 C CALL ERRMSG (NAME, MSG)
C===============================================================================
C INPUT PARAMETERS:
 C NAME - CHARACTER(*) - NAME OF THE ROUTINE WHERE ERROR OCCURRED
 C MSG - CHARACTER(*) - ERROR MESSAGE
C===============================================================================
C OUTPUT PARAMETERS:
C===============================================================================
C COMMON INPUTS:
 C NONE
C COMMON OUTPUTS:
 C NONE
C LOCAL VARIABLES:
C FUNCTIONAL DESCRIPTION:
 C CADMADE WRITE ERROR MESSAGE TO ERROR FILE
===============================================================================
C MODULES CALLED:
 C===============================================================================
C CODED BY: SANKAR JAYARAM
C DATE: 03/04/89 (09:09:09)
C===============================================================================
C SUBROUTINE ERRMSG (NAME, MSG)
 CHARACTER(*) NAME, MSG
 INTEGER NOUT
 PARAMETER (NOUT = 90)
 C WRITE ERROR MESSAGE TO CONSOLE
 WRITE(NOUT,*) 'ERROR (',NAME,') --> ',MSG,'.'
 RETURN
 END

C===============================================================================
C CADMADE MODULE CMMAIT
C PROJECT: CADMADE
C VERSION: X1.0.0
C PART : UIE--
SUBROUTINE CMHAIT (TIME, MKID, ICL, IDNR)
   INTEGER MKID, ICL, IDNR
   REAL TIME
   INTEGER PICKCL, MENUCL, STRNCL
   PARAMETER (PICKCL = 5, STRNCL = 6, MENUCL = 7)
   INTEGER NDEP
   PARAMETER (NDEP = 16)
   INTEGER ISTAT1, PPDI, PP1(3,NDEP)
   COMMON /CMPICK/, ISTAT1, PPDI, PP1
   COMMON /CMSTRN/, LOSTR
   COMMON /CMSTR1/, STRING
   CHARACTER*80 STRING
   INTEGER LOSTR
   CALL PHIGS AHAIT EVENT
100 CALL PMHAIT (TIME, MKID, ICL, IDNR)
   IF (ICL .EQ. PICKCL) THEN
      IF THE EVENT HAS A PICK, GET THE PICK EVENT
      CALL PGTHP (NDEP, ISTAT1, PPDI, PP1)
      CHECK THE PICK STATUS
      IF (ISTAT1 .EQ. 0) THEN
         CHECK THE PICK TO SEE IF IT WAS A MENU PICK
         IF (PP1(1,1) .GE. 10000 .AND. PP1(1,1) .LE. 90000) THEN
         SET THE CLASS TO MENU PICK
         ICL = MENUCL
         ENDFI
      ENDFI
   ENDFI
   IF (ICL .EQ. STRNCL) THEN
      IF THE EVENT HAS A STRING, GET THE STRING EVENT
      CALL PGSTST (LOSTR, STRING)
      CHECK THE STRING TO CHECK THE FIRST CHARACTER
      IF (STRING (1:1) .EQ. '*' ) THEN

C STRIP OUT THE FIRST CHARACTER
STRIN1(1:LO$TR-1) = STRIN1(2:LOSTR)
C EXECUTE SYSTEM COMMAND
CALL SYSCAL (STRIN1, LO$TR-1, IER)
C CLEAR THE SCREEN
CALL SYSCAL ('CLEAR', 5, IER)
C ADD THE STRING TO THE MESSAGES FOR ECHO
CALL CHMSG (MKID, 1, STRING)
C UPDATE THE WORKSTATION
CALL PUNK (MKID, 1)
C GO BACK TO WAIT EVENT
GO TO 100
ENDIF
C ADD THE STRING TO THE MESSAGES
CALL CHMSG (MKID, 1, STRING)
C UPDATE THE WORKSTATION
CALL PUNK (MKID, 1)
ENDIF
RETURN
END

C==============================================================================
C CADMADE MODULE CMGTHP
C==============================================================================
C PROJECT: CADMADE
C VERSION: X1.0.0
C PART : UIE-___
C MODMUE : ___
C==============================================================================
C CALLING SEQUENCE:
C CALL CMGTHP (NDEPTH, ISTAT, PPD, PP)
C==============================================================================
C INPUT PARAMETERS:
C NDEPTH - INTEGER - DEPTH OF PICK PATH TO BE RETURNED
C==============================================================================
C OUTPUT PARAMETERS:
C ISTAT - INTEGER - PICK STATUS
C PPD - INTEGER - PICK DEPTH RETURNED
C PP(3,*), INTEGER - PICK PATH
C==============================================================================
C COMMON INPUTS:
C CMPICK
C==============================================================================
C COMMON OUTPUTS:
C COMMON-NONE
C==============================================================================
C LOCAL VARIABLES:
C==============================================================================
C FUNCTIONAL DESCRIPTION:
C CADMADE GET HIERARCHICAL PICK
C==============================================================================
C MODULES CALLED:
C==============================================================================
C CODED BY: SANKAR JAYARAM
C DATE: 03/06/89 (14:14:51)
C==============================================================================

SUBROUTINE CMGTHP (NDEPTH, ISTAT, PPD, PP)
INTEGER NDEPTH, ISTAT, PPD, PP(3,*

INTEGER NDEP
C PARAMETER (NDEP = 16)
INTEGER ISTAT1, PPD1, PPL3(NDEP)
COMMON /CMPICK/ ISTAT1, PPD1, PPL1

C RETURN THE PICK PARAMETERS
ISTAT = ISTAT1
PPD = PPDI

DO 100 IP = 1,NDEPTH
PP(1,IP) = PP1(1,IP)
PP(2,IP) = PP1(2,IP)
PP(3,IP) = PP1(3,IP)

100 CONTINUE
RETURN
END

C=============================================================================
C CADMADE MODULE CMGTST
C=============================================================================
C PROJECT: CADMADE
C VERSION: X1.0.0
C PART : UIE___
C MODULE : _
C=============================================================================
C COMPILER: FORTVS2 (FORTRAN 77)
C=============================================================================
C CALLING SEQUENCE:
C CALL CMGTST (LOSTR, STRING)
C=============================================================================
C INPUT PARAMETERS:
C=============================================================================
C OUTPUT PARAMETERS:
LOSTR- INTEGER - LENGTH OF STRING RETURNED
STRING- CHARACTER(*) - STRING RETURNED
C=============================================================================
C COMMON INPUTS:
CMSTRN, CMSTR1
C=============================================================================
C COMMON OUTPUTS:
NONE
C=============================================================================
C LOCAL VARIABLES:
C=============================================================================
C FUNCTIONAL DESCRIPTION:
CADMADE GET STRING
C=============================================================================
C MODULES CALLED:
C=============================================================================
C CODED BY: SANKAR JAYARAM
C DATE: 05/06/89 (14:14:51)
C=============================================================================
SUBROUTINE CMGTST (LOSTR, STR)
CHARACTER(*) STR
INTEGER LOST
CHARACTER*80 STRING
INTEGER LOSTR
COMMON /CMSTRN/ LOSTR
COMMON /CMSTR1/ STRING
LOSTR = LOSTR
STR(1:LOSTR) = STRING(1:LOSTR)
RETURN
END

C=============================================================================
C CADMADE MODULE CMCHR T
C=============================================================================
C PROJECT: CADMADE
C VERSION: X1.0.0
C PART : UIE___
C MODULE : _
C COMPILER: FORTVS2 (FORTRAN 77)
C=============================================================================
CALLING SEQUENCE:
CALL CMCHRT (MENUID)

INPUT PARAMETERS:
MENUID - INTEGER - MENU STRUCTURE IDENTIFIER

OUTPUT PARAMETERS:

COMMON INPUTS:
NONE

COMMON OUTPUTS:

LOCAL VARIABLES:

FUNCTIONAL DESCRIPTION:
C CADDMAE CREATE MENU ROOT

MODULES CALLED:

CODED BY: SANJAR JAYARAM
DATE: 05/04/89 (08:46:56)

SUBROUTINE CMCHRT (MENUID)

INTEGER MENUID
INTEGER PHRT, IER
INTEGER NXTSTR
COMMON /PHGSTR/ NXTSTR

GET THE PHIGS ROOT STRUCTURE ID
CALL GTPHRT (MENUID, PHRT, IER)

SET THE PHIGS STRUCTURE ID IN THE COMMON BLOCK
NXTSTR = PHRT

SEARCH THE MENU DATA STRUCTURE AND FIND THE ROOT MENU STRUCTURE
CALL FNDMRT (MENUID, PHRT)

CREATE THE PHIGS STRUCTURES FOR THE ROOT MENU
CALL CRPHRT (MENUID, PHRT)

CLEAR THE MENU ATTRIBUTES AND RESTORE THE DEFAULT VALUES
CALL CLMNAT

RETURN

END

MODULE CLMNAT

PROJECT: CADDMAE
VERSION: X1.0.0
PART: UIE

MODULE:

COMPILER: FORTVS2 (FORTRAN 77)

CALLING SEQUENCE:
CALL CMNAT

INPUT PARAMETERS:

COMMON INPUTS:
NONE

COMMON OUTPUTS:
MENUAT, MNJAT1

LOCAL VARIABLES:

FUNCTIONAL DESCRIPTION:
C   CADMADE CLEAR CURRENT MENU ATTRIBUTE SETTINGS
C================================================================================================
C MODULES CALLED:
C================================================================================================
C CODED BY: SANKAR JAYARAM
C DATE: 05/05/89 (13:45:30)
C================================================================================================

SUBROUTINE CLMNX

**********************************************************************************************
C START OF COMMON BLOCK FOR SAVING MENU ATTRIBUTES
**********************************************************************************************

INTEGER NATT

PARAMETER (MAXN = 50, MAXTM = 20)

INTEGER TPATH(MAXN), THAL(MAXN), TVAL(MAXN), TCI(MAXN),
> TFONT(MAXN), TIND(MAXN), TPREC(MAXN), TBFLAG(MAXN),
> TOFLAG(MAXN), TBCI(MAXN), TBSI(MAXN), TBSI(MAXN),
> TBSI(MAXN), TOLT(MAXN),
> TPATH(MAXN), IHAL(MAXN), IVAL(MAXN), ICI(MAXN),
> IFONT(MAXN), IIND(MAXN), IPREC(MAXN), IBFLAG(MAXN),
> IFLAG(MAXN), IBCI(MAXN), I OCI(MAXN), IBSI(MAXN),
> IBSI(MAXN), IOLT(MAXN),
> DLOGID(MAXN), PRMTLN(MAXN)

REAL TH(TMAXN), TSPACE(MAXN), TEXPAN(MAXN), TUPX(MAXN), TUPY(MAXN),
> ITH(MAXN), ISPACE(MAXN), IEXPAN(MAXN), IUPX(MAXN), IUPY(MAXN),
> IOM(MAXN), IXLOC(MAXN), IYLOC(MAXN),
CHARACTER*80 PROMPT(MAXN)

COMMON //MNATT, NATT,
> TPATH, THAL, TVAL, TCI, TFONT, TIND, TPREC, TBFLAG,
> TOFLAG, TBCI, TBSI, TBSI, TOLT,
> TPATH, IHAL, IVAL, ICI, IFONT, IIND, IPREC, IBFLAG,
> IFLAG, IBCI, I OCI, IBSI, IBSI, IOLT,
> DLOGID, PRMTLN,
> ITH, ISPACE, IEXPAN, IUPX, IUPY, IOM, IXLOC, IYLOC,
COMMON //MNATT/ PROMPT

**********************************************************************************************
C END OF COMMON BLOCK
**********************************************************************************************

C SET THE NUMBER OF WAITING ATTRIBUTE LISTS
NATT = 1

C RESET THE CURRENT ATTRIBUTES

TPATH (1) = 0
THAL  (1) = 2
TVAL (1) = 3
TCI  (1) = 1
TFONT (1) = 2
TIND (1) = 1
TPREC (1) = 1
TBFLAG (1) = 1
TOFLAG (1) = 1
TBCI (1) = 0
TOCI (1) = 1
TBSI (1) = 1
TBSI (1) = 1
TOLT (1) = 1

IPATH (1) = 0
IHAL (1) = 2
IVAL (1) = 3
ICI  (1) = 3
IFONT (1) = 2
IIND (1) = 1
IPREC (1) = 1
IBFLAG (1) = 1
IFLAG (1) = 1
IBCI (1) = 0
IOCI (1) = 3
IBS (1) = 1
IBSI (1) = 1
IGLT (1) = 1
DLOGID (1) = 1
PRMTLN (1) = 14
THT (1) = 0.03
TSPACE (1) = 1.0
TEXPAN (1) = 1.0
TUPX (1) = 0.0
TUPY (1) = 1.0
TON (1) = 1.0
TXLOC (1) = 0.7
TYLOC (1) = 0.85
IHT (1) = 0.02
ISPACE (1) = 1.0
IEKPN (1) = 1.0
IUPX (1) = 0.0
IUPY (1) = 1.0
ICM (1) = 1.0

PROMPT (1) = 'PICK MENU ITEM'

DO 50 ITEM = 1, MAXITEM
   TLOC(ITEM, 1) = 0.7
   ILOC(ITEM, 1) = 0.725 - FLOAT(ITEM-1) * 0.1
50 CONTINUE

RETURN

END

C*************************************************************************
C MODULE GTPHRT
C*************************************************************************
C PROJECT: CADMADE
C VERSION: X1.0.0
C PART : UIE__
C COMPILER : FORTVS2 (FORTRAN 77)
C*************************************************************************
C CALLING SEQUENCE:
C CALL GTPHRT (MENIID, PHRT, IER)
C*************************************************************************
C INPUT PARAMETERS:
C MENIID - INTEGER - MENU IDENTIFIER
C*************************************************************************
C OUTPUT PARAMETERS:
C PHRT - INTEGER - PHIGS ROOT STRUCTURE IDENTIFIER
C IER - INTEGER - ERROR INDICATOR
C*************************************************************************
C COMMON INPUTS:
C MNPCHR
C*************************************************************************
C COMMON OUTPUTS:
C MNPCHR
C*************************************************************************
C LOCAL VARIABLES:
C*************************************************************************
C FUNCTIONAL DESCRIPTION:
C CADMADE SET PHIGS ROOT STRUCTURE ID FOR ROOT MENU
C*************************************************************************
C MODULES CALLED:
C*************************************************************************
C CODED BY: SANKAR JAYARAM
C DATE: 05/04/89 (08:51:53)
C*************************************************************************

SUBROUTINE GTPHRT (MENIID, PHRT, IER)
   INTEGER MENIID, PHRT, IER
   INTEGER NROOTS
   PARAMETER (NROOTS = 9)
   INTEGER AVALRT (NROOTS,2)

Appendix C. Code Listing - Prototype UIE
COMMON /HNPHRT/ AVALRT

C  RESET ERROR FLAG
IER = 1

DO 100 ICOUNT = 1, NRROTS - 1

C  CHECK TO SEE IF THE ROOT IS BEING USED
IF (AVALRT(ICOUNT,1).EQ. 0) THEN
C  IF THE ROOT IS AVAILABLE; TAKE IT
AVALRT (ICOUNT,1) = 1
C  ASSOCIATE THE MENJID WITH THE PHIGS ROOT
AVALRT(ICOUNT,2) = MENJID
C  RETURN THE CORRESPONDING PHIGS ROOT
PHRT = 1000*ICOUNT + 1
C  CLEAR THE ERROR FLAG
IER = 0
C  STOP THE SEARCH
GO TO 200

ENDIF

100  CONTINUE

C  CHECK THE ERROR FLAG
200 IF (IER .EQ. 1) THEN
C  WRITE THE ERROR MESSAGE
CALL ERRMSG ('CMCMRT'; 'GTPHRT - ID IDRE R@l°S AVAILABLE')
ENDIF
RETURN
END

C**********************
C  MODULE FNDMRT
C**********************

C  PROJECT: CADMADEC
C  VERSION: X1.0.0
C  PART : UIE-
C  MODULE :
C  COMPILER: FORTVS2 (FORTRAN 77)
C  CALLING SEQUENCE:
CALL FNDMRT (MENJID, PHRT)
C  INPUT PARAMETERS:
MENJID - INTEGER - MENU STRUCTURE IDENTIFIER
PHRT - INTEGER - PHIGS ROOT STRUCTURE IDENTIFIER
C  OUTPUT PARAMETERS:
C  COMMON INPUTS:
NONE
C  COMMON OUTPUTS:
EXHINST
C  LOCAL VARIABLES:
C  FUNCTIONAL DESCRIPTION:
CADMADEC FIND MENU ROOT STRUCTURE
C  MODULES CALLED:
C  CODED BY: SANKAR JAYARAM
C  DATE: 03/04/89 (09:17:25)

SUBROUTINE FNDMRT (MENJID, PHRT)
    INTEGER MENJID, PHRT
    INTEGER NST, MNPHST(100), MNST(100)
COMMON /EXHNST/ NST, MNPHST, MNST
C ADD THE ROOT MENU AND THE PHIGS STRUCTURE TO THE LIST OF
STRUCTURES WAITING TO BE CREATED AND EXECUTED

NST = NST + 1
MNPHST(NST) = PHRT
MNST(NST) = MENID
RETURN
END

C==================================
C MODULE FNDNST
C==================================
C PROJECT: CADCMADE
C VERSION: X1.0.0
C PART: UIE--
C MODULE:
C==================================
C COMPILER: FORTVS2 (FORTRAN 77)
C==================================
C CALLING SEQUENCE:
C CALL FNDNST (MENID, IER)
C==================================
C INPUT PARAMETERS:
C MENID - INTEGER - MENU STRUCTURE IDENTIFIER
C==================================
C OUTPUT PARAMETERS:
C IER - INTEGER - ERROR INDICATOR
C==================================
C COMMON INPUTS:
C DATSTR
C COMMON OUTPUTS:
C NONE
C==================================
C LOCAL VARIABLES:
C==================================
C FUNCTIONAL DESCRIPTION:
C CADCMADE FIND MENU STRUCTURE FROM THE MENU DATA STRUCTURE
C==================================
C MODULES CALLED:
C==================================
C CODED BY: SANKAR JAYARAM
C DATE: 05/04/89 (09:29:27)
C==================================

SUBROUTINE FNDNST (MENID, IER)
INTEGER MENID, IER
CHARACTER*1 CHAR
INTEGER DATLUN
COMMON /DATSTR/ DATLUN
C RESET ERROR INDICATOR
IER = 1
C REMIND THE MENU DATA STRUCTURE FILE
REMIND(DATLUN)
C SEARCH THE DATA STRUCTURE AND LOCATE THE MENU STRUCTURE
DO 100 ICOUNT = 1, 10000
C READ THE FIRST CHARACTER
READ (DATLUN, '(A)', END=200) CHAR
C CHECK THE CHARACTER AND CONTINUE SEARCH IF IT IS NOT 'S'
IF (CHAR .NE. 'S') GO TO 100
C IF THE CHARACTER IS 'S', CHECK THE MENU STRUCTURE ID
READ (DATLUN, *) MENID
C IF THE MENU IDS MATCH, STOP THE SEARCH
IF (MENID .EQ. MENID) THEN
C CLEAR THE ERROR INDICATOR
IER = 0
50

Appendix C. Code Listing - Prototype UIE
STOP THE SEARCH
GO TO 200
ENDIF
100 CONTINUE
200 CONTINUE
RETURN
END

C******************************************************************************
C MODULE CRPHRT
C******************************************************************************
C PROJECT: CADMADE
C VERSION: X1.0.0
C PART: UIE
C MODULE: CRPHRT
C******************************************************************************
C CALLING SEQUENCE:
C CALL CRPHRT (MENJID, PHRT)
C******************************************************************************
C INPUT PARAMETERS:
C MENJID - INTEGER - MENU STRUCTURE IDENTIFIER
C******************************************************************************
C OUTPUT PARAMETERS:
C PHRT - INTEGER - PHIGS ROOT STRUCTURE IDENTIFIER
C******************************************************************************
C COMMON INPUTS:
C NONE
C******************************************************************************
C COMMON OUTPUTS:
C NONE
C******************************************************************************
C LOCAL VARIABLES:
C******************************************************************************
C FUNCTIONAL DESCRIPTION:
C CADMADE CREATE THE PHIGS ROOT STRUCTURE FOR THE ROOT MENU
C******************************************************************************
C MODULES CALLED:
C******************************************************************************
C CODED BY: SANKAR JAYARAM
C DATE: 03/04/89 (10:06:52)
C******************************************************************************

SUBROUTINE CRPHRT (MENJID, PHRT)

INTEGER MENJID, PHRT
INTEGER ROOT, VIEMID
PARAMETER (VIEMID = 0)

CALL THE PHIGS ROOT STRUCTURE FOR THE MENU ROOT
ROOT = (PHRT -1)*10

PHIGS OPEN STRUCTURE FOR THE PHIGS ROOT THAT WILL BE POSTED
CALL POPST (ROOT)

SET THE VIEM ID FOR THE ROOT
CALL PSVHM (VIEMID)

SET THE PICK ID FOR THE MENU ROOT
CALL PSHPID (MENJID)

EXECUTE THE ROOT MENU PHIGS STRUCTURE
CALL PEXT (PHRT)

CLOSE THE ROOT
CALL PCLST

FILL ALL THE MENU STRUCTURES
CALL FAMNST

RETURN
END

C******************************************************************************
C MODULE FAMNST
C******************************************************************************

Appendix C. Code Listing - Prototype UIE 227
SUBROUTINE FAINST
   INTEGER PHSTR, PENJID
   INTEGER NST, MINPHST(100), MNST(100)
   COMMON /EXPINST/ NST, MINPHST, MNST

C CHECK THE WAITING LIST TO SEE IF ANY MENU STRUCTURES ARE WAITING
IF (NST .NE. 0) THEN
C POP THE NEXT WAITING MENU STRUCTURE
   CALL POPENS (PHSTR, MENJID)
C POP THE MENU ATTRIBUTES FOR THE STRUCTURE
   CALL POPMAT
C FILL THE MENU STRUCTURE
   CALL FAINST (PHSTR, MENJID)
C CHECK THE WAITING LIST AGAIN
   IF (NST .NE. 0) GO TO 50
ENDIF
RETURN
END

MODULE POPEMS

SUBROUTINE FAINST
   INTEGER PHSTR, MENJID
   INTEGER NST, MINPHST(100), MNST(100)
   COMMON /EXPINST/ NST, MINPHST, MNST

C CHECK THE WAITING LIST TO SEE IF ANY MENU STRUCTURES ARE WAITING
IF (NST .NE. 0) THEN
C POP THE NEXT WAITING MENU STRUCTURE
   CALL POPENS (PHSTR, MENJID)
C POP THE MENU ATTRIBUTES FOR THE STRUCTURE
   CALL POPMAT
C FILL THE MENU STRUCTURE
   CALL FAINST (PHSTR, MENJID)
C CHECK THE WAITING LIST AGAIN
   IF (NST .NE. 0) GO TO 50
ENDIF
RETURN
END
SUBROUTINE POPEMS (PHSTR, MENUID)

INTEGER PHSTR, MENUID
INTEGER NST, MNPHST(100), MNST(100)

COMMON /EXINST/ NST, MNPHST, MNST

C PULL OUT THE FIRST MENU STRUCTURE
PHSTR = MNPHST(1)
MENUID = MNST(1)

C MOVE THE QUEUE FORWARD
DO 100 I = 1, NST-1

C MOVE THE MENUS AND PHIGS STRUCTURES
MNPHST(I) = MNPHST(I+1)
MNST(I) = MNST(I+1)

100 CONTINUE

C DECREMENT THE NUMBER OF WAITING MENU STRUCTURES
NST = NST - 1
RETURN
END

MODULE POPMAT

PROJECT: CADMADE
VERSION: X1.0.0
PART : UIE-

COMPIILER: FORTVS2 (FORTRAN 77)

CALLING SEQUENCE:
CALL POPMAT

INPUT PARAMETERS:

OUTPUT PARAMETERS:

COMMON INPUTS:
MINATT, MINATT1

LOCAL VARIABLES:

FUNCTIONAL DESCRIPTION:
CADMADE POP MENU ATTRIBUTES FOR A MENU STRUCTURE

MODULES CALLED:

Coded BY: SANKAR JAYARAH
DATE: 03/04/89 (10:18:01)
SUBROUTINE POPMAT

INTEGER PHSTR, MENUID

C START OF COMMON BLOCK FOR SAVING MENU ATTRIBUTES

INTEGER NATT

INTEGER MAXN, MAXITM

PARAMETER (MAXN = 50, MAXITM = 20)

INTEGER TPATH(MAXN), THAL(MAXN), TVAL(MAXN), TCI(MAXN),
 TPONT(MAXN), TIND(MAXN), TPRECI(MAXN), TBFLAG(MAXN),
 TBSI(MAXN), TOLT(MAXN),

IPATH(MAXN), IHAL(MAXN), IVAL(MAXN), ICI(MAXN),
 IFONT(MAXN), IIND(MAXN), IPREC(MAXN), IBFLAG(MAXN),
 IBSI(MAXN), IBTI(MAXN),

MPATH(MAXN), MPATI(MAXN), TSPACE(MAXN), TEXPAN(MAXN), TUPX(MAXN), TUPY(MAXN),
 TION(MAXN), TXLOC(MAXN), TYLOC(MAXN),

IH(MAXN), ISPSPACE(MAXN), IEKSPAN(MAXN), IUPX(MAXN), IUPY(MAXN),
 IOM(MAXN), IXLOC(MAXITM, MAXN), IYLOC(MAXITM, MAXN)

REAL THT(MAXN), TSPACE(MAXN), TEXPAN(MAXN), TUPX(MAXN), TUPY(MAXN),
 TION(MAXN), TXLOC(MAXN), TYLOC(MAXN),

IHT(MAXN), ISPACE(MAXN), IEXPAN(MAXN), IUPX, IUPY MAXN),
 IOM(MAXN), IXLOC(MAXITM, MAXN), IYLOC(MAXITM, MAXN)

CHARACTER*80 PRGPT(MAXN)

COMMON /MNATT/ NATT,
 TPATH, THAL, TVAL, TCI, TFONT, TIND, TPRECI, TBFLAG,

IPATH, IHAL, IVAL, ICI, IFONT, IIND, IPREC, IBFLAG,
 IBSI, IBTI, IXLOC, IYLOC,

COMMON /MPROMPT/ PROMPT(MAXN)

*****************************************************************************
C END OF COMMON BLOCK
*****************************************************************************

IF (NATT .EQ. 1) RETURN

C MOVE THE QUEUE FORWARD
DO 100 I = 1, NATT-1

C MOVE THE ATTRIBUTES
TPATH(I) = TPATH(I+1)
THAL(I) = THAL(I+1)
TVAL(I) = TVAL(I+1)
TCI(I) = TCI(I+1)
TFONT(I) = TFONT(I+1)
TIND(I) = TIND(I+1)
TPRECI(I) = TPRECI(I+1)
TBFLAG(I) = TBFLAG(I+1)
TOFLAG(I) = TOFLAG(I+1)
TBCI(I) = TBCI(I+1)
TCII(I) = TCII(I+1)
TBSCI(I) = TBSCI(I+1)
TBSI(I) = TBSI(I+1)
TOLT(I) = TOLT(I+1)

IPATH(I) = IPATH(I+1)
IHAL(I) = IHAL(I+1)
IVAL(I) = IVAL(I+1)
ICI(I) = ICI(I+1)
IFONT(I) = IFONT(I+1)
IIND(I) = IIND(I+1)
IPREC(I) = IPREC(I+1)
IBFLAG(I) = IBFLAG(I+1)
IBSCI(I) = IBSCI(I+1)
IBCI(I) = IBCI(I+1)
IBCI(I) = IBCI(I+1)
IBI(I) = IBSI(I+1)
IBSII(I) = IBSI(I+1)
IOLT(I) = IOLT(I+1)

DLOGIDI(I) = DLOGIDI(I+1)
PRMTLN(I) = PRMTLN(I+1)
DECLARE
  IHT(I) = IHT(I+1)
  TSPACE(I) = TSPACE(I+1)
  TEXPAN(I) = TEXPAN(I+1)
  TUPX(I) = TUPX(I+1)
  TUPY(I) = TUPY(I+1)
  TOM(I) = TOM(I+1)
  TXLOC(I) = TXLOC(I+1)
  TYLOC(I) = TYLOC(I+1)
  INT(I) = INT(I+1)
  ISPAC(I) = ISPAC(I+1)
  IEXPAN(I) = IEXPAN(I+1)
  IUPX(I) = IUPX(I+1)
  IUPY(I) = IUPY(I+1)
  IOM(I) = IOM(I+1)

PROMPT(I) = PROMPT(I+1)

DO 50 ITEM = 1, MAXTH
  TXLOC(ITEM, I) = TXLOC(ITEM, I+1)
  TYLOC(ITEM, I) = TYLOC(ITEM, I+1)
50  CONTINUE

C DECREMENT THE NUMBER OF WAITING ATTRIBUTE LISTS
NATT = NATT - 1

RETURN
END

C=====================================================================
C MODULE FLMNST
C=====================================================================

C PROJECT: CADMADE
C VERSION: X1.0.0
C PART : UIE
C MODULE : 
C=====================================================================
C CALLER'S SEQUENCE:
C CALL FLMNST (PHSTR, MENUID)
C=====================================================================
C INPUT PARAMETERS:
C PHSTR - INTEGER - PHIGS STRUCTURE IDENTIFIER
C MENUID - INTEGER - MENU STRUCTURE IDENTIFIER
C=====================================================================
C OUTPUT PARAMETERS:
C=====================================================================
C COMMON INPUTS:
C NONE
C=====================================================================
C COMMON OUTPUTS:
C NONE
C=====================================================================
C LOCAL VARIABLES:
C=====================================================================
C FUNCTIONAL DESCRIPTION:
C CADMADE FILL MENU PHIGS STRUCTURE
C=====================================================================
C MODULES CALLED:
C=====================================================================
C CODED BY: SANKAR JAYARAM
C DATE: 03/04/89 (10:23:52)
C=====================================================================

SUBROUTINE FLMNST (PHSTR, MENUID)
  INTEGER PHSTR, MENUID
  INTEGER VIENID, INVIS, VIS, NAMSET(2)
  PARAMETER (VIENID = 0, INVIS = 126, VIS = 127)

  C FIND THE MENU STRUCTURE IN THE DATA STRUCTURE
  CALL FNDHST (MENUID, IER)

  IF (IER .EQ. 0) THEN
C OPEN THE PHIGS STRUCTURE
CALL POPST (PHSTR)
C SET THE VIEW ID FOR THE PHIGS STRUCTURE
CALL PSVHI (VIEWID)
C SET THE MENU ID AS AN APPLICATIONS DATA IN THE PHIGS STRUCTURE
CALL STIPS (MENJID)
C REMOVE BOTH THE NAMES USED FOR MENUS FROM THE SET
NAMSET(1) = INVIS
NAMSET(2) = VIS
CALL PRES (2, NAMSET)
C ADD THE NAME FOR INVISIBILITY TO THE SET
CALL PSDAS (1, INVIS)
C FILL ALL ATTRIBUTES, PRIMITIVES ETC. FOR THE MENU STRUCTURE
CALL FAPMST (MENJID, PHSTR)
C CLOSE THE PHIGS STRUCTURE
CALL PCST
ENDIF
RETURN
END

C MODULE STIPS
C PROJECT: CADMADE
C VERSION: X1.0.0
C PART : UIE—__
C COMPILER: FORTVS2 (FORTRAN 77)
C CALLING SEQUENCE:
C CALL STIPS (MENJID)
C INPUT PARAMETERS:
MENJID - INTEGER - MENU STRUCTURE IDENTIFIER
C OUTPUT PARAMETERS:
C COMMON INPUTS:
NONE
C COMMON OUTPUTS:
NONE
C LOCAL VARIABLES:
LDR - INTEGER, APPLICATION DATA RECORD LENGTH
DATARC(LDR) - INTEGER, APPLICATION DATA RECORD
C FUNCTIONAL DESCRIPTION:
CADMADE SET MENU ID AS APPLICATIONS DATA IN PHIGS STRUCTURE
C MODLES CALLED:
C Coded BY: SANKAR JAYARAM
C DATE: 03/04/89 (10:42:47)
C
SUBROUTINE STIPS (MENJID)
INTEGER MENJID
INTEGER LDR, LDORYT
PARAMETER (LDR = 5, LDORYT = LDR * 6)
INTEGER DATARC(LDR)
INTEGER PEMNI
PARAMETER (PEMIN = 51)
DATA DATARC / 2, 0, 0, PEMNI, 0 /
C PUT VIEW INDEX INTO DATA RECORD
DATARC(5) = MENJID

Appendix C. Code Listing - Prototype UIE

232
C INSERT APPLICATION DATA INTO STRUCTURE
CALL PAP (LDRBYT, DATARC)
RETURN
END

C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C MODULE FAPMST
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C PROJECT: CADMADE
C VERSION: X1.0.
C PART : UIE--
C MODULE :
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C COMPILER: FORTVS2 (FORTRAN 77)
C CALLING SEQUENCE:
C CALL FAPMST (MENUID, PHSTR)
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C INPUT PARAMETERS:
C MENUID - INTEGER - MENU STRUCTURE IDENTIFIER
C PHSTR - INTEGER - PHIGS STRUCTURE IDENTIFIER
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C COMMON INPUTS:
C DATSTR, MINATT, MINATT1, MINPR1, MINPR2
C COMMON OUTPUTS:
C NONE
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C LOCAL VARIABLES:
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C FUNCTIONAL DESCRIPTION:
CADMADE FILL ATTRIBUTES, PRIMITIVES, ETC. IN A MENU PHIGS STRUCTURE
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C MODULES CALLED:
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C CODED BY: SANJAR JAYARAH
C DATE: 03/04/89 (10:49:13)
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

SUBROUTINE FAPMST (MENUID, PHSTR)
INTEGER MENUID, PHSTR
INTEGER ELEHID
INTEGER DATLUN
COMMON /DATSTR/ DATLUN

******************************************************************************
C START OF COMMON BLOCK FOR SAVING MENU ATTRIBUTES
******************************************************************************
INTEGER NATT
INTEGER MAXN, MAXITM
PARAMETER (MAXN = 50, MAXITM = 20)
INTEGER TPATH(MAXN), THAL(MAXN), TVAL(MAXN), TCI(MAXN),
  > TFLAG(MAXN), TBPR(MAXN), TCIC(MAXN), TBSI(MAXN),
  > TSI(MAXN), TOLT(MAXN),
  > IPATH(MAXN), IHAL[MAXN], IVAL(MAXN), ICIC(MAXN),
  > IFONT(MAXN), INDI(MAXN), IPREC(MAXN), IBFLAG(MAXN),
  > IOFLAG(MAXN), IBIC(MAXN), IOCI(MAXN), IBSI(MAXN),
  > IBSI(MAXN), IOLT(MAXN),
  > DLOGI(MAXN), PRMTLN(MAXN)
REAL THT(MAXN), TSPACE(MAXN), TEXPAN(MAXN), TUPX(MAXN), TUPY(MAXN),
  > TOD(MAXN), TLOC(MAXN), TLOC(MAXN),
  > IHT(MAXN), ISPACE(MAXN), IEXPAN(MAXN), IUPX(MAXN), IUPY(MAXN),
  > IOMAXN, IXLOC(MAXITM, MAXN), IYLOC(MAXITM, MAXN)
CHARACTER*80 PROMPT(MAXN)
COMMON /MINUATT/ NATT,
CEND OF COMMON BLOCK

C START OF COMMON BLOCK FOR SAVING MENU PROMPT PARAMETERS

INTEGER NMMENS
PARAMETER (NMMENS=200)

INTEGER MDGID(NMMENS), MPRMTL(NMMENS)
CHARACTER*80 MPRMTL(MDID(MMMENS))

COMMON /MNUPR/ MDGID, MPRMTL
COMMON /MNUPR2/ MPRMTL

CEND OF COMMON BLOCK

C READ THE NEXT ELEMENT IN THE DATA STRUCTURE
50 READ (DATLUN, *) ELEMID

C CALL THE APPROPRIATE ROUTINE FOR THE STRUCTURE ELEMENT

IF (ELEMID .EQ. 1001) THEN
C CALL FILL MENU EXECUTE SUB-STRUCTURE
CALL FEXMST (MDGID, PHSTR)
GO TO 50

ELSEIF (ELEMID .GE. 1010 .AND. ELEMID .LE. 1019) THEN
C CALL FILL MENU PRIMITIVES
CALL FPRMST (ELEMID)
GO TO 50

ELSEIF (ELEMID .GE. 1020 .AND. ELEMID .LE. 1050) THEN
C CALL FILL MENU PRIMITIVE LOCATIONS
CALL FLMST (ELEMID)
GO TO 50

ELSEIF (ELEMID .GE. 1050 .AND. ELEMID .LE. 1099) THEN
C CALL FILL MENU ATTRIBUTES
CALL FAMST (MDGID, ELEMID)
GO TO 50

ELSEIF (ELEMID .EQ. 2005) THEN
C CALL FILL MENU LABEL
* CALL FLBMS
GO TO 50

ELSEIF (ELEMID .EQ. 2001) THEN
C STORE THE MENU DIALOG ID IN THE COMMON BLOCK
MDGID(MDID(MMMENS)) = DLOGID(1)
C STORE THE MENU PROMPT IN THE COMMON BLOCK
MPRMTL(MDID(MMMENS)) = PRMTLN (1)
MPRMTL(MDID(MMMENS)) = PROMPT (1)
C RETURN FROM THE DATA STRUCTURE
RETURN
ELSE
GO TO 50
ENDIF
END
MODULE FEXMST

PROJECT: CADMADE
VERSION: X1.0.0
PART : UIE — —
MODULE 2 __
COMPILER: FORTVS2 (FORTRAN 77)

CALLING SEQUENCE:
CALL FEXMST (MENUID, PHSTR)

INPUT PARAMETERS:
MENUID - INTEGER - MENU STRUCTURE IDENTIFIER
PHSTR - INTEGER - PHIGS STRUCTURE IDENTIFIER

OUTPUT PARAMETERS:

COMMON INPUTS:
DATSTR, MINUATT, MINUATI, NXTSTR, EXMNST

COMMON OUTPUTS:
DATSTR, MINUATT, NXTSTR, EXMNST

LOCAL VARIABLES:

FUNCTIONAL DESCRIPTION:
CADMADE FILL EXECUTE MENU SUB STRUCTURE

MODULES CALLED:

CODED BY: SANKAR JAYARAH
DATE: 05/05/89 (12:14:33)

INTEGER PHSTR, MENUID
INTEGER MINUATI, NXTSTR
COMON /PHGSTR/ NXTSTR
INTEGER DATSTR, DATLUN
COMMON /DATSTR/ DATLUN
INTEGER NST, MMHSTR(100), MNST(100)
COMMON /EXMNST/ NST, MMHST, MNST

START OF COMMON BLOCK FOR SAVING MENU ATTRIBUTES

INTEGER NATT

PARAMETER (MAXN = 50, MAXITH = 20)

INTEGER TPATH(MAXN), TVAL(MAXN), TCI(MAXN),
> TFLAG(MAXN), TINDI(MAXN), TPRECI(MAXN), TBFLAG(MAXN),
> NXTSTR, MINUATI, TCI(MAXN), TOCI(MAXN), TBSI(MAXN),
> TBSI(MAXN), TOLI(MAXN),
> IPATH(MAXN), IVAL(MAXN), ICN(MAXN),
> IFONT(MAXN), IINDI(MAXN), IPRECI(MAXN), IBFLAG(MAXN),
> IFLAG(MAXN), IBICI(MAXN), IOCI(MAXN), IBSI(MAXN),
> IBSI(MAXN), IOLI(MAXN),
> DLOGI(MAXN), PRMHTN(MAXN)

REAL THTI(MAXN), TSPECE(MAXN), TEXPAN(MAXN), TUPX(MAXN), TUPY(MAXN),
> TONI(MAXN), TXLOC(MAXN), TYLOC(MAXN),
> IHTI(MAXN), ISPACE(MAXN), IXPAN(MAXN), IUPX(MAXN), IUPY(MAXN),
> IONI(MAXN), IXLOC(MAXN), IYLOC(MAXN), IYLOC(MAXIYTH, MAXN)

CHARACTER*80 PROMPT(MAXN)

COMMON /MINUATT/ NATT,
C INCREMENT THE NUMBER OF WAITING ATTRIBUTE LISTS
NATT = NATT + 1

C SAVE THE ATTRIBUTES

TPATH (NATT) = TPATH (NATT-1)
THAL (NATT) = THAL (NATT-1)
TVAL (NATT) = TVAL (NATT-1)
TCI (NATT) = TCI (NATT-1)
TFONT (NATT) = TFONT (NATT-1)
TIND (NATT) = TIND (NATT-1)
TPREC (NATT) = TPREC (NATT-1)
TBFLAG (NATT) = TBFLAG (NATT-1)
TOFLAG (NATT) = TOFLAG (NATT-1)
TIMO (NATT) = TIMO (NATT-1)
TBSI (NATT) = TBSI (NATT-1)
TOLT (NATT) = TOLT (NATT-1)

IPATH (NATT) = IPATH (NATT-1)
IHAL (NATT) = IHAL (NATT-1)
IVAL (NATT) = IVAL (NATT-1)
ICI (NATT) = ICI (NATT-1)
IFONT (NATT) = IFONT (NATT-1)
IIND (NATT) = IIND (NATT-1)
IPREC (NATT) = IPREC (NATT-1)
IBFLAG (NATT) = IBFLAG (NATT-1)
IBCI (NATT) = IBCI (NATT-1)
IOCI (NATT) = IOCI (NATT-1)
IBS (NATT) = IBS (NATT-1)
IBSI (NATT) = IBSI (NATT-1)
IOLT (NATT) = IOLT (NATT-1)

DLOGID (NATT) = DLOGID (NATT-1)
PRMTLM (NATT) = PRMTLM (NATT-1)
THT (NATT) = THT (NATT-1)
TSPACE (NATT) = TSPACE (NATT-1)
TEXPAN (NATT) = TEXPAN (NATT-1)
TUPX (NATT) = TUPX (NATT-1)
TUPY (NATT) = TUPY (NATT-1)
TM (NATT) = TM (NATT-1)
TXLOC (NATT) = TXLOC (NATT-1)
TYLOC (NATT) = TYLOC (NATT-1)
INT (NATT) = INT (NATT-1)
ISPACE (NATT) = ISPACE (NATT-1)
IEXPAN (NATT) = IEXPAN (NATT-1)
IUPX (NATT) = IUPX (NATT-1)
IUPY (NATT) = IUPY (NATT-1)
IDM (NATT) = IDM (NATT-1)

PROMPT (NATT) = PROMPT (NATT-1)

DO 50 ITEM = 1, MAXN

IXLoc(ITEM, NATT) = IXLoc(ITEM, NATT-1)
IYLoc(ITEM, NATT) = IYLoc(ITEM, NATT-1)

50 CONTINUE

100 CONTINUE

C SET THE SUB MENU IDENTIFIER FROM THE MENU DATA STRUCTURE
READ (DATLIN, *) MENMNU

C CALCULATE THE NEW PHIGS STRUCTURE
NXTSTR = NXTSTR + 1

C SET THE PICK IDENTIFIER FOR THE MENU SUB STRUCTURE
CALL PSHPID (NEHMNU)

C EXECUTE THE MENU SUB STRUCTURE
CALL PEXST (NXTSTR)

C ADD THE NEW STRUCTURE TO THE LIST OF MENU STRUCTURES WAITING TO
BE EXECUTED
NST = NST + 1
MNPHST(NST) = NXTSTR
MNST(NST) = NEHMNU

RETURN
END

C=====================================================================
C MODULE FPRMST
C=====================================================================
C PROJECT: CADMATE
C VERSION: X1.0.0
C PART : UIE--_
C MODULE: 
C=====================================================================
C COMPILER: FORTVS2 (FORTRAN 77)
C=====================================================================
C CALLING SEQUENCE:
C CALL FPRMST (ELEMID)
C=====================================================================
C INPUT PARAMETERS:
C ELEMID - INTEGER - ELEMENT IDENTIFIER FOR STRUCTURE ELEMENT
C=====================================================================
C OUTPUT PARAMETERS:
C Coded By: SANKAR JAYARAM
C Date: 03/05/89 (15:20:55)
C=====================================================================
SUBROUTINE FPRMST (ELEMID)
INTEGER ELEMID
INTEGER INT1, NPTBFA
REAL XBFA(20), YBFA(20)
CHARACTER TITLE*20, ITEM1*20
INTEGER NITEMS, ILEPFA(10), NMNUAT1(10)
REAL IXOFPA(50), IYBFA(50)
CHARACTER ITEMS(10)*20
REAL XFA(20), YFA(20)
INTEGER DATLUN
COMMON /DATSTR/, DATLUN

**************************************************************************
C START OF COMMON BLOCK FOR SAVING MENU ATTRIBUTES
**************************************************************************
INTEGER NATT

INTEGER MAXN, MAXTHM
PARAMETER (MAXN = 50, MAXTHM = 20)

INTEGER TPATH(20), THAL(MAXN), TVAL(MAXN), TCI(MAXN),
> TFOV(TFVI(MAXN), TINDI(MAXN), TPREC(MAXN), TFLAG(MAXN),
> TOFLAG(MAXN), TBSI(MAXN), TOCI(MAXN), TBSI(MAXN),
C PROCESS THE TITLE PRIMITIVE

IF (ELEMID .EQ. 1010 .OR. ELEMID .EQ. 1012) THEN

C READ THE INTEGER PARAMETERS
READ(DATLIN*,#) INT1, NPTBFA

C READ THE FILL AREA COORDINATES
READ(DATLIN,1001) (XBFA(I), I=1,NPTBFA)
READ(DATLIN,1001) (YBFA(I), I=1,NPTBFA)

50 CONTINUE

C SET THE PICK IDENTIFIER
CALL PSHPID(-1)

C CHECK THE BACKGROUND FLAG
IF (TBFLAG(1) .NE. 0 .AND. NPTBFA .GT. 2) THEN

C DUMP ALL THE FILL AREA ATTRIBUTES
CALL PSEDFG (TOFLAG(1))
CALL PSICI (TBCI(1))
CALL PSEDCI (TOCI(1))
CALL PSIS (TBSI(1))
CALL PSISI (TBSI(1))
CALL PSEDT (TOLT(1))

CALL GPESC (TOM(1))

C CALCULATE THE ABSOLUTE COORDINATES OF THE FILL AREA VERTICES
DO 75 IFA = 1, NPTBFA
   XBFA(IFA) = XBFA(IFA) + TXLOC(IFA)
   YBFA(IFA) = YBFA(IFA) + TYLOC(IFA)
75 CONTINUE

C DRAM THE FILL AREA
CALL PFA (NPTBFA, XBFA, YBFA)
ENDF

C IF IT IS A TITLE TEXT, READ THE TITLE
IF (ELEMID .EQ. 1010) THEN
   READ (DATLIN, 'A') TITLE
C DUMP ALL THE TEXT ATTRIBUTES
CALL PSTXP (TPATH(1))
CALL PSTXAL (THAL(1), TVAL(1))
CALL PSTXCI (TCI(1))
CALL PSTXFN (TFONT(1))
CALL PSTXI (TIND(1))
CALL PSTXPR (TPREC(1))
# CALL PSCHSP (TSPACE(1))
CALL PSCHXP (TEXPAN(1))
CALL PSCHUP (TUPX, TUPY(1))

C DRAM THE TITLE TEXT
CALL PTXS (TXLOC(1), TYLOC(1), INT1, TITLE)
ENDIF
ENDIF
C PROCESS THE ITEMS PRIMITIVE

IF (ELEMID .EQ. 1014 .OR. ELEMID .EQ. 1016) THEN
C READ THE INTEGER PARAMETERS
READ (DATLUN, *) NITEMS
READ (DATLUN, 1002) (NLENST(I), I=1,NITEMS)
READ (DATLUN, 1002) (NPBFA(I), I=1,NITEMS)
READ (DATLUN, 1002) (MENUID(I), I=1,NITEMS)
C CALCULATE THE TOTAL NUMBER OF POINTS FOR THE BACKGROUND
NTOT = 0
DO 100 I = 1, NITEMS
NTOT = NTOT + NPBFA(I)
100 CONTINUE
C READ THE REAL PARAMETERS
READ (DATLUN, 1001) (IXBFA(I), I=1,NTOT)
READ (DATLUN, 1001) (IYBFA(I), I=1,NTOT)
C READ THE CHARACTER PARAMETERS
DO 200 I = 1, NITEMS
READ (DATLUN, '(A)') ITEMS(I)
200 CONTINUE
C CHECK THE BACKGROUND FLAG
IF (IBFLAG(1) .NE. 0) THEN
C DUMP ALL THE FILL AREA ATTRIBUTES
CALL PSEDFG (IOFLAG(1))
CALL PSICI (IBCI(1))
CALL PSCICI (IOC(1))
CALL PSIS (IBSI(1))
CALL PSISI (IBSI(1))
CALL PSEDT (IOLT(1))

************ GRAPHICS CALL
CALL GPESC (ION(1))

************ ENDIF
C IF THE PRIMITIVE IS AN ITEM TEXT, SET THE ITEM TEXT ATTRIBUTES
IF (ELEMID .EQ. 1014) THEN
C DUMP ALL THE TEXT ATTRIBUTES
CALL PSTXP (IPATH(1))
CALL PSTXAL (IHAL(1), IVAL(1))
CALL PSTXCI (ICX(1))
CALL PSTXFN (IFONT(1))
CALL PSTXI (TIND(1))
CALL PSTXPR (TPREC(1))
# CALL PSCHSP (TSPACE(1))
CALL PSCHXP (TEXPAN(1))
CALL PSCHUP (TUPX, TUPY(1))
ENDIF
C DRAM EACH MENU ITEM
C RESET THE NUMBER OF BACKGROUND POINTS USED
NPOINT = 1
DO 300 ITEM = 1, NITEMS
C
SET THE PICK IDENTIFIER
CALL PSHPID(MENUID(ITEM))
C
DRAM THE FILL AREA
IF (NPBFA(ITEM) .GT. 0 .AND. IBFLAG(1) .NE. 0) THEN
   DO 400 IPOINT = 1, NPBFA(ITEM)
   C
   CALCULATE THE ABSOLUTE COORDINATES OF THE POINTS
   XFA(IPOINT) = IXBFA(IPOINT)+IXLOC(ITEM,1)
   YFA(IPOINT) = IYBFA(IPOINT)+IYLOC(ITEM,1)
   C
   INCREASE THE NUMBER OF POINTS USED
   NPOINT = NPOINT + 1
400    CONTINUE
C
DRAM THE FILL AREA
CALL PFA (NPBFA(ITEM), XFA, YFA)
ENDIF
C
IF IT IS AN ITEM TEXT, DRAM THE ITEM
IF (ELEMID .EQ. 1014) THEN
   ITEM1 = ITEMS(ITEM)
C
DRAM THE ITEM TEXT
CALL PTXS (IXLOC(ITEM,1),IYLOC(ITEM,1), ILENB(ITEM), ITEMS(ITEM))
ENDIF
300    CONTINUE
ENDIF
1001 FORMAT(6E12.5)
1002 FORMAT(6I12)
RETURN
END
C****************************************************************************
C   M O D U L E   F A T H S T
C*****************************************************************************
C   P R O J E C T:   C A D M A D E
C   V E R S I O N:   X 1 . 0 . 0
C   P A R T :   U I E--
C   M O D U L E :   __
C*****************************************************************************
C   C O M P I L E R:   F O R T V S 2   ( F O R T R A N 7 7 )
C*****************************************************************************
C   C A L L I N G   S E Q U E N C E:
C       CALL FATMST (MENUID, ELEMID)
C*****************************************************************************
C   I N P U T   P A R A M E T E R S:
C       MENUID - INTEGER - MENU STRUCTURE IDENTIFIER
C       ELEMID - INTEGER - ELEMENT IDENTIFIER FOR STRUCTURE ELEMENT
C*****************************************************************************
C   O U T P U T   P A R A M E T E R S:
C*****************************************************************************
C   C O M M O N   I N P U T S:
C       DATSTR
C*****************************************************************************
C   C O M M O N   O U T P U T S:
C       MNUATT, MNUAT1
C*****************************************************************************
C   L O C A L   V A R I A B L E S:
C*****************************************************************************
C   F U N C T I O N A L   D E S C R I P T I O N:
C       CADMADE FILL MENU ATTRIBUTES IN ATTRIBUTE COMMON BLOCK
C*****************************************************************************
C   M O D U L E S   C A L L E D:
SUBROUTINE FATHST (MENUID, ELEMID)

INTEGER MENUID, ELEMID

INTEGER DATLUN
COMMON /DATSTR/ DATLUN

C START OF COMMON BLOCK FOR SAVING MENU ATTRIBUTES

INTEGER NATT

PARAMETER (MAXN = 50, MAXTH = 20)

INTEGER TPATH(MAXN), THAL(MAXN), TVAL(MAXN), TCI(MAXN),
   TFONT(MAXN), TIND(MAXN), TPREC(MAXN), TBFLAG(MAXN),
   TBSI(MAXN), TOLT(MAXN),
   TPATHIHAXNM, THALIHAXNM, TVALIHAXNM, TCIIHAXNM,
   TFINNIHMXNM, TINDIHAXNM, TPINC(IMAXN), TBFLAGIHAXNM,
   TBSIHAXNM, TOLTIHMXNM,
   DLOGIDIHMXNM, PRMTLNHAXNM

REAL THTIHAXNM, TSPACEIHAXNM, TXPANIHMXNM, TUPXIHAXNM, TUPYIHAXNM,
   TCIHAXNM, IXLOCIHAXNM, IXLOCIHAXNM, IXLOCIHAXNM

CHARACTER*80 PRMTIHAXNM

COMMON /MNUATT/ NATT,
   TPATH, THAL, TVAL, TCI, TFONT, TIND, TPREC, TBFLAG,
   TBSI, TOLT,
   TPATHIHAXNM, THALIHAXNM, TVALIHAXNM, TCIIHAXNM,
   TFINNIHMXNM, TINDIHAXNM, TPINC(IMAXN), TBFLAGIHAXNM,
   TBSIHAXNM, TOLTIHMXNM,
   DLOGIDIHMXNM, PRMTLNHAXNM

COMMON /MNUATT/ PROMPT

C END OF COMMON BLOCK

C PROCESS THE MENU TITLE ATTRIBUTE

IF (ELEMID .EQ. 1050) THEN
   READ (DATLUN,*) TPATH (1)
ELSEIF (ELEMID .EQ. 1051) THEN
   READ (DATLUN,*) THT (1)
ELSEIF (ELEMID .EQ. 1052) THEN
   READ (DATLUN,*) TSPACE (1)
ELSEIF (ELEMID .EQ. 1053) THEN
   READ (DATLUN,*) TXPAN (1)
ELSEIF (ELEMID .EQ. 1054) THEN
   READ (DATLUN,*) THAL (1), TVAL (1)
ELSEIF (ELEMID .EQ. 1055) THEN
   READ (DATLUN,*) TCI (1)
ELSEIF (ELEMID .EQ. 1056) THEN
   CALL ERRMSG ('FATHST', 'CHARACTER SET NOT SUPPORTED')

READ (DATLUN,*) JUNK
ELSEIF (ELEMID .EQ. 1057) THEN
  READ (DATLUN,*) TFONT (1)
ELSEIF (ELEMID .EQ. 1058) THEN
  READ (DATLUN,*) TIND (1)
ELSEIF (ELEMID .EQ. 1059) THEN
  READ (DATLUN,*) TPREC (1)
ELSEIF (ELEMID .EQ. 1060) THEN
  READ (DATLUN,*) TUPX (1), TUPY(1)
ELSEIF (ELEMID .EQ. 1061) THEN
  READ (DATLUN,*) TBFLAG (1)
ELSEIF (ELEMID .EQ. 1062) THEN
  READ (DATLUN,*) TOFLAG (1)
ELSEIF (ELEMID .EQ. 1063) THEN
  READ (DATLUN,*) TBCI (1)
ELSEIF (ELEMID .EQ. 1064) THEN
  READ (DATLUN,*) TOCI (1)
ELSEIF (ELEMID .EQ. 1065) THEN
  READ (DATLUN,*) TBS (1)
ELSEIF (ELEMID .EQ. 1066) THEN
  READ (DATLUN,*) TBSI (1)
ELSEIF (ELEMID .EQ. 1067) THEN
  READ (DATLUN,*) TOLT (1)
ELSEIF (ELEMID .EQ. 1068) THEN
  READ (DATLUN,*) TOM (1)
ENDIF
ELSEIF (ELEMID .EQ. 1069) THEN
  READ (DATLUN,*)
ENDIF
ELSEIF (ELEMID .EQ. 1070) THEN
  READ (DATLUN,*) IPATH (1)
ELSEIF (ELEMID .EQ. 1071) THEN
  READ (DATLUN,*) IHT (1)
ELSEIF (ELEMID .EQ. 1072) THEN
  READ (DATLUN,*) ISPACE (1)
ELSEIF (ELEMID .EQ. 1073) THEN
  READ (DATLUN,*) IEXPAN (1)
ELSEIF (ELEMID .EQ. 1074) THEN
  READ (DATLUN,*) IHAL (1), IVAL (1)
ELSEIF (ELEMID .EQ. 1075) THEN
  READ (DATLUN,*) ICI (1)
ELSEIF (ELEMID .EQ. 1076) THEN
  READ (DATLUN,*)
CALL ERRMSG ('FATMST'; 'CHARACTER SET NOT SUPPORTED')
READ (DATLUN,*) JUNK

ELSEIF (ELEMID .EQ. 1077) THEN
  READ (DATLUN,*) IFONT (1)
ELSEIF (ELEMID .EQ. 1078) THEN
  READ (DATLUN,*) IIND (1)
ELSEIF (ELEMID .EQ. 1079) THEN
  READ (DATLUN,*) IPREC (1)
ELSEIF (ELEMID .EQ. 1080) THEN
  READ (DATLUN,*) IUPX (1), IUPY(1)
ELSEIF (ELEMID .EQ. 1081) THEN
  READ (DATLUN,*) IBFLAG (1)
ELSEIF (ELEMID .EQ. 1082) THEN
  READ (DATLUN,*) IIOFLAG (1)
ELSEIF (ELEMID .EQ. 1083) THEN
  READ (DATLUN,*) IBCI (1)
ELSEIF (ELEMID .EQ. 1084) THEN
  READ (DATLUN,*) IBF (1)
ELSEIF (ELEMID .EQ. 1086) THEN
  READ (DATLUN,*) IBSI (1)
ELSEIF (ELEMID .EQ. 1087) THEN
  READ (DATLUN,*) IOLT (1)
ELSEIF (ELEMID .EQ. 1088) THEN
  READ (DATLUN,*) ICM (1)
ELSEIF (ELEMID .EQ. 1089) THEN  
  READ (DATLUN,*)
  CALL PRINTDIAGR (ATTRIBUTES
ELSEIF (ELEMID .EQ. 1091) THEN
  READ (DATLUN,*) DLOGID (1)
ELSEIF (ELEMID .EQ. 1092) THEN
  READ (DATLUN,*) PRMTLN (1)
  READ (DATLUN,*'PRMTLN') PROMPT (1)
ENDIF

RETURN
END

C=====================================================================================================
C MODULE FLCMST
C=====================================================================================================
C PROJECT: CADMADE
C VERSION: X1.0.0
C PART : UIE--
C MODULE :
C=====================================================================================================
C COMPILER: FORTVSZ (FORTRAN 77)

Appendix C. Code Listing - Prototype UIE 243
SUBROUTINE FLCHST (ELEMID)
INTEGER ELEMID
INTEGER DATLUN
COMMON /DATSTR/ DATLUN

C START OF COMMON BLOCK FOR SAVING MENU ATTRIBUTES
C
INTEGER NATT
PARAMETER (MAXN = 50, MAXITM = 20)
INTEGER TPATH(MAXN), THAL(MAXN), TVAL(MAXN), TCI(MAXN),
    TFONT(MAXN), TIND(MAXN), TPREC(MAXN), TBFLAG(MAXN),
    TFLAG(MAXN), TBCI(MAXN), TOCI(MAXN), TBS(MAXN),
    TBSI(MAXN), TLT(MAXN),
    IPATH(MAXN), IHAL(MAXN), IVAL(MAXN), ICII(MAXN),
    IFONT(MAXN), IIND(MAXN), IPREC(MAXN), IBFLAG(MAXN),
    IFLAG(MAXN), IBCI(MAXN), IOCI(MAXN), IBSI(MAXN),
    IBSI(MARK), IOLT(MAXN),
    DLOGI(DMAXN), PRMTLN(MAXN)
REAL THT(MAXN), TSPACE(MAXN), TEXPAN(MAXN), TUPX(MAXN), TUPY(MAXN),
    TOM(MAXN), TXLOC(MAXN), TYLOC(MAXN),
    INT(MAXN), ISPACE(MAXN), IEXPAN(MAXN), IUPX(MAXN), IUPY(MAXN),
    IDM(MAXN), IXLOC(MAXTM, MAXN), IYLOC(MAXTM, MAXN)
CHARACTER*80 PROMPT(MAXN)

COMMON /HNUATT/ NATT,
    TPATH, THAL, TVAL, TCI, TFONT, TIND, TPREC, TBFLAG,
    TFLAG, TBCI, TOCI, TBS, TBSI, TLT,
    IPATH, IHAL, IVAL, ICII, IFONT, IIND, IPREC, IBFLAG,
    IFLAG, IBCI, IOCI, IBSI, IOLT,
    DLOGI, PRMTLN,
    THT, TSPACE, TEXPAN, TUPX, TUPY, TOM, TXLOC, TYLOC,
    IHIT, ISPAC, IUPX, IUPY, IDM, IXLOC, IYLOC
COMMON /HNUATT/ PROMPT

C END OF COMMON BLOCK

C PROCESS THE MENU TITLE LOCATION
IF (ELEMID .EQ. 1020) THEN
    READ (DATLUN,*) TXLOC(1), TYLOC(1)
ELSEIF (ELEMID .EQ. 1030) THEN

READ (DATLUN,*) NUMBER
READ (DATLUN,1001) (IXLOC(I,1), I=1,NUMBER)
READ (DATLUN,1001) (IXLOC(I,1), I=1,NUMBER)
ENDF

1001 FORMAT(6E12.5)
RETURN
END

C=============================================================================
C CADMADE MODULE CMPOMN
C=============================================================================
C PROJECT: CADMADE
C VERSION: X1.0.0
C PART : UIE-
C MODULE :
C=============================================================================
C CALLING SEQUENCE:
C CALL CMPOMN (MKID, MENUID, PRIORi)
C=============================================================================
C INPUT PARAMETERS:
C MKID - INTEGER - WORKSTATION IDENTIFIER
C MENUID - INTEGER - MENU STRUCTURE IDENTIFIER
C PRIORIT - REAL - STRUCTURE PRIORITY
C=============================================================================
C OUTPUT PARAMETERS:
C=============================================================================
C COMMON INPUTS:
C MNPHRT
C=============================================================================
C COMMON OUTPUTS:
C NONE
C=============================================================================
C LOCAL VARIABLES:
C=============================================================================
C FUNCTIONAL DESCRIPTION:
C CADMADE POST MENU ROOT
C=============================================================================
C MODULES CALLED:
C=============================================================================
C CODED BY: SANKAR JAYARAM
C DATE: 03/05/89 (18:48:49)
C=============================================================================
SUBROUTINE CMPOMN (MKID, MENUID, PRIORIT)
INTEGER MKID, MENUID
REAL PRIORIT
INTEGER PHRT, ROOT
INTEGER NRROOTS
PARAMETER (NRROOTS = 9)
INTEGER AVALRT (NRROOTS,2)
COMMON /MNPHRT/ AVALRT
C SET THE ERROR INDICATOR
IER = 1
C FIND THE ROOT WHICH HAS THE MENU
DO 100 IROOT = 1, NRROOTS
   IF (AVALRT(IROOT, 2) .EQ. MENUID) THEN
C RESET THE ERROR INDICATOR
   IER = 0
C STOP THE SEARCH
GO TO 200
ENDF

Appendix C. Code Listing - Prototype UIE 245
CONTINUE

IF (IER .EQ. 0) THEN

C CALCULATE THE STRUCTURE TO BE MODIFIED
PHRT = 1000*ROOT + 1
C CALCULATE THE ROOT TO BE POSTED
ROOT = IROOT*10000
C POST THE ROOT
CALL PPRT (MKID, ROOT, PRIORT)
C DISPLAY THE MENU PROMPT
CALL DSPRMT (MKID, MENUID)
ELSE
C WRITE ERROR MESSAGE
CALL ERRMSG ('CMPOHN', 'MENU ROOT HAS NOT BEEN CREATED')
ENDIF
RETURN
END

SUBROUTINE CMPOHN (MKID, MENUID)
INTEGER MKID, MENUID
INTEGER PHRT, ROOT
INTEGER NROOTS
PARAMETER (NROOTS = 9)
INTEGER AVALRT (NROOTS, 2)
COMMON /MNPHRT/ AVALRT
C SET THE ERROR INDICATOR
IER = 1
C FIND THE ROOT WHICH HAS THE MENU
DO 100 IROOT = 1, NROOTS
100 CONTINUE
IF (AVALRT(IROOT, 2) .EQ. MENJID) THEN
  C    RESET THE ERROR INDICATOR
    IER = 0
  C    STOP THE SEARCH
    GO TO 200
ENDIF

100 CONTINUE

200 IF (IER .EQ. 0) THEN
    C    CALCULATE THE STRUCTURE TO BE MODIFIED
    PHRT = 1000*IROOT + 1
    C    CALCULATE THE ROOT TO BE POSTED
    ROOT = IROOT*10000
    C    OPEN THE STRUCTURE
    CALL POPST (PHRT)
    C    SET THE ELEMENT POINTER
    CALL PSEP (4)
    C    DELETE THE ELEMENT
    CALL PDEL
    C    CHANGE THE NAME TO 126
    CALL PADS (1,126)
    C    CLOSE THE STRUCTURE
    CALL PCLST
    C    UNPOST THE ROOT
    CALL PUPORT (MKID, ROOT)
ELSE
    C    WRITE ERROR MESSAGE
    CALL ERRMSG ('Call GTPNID,'MENU ROOT HAS NOT BEEN CREATED')
ENDIF
    RETURN
END

C*******************************************************************************
C MODU LE G T M I D
C*******************************************************************************
C PROJECT: CADMADE
C VERSION: X1.0.0
C PART : UIE--
C*******************************************************************************
C COMPILER: FORTVS2 (FORTRAN 77)
C*******************************************************************************
C CALLING SEQUENCE:
C CALL GTMID (PHSTR, MENJID)
C*******************************************************************************
C INPUT PARAMETERS:
C PHSTR - INTEGER - PHIGS STRUCTURE IDENTIFIER
C MENJID - INTEGER - MENU STRUCTURE IDENTIFIER
C*******************************************************************************
C OUTPUT PARAMETERS:
C*******************************************************************************
C COMMON INPUTS:
C NONE
C*******************************************************************************
C COMMON OUTPUTS:
C NONE
C*******************************************************************************
C LOCAL VARIABLES:
C*******************************************************************************
C*******************************************************************************
C FUNCTIONAL DESCRIPTION:
C CADMADE GET MENU ID FROM PHIGS STRUCTURE
C*******************************************************************************
C MODULES CALLED:
C*******************************************************************************
SUBROUTINE GTHNID (PHSTR, MENJID)
INTEGER PHSTR, MENJID
INTEGER TYPE, SIZE, APPDAT
INTEGER LDR
PARAMETER (LDR=5, APPDAT=5370)
INTEGER DATARC(LDR)
C OPEN THE STRUCTURE
CALL POPST (PHSTR)
C SET THE ELEMENT POINTER AT THE CORRECT LOCATION
CALL PSEP (2)
C SET THE ELEMENT SIZE AND TYPE
CALL PGEL (IER, TYPE, SIZE)
IF (TYPE .EQ. APPDAT) THEN
C RETRIEVE THE DATA
CALL GPHE (1, SIZE, IER, MLDR, DATARC)
MENJID = DATARC(5)
ELSE
C WRITE ERROR MESSAGE
CALL ERRMSG ('GTHNID', 'ERROR IN MENU PHIGS STRUCTURE')
MENJID = 0
ENDIF
C CLOSE THE STRUCTURE
CALL PCLST
RETURN
END

C====================================================================
C Coded By: SANKAR JAYARAM
C Date: 03/06/89 (09:51:11)
C====================================================================
C PROJECT: CADMADE
C Version: X1.0.0
C Part: UIE_-
C Module:
C Compiler: FORTS2 (FORTRAN 77)
C====================================================================
C Calling Sequence:
C CALL CHFNDF (NCID, PPD, PP)
C====================================================================
C Input Parameters:
C NCID - INTEGER - WORKSTATION IDENTIFIER
C PPD - INTEGER - PICK PATH DEPTH
C PP(3,m)-INTEGER- PICK PATH
C====================================================================
C Output Parameters:
C====================================================================
C Common Inputs:
C NONE
C====================================================================
C Common Outputs:
C NONE
C====================================================================
C Local Variables:
C====================================================================
C Functional Description:
C CADMADE NEXT MENU
C====================================================================
C Modules Called:
C
SUBROUTINE CHPGNS (RKID, PPD, PP)
INTEGER MKID, PPD, PP(3,*)
INTEGER IER
INTEGER ERROR, NSTR, STRLST(20)
INTEGER NERO!
C SET THE CURRENT MENU INVISIBLE
CALL STINVS (PP(1,PPD), 0)
C SET THE ERROR INDICATOR
IER = 1
C IF THE MENU PICKED IS 0, DISPLAY THE ROOT MENU OF THE TREE
IF (PP(2,PPD) .EQ. 0) THEN
C SET THE ROOT VISIBLE
CALL STINVS (PP(1,2), 1)
C RESET THE ERROR INDICATOR
IER = 0
C SET THE MENU ID FOR THE MENU PROMPT
NEMNU = PP(2,1)
ENDIF
C IF THE MENU PICKED IS -1, DISPLAY THE PARENT OF THE CURRENT MENU
IF (PP(2,PPD) .EQ. -1 .AND. PP(1,PPD-1) .LT. 10000) THEN
C SET THE PARENT VISIBLE
CALL STINVS (PP(1,PPD-1), 1)
C RESET THE ERROR INDICATOR
IER = 0
C SET THE MENU ID FOR THE MENU PROMPT
NEMNU = PP(2,PPD-2)
ENDIF
C IF THE MENU PICKED IS > 0, DISPLAY THE NEXT MENU
IF (PP(2,PPD) .GT. 0) THEN
C FIND THE EXECUTED STRUCTURES IN THE PHIGS STRUCTURE
****** GRAPHICS CALL
CALL GRPREX (PP(1,PPD), 1, 20, ERROR, NSTR, STRLST)
C LOOP THROUGH THE EXECUTED STRUCTURES AND FIND THE ONE WHICH
HAS THE MENU ID = THE PICKED MENU ID
DO 100 ISTR = 1, NSTR
C GET THE MENU ID
CALL GMNID (STRLST(ISTR), MENU)
C CHECK THE MENUID IN THE PHIGS SUB STRUCTURE
IF (MENU .EQ. PP(2,PPD)) THEN
C MAKE THE MENU VISIBLE
CALL STINVS (STRLST(ISTR), 1)
C RESET THE ERROR INDICATOR
IER = 0
C SET THE MENU ID FOR THE MENU PROMPT
NEMNU = PP(2,PPD)
C STOP THE SEARCH
GO TO 200
ENDIF
100 CONTINUE
PROCEDURE STINVS (PHSTR, VIS)

INTEGER PHSTR, VIS

OPEN THE STRUCTURE
CALL POPST (PHSTR)

SET THE ELEMENT POINTER
CALL PSEP (4)

DELETE THE ELEMENT
CALL PDEL

CHANGE THE NAME TO 126 FOR INVISIBLE AND 127 FOR VISIBLE
IVIS = 126 + VIS
CALL PADS (1, IVIS)

CLOSE THE STRUCTURE
CALL PCLST

RETURN
END

SUBROUTINE DSPRMT (MKID, NEMNU)

RETURN
END

MODULE STINVS

C PROJECT: CADMADE
C VERSION: X1.0.0
C PART: UIE-
C MODULE:

C CALLING SEQUENCE:
CALL STINVS (PHSTR, VIS)

C INPUT PARAMETERS:
PHSTR - INTEGER - PHIGS STRUCTURE IDENTIFIER
VIS - INTEGER - VISIBILITY FLAG

C OUTPUT PARAMETERS:

C COMMON INPUTS:
NONE

C COMMON OUTPUTS:
NONE

C LOCAL VARIABLES:

C FUNCTIONAL DESCRIPTION:
CADMADE SET MENU PHIGS STRUCTURE INVISIBLE

C MODULES CALLED:

C CODED BY: SANKAR JAYARAM
C DATE: 05/06/89 (23:12:44)

SUBROUTINE DSPRMT (MKID, NEMNU)

RETURN
END

MODULE DSPRMT
SUBROUTINE DSPRT (MKID, MENUID)
INTEGER MKID, MENUID

C START OF COMMON BLOCK FOR SAVING MENU PROMPT PARAMETERS
C
INTEGER NMENUS
PARAMETER (NMENUS=200)

INTEGER HDGID(NMENUS), MPRI(NMENUS)
CHARACTER*80 MPRPT(NMENUS)

C COMMON /MNUP1, HDGID, MPRI/ MPRPT
C COMMON /MNUP2/ MPRPT

C END OF COMMON BLOCK
C

C DISPLAY THE MENU PROMPT
CALL CHGE (MKID, HDGID(MENUID), MPRPT(MENUID))

RETURN
END
SUBROUTINE CMST (MSTRID)

INTEGER MSTRID
INTEGER ELEMID, DATUM
COMMON /DATSTR/ DATUM
PARAMETER (ELEMID = 2000)

OPEN MENU STRUCTURE
WRITE (DATUM, 1001) 'S - OPEN MENU STRUCTURE'

WRITE THE STRUCTURE IDENTIFIER INTO THE DATA STRUCTURE
WRITE (DATUM, M) MSTRID

1001 FORMAT (A24)
RETURN
END

SUBROUTINE CMST (MSTRID)

INTEGER MSTRID
INTEGER ELEMID, DATUM

C PROJECT: CADMADE
C VERSION: X1.0.0
C PART: UIE
C MODULE: ...
C COMPILER: FORTYS2 (FORTRAN 77)
C CALLING SEQUENCE:
C CALL CMST
C INPUT PARAMETERS:
C
C OUTPUT PARAMETERS:
C
COMMON INPUTS:
C DATSTR
COMMON OUTPUTS:
C
LOCAL VARIABLES:
C
FUNCTIONAL DESCRIPTION:
C CADMADE CLOSE MENU STRUCTURE
C MODULES CALLED:
C
CODED BY: SANKAR JAYARAM
C DATE: 02/19/89 (11:46:21)

SUBROUTINE CMST

INTEGER ELEMID, DATUM
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 2001)

WRITE(DATLUN, 1003) ELEMID, ' - CLOSE MENU STRUCTURE'

1003 FORMAT (I4)
RETURN
END

C SUBROUTINE CMMLB (LABEL)
INTEGER LABEL
INTEGER ELEMID, DATLUN
COMMON /DATSTR/ DATLUN
PARAMETER (ELEMID = 2005)
WRITE(DATLUN, *) ELEMID
WRITE THE LABEL IDENTIFIER INTO THE DATA STRUCTURE
WRITE(DATLUN, *) LABEL

RETURN
END
C MODULE : _
C==================================================================================================
C COMPILER: FORTVS2 (FORTRAN 77)
C CALLING SEQUENCE:
C CALL CMIDEL (LABEL)
C==================================================================================================
C INPUT PARAMETERS:
C LABEL - INTEGER - LABEL
C==================================================================================================
C OUTPUT PARAMETERS:
C==================================================================================================
C COMMON INPUTS:
C NONE
C==================================================================================================
C COMMON OUTPUTS:
C NONE
C==================================================================================================
C LOCAL VARIABLES:
C==================================================================================================
C FUNCTIONAL DESCRIPTION:
C 2020 - CADMADE SET ELEMENT POINTER AT LABEL IN MENU STRUCTURE
C==================================================================================================
C MODULES CALLED:
C==================================================================================================
C CODED BY: SANKAR JAYARAM
C DATE: 02/19/89 (11:52:42)
C==================================================================================================
SUBROUTINE CMIDEL (LABEL)
*****THIS SUBROUTINE IS NOT SUPPORTED YET

RETURN
END

C==================================================================================================
C CADMADE MODULE CMIDEL
C==================================================================================================
C PROJECT: CADMADE
C VERSION: X1.0.0
C PART : UIE-__
C MODULE : _
C==================================================================================================
C COMPILER: FORTVS2 (FORTRAN 77)
C CALLING SEQUENCE:
C CALL CMIDEL
C==================================================================================================
C INPUT PARAMETERS:
C C
C==================================================================================================
C OUTPUT PARAMETERS:
C C
C==================================================================================================
C COMMON INPUTS:
C NONE
C==================================================================================================
C COMMON OUTPUTS:
C NONE
C==================================================================================================
C LOCAL VARIABLES:
C==================================================================================================
C FUNCTIONAL DESCRIPTION:
C 2025 - CADMADE DELETE MENU STRUCTURE ELEMENT
C==================================================================================================
C MODULES CALLED:
C==================================================================================================
C CODED BY: SANKAR JAYARAM
C DATE: 02/19/89 (11:52:42)
C==================================================================================================
SUBROUTINE CMIDEL
*****THIS SUBROUTINE IS NOT SUPPORTED YET

Appendix C. Code Listing - Prototype UIE 255
RETURN
END

SUBROUTINE CMDER (EP1, EP2)
*** THIS SUBROUTINE IS NOT SUPPORTED YET

RETURN
END
**FUNCTIONAL DESCRIPTION:**
2035 - CADMADE DELETE MENU STRUCTURE ELEMENTS BETWEEN LABELS

**MODULES CALLED:**

**CODED BY:** SANKAR JAYARAM
**DATE:** 02/19/89 (11:53:42)

**SUBROUTINE CMDLL (LABEL1, LABEL2)**

**DESCRIPTION:** THIS SUBROUTINE IS NOT SUPPORTED YET

RETURN
END

**MODULE FNOLST**

**PROJECT:** CADMADE
**VERSION:** X3.0.0
**PART:** UIE
**MODULE:**

**COMPILED:** FORTVS2 (FORTRAN 77)

**CALLING SEQUENCE:**
CALL FNOLST

**INPUT PARAMETERS:**

**OUTPUT PARAMETERS:**

**COMMON INPUTS:**
DATSTR

**COMMON OUTPUTS:**
NONE

**LOCAL VARIABLES:**

**FUNCTIONAL DESCRIPTION:**
CADMADE - FIND THE LAST ELEMENT IN THE STRUCTURE AND MOVE THE ELEMENT POINTER TO IT

**SUBROUTINE FNOLST**

INTEGER DATLUN
COMMON /DATSTR/ DATLUN
CHARACTER*4 INCHAR

DO 100 ILINE = 1, 1000

C READ THE NEXT ELEMENT IN THE DATA STRUCTURE
READ (DATLUN,'(A)') INCHAR

IF (INCHAR .EQ. '2001') THEN

C IF IT IS A "CLOSE MENU STRUCTURE" BACKSPACE (DATLUN)
C STOP THE SEARCH
GOTO 200

ENDIF

100 CONTINUE

Appendix C. Code Listing - Prototype UIE 257
SUBROUTINE CMDGCH (DGID, HEIGHT)

INTEGER DGID
REAL HEIGHT

C START OF COMMON BLOCK FOR DIALOG AREA ATTRIBUTES

INTEGER MAXN
PARAMETER (MAXN = 5)

INTEGER DCI(MAXN),
> DFONT(MAXN), DIND(MAXN), DPREC(MAXN), DBFLAG(MAXN),
> DOUTL(MAXN), DOCI(MAXN), DBS(MAXN),
> DBSI(MAXN), DOLT(MAXN),
> DNBUFF(MAXN), DNL(MAXN), DNICHAR(MAXN), DMODE(MAXN), DVIS(MAXN)

REAL DHT(MAXN), DSPACE(MAXN), DEXPAN(MAXN),
> DOM(MAXN), DXLOC(MAXN,2), DYLOC(MAXN,2)

COMMON /DLGATT/
> DCI, DFONT, DIND, DPREC, DBFLAG,
> DOUTL, DOCI, DBS, DBSI, DOLT,
> DNBUFF, DNL, DNICHAR, DMODE, DVIS

C END OF COMMON BLOCK

DHT(DGID) = HEIGHT

RETURN
SUBROUTINE CHDGCS (DGID, SPACE)

INTEGER DGID
REAL SPACE

C START OF COMMON BLOCK FOR DIALOG AREA ATTRIBUTES

INTEGER MAXN
PARAMETER (MAXN = 5)

INTEGER DCI(MAXN),
  > DFONT(MAXN), DIND(MAXN), DPREC(MAXN), DBFLAG(MAXN),
  > DFLAG(MAXN), DBCI(MAXN), DOCI(MAXN), DBSI(MAXN),
  > DSB(MAXN), DOLT(MAXN),
  > DBUF(MAXN), DNL(MAXN), DNBCAD(MAXN), DMODE(MAXN), DVIS(MAXN)

REAL DHT(MAXN), DSPACE(MAXN), DEXPAN(MAXN),
  > DOM(MAXN), DLOC(MAXN,2), DYLOC(MAXN,2)

COMMON /DLGATT/
  > DCI, DFONT, DIND, DPREC, DBFLAG,
  > DFLAG, DBCI, DOCI, DBSI, DOLT,
  > DHT, DSPACE, DEXPAN, DOM, DLOC, DYLOC,
  > DBUF, DNL, DNCHAR, DMODE, DVIS

C END OF COMMON BLOCK

RETURN

END
C CALLING SEQUENCE:
  CALL CMDGXP (DGID, EXPANS)

C INPUT PARAMETERS:
  DGID - INTEGER - DIALOG AREA IDENTIFIER
  EXPANS - REAL - CHARACTER EXPANSION FACTOR

C OUTPUT PARAMETERS:

C COMMON INPUTS:
  NONE

C COMMON OUTPUTS:
  DLGATT

C LOCAL VARIABLES:

C FUNCTIONAL DESCRIPTION:
  CADMADE SET DIALOG AREA CHARACTER EXPANSION FACTOR

C MODULES CALLED:

C CODED BY: SANKAR JAYARAM
C DATE: 02/18/89 (19:56:04)

SUBROUTINE CMDGXP (DGID, EXPANS)

INTEGER DGID
REAL EXPANS

*******************************************************************************
C START OF COMMON BLOCK FOR DIALOG AREA ATTRIBUTES
*******************************************************************************

C PARAMETER (MAXN = 5)

INTEGER DCI(MAXN), DFONT(MAXN), DIND(MAXN), DPREC(MAXN), DBFLAG(MAXN),
  DDBL(MAXN), DBCI(MAXN), DOCI(MAXN), DBS(MAXN),
  DBSS(MAXN), DOLT(MAXN),
  DNBUFF(MAXN), DNL(MAXN), DNCHAR(MAXN), DMODE(MAXN), DVIS(MAXN)

REAL DHT(MAXN), DSPACE(MAXN), DEXPAN(MAXN),
  DOM(MAXN), DXLOC(MAXN,2), DYLOC(MAXN,2)

COMMON /DLGATT/,
    > DCI, DFONT, DIND, DPREC, DBFLAG,
    > DDBL, DBCI, DOCI, DBS, DSS, DOLT,
    > DHT, DSPACE, DEXPAN, DOM, DXLOC, DYLOC,
    > DNBUFF, DNL, DNCHAR, DMODE, DVIS

*******************************************************************************
C END OF COMMON BLOCK
*******************************************************************************

DEXPAN(DGID) = EXPANS

RETURN
END

C -----------------------------------------------
C CADMADE MODULE CMDGCI
C -----------------------------------------------
C PROJECT: CADMADE
C VERSION: X1.0.0
C PART : UIE-
C MODULE : ...-
C COMPILER: FORTVSZ (FORTRAN 77)
C -----------------------------------------------
C CALLING SEQUENCE:
  CALL CMDGCI (DGID, COLOR)
C -----------------------------------------------
C INPUT PARAMETERS:
  DGID - INTEGER - DIALOG AREA IDENTIFIER
  COLOR - INTEGER - COLOR INDEX
C -----------------------------------------------

Appendix C. Code Listing - Prototype UIE
SUBROUTINE CMDGFO (DGID, COLOR)

INTEGER DGID, COLOR

PARAMETER (MAXN = 5)

INTEGER DCI(MAXN), DFONT(MAXN), DIND(MAXN), DPREC(MAXN), DFLAG(MAXN),
   DBCI(MAXN), DBSI(MAXN), DOLT(MAXN),
   DBUF(MAXN), DNL(MAXN), DICHAR(MAXN), DMODE(MAXN), DVIS(MAXN)

REAL DNT(MAXN), DSPAC(MAXN), DEXPAN(MAXN),
   DOM(MAXN), DLOC(MAXN,2), DYLN(MAXN,2)

COMMON /DLGATT/

DCI(DGID) = COLOR

RETURN
END

C ADMADE MODULE CMDGFO

C PROJECT: ADMADE
C VERSION: X1.0.0
C PART : UIE
C MODULE :
C COMPILER: FORTRNS77 (FORTRAN 77)
C CALLING SEQUENCE:
C CALL CMDGFO (DGID, FONT)
C INPUT PARAMETERS:
DGID - INTEGER - DIALOG AREA IDENTIFIER
FONT - INTEGER - TEXT FONT
C OUTPUT PARAMETERS:
C COMMON INPUTS:
C NONE
C COMMON OUTPUTS:
C DLGATT
C LOCAL VARIABLES:
C
SUBROUTINE CMDGFO (DGID, FONT)

INTEGER DGID, FONT

*******************************************************************************
C START OF COMMON BLOCK FOR DIALOG AREA ATTRIBUTES
*******************************************************************************

INTEGER MAXN
PARAMETER (MAXN = 5)
INTEGER DCI(MAXN), DIND(MAXN), DPREC(MAXN), DBFLAG(MAXN),
  DOFLAG(MAXN), DBCI(MAXN), DOCI(MAXN), DBS(MAXN),
  DBSI(MAXN), DOLT(MAXN),
REAL DHT(MAXN), DSPACE(MAXN), DEXPAN(MAXN),
  DNBUFF(MAXN), DNL(MAXN), DNLCHAR(MAXN), DMODE(MAXN), DVIS(MAXN)

COMMON /DLGATT/
  DDI, OFONT, DIND, DPRE, DBFLAG,
  DOFLAG, DBCI, DOCI, DBS, DBSI, DOLT,
  DHT, DSPACE, DEXPAN, DOM, DXMLC, DVLYC,
  DNBUFF, DNL, DNLCHAR, DMODE, DVIS

*******************************************************************************
C END OF COMMON BLOCK
*******************************************************************************

RETURN
END

C==============================================================================
C CADMADE MODULE CMDGFX
C==============================================================================

C PROJECT: CADMADE
C VERSION: XI.0.0
C PART : UIE-
C MODULE :
C COMPILER: FORTVS2 (FORTRAN 77)
C CALLING SEQUENCE:
C CALL CMDGFX (DGID, INDEX)
C INPUT PARAMETERS:
C DGID - INTEGER - DIALOG AREA IDENTIFIER
  INDEX - INTEGER - ATTRIBUTE INDEX
C OUTPUT PARAMETERS:
C
C COMMON INPUTS:
C NONE
C COMMON OUTPUTS:
C DLGATT
C LOCAL VARIABLES:
C
C FUNCTIONAL DESCRIPTION:
C CADMADE SET DIALOG AREA TEXT INDEX
C==============================================================================
C MODULES CALLED:
C==============================================================================

C CODED BY: SANKAR JAYARAM
C DATE: 02/18/89 (19:51:03)
SUBROUTINE CMDGPR (DGID, PREC)

INTEGER DGID, PREC

*****************************************************************************
C START OF COMMON BLOCK FOR DIALOG AREA ATTRIBUTES
*****************************************************************************

INTEGER MAXN

PARAMETER (MAXN = 5)

INTEGER DICI(MAXN), DIND(MAXN), DPREC(MAXN), DBFLAG(MAXN),
> DDFONT(MAXN), DBSCI(MAXN), DOCI(MAXN), DBSI(MAXN),
> DINDUAXN), DBSTMAXN), DMODE(MAXN), DVIS(MAXN),
> REAL DHT(MAXN), DSPACE(MAXN), DEXPAN(MAXN),
> MAXN), DXLOC(MAXN,2), DYLOC(MAXN,2)

COMMON /DLGATT/
> DICI, DDFONT, DIND, DPREC, DBFLAG,
> DDFLAG, DBCI, DOCI, DBSI, DOLT,
> DHT, DSPACE, DEXPAN, DOM, DXLOC, DYLOC,
> DNUMF, DNFL, DCHARK, DMODE, DVIS

*****************************************************************************
C END OF COMMON BLOCK
*****************************************************************************

RETURN
END

C=============================================================================
C CADMIN MODULE CMDGPR
C=============================================================================

C PROJECT: CADMADE
C VERSION: X1.0.0
C PART : UIE---
C MODULE :
C=============================================================================
C COMPILER: FORTVS2 (FORTRAN 77)
C=============================================================================
C CALLING SEQUENCE:
C CALL CMDGPR (DGID, PREC)
C=============================================================================
C INPUT PARAMETERS:
C DGID - INTEGER - DIALOG AREA IDENTIFIER
C PREC - INTEGER - TEXT PRECISION
C=============================================================================
C OUTPUT PARAMETERS:
C=============================================================================
C COMMON INPUTS:
C NONE
C COMMON OUTPUTS:
C DLGATT
C=============================================================================
C LOCAL VARIABLES:
C=============================================================================
C FUNCTIONAL DESCRIPTION:
C CADMADE SET DIALOG AREA TEXT PRECISION
C=============================================================================
C MODULES CALLED:
C=============================================================================
C CODED BY: SANKAR JAYARAM
C DATE: 02/18/89 (19:49:58)
C=============================================================================

SUBROUTINE CMDGPR (DGID, PREC)

INTEGER DGID, PREC

*****************************************************************************
C START OF COMMON BLOCK FOR DIALOG AREA ATTRIBUTES
*****************************************************************************

INTEGER MAXN

PARAMETER (MAXN = 5)
INTEGER DCI(MAXN),
> DFONT(MAXN), DIND(MAXN), DPREC(MAXN), DBFLAG(MAXN),
> DFLA(MAXN), DBCI(MAXN), DOCI(MAXN), DBS(MAXN),
> DBSI(MAXN), DOLT(XAXN),
> DBUF(MAXN), DNL(MAXN), DNCHAI(MAXN), DMODE(MAXN), DVIS(MAXN)

REAL DHT(MAXN), DSPACE(MAXN), DEXPAN(MAXN),
> DOM(MAXN), DXLOC(MAXN,2), DYLOC(MAXN,2)

COMMON /DLGATT/
> DCI, DFONT, DIND, DPREC, DBFLAG,
> DFLA, DBCI, DOCI, DBS, DBSI, DOLT,
> DHT, DSPACE, DEXPAN, DOM, DXLOC, DYLOC,
> DBUF, DNL, DNCHAI, DMODE, DVIS

******************************************************************************
C END OF COMMON BLOCK
******************************************************************************

RETURN
END

C==============================================================================
C PROJECT: CADMADe
C VERSION: X1.0.0
C PART: UIE___
C MODULE : C==============================================================================
C COMPILER: FORTGSZ (FORTRAN 77)
C==============================================================================
C CALLING SEQUENCE: CALL CMDBF (DGID, FLAG)
C==============================================================================
C INPUT PARAMETERS:
> DGID - INTEGER - DIALOG AREA IDENTIFIER
> FLAG - INTEGER - AREA BACKGROUND FLAG
C==============================================================================
C OUTPUT PARAMETERS:
C==============================================================================
C COMMON Inputs:
> NONE
C==============================================================================
C COMMON OUTPUTS:
> DLGATT
C==============================================================================
C LOCAL VARIABLES:
> DLGATT
C==============================================================================
C FUNCTIONAL DESCRIPTION:
> CADMADe SET DIALOG AREA AREA BACKGROUND FLAG
C==============================================================================
C MODULES CALLED:
C==============================================================================
C CODED BY: SANKAR JAYARAM
C DATE: 02/18/89 (19:47:40)
C==============================================================================

SUBROUTINE CMDBF (DGID, FLAG)
 INTEGER DGID, FLAG
******************************************************************************
C START OF COMMON BLOCK FOR DIALOG AREA ATTRIBUTES
******************************************************************************
INTEGER MAXN
PARAMETER (MAXN = 5)

INTEGER DCI(MAXN),
> DFONT(MAXN), DIND(MAXN), DPREC(MAXN), DBFLAG(MAXN),
> DFLA(MAXN), DBCI(MAXN), DOCI(MAXN), DBS(MAXN),
> DBSI(MAXN), DOLT(XAXN),
> DBUF(MAXN), DNL(MAXN), DNCHAI(MAXN), DMODE(MAXN), DVIS(MAXN)

REAL DHT(MAXN), DSPACE(MAXN), DEXPAN(MAXN),
> DOM(MAXN), DXLOC(MAXN,2), DYLOC(MAXN,2)

COMMON /DLGATT/
> DCI, DFONT, DIND, DPREC, DBFLAG,
CALL IN'UT'S:

CGIIN GJTPUTS:

DLGATT

C LOCAL VARIABLES:

C FIRSTIGJAL DESCRIPTICN:

C CADMADE SET DIAL® AREA AREA GITLIE FLAG

CALLED:

C CODED BY: SANKAR JAYARAM

C DATE: 02/18/89 (19:46:27)

SUBROUTINE CMDGOF (DGID, FLAG)

INTEGER DGID, FLAG

**********************************************************************
C START OF COMMON BLOCK FOR DIALOG AREA ATTRIBUTES
**********************************************************************

INTEGER MAXN

PARAMETER (MAXN = 5)

INTEGER DCI(MAXN),
> DFONT(MAXN), DIND(MAXN), DPREC(MAXN), DBFLAG(MAXN),
> DHT, DSPACE, DEXPAN, DOM, DXLOC, DVIS

REAL DHT(MAXN), DSPACE(MAXN), DEXPAN(MAXN)
> DOM(MAXN), DXLOC(MAXN), DVIS

COMMON /DLGATT/

> DCI, DFONT, DIND, DPREC, DBFLAG,
> DHT, DSPACE, DEXPAN, DOM, DXLOC, DVIS

**********************************************************************
C END OF COMMON BLOCK
**********************************************************************

DBFLAG(DGID) = FLAG

RETURN

END
SUBROUTINE CMDGBC (DGID, COLOR)

INTEGER DGID, COLOR

C START OF COMMON BLOCK FOR DIALOG AREA ATTRIBUTES

INTEGER MAXN

PARAMETER (MAXN = 5)

INTEGER DCI(MAXN),
    DFLAG(MAXN), DIND(MAXN), DPREC(MAXN), DBS(MAXN),
    DBCI(MAXN), DOCI(MAXN), DBS(MAXN),
    DMAXN, DMAXN, DMAXN, DMAXN, DMAXN, DMAXN

REAL DHT(MAXN), DSPACE(MAXN), DEXPAN(MAXN),
    DMIN(MAXN), DXLOC(MAXN,2), DYL(MAXN,2)

COMMON /DLGATT/,
    DCI, DFLAG, DIND, DPREC, DBS,
    DMAXN, DMAXN, DMAXN, DMAXN, DMAXN

C END OF COMMON BLOCK

DBCI(DGID) = COLOR

RETURN
END

Appendix C. Code Listing - Prototype UIE
CALLING SEQUENCE:

CALL CMDGOC (DGID, COLOR)

INPUT PARAMETERS:
- DGID - INTEGER - DIALOG AREA IDENTIFIER
- COLOR - INTEGER - AREA OUTLINE COLOR INDEX

OUTPUT PARAMETERS:

COMMON INPUTS:
- NONE

COMMON OUTPUTS:
- DLGATT

LOCAL VARIABLES:

FUNCTIONAL DESCRIPTION:
- CADMADE SET DIALOG AREA AREA OUTLINE COLOR INDEX

MODULES CALLED:

CODED BY: SANKAR JAYARAM
DATE: 02/18/89

SUBROUTINE CMDGOC (DGID, COLOR)

INTEGER DGID, COLOR

C START OF COMMON BLOCK FOR DIALOG AREA ATTRIBUTES

INTEGER MAXN

PARAMETER (MAXN = 5)

INTEGER DCI(MAXN),
> DFONT(MAXN), DIND(MAXN), DPREC(MAXN), DBFLAG(MAXN),
> DBCI[MAXN], DCI[MAXN], DBS(MAXN),
> DBSII[MAXN], DOLT[MAXN],
> DBSPF(MAXN), DNL[MAXN], DCHAR[MAXN], DMODE[MAXN], DVIS[MAXN]

REAL DHT(MAXN), DSPACE(MAXN), DEXPAN(MAXN),
> DOM(MAXN), DLLOC(MAXN,2), DLLOC(MAXN,2)

COMMON /DLGATT/
> DCI, DFONT, DIND, DPREC, DBFLAG,
> DBCI, DCI, DBS, DOLT,
> DHT, DSPACE, DEXPAN, DOM, DLLOC, DLOC,
> DNL, DCHAR, DMODE, DVIS

C END OF COMMON BLOCK

DOCI(DGID) = COLOR

RETURN
END

C CALLING SEQUENCE:

CALL CMDGOC (DGID, COLOR)

CALLING SEQUENCE:

CALL CMDGBS (DGID, STYLE)

INPUT PARAMETERS:
- DGID - INTEGER - DIALOG AREA IDENTIFIER
- STYLE - INTEGER - AREA BACKGROUND INTERIOR STYLE

OUTPUT PARAMETERS:
SUBROUTINE CMDGBS (DGID, STYLE)
INTEGER DGID, STYLE

C START OF COMMON BLOCK FOR DIALOG AREA ATTRIBUTES
C CADMADE MODULE CMDGBS
C PROJECT: CADMADE
C VERSION: X1.0.0
C PART : UIE_
C MODULE : 
C CALLER: FORTVS2 (FORTRAN 77)
C CALLING SEQUENCE:
C CALL CMDGBS (DGID, INDEX)
C COMMON INPUTS:
C DGID - INTEGER - DIALOG AREA IDENTIFIER
C INDEX - INTEGER - AREA BACKGROUND INTERIOR STYLE
C COMMON OUTPUTS:
C COMMON VARIABLES:
C LOCAL VARIABLES:
C FUNCTIONAL DESCRIPTION:
C CADMADE SET DIALOG AREA AREA BACKGROUND INTERIOR STYLE
SUBROUTINE CMDGOBI (DGID, INDEX)

SUBROUTINE CMDGOT (DGID, TYPE)

C START OF COMMON BLOCK FOR DIALOG AREA ATTRIBUTES

INTEGER MAXN
PARAMETER (MAXN = 5)

INTEGER DCI(MAXN), DNAME(MAXN), DPREC(MAXN), DBFLAG(MAXN),
    DNAME(MAXN), DOCI(MAXN), DNAME(MAXN), DBNAME(MAXN),
    DNAME(MAXN), DNAME(MAXN), DNAME(MAXN), DNAME(MAXN), DNAME(MAXN),
    DNAME(MAXN), DNAME(MAXN), DNAME(MAXN), DNAME(MAXN),

REAL DHT(MAXN), DSPACE(MAXN), DEXPAN(MAXN),
    DNAME(MAXN), DNAME(MAXN), DNAME(MAXN), DNAME(MAXN),

COMMON /DLGATT/
    DCI, DNAME, DNAME, DNAME, DNAME, DNAME, DNAME,
    DNAME, DNAME, DNAME, DNAME, DNAME

******************************************************************************
C END OF COMMON BLOCK
******************************************************************************

RETURN

END

C =********************************************************************************
C C A D M A D E  M O D U L E  C M D G O T
C =********************************************************************************
C PROJECT: CADMADE
C VERSION: X1.0.0
C PART: UIE
C MODULE:
C =********************************************************************************
C CALLING SEQUENCE:
C CALL CMDGOT (DGID, TYPE)
C =********************************************************************************
C INPUT PARAMETERS:
C DGID - INTEGER - DIALOG AREA IDENTIFIER
C TYPE - INTEGER - AREA OUTLINE LINE TYPE
C =********************************************************************************
C OUTPUT PARAMETERS:
C =********************************************************************************
C COMMON INPUTS:
C NONE
C =********************************************************************************
C COMMON OUTPUTS:
C DLGATT
C =********************************************************************************
C LOCAL VARIABLES:
C =********************************************************************************
C FUNCTIONAL DESCRIPTION:
C CADMADE SET DIALOG AREA AREA OUTLINE LINE TYPE
C =********************************************************************************
C MODULES CALLED:
C =********************************************************************************
C Coded by: Sankar Jayaram
C Date: 02/18/89 (18:56:55)
INTEGER DGID, TYPE

C START OF COMMON BLOCK FOR DIALOG AREA ATTRIBUTES

INTEGER MAXN
PARAMETER (MAXN = 5)

INTEGER DCI(MAXN), DFONT(MAXN), DPREC(MAXN), DBFLAG(MAXN),
             DOFLAG(MAXN), DBCI(MAXN), DDCI(MAXN), DBS(MAXN),
             DBSI(MAXN), DOLT(MAXN),
             DNBUF(MAXN), DNL(MAXN), DNCHAR(MAXN), DMSK(MAXN), DVIS(MAXN)

REAL DHT(MAXN), DSPACE(MAXN), DEXPAN(MAXN),
            DOM(MAXN), DXLOC(MAXN,2), DYLOC(MAXN,2)

COMMON /DLGATT/
                         > DCI, DFONT, DIND, DPREC, DBFLAG,
                         > DOFLAG, DBCI, DDCI, DBS, DBSI, DOLT,
                         > DNBUF, DNL, DNCHAR, DMSK, DVIS

C END OF COMMON BLOCK

RETURN

C PROJECT: CADHADE
C VERSION: X1.0.0
C PART : UIE-__
C PDDULE : __
C CGIPILER: FORTVS2 (FORTRAN 77)
C CALLIE SEHJEFCE:
C CALL CHDGGI (DGID, HIDTH)
C INPUT PARAMETERS:
C DGID — INTEGER - DIALOG AREA IDENTIFIER
C HIDTH · REAL · AREA GJTLIE HIDTH FACTOR
C COMMON INPUTS:
C · NONE
C COMMON OUTPUTS:
C · DLGATT,
C LOCAL VARIABLES:
C C FUNCTIONAL DESCRIPTION:
C CADHADE SET DIALOG AREA AREA OUTLINE WIDTH FACTOR
C MODULES CALLED:
C C CODED BY: SAN(AR JAYARAH
C DATE: 02/18/89 (18:56:08)

CALL CHDGGI (DGID, HIDTH)

INTEGER DGID
REAL WIDTH

C START OF COMMON BLOCK FOR DIALOG AREA ATTRIBUTES

INTEGER MAXN
PARAMETER (MAXN = 5)
INTEGER DCI(MAXN),
> DFONT(MAXN), DIND(MAXN), DPREC(MAXN), DBFLAG(MAXN),
> DOFLAG(MAXN), DBCI(MAXN), DOCI(MAXN), DBSI(MAXN),
> DOTT(MAXN), DOLT(MAXN),
> DNBUF(MAXN), DNL(MAXN), DNCHAR(MAXN), DMODE(MAXN), DVIS(MAXN)

REAL DHT(MAXN), DSPACE(MAXN), DEXPAN(MAXN),
> DOM(MAXN), DXLOC(MAXN,2), DYLOC(MAXN,2)

COMMON /DLGATT/
> DCI, DFONT, DIND, DPREC, DBFLAG,
> DOFLAG, DBCI, DOCI, DBS, DBSI, DOTT,
> DOLT, DSPACE, DEXPAN, DOM, DXLOC, DYLOC,
> DNBUF, DNL, DNCHAR, DMODE, DVIS

*******************************************************************************
C END OF COMMON BLOCK
*******************************************************************************

DOM(DGID) = MWIDTH

RETURN

C===============================================================================
C CADMADE MODULE CMDGBU
C===============================================================================
C PROJECT: CADMADE
C VERSION: V1.0.0
C PART : UIE___
C MODLE : 
C===============================================================================
C CALLING SEQUENCE:
C CALL CMDGBU (DGID, NBUF)
C===============================================================================
C CALLING SEQUENCE:
C DGID - INTEGER - DIALOG AREA IDENTIFIER
C NBUF - INTEGER - DIALOG AREA BUFFER SIZE

C===============================================================================
C OUTPUT PARAMETERS:
C===============================================================================
C COMMON INPUTS:
C NONE
C===============================================================================
C COMMON OUTPUTS:
C DLGATT
C===============================================================================
C LOCAL VARIABLES:
C===============================================================================
C FUNCTIONAL DESCRIPTION:
C CADMADE SET DIALOG AREA BUFFER SIZE
C===============================================================================
C MODULES CALLED:
C===============================================================================
C CODED BY: SANKAR JAYARAM
C DATE: 02/18/99 (18:56:08)
C===============================================================================

SUBROUTINE CMDGBU (DGID, NBUF)

INTEGER DGID, NBUF

C START OF COMMON BLOCK FOR DIALOG AREA ATTRIBUTES
C===============================================================================

INTEGER MAXN

PARAMETER (MAXN = 5)

INTEGER DCI(MAXN),
> DFONT(MAXN), DIND(MAXN), DPREC(MAXN), DBFLAG(MAXN),
> DOFLAG(MAXN), DBCI(MAXN), DOCI(MAXN), DBSI(MAXN),
> DOTT(MAXN), DOLT(MAXN),
> DNBUF(MAXN), DNL(MAXN), DNCHAR(MAXN), DMODE(MAXN), DVIS(MAXN)

REAL DHT(MAXN), DSPACE(MAXN), DEXPAN(MAXN),
> DOM(MAXN), DXLOC(MAXN,2), DYLOC(MAXN,2)

COMMON /DLGATT/
**C END OF COMMON BLOCK**

**********************************************************************************

DNBUF(DGID) = NBUF

RETURN

END

**********************************************************************************

**C C A D M A D E M O D U L E C M D G N L**

C_*******************************************************************************

C PROJECT: CADMADE
C VERSION: V1.0.0
C PART : UIE
C MODULE : 

C*******************************************************************************

C COMPILER: FORTVS2 (FORTRAN 77)

C*******************************************************************************

C CALLING SEQUENCE:
C CALL CMDGNL (DGID, NLINE)

C*******************************************************************************

C INPUT PARAMETERS:
C DGID - INTEGER - DIALOG AREA IDENTIFIER
C NLINE - INTEGER - DIALOG AREA NO. OF DISPLAYED LINES

C*******************************************************************************

C OUTPUT PARAMETERS:

C*******************************************************************************

C COMMON INPUTS:
C NONE

C*******************************************************************************

C COMMON OUTPUTS:
C DLGATT

C*******************************************************************************

C LOCAL VARIABLES:
C

C*******************************************************************************

C FUNCTIONAL DESCRIPTION:
C CADMADE SET DIALOG AREA NO. OF DISPLAYED LINES

C*******************************************************************************

C MODULES CALLED:
C

C*******************************************************************************

C CODED BY: SANKAR JAYARAM
C DATE: 02/18/89 (18:56:08)

C*******************************************************************************

**SUBROUTINE CMDGNL (DGID, NLINE)**

INTEGER DGID, NLINE

*******************************************************************************

C START OF COMMON BLOCK FOR DIALOG AREA ATTRIBUTES

*******************************************************************************

INTEGER MAXN

INTEGER DICI(MAXN),
    > DFIN(MAXN), DIND(MAXN), DPREC(MAXN), DBFLAG(MAXN),
    > DBFLAG(MAXN), DBCI(MAXN), DOCI(MAXN), DSS(MAXN),
    > DBSI(MAXN), DOLT(MAXN),
    > DYNBUF(MAXN), DNL(MAXN), DNMCHAR(MAXN), DMODE(MAXN), DVIS(MAXN)

REAL DHT(MAXN), DSPACE(MAXN), DEXPAN(MAXN),
    > DOMAXN, DXLOC(MAXN,2), DYLOC(MAXN,2)

COMMON /DLGATT/
    > DICI, DFIN, DIND, DPREC, DBFLAG,
    > DBFLAG, DBCI, DOCI, DSS, DBSI, DOLT,
    > DHT, DSPACE, DEXPAN, DOM, DXLOC, DYLOC,
    > DYNBUF, DNL, DNMCHAR, DMODE, DVIS

*******************************************************************************

C END OF COMMON BLOCK

*******************************************************************************

DNL(DGID) = NLINE
SUBROUTINE CMDGN (DGID, NCHAR)

INTEGER DGID, NCHAR

C START OF COMMON BLOCK FOR DIALOG AREA ATTRIBUTES

PARAMETER (MAXN = 5)

INTEGER DCI(MAXN), DIFONT(MAXN), DIND(MAXN), DPREC(MAXN), DBFLAG(MAXN),
  DOFLAG(MAXN), DBCI(MAXN), DOCI(MAXN), DBS(MAXN),
  DBSI(MAXN), DOLT(MAXN),
  DNBUF(MAXN), DNL(MAXN), DCHAR(MAXN), DMODE(MAXN), DVIS(MAXN)

REAL DHT(MAXN), DSPACE(MAXN), DekPAN(MAXN),
  DOM(MAXN), DXLOC(MAXN,2), DYLOC(MAXN,2)

COMMON /DLGATT/
  DCI, DIFONT, DIND, DPREC, DBFLAG,
  DOFLAG, DBCI, DOCI, DBS, DBSI, DOLT,
  DHT, DSPACE, DekPAN, DOM, DXLOC, DYLOC,
  DNBUF, DNL, DCHAR, DMODE, DVIS

C END OF COMMON BLOCK

RETURN

END
C MODULE =
C---------------------------
C COMPILER: FORTVS2 (FORTRAN 77)
C---------------------------
C CALLING SEQUENCE:
C CALL CMDGVS (DGID, VIS)
C---------------------------
C INPUT PARAMETERS:
C DGID - INTEGER - DIALOG AREA IDENTIFIER
C VIS - INTEGER - DIALOG AREA VISIBILITY
C---------------------------
C OUTPUT PARAMETERS:
C---------------------------
C COMMON INPUTS:
C NONE
C---------------------------
C COMMON OUTPUTS:
C NONE
C---------------------------
C LOCAL VARIABLES:
C---------------------------
C FUNCTIONAL DESCRIPTION:
C CADMADE SET DIALOG AREA SCROLL MODE
C---------------------------
C MODULES CALLED:
C---------------------------
C Coded by: Sankar Jayaram
C Date: 02/18/89 (18:56:08)

SUBROUTINE CMDGVS (DGID, VIS)

INTEGER DGID, VIS

***************************************************************************
C START OF COMMON BLOCK FOR DIALOG AREA ATTRIBUTES
***************************************************************************
INTEGER MAXN
PARAMETER (MAXN = 5)
INTEGER DCl(MAXN), DONT(MAXN), DIND(MAXN), DPREC(MAXN), DBRAT(MAXN),
> DFLAG(MAXN), DBCI(MAXN), DABS(MAXN),
> DOL(MAXN), DOLT(MAXN),
> DNBUF(MAXN), DNL(MAXN), DCHAR(MAXN), DMODE(MAXN), DVIS(MAXN)
REAL DHT(MAXN), DSPACE(MAXN), DEPXN(MAXN),
> DOMAXN), DXLOC(MAXN,2), DYLOC(MAXN,2)

COMMON /DGLATT/
> DCI, DONT, DIND, DPREC, DFLAG,
> DFLAG, DBCI, DCC, DBS, DOL, DHT, DSPACE, DEPXN, DOM, DXLOC, DYLOC,
> DNBUF, DNL, DCHAR, DMODE, DVIS
***************************************************************************
C END OF COMMON BLOCK
***************************************************************************

RETURN
END

***************************************************************************
C CADMADE MODULE CMDGVS
***************************************************************************
C PROJECT: CADMADE
C VERSION: V1.0.0
C PART : UIE-
C MODUHE:
C***************************************************************************
C COMPILER: FORTVS2 (FORTRAN 77)
C***************************************************************************
C CALLING SEQUENCE:
C CALL CMDGVS (DGID, VIS)
C***************************************************************************
C INPUT PARAMETERS:
C DGID - INTEGER - DIALOG AREA IDENTIFIER
C VIS - INTEGER - DIALOG AREA VISIBILITY
Appendix C. Code Listing - Prototype UIE 274
SUBROUTINE CMDGVL (DGID, VIS)

INTEGER DGID, VIS

******************************************************************************
C START OF COMMON BLOCK FOR DIALOG AREA ATTRIBUTES
******************************************************************************

INTEGER MAXN
PARAMETER (MAXN = 5)

INTEGER DCI(MAXN), DFIN(MAXN), DPREF(MAXN), DBFLAG(MAXN),
> DOFLAG(MAXN), DBCI(MAXN), DOCI(MAXN), DBS(MAXN),
> DSI(MAXN), DOLT(MAXN),
> DNSFU(MAXN), DNL(MAXN), DNLCHR(MAXN), DMOD(MAXN), DVIS(MAXN)

REAL DHT(MAXN), DSPACE(MAXN), DEXPA(NAXN),
> DOM(MAXN), DXLOC(MAXN,2), DYLOC(MAXN,2)

COMMON /DLGATT/
> DCI, DFIN, DFIN, DPREF, DBFLAG,
> DOFLAG, DBCI, DOCI, DBS, DSI, DOLT,
> DNSFU, DNL, DNLCHR, DMOD, DVIS

******************************************************************************

DVIS(DGID) = VIS

RETURN
END

C==================================
C C A D M A D E M O D U L E C M D G L C
C==================================

C PROJECT: CADMADE
C VERSION: XI.0.0
C PART : UIE__
C MODULE : C

C COMPILER: FORTVS2 (FORTRAN 77)
C CALLING SEQUENCE:
CALL CMDGLC (DGID, XI, X2, Y1, Y2)
C INPUT PARAMETERS:
DGID - INTEGER - DIALOG AREA IDENTIFIER
XI, X2, Y1, Y2 - REAL - CORNER POINTS OF THE RECTANGLE
C OUTPUT PARAMETERS:

C COMMON INPUTS:
C NONE
C COMMON OUTPUTS:
C DLGATT

Appendix C. Code Listing - Prototype UIE 275
LOCAL VARIABLES:

C
C====================================================================
C FUNCTIONAL DESCRIPTION:
C CADMADE SET DIALOG AREA LOCATION
C====================================================================
C MODULES CALLED:
C
C CODED BY: SANKAR JAYARAM
C DATE: 02/18/89 (18:56:08)
C====================================================================

SUBROUTINE CMDLGAT (DGID, X1, X2, Y1, Y2)

INTEGER DGID
REAL X1, X2, Y1, Y2

**************************************************************************
C START OF COMMON BLOCK FOR DIALOG AREA ATTRIBUTES
**************************************************************************

INTEGER MAXN
PARAMETER (MAXN = 5)

INTEGER DCI(MAXN),
> DFONT(MAXN), DIND(MAXN), DPREC(MAXN), DBFLAG(MAXN),
> DDPF(MAXN), DBCI(MAXN), DDCI(MAXN), DBS(MAXN),
> DBSI(MAXN), DOLI(MAXN),
> DNBUFF(MAXN), DNL(MAXN), DNCCHAR(MAXN), DMODE(MAXN), DVISI(MAXN)
REAL DHT(MAXN), DSPACI(MAXN), DEXPANI(MAXN),
> DOMI(MAXN), DXLOC(MAXN,2), DYLOC(MAXN,2)

COMMON /DLGATT/
> DCI, DFIN, DIND, DPREC, DBLAG,
> DDPF, DBCI, DDCI, DBS, DBSI, DOLI,
> DHT, DSPACI, DEXPAN, DOM, DXLOC, DYLOC,
> DNBUFF, DNL, DNCCHAR, DMODE, DVIS

**************************************************************************
C END OF COMMON BLOCK
**************************************************************************

DXLOC(DGID,1) = X1
DXLOC(DGID,2) = X2
DYLOC(DGID,1) = Y1
DYLOC(DGID,2) = Y2

RETURN
END

====================================================================

MODULE CLDGAT

====================================================================

PROJECT: CADMADE
VERSION: X1.0.0
PART : UIE___
MODULE : 

====================================================================

CALLING SEQUENCE:
CALL CLDGAT

INPUT PARAMETERS:

OUTPUT PARAMETERS:

COMMON INPUTS:
NONE

COMMON OUTPUTS:

COMMON CALLS:

LOCAL VARIABLES:

====================================================================

Appendix C. Code Listing - Prototype UIE

276
SUBROUTINE CLDGT

******************************************************************************
C START OF COMMON BLOCK FOR DIALOG AREA ATTRIBUTES
******************************************************************************

INTEGER MAXN

PARAMETER (MAXN = 5)

INTEGER DCI(MAXN), DFLAG(MAXN), DPREC(MAXN), DBFLAG(MAXN),
> DBCI(MAXN), DOCI(MAXN), DBS(MAXN),
> DIND[MAXN], DOLT(MAXN),
> DNCHAR[MAXN], DNBUFFA(MAXN), DNSPACEA(MAXN), DMODEA(MAXN), DVIS(MAXN)

REAL DHT(MAXN), DSPACE(MAXN), DEXPAN(MAXN),
> DOMA(MAXN), DXLOC(MAXN,Z), DNL(MAXN),
> DBUF(MAXN), DNCHAR, DNBUFFA, DNSPACEA, DMODEA, DVIS

******************************************************************************
C END OF COMMON BLOCK
******************************************************************************

C RESET THE ATTRIBUTES

DO 100 I = 1, MAXN
  DCI(I) = 1
  DFLAG(I) = 2
  DIND(I) = 1
  DPREC(I) = 1
  DBFLAG(I) = 1
  DBCI(I) = 0
  DOCI(I) = 1
  DBS(I) = 1
  DBSI(I) = 1
  DOLT(I) = 1
  DHT(I) = 0.02
  DSPACE(I) = 1.0
  DEXPAN(I) = 1.0
  DOM(I) = 1.0
  DXLOC(I,1) = -0.8
  DXLOC(I,2) = 0.9
  DYLX(I,1) = -0.40
  DYT(I) = 15
  DNL(I) = 6
  DNCHAR(I) = 80
  DMODE(I) = 1
  DVIS(I) = 1

100 CONTINUE
RETURN
END
C INPUT PARAMETERS:
C
PKID
INTEGER; KJRKSTATIUI IDEN'|'IFIER
C
DGID - INTEGER, DIALOG AREA IDENTIFIER
C
INSTR -- CHARACTER*(·l·)• MESSAGE STRING
C (UTPUT PARAMETERS:
C
NONE
C CGYDN INPUTS:
C
IDNE
C CGN!) GITPUTS:
C
FINE
¢s:•::=========sssss===:==ss=s¤¤s¤•:=:=s:s¤:=:::s:sssssznzssasassssssu

C LOCAL VARIABLES:
•
C
NLINES
INTEGER• MRBER OF MESSAGE LINES IN TEXT ARRAY
C
TEXTINLINES) - CHARACTER; IESSAGE STORAGE ARRAY

ßs|sss======:==============:=s=s==s===============:==:s:::¤=s¤:=s¤==ss

C FUCTIGIAL DESCRIPTION:
C
HRITES THE SPECIFIED MESSAGE
C
ALL PREVIGJS MESSAGE LINES.

THE SCREEN AFTER SCROLLHG LP

C IDDULES CALLED:
—
C
SCRMSG
SCROLLS MESSAGE ARRAY
C
HRTMSG - NRITES NEM MESSAGE
C CODED BY:
C
C
DATE:

SANKAR JAYARAM
ADAPTED FROM ACSYNT/VPI CODE HRITTEN BY MICHELE GRIESHABER
09/11/87 (13:01:08)

ßssuzszsszsss==:===ss=====¤==:=sass:¤:s=¤=s=sss:::¤:s¤¤:ss¤nsszzssszss

S\BR(IJTINE C)|1SG()|(ID• DGID• INSTR)
INTEGER II(ID• DGID

CHARACTERx(•~) INSTR
INTEGER NLINES
PARA|£TER(NLINES = 15)
CHARACTERN80) TEXT ( NLITES)
'•
•,
•’
•,
u,
•'
•’
•,
u,
u,
n,
•’
•,
•,
•
[

DATA TEXT / '
¤•
>
>
•
>
•
>
¤
>
n
>
>
•u
>
•
>
•
>
¤
>
u•
>
>
•
>
C

SCROLL TEXT
CALL SCRMSG(INSTR» NLINES; TEXT)

C

PROCESS MESSAGE LINES
CALL NRTH$GlH(IÜ; DGID• NLI§$» TEXT)
RETURN
END

CSIIIISIBIIISIS!222S22232S=3IBIIIIIIISSSSESSESSSSBIIIIIIIIIISSBBSSSSII

C

MODULE

C PROJECT:
C VERSION:
C
PART 8:
C FDDULE Q:

CADMADE
X1.D.O

C CCMPILER:

FORTVS2 (FORTRAN 77)

SCRMSG

'

ß=====:================sz=======sasss=:==:ssass¤===s¤::::¤s¤¤ss=s=s===

C CALLING SEQUEMZE:
C
CALL SCRMSG(INSTR• NLINES: TEXT)

C INPUT PARAMETERS:

C

C
C

INSTR

- CHARACTER*(I)» INPUT MESSAGE STRI)8

NLINES
MJIBER OF STRINGS IN TEXT ARRAY.
- INTEGER•
TEXT(NLINES)
- CHARACTER*l·l·)• MESSAGE TEXT ARRAY

Appendix C. Code Listing - Prototype UIE

278


SUBROUTINE SFTMSG(INSTR, NLINES, TEXT)

INTEGER NLINES
CHARACTER*(*) INSTR
CHARACTER*(*) TEXT(*)

CALL SFTMSG(NLINES, TEXT)

CALL PLGMSG(INSTR, TEXT(1))

RETURN

END

MODULE SFTMSG

PROJECT: CADMADE
VERSION: X1.0.0
PART #: 
MODULE #: 

CALLING SEQUENCE:
CALL SFTMSG(NLINES, TEXT)

INPUT PARAMETERS:
NLINES - INTEGER, NUMBER OF MESSAGE LINES CONTAINED IN TEXT
TEXT(NLINES) - INTEGER, MESSAGE ARRAY AFTER SHIFTING LINES

OUTPUT PARAMETERS:

COMMON INPUTS:
NONE

COMMON OUTPUTS:
NONE

LOCAL VARIABLES:

FUNCTIONAL DESCRIPTION:
SHIFTS THE ITH LINE TO THE I+1 LINE

MODULES CALLED:
NONE

CODED BY: SANKAR JAYARAM
ADAPTED FROM ACSYNT/VPI CODE WRITTEN BY STEVE NAMPLER
DATE: 09/29/87 (14:30:53)

SUBROUTINE SFTMSG(NLINES, TEXT)

INTEGER NLINES

Appendix C. Code Listing - Prototype UIE
CHARACTER(*) TEXT(*)

INTEGER LINE

DO 100 LINE = NLLINES, 2, -1
C
  COPY PREVIOUS LINE
  TEXT(LINE) = TEXT(LINE - 1)
100 CONTINUE
C
RETURN
END

C*****************************************************************************
C MODULE PLGMSG
C*****************************************************************************
C PROJECT: CADMADE
C VERSION: X1.0.0
C PART #: 8
C MODULE #: 8
C*****************************************************************************
C CALLING SEQUENCE:
C CALL PLGMSG(INSTR, TEXT)
C*****************************************************************************
C INPUT PARAMETERS:
C INSTR - CHARACTER(*), MESSAGE STRING TO BE COPIED TO TEXT
C*****************************************************************************
C OUTPUT PARAMETERS:
C TEXT - CHARACTER(*), NEW MESSAGE TEXT LINE
C*****************************************************************************
C COMMON INPUTS:
C NONE
C*****************************************************************************
C COMMON OUTPUTS:
C NONE
C*****************************************************************************
C LOCAL VARIABLES:
C LTXT - INTEGER, MINIMUM OF THE LENGTH OF TEXT OR INSTR
C I - INTEGER, CHARACTER COUNTER
C*****************************************************************************
C FUNCTIONAL DESCRIPTION:
C PLUGS NEW MESSAGE STRING INTO MESSAGE ARRAY
C*****************************************************************************
C MODULES CALLED:
C NONE
C*****************************************************************************
C CODED BY: SANKAR JAYARAM
C ADAPTED FROM ACSYNT/VPI CODE WRITTEN BY STEVE HAMPLER
C DATE: 09/29/87 (14:37:17)
C*****************************************************************************
SUBROUTINE PLGMSG(INSTR, TEXT)
  CHARACTER(*) INSTR
  CHARACTER(*) TEXT
  INTEGER LTXT
C
  CALCULATE MINIMUM STRING LENGTH
  LTXT = MIN LEN(TEXT), LEN(INSTR)
C
  COPY INPUT STRING TO TEXT STRING
  TEXT(1:LTXT) = INSTR(1:LTXT)
C
  CLEAR TEXT STRING NOT OCCUPIED BY INSTR
  DO 100 I = LTXT+1, LEN(TEXT)
  TEXT(I:I) = ' '
100 CONTINUE
C
RETURN
END

C*****************************************************************************
C MODULE MRTMSG
C*****************************************************************************
C PROJECT: CADMADE
C VERSION: X1.0.0
C PART #: 8
C MODULE #: 8
C*****************************************************************************
C CALLING SEQUENCE:
C*****************************************************************************
SUBROUTINE DRMSG(DGID, NLINES, TEXT)
INTEGER DGID, NLINES
CHARACTER*(*) TEXT(*)
INTEGER MSGSTR
REAL PRIORT
PARAMETER(PRIORT = 1.0)
WRITE(6,*),'DGID', DGID
C CALCULATE THE DIALOG AREA STRUCTURE
MSGSTR = 100000 + DGID
C EMPTY STRUCTURE
CALL PEMST(MSGSTR)
C OPEN STRUCTURE
CALL POPST(MSGSTR)
C DRAW MESSAGE
CALL DRMSG(DGID, NLINES, TEXT)
C CLOSE STRUCTURE
CALL PCLST
C POST ROOT
CALL PPORT(MKID, MSGSTR, PRIORT)
RETURN
END

MODULE DRMSG
C PROJECT: CADMADE
C VERSION: X1.0.0
C PART #: 
C MODULE #: 
C COMPILER: FORTVS2 (FORTRAN 77)
C CALLING SEQUENCE:
CALL DRMSG(DGID, NLINES, TEXT)
C INPUT PARAMETERS:
C DGID - INTEGER, DIALOG AREA IDENTIFIER
C NLINES - INTEGER, NUMBER OF MESSAGE TEXT LINES
C TEXT(NLINES) - CHARACTER(*), MESSAGE TEXT ARRAY
SUBROUTINE DRMSG(DGID, N LINES, TEXT)

INTEGER DGID, N LINES
CHARACTER*(*) TEXT(*)

INTEGER VIEHI
PARAMETER(VIEHI = 0)

C SET VIEW INDEX
CALL PSVNI(VIEHI)

C DRAW MESSAGE TEXT
CALL DRSTX(DGID, N LINES, TEXT)

RETURN
END

MODULE DRSTX

C Project: CADMADE
C Version: X1.0.0
C Part #: C
C Module #: M
C Compiler: FORTVS2 (FORTRAN 77)
C Calling Sequence:
C CALL DRSTX(DGID, N LINES, TEXT)
C
C Input Parameters:
DGID - INTEGER, DIALOG AREA IDENTIFIER
N LINES - INTEGER, NUMBER OF MESSAGE TEXT LINES
TEXT(N LINES) - CHARACTER(*), MESSAGE TEXT LINES
C
C Output Parameters:
C
C Common Inputs:
C
C Common Outputs:
C
C Local Variables:
COLORS(IO) - INTEGER, ARRAY OF TEXT COLOR INDICES
GAP - REAL, SIZE OF GAP BETWEEN TEXT LINES
PX - REAL, CALCULATED POSITION OF TEXT
PY -
XBLK - REAL, LOCATION OF LOWER LEFT CORNER OF MESSAGE BLOCK
YBLK -
BLKHT - REAL, HEIGHT OF BLOCK IN WHICH MESSAGE LINES MUST FIT
C
C Functional Description:
C DRAM THE MESSAGE TEXT LINES
C
C=========================================================================
C MODULES CALLED:
C PSTXCI - PHIGS SET TEXT COLOR INDEX
C PTX - PHIGS TEXT
C=========================================================================
C CODED BY: SANKAR JAYARAM
C ADAPTED FROM ACSYNT/VPI CODE WRITTEN BY STEVE HAFPLER
C DATE: 09/29/87 (15:14:14)
C=========================================================================

SUBROUTINE DRAMTX(DGID, NLINES, TEXT)

INTEGER DGID, NLINES
CHARACTER(*) TEXT(*)

*************************************************************************
C START OF COMMON BLOCK FOR DIALOG AREA ATTRIBUTES
*************************************************************************

INTEGER MAXN
PARAMETER (MAXN = 5)
INTEGER DCI(MAXN), DIND(MAXN), DPREC(MAXN), DBCI(MAXN),
      DFLAG(MAXN), DDCI(MAXN), DBS(MAXN),
      DBSI(MAXN), DOLT(MAXN),
      D Buf(MAXN), DNL(MAXN), DCHI(MAXN), DMODE(MAXN), DVIS(MAXN)

REAL DHT(MAXN), DSPACE(MAXN), DEXPAN(MAXN),
      DOM(MAXN), DXLOC(MAXN), DYLOC(MAXN),
      DHT, DSPACE, DEXPAN, DOM, DXLOC, DYLOC,
      D Buf, DNL, DCHI, DMODE, DVIS

*************************************************************************
C END OF COMMON BLOCK
*************************************************************************

INTEGER NPBFA
PARAMETER (NPBFA=6)
REAL XBFA(NPBFA), YBFA(NPBFA)

INTEGER COLORS(10); WHITE, TEXTCL
PARAMETER(WHITE = 7; TEXTCL = 11)

REAL GAP, PX, PY, XBK, YBLK, BLKHT

C SET THE DIALOG AREA VISIBILITY
IVIS = 126 + DVIS(DGID)
CALL PADS (1, IVIS)

C CHECK THE BACKGROUND FLAG
IF (DBFLAG(DGID) .NE. 0) THEN

C SET ALL THE FILL AREA ATTRIBUTES
CALL PSEDFG (DFLAG(DGID))
CALL PSICI (DCCI(DGID))
CALL PSEDIC (DCDI(DGID))
CALL PSISI (DBS(DGID))
CALL PSEDES (DBSI(DGID))
CALL PSEDIT (DOLT(DGID))

*************************************************************************
GRAPHICS CALL
*************************************************************************

CALL GPESC (DOM(DGID))

*************************************************************************
C DRAM THE FILL AREA
XBFA(1) = DXLOC(DGID,1)
XBFA(2) = DXLOC(DGID,2)
XBFA(3) = DXLOC(DGID,2)
XBFA(4) = DXLOC(DGID,1)

YFA(1) = DYLOC(DGID,1)
YFA(2) = DYLOC(DGID,1)
YFA(3) = DYLOC(DGID,2)
YFA(4) = DYLOC(DGID,2)

CALL PFA (NPBFA, XBFA, YBFA)

ENDIF

C CALCULATE TEXT GAP AND INITIAL TEXT POSITION

Appendix C. Code Listing - Prototype UIE
GAP = (DXLOC(DGID,2) - DXLOC(DGID,1)) / FLOAT(DNL(DGID)+1)
PX = DXLOC(DGID,1) + DHT(DGID)*DEXPAN(DGID)
PY = DXLOC(DGID,1) + GAP/3.0 - DHT(DGID)

CALL PSTXCI (DCI(1))
CALL PSTXFN (DFONT(1))
CALL PSTXI (DIND(1))
CALL PSTXPR (DPREC(1))
CALL PSCHHS (DHT(1))
# CALL PSCHSP (DSPACE(1))
CALL PSCHXP (DEXPAN(1))

PY = PY + GAP

DO 100 I = 1, DNL(DGID)-1
C CALCULATE Y POSITION OF TEXT
PY = PY + GAP
C DRAM TEXT
CALL PTXIPX (PX, PY, TEXT(I))
100 CONTINUE
RETURN
END

C==========================================================================
C CADMADE MODULE CHINST
C==========================================================================
C PROJECT: CADMADE
C VERSION: X1.0.0
C PART #: 1
C MODULE #: 5
C==========================================================================
C COMPILER: FORTVS2 (FORTRAN 77)
C==========================================================================
C CALLING SEQUENCE:
CALL CHINST(MKID, DGID, STDNR, LENT, TEXT, PET, XHIN, XMAX, YMIN, YMAX,  
> LBUF, INIPOS, LDR, DATREC)
C==========================================================================
C INPUT PARAMETERS:
MKID - INTEGER, WORKSTATION IDENTIFIER
DGID - INTEGER, DIALOG AREA IDENTIFIER
STDNR - INTEGER, STRING DEVICE NUMBER
LENT - LENGTH OF INITIAL STRING
TEXT - CHARACTER(*), INITIAL (DEFAULT) CHARACTER STRING
PET - INTEGER, ECHO TYPE
XHIN - REAL, STRING ECHO AREA LIMITS
XMAX
YMIN
YMAX
LBUF - INTEGER, BUFFER LENGTH
LDR - INTEGER, LENGTH OF DATA RECORD
DATREC - CHARACTER*80, DATA RECORD
C==========================================================================
C OUTPUT PARAMETERS:
C
C==========================================================================
C COMMON INPUTS:
C DLGATT, MNUMPR1, MNUMPR2
C==========================================================================
C COMMON OUTPUTS:
C
C==========================================================================
C LOCAL VARIABLES:
C==========================================================================
C FUNCTIONAL DESCRIPTION:
C CADMADE INITIALIZES STRING INPUT DEVICE
C==========================================================================
C MODULES CALLED:
C AGDSP - INQUIRE ACTUAL DISPLAY SIZE
C PNST - PHIGS INITIALIZE STRING
C==========================================================================
C Coded By: Sankar Jayaram
C Date:
C==========================================================================

SUBROUTINE CHINST(MKID, DGID, STDNR, LENT, TEXT, PET,  
> XHIN, XMAX, YMIN, YMAX,  
> LBUF, INIPOS, LDR, DATREC)

INTEGER MKID, DGID
CHARACTER(*) TEXT
INTEGER ERRIND, STDNR, PREQU, PEVENT, PECOH
INTEGER PET, INIPOS, LDR, DATREC
REAL RX, RY, XMIN, XMAX, YMIN, YMAX
INTEGER DCUNIT, Lx, Ly

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C START OF COMMON BLOCK FOR DIALOG AREA ATTRIBUTES
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
INTEGER MAXN

PARAMETER (MAXN = 5)

INTEGER DCI(MAXN),
   > DONT(MAXN), DIND(MAXN), DPREC(MAXN), DBFLAG(MAXN),
   > DOFLAG(MAXN), DBCI(MAXN), DOCI(MAXN), DBS(MAXN),
   > DSI(MAXN), DOL(MAXN),
   > DBUF[MAXN], DNL(MAXN), DCHIEF(MAXN), DMODE(MAXN), DVIS(MAXN),

REAL DHT[MAXN], DSPACE[MAXN], DEXPAN[MAXN],
   > DOM[MAXN], DXLOC(MAXN, 2), DYLOC(MAXN, 2)

COMMON /DLGATT/
   > DCI, DONT, DIND, DPREC, DBFLAG,
   > DOFLAG, DBCI, DOCI, DBS, DBTI, DOLT,
   > DHT, DSPACE, DEXPAN, DOM, DXLOC, DYLOC,
   > DBUF, DNL, DCHIEF, DMODE, DVIS

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C END OF COMMON BLOCK
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C START OF COMMON BLOCK FOR SAVING MENU P|ONTPT PARAMETERS
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
INTEGER NMENUS

PARAMETER (NMENUS=200)

INTEGER MDGID(NMENUS), MPRMTL(NMENUS)
CHARACTER*80 MPRMTL(NMENUS)

COMMON /MNPR1/ MDGID, MPRMTL
COMMON /MNPR2/ MPRMTL

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C END OF COMMON BLOCK
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

C INQUIRE DISPLAY SPACE SIZE
CALL AQDSP(MKID, ERRIND, DCUNIT, RX, RY, LX, LY)

C CALCULATE COORDINATE CONVERSION FACTORS
XCONV = RX/2.0
YCONV = RY/2.0

C FIND THE ECHO AREA FOR THE DIALOG ID OF 1
XMIN = (DXLOC(1,1) + 0.01*(DXLOC(1,2)-DXLOC(1,1)) +1)*XCONV
YMIN = (DYLOC(1,1) + 0.01*(DYLOC(1,2)-DYLOC(1,1)) +1)*YCONV
XMAX = RX
YMAX = RY

C INITIALIZE STRING
CALL PINS1(MKID, STDNR, LENT, TEXT, PET,
   > XMIN, XMAX, YMIN, YMAX,
   > LENT(TEXT), INIPOS, LDR, DATREC)

RETURN
END

C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C MODULE AQDSP
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C PROJECT: CADMADE
C VERSION: XL1.0
C PART #: C
C MODULE #: C
C COMPILER: FORTV52 (FORTRAN 77)
C CALLING SEQUENCE:
C CALL AQDSP(MKID, ERRIND, DCUNIT, RX, RY, LX, LY)
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

Appendix C. Code Listing - Prototype UIE
SUBROUTINE AGDSP(MKID, ERRIND, DClNIT, RX, RY, LX, LY)

INTEGER MKID, ERRIND, DClNIT, LX, LY
REAL RX, RY

INTEGER CONID, MTYPE

BEGIN

C INQUIRE WORKSTATION CONNECTION AND TYPE
CALL PQWKIC(MKID, ERRIND, CONID, MTYPE)

IF (ERRIND .EQ. 0) THEN
C INQUIRE DISPLAY SPACE SIZE
CALL PQDSP(MTYPE, ERRIND, DClNIT, RX, RY, LX, LY)
ENDIF

RETURN

END
This section contains a listing of the code written to implement the sample CAD program. This program also uses modules developed by Gandhi (1988) for implementing a prototype feature-based modeler. The feature-based modeling routines are not included in this listing.
C MODULE FEATURE
C
C COMPILER: FORTVS2 (FORTRAN 77)
C CALLING SEQUENCE:
C CALLED FROM SYSTEM
C
C LOCAL VARIABLES:
C MKID - INTEGER, PRIMARY WORKSTATION IDENTIFIER
C
C FUNCTIONAL DESCRIPTION:
C MAIN MODULE FOR FEATURE
C
C MODULES CALLED:
C OPMAIN - OPENS FEATURE
C DRFEAT - FEATURE BASED MODELER
C CLMAIN - CLOSES FEATURE
C
C CODED BY: SANKAR JAYARAM
C DATE: 09/01/87
C
PROGRAM FEATURE
INTEGER MKID
PARAMETER(MKID = 1)

C OPEN WORKSTATION
CALL OPMAIN(MKID)

C CALL THE FEATURE BASED MODELER ROUTINES
CALL DRFEAT(MKID)

C CLOSE WORKSTATION
CALL CLMAIN(MKID)

STOP
END

C=================================================================
C MODULE OPMAIN
C=================================================================
C COMPILER: FORTVS2 (FORTRAN 77)
C CALLING SEQUENCE:
C CALLED OPMAIN(MKID)
C
C INPUT PARAMETERS:
C MKID - INTEGER, WORKSTATION IDENTIFIER
C
C OUTPUT PARAMETERS:
C NONE
C
C COMMON INPUTS:
C NONE
C
C COMMON OUTPUTS:
C NONE
C
C LOCAL VARIABLES:
C ERRFIL - INTEGER, ERROR FILE IDENTIFIER
C BUFA - INTEGER, NUMBER OF MEMORY UNITS
C CONSID - INTEGER, WORKSTATION CONNECTION IDENTIFIER
C NTYPE - INTEGER, WORKSTATION TYPE
C
C FUNCTIONAL DESCRIPTION:
C OPENS FEATURE
C
C MODULES CALLED:
C GWINFO - GET WORKSTATION CONNECTION ID AND TYPE
C POPPH - PHIGS OPEN PHIGS
C POPPK - PHIGS OPEN WORKSTATION
C INITPK - INITIALIZE WORKSTATION
C
C CODED BY: SANKAR JAYARAM
C ADAPTED FROM ACSYNT/VPI CODE WRITTEN BY STEVE MAMPLER
C DATE: 09/01/87 (15:05:17)
C
SUBROUTINE OPMAIN(MKID)
INTEGER PKID
INTEGER ERRFIL, BUFA
PARAMETER(ERRFIL = 0, BUFA = 0)
INTEGER CONID, MTYPE

C GET WORKSTATION TYPE AND CONNECTION ID
CALL GMINFO(MTYPE, CONID)

C OPEN PHIGS
CALL POPPHI(ERRFIL, BUFA)

C OPEN WORKSTATION
CALL POPMK(PKID, CONID, MTYPE)

C INITIALIZE WORKSTATION
CALL INITMK(PKID)

RETURN
END

C MODULE GMINFO
C COMPILER: FORTRNS (FORTRAN 77)
C CALLING SEQUENCE:
C CALL GMINFO(MTYPE, CONID)
C INPUT PARAMETERS:
C NTYPE - INTEGER, WORKSTATION TYPE
C CWID ·· INTEGER, CONNECTION ID

C COMMON INPUTS:
C NONE

C COMMON OUTPUTS:
C NONE

C LOCAL VARIABLES:
C NIN - INTEGER, INPUT LOGICAL UNIT NUMBER
C NOUT · INTEGER, OUTPUT LOGICAL UNIT NUMBER

C FUNCTIONAL DESCRIPTION:
C ASK USER TO INPUT WORKSTATION TYPE AND CONNECTION ID

C MODULES CALLED:
C Coded by: Sankar Jayaram
C DATE:

SUBROUTINE GMINFO(MTYPE, CONID)
INTEGER MTYPE, CONID
INTEGER NIN, NOUT, DEFAULT
PARAMETER(NIN = 5, NOUT = 6, DEFAULT = 1)

C PROMPT FOR WORKSTATION TYPE
WRITE(NOUT,*)'ENTER WORKSTATION TYPE (INTEGER):'

C GET INTEGER INPUT FROM CONSOLE
READ (NIN, *) MTYPE

C PROMPT FOR WORKSTATION CONNECTION ID
WRITE(NOUT,*)'ENTER WORKSTATION CONNECTION ID (INTEGER):'

C GET INTEGER INPUT FROM CONSOLE
READ (NIN, *) CONID

RETURN
END
C CALLING SEQUENCE:
C CALL INITPKUKID(KID)
C
C INPUT PARAMETERS:
C KID - INTEGER, WORKSTATION IDENTIFIER
C
C COMMON INPUTS:
C NONE
C
C COMMON OUTPUTS:
C NONE
C
C LOCAL VARIABLES:
C NONE
C
C FUNCTIONAL DESCRIPTION:
C CALLS Routine TO INITIALIZE THE WORKSTATION
C
C MODULES CALLED:
C INITCL - INITIALIZE WORKSTATION COLOR TABLE
C INITIN - INITIALIZE INPUT DEVICES
C
C CODED BY: SANKAR JAYARAM
C ADAPTED FROM ACSYNT/VPI CODE WRITTEN BY STEVE HAMPLER
C DATE: 09/01/87 (15:19:29)
C
SUBROUTINE INITPKUKID(KID)
INTEGER KID
C
CALL INITIALIZE COLORS
CALL INITCL(KID)
C
CALL INITIALIZE INPUT DEVICES
CALL INITIN(KID)
C
RETURN
END

MODULE INITCL
C
C CALLING SEQUENCE:
C CALL INITCL(KID)
C
C INPUT PARAMETERS:
C KID - INTEGER, WORKSTATION IDENTIFIER
C
C COMMON INPUTS:
C NONE
C
C COMMON OUTPUTS:
C NONE
C
C LOCAL VARIABLES:
C NCOLI - INTEGER, NUMBER COLOR INDICES TO BE DEFINED
C CI - INTEGER, COLOR INDEX
C RGB(3) - REAL, RED, GREEN, BLUE COLOR VALUES FOR EACH COLOR INDEX
C
C FUNCTIONAL DESCRIPTION:
C Initializes WORKSTATION COLOR TABLE
C
C MODULES CALLED:
C PSCMO - PHIGS SET COLOR MODEL
C PSCR - PHIGS SET COLOR REPRESENTATION
C
C CODED BY: SANKAR JAYARAM
C ADAPTED FROM ACSYNT/VPI CODE WRITTEN BY STEVE HAMPLER
C DATE: 09/01/87 (15:39:10)
C
SUBROUTINE INITCL(KID)
INTEGER KID
C
Appendix D. Code Listing - Sample CAD Program 290
INTEGER CI, NCOLI, PRGB
PARAMETER(NCOLI = 16, PRGB = 0)

REAL RGB(3, NCOLI)

DATA RGB / 0.00, 0.00, 0.00,
> 1.00, 1.00, 1.00,
> 1.00, 1.00, 1.00,
> 1.00, 1.00, 0.00,
> 1.00, 1.00, 0.00,
> 0.00, 1.00, 1.00,
> 1.00, 1.00, 0.00,
> 0.00, 1.00, 0.00,
> 0.50, 0.50, 0.50,
> 0.00, 0.00, 0.00,
> 0.00, 1.00, 0.00,
> 0.00, 1.00, 0.00,
> 1.00, 1.00, 0.00/

C SET COLOR MODEL
CALL PSCMD(MKID, PRGB)

C DO 100 CI = 0, NCOLI-1
CALL PSRUKID(MKID, CI, RGB(1,CI+1), RGB(2, CI+1), RGB(3, CI+1))
100 CONTINUE

RETURN
END

C MODULE INITIN
C=============================================================================
C_COMPILER: FORTVS2 (FORTRAN 77)
C CALLING SEQUENCE:
C CALL INITIN(MKID)
C=============================================================================
C INPUT PARAMETERS:
C MKID - INTEGER, WORKSTATION IDENTIFIER
C=============================================================================
C OUTPUT PARAMETERS:
C None
C=============================================================================
C COMMON INPUTS:
C None
C=============================================================================
C COMMON OUTPUTS:
C None
C=============================================================================
C LOCAL VARIABLES:
C None
C=============================================================================
C FUNCTIONAL DESCRIPTION:
C Initializes input devices
C=============================================================================
C MODULES CALLED:
C AINVl - Initializes valuator device
C INITPK - Initializes pick device
C=============================================================================
C CODED BY: SANKAR JAYARAM
C DATE:
C=============================================================================

SUBROUTINE INITIN(MKID)
 INTEGER MKID

C INITIALIZE VALUATOR DEVICES
DO 100 IDNR = 1, 8
C INITIALIZE VALUATOR DEVICE
CALL AINVl(MKID, IDNR)
100 CONTINUE
C INITIALIZE PICK DEVICE
CALL INITPK(MKID)

RETURN
END

Appendix D. Code Listing - Sample CAD Program 291
SUBROUTINE INITPK(MKID)

INTEGER MKID

COMMON /PICKCL/ NPKCL, PKCL
INTEGER MAXNCL
PARAMETER(MAXNCL = 255)
INTEGER PKCL(MAXNCL)

INTEGER HPDNR, ESN
PARAMETER(HPDNR = 1, ESN = 0)
INTEGER PEVENT, PECHO
PARAMETER(PEVENT = 2, PECHO = 1)

C SET NUMBER OF PICKABLE CLASSES

NPKCL = MAXNCL
DO 100 I = 1, NPKCL
C PUT ITH CLASS INTO PICK FILTER LIST
PKCL(I) = I
100 CONTINUE
C
C SET PICK FILTER

CALL P$HPFT(MKID, HPDNR, NPKCL, PKCL, ESN, ES)
C
C SET PICK MODE TO EVENT

CALL P$HPH(MKID, HPDNR, PEVENT, PECHO)

RETURN
END

SUBROUTINE CLMAIN(MKID)

INTEGER MKID

COMMON /PICKCL/ NPKCL, PKCL
INTEGER MAXNCL
PARAMETER(MAXNCL = 255)
INTEGER PKCL(MAXNCL)

INTEGER HPDNR, ESN
PARAMETER(HPDNR = 1, ESN = 0)
INTEGER PEVENT, PECHO
PARAMETER(PEVENT = 2, PECHO = 1)

C CALL CLMAIN(MKID)

RETURN
END
SUBROUTINE CLMAIN(KID)

INTEGER KID

CLOSE THE WORKSTATION
CALL PCLMK(KID)

CLOSE PHIGS
CALL PCLPH

RETURN

END

MODULE AINV

CALL AINV(LKID, IDNR)

COMMON/VAULUT/ VALUE(N) - REAL, STORAGE FOR CURRENT VALUATOR VALUES

LOCAL VARIABLES:

ERRIND - INTEGER, ERROR INDICATOR RETURNED BY PGDSP
DCLNIT - INTEGER, DEVICE COORDINATE UNITS RETURNED BY PGDSP
LX - INTEGER, MAXIMUM DISPLAY SURFACE SIZE IN RASTER UNITS RETURNED BY PGDSP
LY - INTEGER, MAXIMUM DISPLAY SURFACE SIZE IN RASTER UNITS RETURNED BY PGDSP
RX - REAL, MAXIMUM DISPLAY SURFACE SIZE IN DEVICE COORDINATES RETURNED BY PGDSP
RY - REAL, MAXIMUM DISPLAY SURFACE SIZE IN DEVICE COORDINATES RETURNED BY PGDSP
XMIN - REAL, ECHO AREA EXTENTS
XMAX -
YMIN -
YMAX -
IVAL - REAL, INITIAL VALUATOR VALUE
LOVAL - REAL, LOWER LIMIT ON VALUATOR VALUE
HIVAL - REAL, UPPER LIMIT ON VALUATOR VALUE

FUNCTIONAL DESCRIPTION:

INITIALIZES A SPECIFIED VALUATOR DEVICE

SUBROUTINE AINV(LKID, IDNR)

Appendix D. Code Listing - Sample CAD Program 293
COMMON VALUE
REAL VALUE(8)
INTEGER PKID, IDNR

INTEGER ERRIND, DCUNIT, LX, LY
REAL RX, RY, XMIN, XMAX, YMIN, YMAX
REAL IVAL, LOVAL, HIVAL
PARAMETER(IVAL = 0.5, LOVAL = 0.0, HIVAL = 1.0)
INTEGER PET, PREGU, PEVENT, PECCHO, PNECHO
PARAMETER(PET = 1, PREGU = 0, PEVENT = 2, PECCHO = 0, PNECHO = 1)

C INQUIRE DISPLAY SPACE SIZE
CALL AGDSPKID, ERRIND, DCUNIT, RX, RY, LX, LY)
C CALCULATE ECHO AREA
XMIN = 0.0
XMAX = RX * 0.81
YMIN = RY * (0.95 - 0.02 * FLOAT(IDNR))
YMAX = RY
C INITIALIZE COMMON BLOCK
VALUE(IDNR) = IVAL
C SET VALUATOR MODE TO REQUEST
CALL PSVLHMKID, IDNR, PREGU, PNECHO)
C INITIALIZE VALUATOR
CALL PINVLMKID, IDNR, IVAL, PET, PECUID, XMIN, XMAX, YMIN, YMAX,
> LDVAL, HIVAL, 0, DUHY)
C SET VALUATOR FUNCTION TO EVENT
CALL PSVLHMKID, IDNR, PEVENT, PECCHO)
RETURN
END

C==============================================================================
C MODULE AINST
C==============================================================================
C COMPILER: FORTVSZ (FORTRAN 77)
C CALLING SEQUENCE:
C CALL AINST(MKID, TEXT)
C==============================================================================
C INPUT PARAMETERS:
C MKID - INTEGER, WORKSTATION IDENTIFIER
C TEXT - CHARACTER(*), INITIAL (DEFAULT) CHARACTER STRING
C==============================================================================
C OUTPUT PARAMETERS:
C NONE
C==============================================================================
C COMMON INPUTS:
C NONE
C==============================================================================
C COMMON OUTPUTS:
C NONE
C==============================================================================
C LOCAL VARIABLES:
C STDNR - INTEGER, STRING DEVICE NUMBER
C RX - REAL, DISPLAY SIZE (DC)
C RY
C XMIN - REAL, STRING ECHO AREA LIMITS
C XMAX
C YMIN
C YMAX
C==============================================================================
C FUNCTIONAL DESCRIPTION:
C INITIALIZES STRING INPUT DEVICE
C==============================================================================
C MODULES CALLED:
C AGDSP - INQUIRE ACTUAL DISPLAY SIZE
C PSSTM - PHIGS SET STRING MODE
C CMINST - PHIGS INITIALIZE STRING
C==============================================================================
C CODED BY: SANKAR JAYARAH
C ADAPTED FROM ACSYST/VPI CWE WRITTEN BY STEVE HAMPLER
C DATE: 10/05/87 (19:42:48)
C==============================================================================

SUBROUTINE AINST(MKID, TEXT)
INTEGER MKID, DGD
C INITIALIZE THE VARIABLES FOR THE FEATURE BASED MODELER

INTEGER INSNUM, ASSNUM, OBJNUM, UNUM, MNUM, VIENI, ERFILE
REAL PRIORT
CHARACTER*8 FNAME, ASNAME, OBNAME

C INITIALIZE THE VARIABLES FOR THE FEATURE BASED MODELER

INSNUM = 52
ASSNUM = 50
OBJNUM = 51
ASNAME = 'ASSEMBLY'
OBNAME = 'OBJECT'

NUM = 8
MNUN = 8
VIEHID = 1
PRIORT = 1.0
C SET UP THE FEATURE MENUS
CALL FEATH
C SET UP THE GEOMETRY VIEW
CALL STGMVM(1)
C OPEN THE DESIGN AND MODELING ENVIRONMENT
CALL CKOPDM (MKID, ERFILE, FNAME)
C INITIALIZE THE FEATURE BASED MODELER
CALL CHIFBM (MKID, ASSN, OBJNUM, INNUM, ASNAME, OBNAME,
> UNUM, MNUN, VIEHID, PRIORT)
C CALL THE MENU CONTROL ROUTINE
CALL MCONT
RETURN
END

C=================================================================
C MODULE MCONT
C=================================================================
C CALLING SEQUENCE:
C CALL MCONT
C=================================================================
C INPUT PARAMETERS:
C=================================================================
C OUTPUT PARAMETERS:
C=================================================================
C COMMON INPUTS:
C=================================================================
C COMMON OUTPUTS:
C=================================================================
C LOCAL VARIABLES:
C=================================================================
C FUNCTIONAL DESCRIPTION:
C CONTROLS THE MENU FUNCTIONS
C=================================================================
C MODULES CALLED:
C=================================================================
C CODED BY: SANKAR JAYARAM
C DATE: 03/05/89 (14:07:17)
C=================================================================
CALL CPPUN(1, 2, 1.0)

UPDATE THE WORKSTATION

CALL PLUNK(I, I)

CADMADE AHAIT EVENT
CALL CHMAIT(10000, MKID, ICL, IDNR)

IF (ICL .EQ. 5 .OR. ICL .EQ. 7) THEN

IF THE EVENT IS A PICK OR A MENU PICK, GET THE PICK
CALL CMTTHP(10, ISTAT, IPPD, IPP)

PROCESS A MENU PICK
IF (ICL .EQ. 7) CALL PRMNPK (MKID, IPPD, IPP, EXIT)

ELSEIF (ICL .EQ. 6) THEN

PROCESS A STRING INPUT
CALL CMTSTL (L, STR)

HARD COPY GENERATOR
IF (STR(1:4) .EQ. 'COPY') CALL HRDCPY (MKID, MKID+1)

ELSEIF (ICL .EQ. 3) THEN

PROCESS A VALUATOR INPUT FOR VIEWS TRANSFORMATIONS
CALL PGMVLA (MKID, IDNR)

ENDIF

CHECK THE EXIT FLAG
IF (EXIT .NE. 1) GO TO 200

RETURN

END

=====================================================================

MODULE PRMNPK

COMPILER: FORTY2 (FORTRAN 77)

CALLING SEQUENCE:
CALL PRMNPK (MKID, IPPD, IPP, EXIT)

INPUT PARAMETERS:

OUTPUT PARAMETERS:

COMMON INPUTS:
NONE

COMMON OUTPUTS:
NONE

LOCAL VARIABLES:

FUNCTIONAL DESCRIPTION:
PROCESS A MAIN MENU PICK

MODULES CALLED:
FEATURE BASED MODELING ROUTINES ARE USED IN THIS MODULE

CODED BY: SANKAR JAYARAM
DATE: 03/05/89 (14:07:17)

SUBROUTINE PRMNPK (MKID, IPPD, IPP, EXIT)

INTEGER MKID, IPPD, IPP(3,10), EXIT
REAL ASIZE(6), CSIZE(3), DATA(12), PAREA(6)
INTEGER PPATH(3)
REAL LOC(3), ORI(3), A, B, C, D, AR
INTEGER COLOR, IFL, IFL1
CHARACTER BOOL*1

PROCESS A MAIN MENU PICK
IF (IPP(1, IPP) .LT. 1000) THEN
IF (IPP(2, IPPD) .EQ. 15) THEN

C  EXIT THE ROUTINE
EXIT = 1

ELSE

C  PROCESS THE MENU PICK TO DISPLAY THE NEXT MENU
CALL CH00DN (MKID, IPPD, IPP)

ENDIF

ELSEIF (IPP(2, IPPD) .GT. 1110 .AND. IPP(2, IPPD) .LT. 1120) THEN

C  CALL THE TEMPLATE FOR THE CYLINDER
CALL CYNDR (A, B, LOC, ORI, COLOR, BOOL)

ELSEIF (IPP(2, IPPD) .GT. 1120 .AND. IPP(2, IPPD) .LT. 1130) THEN

C  CALL THE TEMPLATE FOR THE BUSHING
CALL MBUSH (A, B, C, LOC, ORI, COLOR, BOOL)

ELSEIF (IPP(2, IPPD) .GT. 1130 .AND. IPP(2, IPPD) .LT. 1140) THEN

C  CALL THE TEMPLATE FOR THE TUNNEL
CALL TUNPL (A, B, C, LOC, ORI, COLOR, BOOL)

ELSEIF (IPP(2, IPPD) .GT. 1140 .AND. IPP(2, IPPD) .LT. 1150) THEN

C  CALL THE TEMPLATE FOR THE SPHERE
CALL SPHE (A, LOC, COLOR, BOOL)

ELSEIF (IPP(2, IPPD) .GT. 1150 .AND. IPP(2, IPPD) .LT. 1160) THEN

C  CALL THE TEMPLATE FOR THE HEMISPHERE
CALL HEMSPH (A, LOC, ORI, COLOR, BOOL)

ELSEIF (IPP(2, IPPD) .GT. 1160 .AND. IPP(2, IPPD) .LT. 1170) THEN

C  CALL THE TEMPLATE FOR THE ELLIPSOID
CALL ELLIPSOID (A, B, LOC, ORI, COLOR, BOOL)

ELSEIF (IPP(2, IPPD) .GT. 1170 .AND. IPP(2, IPPD) .LT. 1180) THEN

C  CALL THE TEMPLATE FOR THE SLAB
CALL BLAB (A, B, C, LOC, ORI, COLOR, BOOL)

ELSEIF (IPP(2, IPPD) .GT. 1180 .AND. IPP(2, IPPD) .LT. 1190) THEN

C  CALL THE TEMPLATE FOR THE PRISM
CALL PRISM (A, B, C, AR, IFL, LOC, ORI, COLOR, BOOL)

ELSEIF (IPP(2, IPPD) .GT. 1190 .AND. IPP(2, IPPD) .LT. 1200) THEN

C  CALL THE TEMPLATE FOR THE MEGE
CALL MEGE (A, B, C, LOC, ORI, COLOR, BOOL)

ELSEIF (IPP(2, IPPD) .GT. 1200 .AND. IPP(2, IPPD) .LT. 1210) THEN

C  CALL THE TEMPLATE FOR THE RING
CALL RING (A, B, LOC, ORI, COLOR, BOOL)

ELSEIF (IPP(2, IPPD) .GT. 1210 .AND. IPP(2, IPPD) .LT. 1220) THEN

C  CALL THE TEMPLATE FOR THE COUNTERSINK OR COUNTERBORE
CALL MOUNT (A, B, C, D, IFL, IFL1, LOC, ORI, COLOR, BOOL)

ELSEIF (IPP(2, IPPD) .GT. 1220 .AND. IPP(2, IPPD) .LT. 1230) THEN

C  CALL THE TEMPLATE FOR THE CONE
CALL CONE (A, B, LOC, ORI, COLOR, BOOL)

C  PROCESS A UTILITY MENU PICK
ELSEIF (IPP(2, IPPD) .GT. 2000) THEN

C  CALL THE ERASE MODULE
CALL ERASE (MKID, IPPD, IPP)

ENDIF

IF (IPP(2, IPPD) .GT. 1000 .AND. IPP(2, IPPD) .LT. 2000) THEN

C  RE-INITIALIZE THE PICK DEVICE AND THE PICK FILTERS
*GRAPHICS CALLS
CALL PSHPF7 (1, 1, 1, 127, 1, 126)
CALL GPQADS(1, IER, UNITS, CSIZE, ASIZE)
PAREA(1) = 0.0
PAREA(2) = CSIZE(1)
PAREA(3) = 0.0
PAREA(4) = CSIZE(2)
PAREA(5) = 0.0
PAREA(6) = CSIZE(3)
CALL GPKKM0 (1, 1, 1, 2)
CALL GPINPK(1, 1, 0, PPATH, 1, PAREA, 0, DATA, 1)
CALL GPKKM0 (1, 1, 5, 2)
ENDIF
RETURN
END

C MODULE MERASE
C COMPILER: FORTVS2 (FORTRAN 77)
C CALLING SEQUENCE:
CALL MERASE (PKID, IPPD, IPP)
C INPUT PARAMETERS:
C OUTPUT PARAMETERS:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C FUNCTIONAL DESCRIPTION:
C PROCESS A UTILITY MENU PICK
C MODULES CALLED:
C CODED BY: SANKAR JAYARAM
C DATE: 03/05/89 (14:07:17)
SUBROUTINE MERASE (MKID, IPPD, IPP)
INTEGER MKID, IPPD, IPP(3,10)
INTEGER IPP1, IPP1(3,10)
INTEGER NSNUM, ASSNUM, OBJNUM, UNUM, MINUM, VIENID
REAL PRIORIT
CHARACTER*8 ASNAME, OBNAME
C SET UP THE VARIABLES TO INITIALIZE THE FEATURE BASED MODELER
INSNUM = 52
ASSNUM = 50
OBJNUM = 51
ASNAME = 'ASSEMBLY'
OBNAME = 'OBJECT'
UNUM = 8
MNUN = 8
VIENID = 1
PRIORIT = 1.0
C WRITE A MESSAGE
CALL CMMSG (MKID, 1, 'SELECT ITEM TO ERASE OR SELECT MENU')
C UPDATE THE WORKSTATION
CALL PUMP (MKID, 1)
C WAIT FOR CADMADE EVENT
CALL CMWAI(I10000., MKID, ICL, IDNR)
C SET THE PICK
CALL CMTDH(10, ISTAT, IPPD1, IPP1)

IF (ICL .EQ. 5) THEN

C ERASE THE APPROPRIATE OBJECT OR INSTANCE
IF (IPP(2, IPPD) .EQ. 2001) THEN
CALL PEMST (IPP1(2, IPPD1))
ELSEIF (IPP(2, IPPD) .EQ. 2002) THEN
CALL PEMST (IPP1(2, IPPD1-1))

C RE-INITIALIZE THE FEATURE BASED MODELER
CALL CHIFBH (MKID, ASSNUM, OBJNUM, INSHM, ASNAME, OBNAME, >
  LINUM, MNAM, VIEMID, PRIORT)
ENDIF
ELSEIF (ICL .EQ. 7) THEN

C DISPLAY NEXT MENU IF A MENU WAS PICKED
CALL CMNNCH (MKID, IPPD1, IPP1)
ENDIF

C H O D U L E
C CGIPILER: FORTVSZ FORTRAN 77
C CALLING SEQUENCE:
C CALL FEATH

INPUT PARAMETERS:

ROUTINE FEATH

SUBROUTINE FEATH

INTEGER LENGTH, NPTBFA, NPBFALO, ILENGTH(10), MENUID(10)
REAL XBFALO(50), YBFALO(50)
REAL XBFAT(10), YBFAT(10)
REAL XLOC(20), YLOC(20)
CHARACTER TITLE*15, ITEMS(10)*10
CHARACTER PROMPT*80

INTEGER KERFIL, MKID, HPDNR
CHARACTER*7 FILNAM
CHARACTER*10 CLEAR
PARAMETER(CLEAR = ' ')

PARAMETER (MKID = 1, HPDNR = 1)
PARAMETER (KERFIL=90, FILNAM='MENUEIR')

C OPEN THE USER INTERFACE ENVIRONMENT
CALL CHOPUI (MKID, HPDNR, KERFIL, FILNAM)

C INITIALIZE STRING INPUT
CALL AINST (MKID, CLEAR)
C OPEN MENU STRUCTURE
CALL CHMST (1)

C MENU PRIMITIVES FOR MENU TITLE

C DEFINE THE LENGTH OF THE MENU TITLE TEXT
LENGTH = 12

C DEFINE THE NO. OF POINTS IN THE BACKGROUND FILL AREA
NPTBFA = 4

C DEFINE THE BACKGROUND AREA
XBFAT (1) = -.15
XBFAT (2) = +.15
XBFAT (3) = +.15
XBFAT (4) = -.15
YBFAT (1) = -.05
YBFAT (2) = -.05
YBFAT (3) = +.05
YBFAT (4) = +.05

C DEFINE THE TITLE TEXT
TITLE = ' SHAPES '

C DRAM THE MENU TITLE TEXT
CALL CHMTXT (LENGTH, NPTBFA, XBFAT, YBFAT, TITLE)

C MENU ITEMS

C SET THE NUMBER OF MENU ITEMS
NITEMS = 5

C SET THE ITEM TEXT LENGTHS
ILENGT(1) = 12
ILENGT(2) = 12
ILENGT(3) = 12
ILENGT(4) = 12
ILENGT(5) = 12
ILENGT(6) = 12
ILENGT(7) = 12
ILENGT(8) = 12

C DEFINE THE NO. OF VERTICES FOR THE FILL AREAS
NPBFA(1) = 4
NPBFA(2) = 4
NPBFA(3) = 4
NPBFA(4) = 4
NPBFA(5) = 4
NPBFA(6) = 4
NPBFA(7) = 4
NPBFA(8) = 4

C SET THE MENU IDENTIFIERS FOR THE MENU ITEMS
MENUID(1) = 11
MENUID(2) = 12
MENUID(3) = 13
MENUID(4) = 14
MENUID(5) = 15

C DEFINE THE FILL AREAS
XBF(1) = -.15
XBF (2) = +.15
XBF (3) = +.15
XBF (4) = -.15
XBF (5) = -.15
XBF (6) = +.15
XBF (7) = +.15
XBF (8) = -.15
XBF (9) = -.15
XBF (10) = +.15
XBF (11) = +.15
XBF (12) = -.15
XBF (13) = -.15
XBF (14) = +.15
XBF (15) = +.15
XBF (16) = +.15
XBF (17) = +.15
XBF (18) = +.15
XBF (19) = +.15
XBF (20) = -.15
XBF (21) = -.15
XBF (22) = +.15
XBF (23) = +.15
C DEFINE THE ITEM TEXT STRINGS
ITEMS(1) = 'CYLINDRICAL'
ITEMS(2) = 'ELLIPICAL'
ITEMS(3) = 'FLAT'
ITEMS(4) = 'OTHERS'
ITEMS(5) = 'EXIT'

C DRAW THE MENU ITEMS
CALL CMMTX (NITEMS, ILEIG, NPBFA, MEMID, XBFA, YBFA, ITEMS)

C SET MENU PROMPT STRING
PROMPT = 'SELECT SHAPE'
CALL CIMPST (80, PROMPT)

C EXECUTE MENU SUB STRUCTURES
CALL CHELHN(11)
CALL CHELHN(12)
CALL CHELHN(13)
CALL CHELHN(14)
CALL CHELHN(15)

C CLOSE MENU STRUCTURE
CALL CMCHST

C OPEN MENU STRUCTURE
CALL CMCHST (11)

C MENU PRIMITIVES
C DEFINE THE TITLE TEXT
TITLE = 'CYLINDRICAL'
C DRAW THE MENU TITLE TEXT
CALL CMMTTX (LENGTH, NPTBFA, XBFA, YBFA, TITLE)
C SET THE NUMBER OF MENU ITEMS
ITEMS = 4
C SET THE MENU IDENTIFIERS FOR THE MENU ITEMS
MEMID(1) = 1

Appendix D. Code Listing - Sample CAD Program 302
C DEFINE THE ITEM TEXT STRINGS
ITEMS(1) = ' CYLINDER ';
ITEMS(2) = ' BUSHING ';
ITEMS(3) = ' TUNNEL ';
ITEMS(4) = ' RETURN ';

C DRAM THE MENU ITEMS
CALL CHMTX (NITEMS, ILNGTH, NPBFRA, MENUID, XBFRA, YBFRA, ITEMS)

C SET MENU PROMPT STRING
PROMPT = 'SELECT CYLINDRICAL SHAPE'
CALL CHMPS (80, PROMPT)

C EXECUTE MENU SUB STRUCTURES
CALL CMEX(111)
CALL CMEX(112)
CALL CMEX(113)

C CLOSE MENU STRUCTURE
CALL CMCHST

C OPEN MENU STRUCTURE
CALL CMCHST (12)

C MENU PRIMITIVES

C DEFINE THE TITLE TEXT
TITLE = ' CYLINDRICAL ';

C DRAM THE MENU TITLE TEXT
CALL CHMTX (ILNGTH, NPBFRA, XBFAT, YBFAT, TITLE)

C SET THE NUMBER OF MENU ITEMS
NITEMS = 4

C SET THE MENU IDENTIFIERS FOR THE MENU ITEMS
MENUID(1) = 121
MENUID(2) = 122
MENUID(3) = 123
MENUID(4) = -1

C DEFINE THE ITEM TEXT STRINGS
ITEMS(1) = ' SPHERE ';
ITEMS(2) = ' HEMISPHERE ';
ITEMS(3) = ' ELLIPSOID ';
ITEMS(4) = ' RETURN ';

C DRAM THE MENU ITEMS
CALL CHMTX (NITEMS, ILNGTH, NPBFRA, MENUID, XBFRA, YBFRA, ITEMS)

C SET MENU PROMPT STRING
PROMPT = 'SELECT CYLINDRICAL SHAPE'
CALL CHMPS (80, PROMPT)

C EXECUTE MENU SUB STRUCTURES
CALL CMEX(111)
CALL CMEX(112)
CALL CMEX(113)

C CLOSE MENU STRUCTURE
CALL CMCHST

C OPEN MENU STRUCTURE
CALL CMCHST (13)

C MENU PRIMITIVES

C DEFINE THE TITLE TEXT
TITLE = ' ELLIPTICAL ';

C DRAM THE MENU TITLE TEXT
CALL CHMTX (ILNGTH, NPBFRA, XBFAT, YBFAT, TITLE)

C DEFINE THE NUMBER OF ITEMS
NITEMS = 5

C SET THE MENU IDENTIFIERS FOR THE MENU ITEMS
MENUID(1) = 131
MENUID(2) = 132
MENUID(3) = 133

 Appendix D. Code Listing - Sample CAD Program 303
MENJID(4) = 134
MENJID(5) = -1

C DEFINE THE ITEM TEXT STRINGS
ITEMS(1) = 'SLAB'
ITEMS(2) = 'PRISM'
ITEMS(3) = 'MEDGE'
ITEMS(4) = 'RING'
ITEMS(5) = 'RETURN'

C DRAW THE MENU ITEMS
CALL CMMITX (NITEMS, ILENGT, NBFA, MENJID, XBFA, YBFA, ITEMS)

C SET MENU PROMPT STRING
PROMPT = 'SELECT FLAT SHAPE'
CALL CHMPST(80, PROMPT)

C EXECUTE MENU SUB STRUCTURES
CALL CMEXMN(131)
CALL CMEXMN(132)
CALL CMEXMN(133)
CALL CMEXMN(134)

C CLOSE MENU STRUCTURE
CALL CMCHST

C---- OPEN MENU STRUCTURE
CALL CMCHST (14)

C MENU PRIMITIVES

C DEFINE THE TITLE TEXT
TITLE = 'OTHER'

C DRAW THE MENU TITLE TEXT
CALL CMMTTX (LENGTH, NPTBFA, XBFA, YBFA, TITLE)

C DEFINE THE NUMBER OF ITEMS
NITEMS = 4

C SET THE MENU IDENTIFIERS FOR THE MENU ITEMS
MENJID(1) = 141
MENJID(2) = 142
MENJID(3) = -1

C DEFINE THE ITEM TEXT STRINGS
ITEMS(1) = 'HOLE'
ITEMS(2) = 'CONE'
ITEMS(3) = 'RETURN'

C DRAW THE MENU ITEMS
CALL CMMITX (NITEMS, ILENGT, NBFA, MENJID, XBFA, YBFA, ITEMS)

C SET MENU PROMPT STRING
PROMPT = 'SELECT OTHER SHAPE'
CALL CHMPST(80, PROMPT)

C EXECUTE MENU SUB STRUCTURES
CALL CMEXMN(141)
CALL CMEXMN(142)

C CLOSE MENU STRUCTURE
CALL CMCHST

C---- OPEN MENU STRUCTURE
CALL CMCHST (111)

C MENU PRIMITIVES

C DEFINE THE TITLE TEXT
TITLE = 'CYLINDER'

C DRAW THE MENU TITLE TEXT
CALL CMMTTX (LENGTH, NPTBFA, XBFA, YBFA, TITLE)

C DEFINE THE NUMBER OF ITEMS
NITEMS = 8

C SET THE MENU IDENTIFIERS FOR THE MENU ITEMS
MENJID(1) = 1111
MENJID(2) = 1112
MENJID(3) = 1113
MENJID(4) = 1114
C DEFINE THE ITEM TEXT STRINGS
ITEMS(1) = ' BAR '
ITEMS(2) = ' BEAM '
ITEMS(3) = ' BOSS '
ITEMS(4) = ' BUTTON '
ITEMS(5) = ' DISC '
ITEMS(6) = ' PIN '
ITEMS(7) = ' RETURN '
ITEMS(8) = ' ROOT '

C DRAW THE MENU ITEMS
CALL CMBITX (NITEMS, ILENGT, NPBFA, MENUID, XBFA, YBFA, ITEMS)

C SET MENU PROMPT STRING
PROMPT = 'SELECT PARAMETER'
CALL CMHPST (80, PROMPT)

C CLOSE MENU STRUCTURE
CALL CMCHST

C MENU PRIMITIVES

C DEFINE THE TITLE TEXT
TITLE = ' BUSHING '

C DRAW THE MENU TITLE TEXT
CALL CMBTIX (LENGTH, NPTBFA, XBFA, YBFA, TITLE)

C DEFINE THE NUMBER OF ITEMS
NITEMS = 8

C SET THE MENU IDENTIFIERS FOR THE MENU ITEMS
MENUID(1) = 1121
MENUID(2) = 1122
MENUID(3) = 1123
MENUID(4) = 1124
MENUID(5) = 1125
MENUID(6) = 1126
MENUID(7) = -1
MENUID(8) = 0

C DEFINE THE ITEM TEXT STRINGS
ITEMS(1) = ' COLLAR '
ITEMS(2) = ' GASKET '
ITEMS(3) = ' GLAND '
ITEMS(4) = ' PACKING '
ITEMS(5) = ' PIPE '
ITEMS(6) = ' TUBE '
ITEMS(7) = ' RETURN '
ITEMS(8) = ' ROOT '

C DRAW THE MENU ITEMS
CALL CMBITX (NITEMS, ILENGT, NPBFA, MENUID, XBFA, YBFA, ITEMS)

C SET MENU PROMPT STRING
PROMPT = 'SELECT PARAMETER'
CALL CMHPST (80, PROMPT)

C CLOSE MENU STRUCTURE
CALL CMCHST

C MENU PRIMITIVES

C DEFINE THE TITLE TEXT
TITLE = ' TUNNEL '

C DRAW THE MENU TITLE TEXT
CALL CMBTIX (LENGTH, NPTBFA, XBFA, YBFA, TITLE)

C DEFINE THE NUMBER OF ITEMS
NITEMS = 6
C SET THE MENU IDENTIFIERS FOR THE MENU ITEMS
   MENUID(1) = 1131
   MENUID(2) = 1132
   MENUID(3) = 1133
   MENUID(4) = 1134
   MENUID(5) = -1
   MENUID(6) = 0

C DEFINE THE ITEM TEXT STRINGS
   ITEMS(1) = ' RACE '
   ITEMS(2) = ' ARCH '
   ITEMS(3) = ' BOM '
   ITEMS(4) = ' CRESCENT '
   ITEMS(5) = ' RETURN '
   ITEMS(6) = ' ROOT '

C DRAW THE MENU ITEMS
   CALL CMHITX (NITEMS, ILENGT, NPBF, MENUID, XBFA, YBFA, ITEMS)

C SET MENU PROMPT STRING
   PROMPT = 'SELECT PARAMETER'
   CALL CMHPST (80, PROMPT)

C CLOSE MENU STRUCTURE
   CALL CMCHST

C OPEN MENU STRUCTURE
   CALL CMCHST (121)

C MENU PRIMITIVES

C DEFINE THE TITLE TEXT
   TITLE = ' SPHERE '

C DRAW THE MENU TITLE TEXT
   CALL CMHTTX (LENGTH, NPBTFA, XBFA, YBFA, TITLE)

C DEFINE THE NUMBER OF ITEMS
   NITEMS = 6

C SET THE MENU IDENTIFIERS FOR THE MENU ITEMS
   MENUID(1) = 1211
   MENUID(2) = 1212
   MENUID(3) = 1213
   MENUID(4) = 1214
   MENUID(5) = -1
   MENUID(6) = 0

C DEFINE THE ITEM TEXT STRINGS
   ITEMS(1) = ' BALL '
   ITEMS(2) = ' BULB '
   ITEMS(3) = ' GLOBULE '
   ITEMS(4) = ' ROUNDING '
   ITEMS(5) = ' RETURN '
   ITEMS(6) = ' ROOT '

C DRAW THE MENU ITEMS
   CALL CMHITX (NITEMS, ILENGT, NPBF, MENUID, XBFA, YBFA, ITEMS)

C SET MENU PROMPT STRING
   PROMPT = 'SELECT PARAMETER'
   CALL CMHPST (80, PROMPT)

C CLOSE MENU STRUCTURE
   CALL CMCHST

C OPEN MENU STRUCTURE
   CALL CMCHST (122)

C MENU PRIMITIVES

C DEFINE THE TITLE TEXT
   TITLE = ' HEMISPHERE '

C DRAW THE MENU TITLE TEXT
   CALL CMHTTX (LENGTH, NPBTFA, XBFA, YBFA, TITLE)

C DEFINE THE NUMBER OF ITEMS
   NITEMS = 7

C SET THE MENU IDENTIFIERS FOR THE MENU ITEMS
MENJID(1) = 1221
MENJID(2) = 1222
MENJID(3) = 1223
MENJID(4) = 1224
MENJID(5) = 1225
MENJID(6) = -1
MENJID(7) = 0

C DEFINE THE ITEM TEXT STRINGS
ITEMS(1) = ' CUP '
ITEMS(2) = ' DOME '
ITEMS(3) = ' CAP '
ITEMS(4) = ' SCOOH '
ITEMS(5) = ' LADLE '
ITEMS(6) = ' RETURN '
ITEMS(7) = ' ROOT '

C DRAM THE MENU ITEMS
CALL CMNITX (NITEMS, ILENGT, NPBFPA, MENJID, XBFA, YBFA, ITEMS)

C SET MENU PROMPT STRING
PROMPT = 'SELECT PARAMETER'
CALL CHMPST (80, PROMPT)

C CLOSE MENU STRUCTURE
CALL CMCHST

C OPEN MENU STRUCTURE
CALL CMHNSR (123)

C MENU PRIMITIVES
C DEFINE THE TITLE TEXT
TITLE = ' ELLIPSOID '
C DRAM THE MENU TITLE TEXT
CALL CMNITX (ILENGT, NPBFPA, XBFA, YBFA, TITLE)

C DEFINE THE NUMBER OF ITEMS
NITEMS = 3
C SET THE MENU IDENTIFIERS FOR THE MENU ITEMS
MENJID(1) = 1231
MENJID(2) = -1
MENJID(3) = 0

C DEFINE THE ITEM TEXT STRINGS
ITEMS(1) = ' OVAL '
ITEMS(2) = ' RETURN '
ITEMS(3) = ' ROOT '

C DRAM THE MENU ITEMS
CALL CMNITX (NITEMS, ILENGT, NPBFPA, MENJID, XBFA, YBFA, ITEMS)

C SET MENU PROMPT STRING
PROMPT = 'SELECT PARAMETER'
CALL CHMPST (80, PROMPT)

C CLOSE MENU STRUCTURE
CALL CMCHST

C OPEN MENU STRUCTURE
CALL CMHNSR (131)

C MENU PRIMITIVES
C DEFINE THE TITLE TEXT
TITLE = ' SLAB '
C DRAM THE MENU TITLE TEXT
CALL CMNITX (ILENGT, NPBFPA, XBFA, YBFA, TITLE)

C DEFINE THE NUMBER OF ITEMS
NITEMS = 7
C SET THE MENU IDENTIFIERS FOR THE MENU ITEMS
MENJID(1) = 1311
MENJID(2) = 1312
MENJID(3) = 1313
MENJID(4) = 1314
MENJID(5) = 1315
MENJID(6) = -1
C DEFINE THE ITEM TEXT STRINGS
ITEMS(1) = 'BASE'
ITEMS(2) = 'BILLET'
ITEMS(3) = 'BLOCK'
ITEMS(4) = 'CUBE'
ITEMS(5) = 'PAD'
ITEMS(6) = 'RETURN'
ITEMS(7) = 'ROOT'

C DRAW THE MENU ITEMS
CALL CHMITX (NITEMS, ILENGT, NPBFA, MENUID, XBFA, YBFA, ITEMS)

C SET MENU PROMPT STRING
PROMPT = 'SELECT PARAMETER'
CALL CHMPST (80, PROMPT)

C CLOSE MENU STRUCTURE
CALL CHMST

C OPEN MENU STRUCTURE
CALL CHMST (132)

C MENU PRIMITIVES
C DEFINE THE TITLE TEXT
TITLE = 'PRISM'
C DRAW THE MENU TITLE TEXT
CALL CHMITX (LENGTH, NPFBFA, XBFA, YBFAT, TITLE)
C DEFINE THE NUMBER OF ITEMS
NITEMS = 3
C SET THE MENU IDENTIFIERS FOR THE MENU ITEMS
MENUID(1) = 1321
MENUID(2) = -1
MENUID(3) = 0
C DEFINE THE ITEM TEXT STRINGS
ITEMS(1) = 'PRISM'
ITEMS(2) = 'RETURN'
ITEMS(3) = 'ROOT'
C DRAW THE MENU ITEMS
CALL CHMITX (NITEMS, ILENGT, NPBFA, MENUID, XBFA, YBFA, ITEMS)
C SET MENU PROMPT STRING
PROMPT = 'SELECT PARAMETER'
CALL CHMPST (80, PROMPT)

C CLOSE MENU STRUCTURE
CALL CHMST

C OPEN MENU STRUCTURE
CALL CHMST (133)

C MENU PRIMITIVES
C DEFINE THE TITLE TEXT
TITLE = 'MEDGE'
C DRAW THE MENU TITLE TEXT
CALL CHMITX (LENGTH, NPFBFA, XBFA, YBFAT, TITLE)
C DEFINE THE NUMBER OF ITEMS
NITEMS = 3
C SET THE MENU IDENTIFIERS FOR THE MENU ITEMS
MENUID(1) = 1331
MENUID(2) = -1
MENUID(3) = 0
C DEFINE THE ITEM TEXT STRINGS
ITEMS(1) = 'MEDGE'
ITEMS(2) = 'RETURN'
ITEMS(3) = 'ROOT'
C DRAW THE MENU ITEMS
CALL CHMITX (NITEMS, ILENGT, NPBFA, MENUID, XBFA, YBFA, ITEMS)
C SET MENU PROMPT STRING
PROMPT = 'SELECT PARAMETER'
CALL CHMPST (80, PROMPT)

C CLOSE MENU STRUCTURE
CALL CHCMST

C OPEN MENU STRUCTURE
CALL CHCMST (134)

C MENU PRIMITIVES

C DEFINE THE TITLE TEXT,
TITLE = 'RING'

C DRAW THE MENU TITLE TEXT
CALL CMITTX (LENGTH, NPTBFA, XBFAT, YBFAT, TITLE)

C DEFINE THE NUMBER OF ITEMS
NITEMS = 3

C SET THE MENU IDENTIFIERS FOR THE MENU ITEMS
MENUID(1) = 1341
MENUID(2) = -1
MENUID(3) = 0

C DEFINE THE ITEM TEXT STRINGS
ITEMS(1) = 'RING'
ITEMS(2) = 'RETURN'
ITEMS(3) = 'ROOT'

C DRAW THE MENU ITEMS
CALL CMITX (NITEMS, ILENGTH, NPTBFA, MENUID, XBFAT, YBFAT, ITEMS)

C SET MENU PROMPT STRING
PROMPT = 'SELECT PARAMETER'
CALL CHMPST (80, PROMPT)

C CLOSE MENU STRUCTURE
CALL CHCMST

C OPEN MENU STRUCTURE
CALL CHCMST (141)

C MENU PRIMITIVES

C DEFINE THE TITLE TEXT,
TITLE = 'HOLE'

C DRAW THE MENU TITLE TEXT
CALL CMITTX (LENGTH, NPTBFA, XBFAT, YBFAT, TITLE)

C DEFINE THE NUMBER OF ITEMS
NITEMS = 4

C SET THE MENU IDENTIFIERS FOR THE MENU ITEMS
MENUID(1) = 1411
MENUID(2) = 1412
MENUID(3) = -1
MENUID(4) = 0

C DEFINE THE ITEM TEXT STRINGS
ITEMS(1) = 'COUNTERBORE'
ITEMS(2) = 'COUNTERSINK'
ITEMS(3) = 'RETURN'
ITEMS(4) = 'ROOT'

C DRAW THE MENU ITEMS
CALL CMITX (NITEMS, ILENGTH, NPTBFA, MENUID, XBFAT, YBFAT, ITEMS)

C SET MENU PROMPT STRING
PROMPT = 'SELECT PARAMETER'
CALL CHMPST (80, PROMPT)

C CLOSE MENU STRUCTURE
CALL CHCMST

C OPEN MENU STRUCTURE
CALL CHCMST (142)

Appendix D. Code Listing - Sample CAD Program 309
C MENU PRIMITIVES
C DEFINE THE TITLE TEXT
    TITLE = 'CONE';
C DRAW THE MENU TITLE TEXT
    CALL CMNTTX (LENGTH, NPTBFA, XBFAT, YBFAT, TITLE)
C DEFINE THE NUMBER OF ITEMS
    NITEMS = 3
C SET THE MENU IDENTIFIERS FOR THE MENU ITEMS
    MENUID(1) = 1421
    MENUID(2) = -1
    MENUID(3) = 0
C DEFINE THE ITEM TEXT STRINGS
    ITEMS(1) = 'CONE'
    ITEMS(2) = 'RETURN'
    ITEMS(3) = 'ROOT'
C DRAW THE MENU ITEMS
    CALL CMHITX (NITEMS, ILEFST, IPBFA, MENUID, XBFAT, YBFAT, ITEMS)
C SET MENU PROMPT STRING
    PROMPT = 'SELECT PARAMETER'
    CALL CMPPST (80, PROMPT)
C CLOSE MENU STRUCTURE
    CALL CMCMST
C DEFINE THE SECOND MENU TREE
C C MENU PRIMITIVES
C CALL CMCKST (2)
C C MENU PRIMITIVES STRUCTURES
C CALL CMCMST
C C DEFINE THE SECOND MENU STRUCTURE
C C MENU PRIMITIVES
C SET THE NUMBER OF MENU ITEMS
    NITEMS = 1
C SET THE MENU IDENTIFIER
    MENUID(1) = 21
C DEFINE THE MENU ITEM TEXT
    ITEMS(1) = 'UTILITIES'
C SET THE MENU LOCATION
    XLOC(1) = 0.7
    YLOC(1) = -0.1
    CALL CMHIC (1, XLOC, YLOC)
C DRAW THE MENU ITEM
    CALL CMHITX (NITEMS, ILEFST, IPBFA, MENUID, XBFAT, YBFAT, ITEMS)
C EXECUTE MENU SUB STRUCTURES
    CALL CMEXMN(21)
C C MENU PRIMITIVES
C DEFINE THE MENU TITLE LOCATION
    XLOC(1) = 0.7
    YLOC(1) = -0.1
    CALL CMNTLC (XLOC(1), YLOC(1))
C C MENU PRIMITIVES
C DEFINE THE MENU TITLE TEXT
    TITLE = 'UTILITIES'
C DRAW THE MENU TITLE TEXT
    CALL CMNTTX (LENGTH, NPTBFA, XBFAT, YBFAT, TITLE)
C DEFINE THE NUMBER OF MENU ITEMS
    NITEMS = 3
C SET THE MENU IDENTIFIERS
    MENUID(1) = 2001
    MENUID(2) = 2002
    MENUID(3) = -1

Appendix D. Code Listing - Sample CAD Program
C DEFINE THE MENU ITEM TEXT STRINGS
ITEMS(1) = 'ERASE INST.'
ITEMS(2) = 'ERASE OBJECT'
ITEMS(3) = 'RETURN'
C DEFINE THE MENU ITEM LOCATIONS
XLOC(1) = 0.7
YLOC(1) = -0.225
XLOC(2) = 0.7
YLOC(2) = -0.325
XLOC(3) = 0.7
YLOC(3) = -0.425
CALL CHMILC (3, XLOC, YLOC)
C DRAM THE MENU ITEMS
CALL CHMITX (NITEMS, ILENGT, NPBFA, MENUID, XBFA, YBFA, ITEMS)
C SET MENU PROMPT STRING
PROMPT = 'SELECT UTILITY ITEM'
CALL CHMPST (80, PROMPT)
C CLOSE MENU STRUCTURE
CALL CMMCST
RETURN
END

C=======================================================================
C MODULE ASVM3
C=======================================================================
C COMPILER: FORTRAN 77
C CALLING SEQUENCE:
C CALL ASVM3(MKID, VIENID, VALUE)
C INPUT PARAMETERS:
C MKID - INTEGER, WORKSTATION IDENTIFIER
C VIENID - INTEGER, VIEN IDENTIFIER
C VALUE(*) - REAL, VIEN TRANSFORMATION VALUES
C=======================================================================
C OUTPUT PARAMETERS:
C NONE
C=======================================================================
C COMMON INPUTS:
C NONE
C=======================================================================
C COMMON OUTPUTS:
C NONE
C=======================================================================
C LOCAL VARIABLES:
C MAT(4,4) - REAL, VIEN MATRIX
C=======================================================================
C FUNCTIONAL DESCRIPTION:
C Computes and sets a VIEN matrix given the VIEN transformation values
C=======================================================================
C CODED BY: SANKAR JAYARAM
C ADAPTED FROM ACSTN/VPI CODE WRITTEN BY STEVE NANDLER
C DATE: 09/22/87 (12:39:57)
C=======================================================================

SUBROUTINE ASVM3(MKID, VIENID, VALUE)
INTEGER MKID, VIENID
REAL VALUE(*)
REAL MAT(4,4)
C GET XFORM MATRIX
CALL GTXFM1(VALUE, MAT)
C SET VIEN MATRIX
CALL PSVHM3(MKID, VIENID, MAT)
RETURN
END

C=======================================================================
C MODULE PGMVL
C=======================================================================
C COMPILER: FORTRAN 77
C CALLING SEQUENCE:
C CALL PGMVL(MKID, IDNR)
SUBROUTINE PGMVL(VKID, IDNR)

INTEGER VKID, IDNR

COMMON /VALUAT/ VALUE
REAL VALUE
COMMON /GMVFL/ VMFL
REAL VMFL
COMMON /GMVIEW/ CURVM
INTEGER CURVM

INTEGER NVALS
PARAMETER(NVALS = 8)
REAL VLVAL, MAX(NVALS), MIN(NVALS)

DATA MIN / 0.0, 0.0, 0.0, -30.0, -30.0, -30.0, 0.1, 0.0 /
DATA MAX / 360.0, 360.0, 360.0, 30.0, 30.0, 30.0, 5.0, 1.0 /
CURVM = 1

C GET VALUATOR INPUT
CALL AGTVL(VLVAL)

C CALCULATE NEW VIEW TRANSFORMATION VALUE
VMFL(1, IDNR, CURVM) = VMFL(1, IDNR, CURVM) > *
VALUE(1, VALUE)
> *
(MAX(IDNR) - MIN(IDNR))

C CHECK VALUATOR VALUE LIMITS
CALL CKVLVL(VKID, IDNR, VLVAL)

C SET VIEW MATRIX
CALL ASVHL(VKID, IDNR, VMFL, CURVM)

RETURN
END
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C LOCAL VARIABLES:
C COMMON INPUTS:
C COMMON OUTPUTS:
C CODED BY: SANJAR JAYARAM
C ADAPTED FROM ACSYN/T/PI CODE WRITTEN BY STEVE WABLER
C DATE: 02/03/88 (21:35:54)

SUBROUTINE STMCS(MKID, VMID)

INTEGER MKID, VMID
INTEGER PNCLIP, PCLIP, PHIGHR, PLOMER
PARAMETER(PNCLIP = 0, PCLIP = 1, PHIGHR = 0, PLOMER = 1)
INTEGER MCLIP, FCLIP, BCLIPI, IPRN, RIPR
PARAMETER(MCLIP = PCLIP, FCLIP = PCLIP)
PARAMETER(BCLIPI = PNCLIP)
INTEGER ACTITM, NITEMS, ACTIVE, ERRIND
PARAMETER(ACTITM = 1, NITEMS = 1)
INTEGER TOPIPR

DATA TOPIPR / 1 /

C SET THE VIEW TO BE ACTIVE
ACTIVE = 1

IF (ACTIVE .EQ. 1) THEN

C SET RELATIVE INPUT PRIORITY VIEW
IPRN = TOPIPR

C SET RELATIVE INPUT PRIORITY
RIPR = PHIGHR

C CHANGE TOP INPUT PRIORITY VIEW
TOPIPR = VMID

ELSE

C SET RELATIVE INPUT PRIORITY VIEW
IPRN = TOPIPR

C SET RELATIVE INPUT PRIORITY
RIPR = PLOMER

ENDIF

C SET VIEW CHARACTERISTICS
CALL PSVMCS(MKID, VMID, MCLIP, FCLIP, BCLIPI, IPRN, RIPR)

RETURN
END

C===============================================================================
C MODULE STVMPS
C===============================================================================
C COMPILER: FORTRAN77 (FORTRAN 77)
C===============================================================================
C CALLING SEQUENCE:
C CALL STVMPS(MKID, VMID)
C===============================================================================
C INPUT PARAMETERS:
C MKID - INTEGER, WORKSTATION IDENTIFIER
C VMID - INTEGER, PHIGS VIEW IDENTIFIER
C===============================================================================
C OUTPUT PARAMETERS:
C NONE
C===============================================================================
C COMMON INPUTS:
C NONE
C===============================================================================
C COMMON OUTPUTS:
C NONE
C===============================================================================
C LOCAL VARIABLES:
C PTITM - INTEGER, VIEW DATABASE ITEM NUMBER FOR PROJECTION TYPE
C ONE - INTEGER, NUMBER OF PROJECTION TYPE ITEMS
C PJTYPE - INTEGER, VIEW PROJECTION TYPE (PARALLEL OR PERSPECTIVE)
C ERRIND - INTEGER, ERROR INDICATOR RETURNED BY DATABASE Routines
C VPITM - INTEGER, VIEW DATABASE ITEM NUMBER FOR VIEWPORT DATA
C NVPITM - INTEGER, NUMBER OF VIEWPORT DATA ITEMS
C PJVP1m() - REAL, PROJECTION VIEWPORT LIMITS
C VMIH - INTEGER, VIEW DATABASE ITEM NUMBER FOR VIEW WINDOW LIMITS
C NVMIH - INTEGER, NUMBER OF VIEW WINDOW LIMIT DATA ITEMS
C VMNLM() - REAL, VIEW WINDOW LIMITS
C PRITM - INTEGER, VIEW DATABASE ITEM NUMBER FOR PROJECTION REFERENCE POINT DATA
C NRPRITM - INTEGER, NUMBER OF PROJECTION REFERENCE POINT DATA ITEMS
C PJRP() - REAL, PROJECTION REFERENCE POINT
C PLITM - INTEGER, VIEW DATABASE ITEM NUMBER FOR VIEW AND
C CLIPPING PLANE DATA

Appendix D. Code Listing - Sample CAD Program 314
C NPLITM - INTEGER, NUMBER OF VIEW AND CLIPPING PLANE DATA ITEMS
C PLANE(1) - REAL, VIEW PLANE AND CLIPPING PLANE DISTANCES
C=============================================================================
C FUNCTIONAL DESCRIPTION:
C SETS GEOMETRY VIEN MAPPING
C=============================================================================
C MODULES CALLED:
C PSVMP3 - PHIES SET VIEW MAPPING
C CRVHFM - CREATES VIEW FRAME STRUCTURES
C=============================================================================
C CODED BY: SANKAR JAYARAM
C ADAPTED FROM ACSYNT/VPI CODE WRITTEN BY STEVE HAMPER
C DATE: 02/01/88 (16:10:02)
C=============================================================================

SUBROUTINE STVMP3(MKID, VMID)
    INTEGER MKID, VMID
    INTEGER PTITM, ONE, PJTYPE, ERRIND
    PARAMETER(PTITM = 5, ONE = 1)
    INTEGER VPITM, NVPITM
    PARAMETER(VPITM = 8, NVPITM = 6)
    REAL PJVPLM(VPITM)
    INTEGER VVITM, NVITM
    PARAMETER(VVITM = 14, NVITM = 4)
    REAL VNML(NVITM)
    INTEGER PRITM, NPRITM
    PARAMETER(PRITM = 16, NPRITM = 3)
    REAL PJRP(NPRITM)
    INTEGER PLITM, NPLITM
    PARAMETER(PLITM = 21, NPLITM = 3)
    REAL PLANE(NPLITM)
C SET THE PROJECTION TYPE
    PJTYPE = 0
C SET THE VIEWPORT
    PJVPLM(1) = 0.0
    PJVPLM(2) = 0.75
    PJVPLM(3) = 0.35
    PJVPLM(4) = 0.95
    PJVPLM(5) = 0.00
    PJVPLM(6) = 1.00
C CREATE VIEW FRAME STRUCTURES
    CALL CRVHFM(MKID, VMID, PJVPLM)
C SET THE MINDOM
    VMNLMLM(1) = -25.0
    VMNLMLM(2) = 25.0
    VMNLMLM(3) = -25.0
    VMNLMLM(4) = 25.0
C SET THE PROJECTION REFERENCE POINT
    PJRP(1) = 0.0
    PJRP(2) = 0.0
    PJRP(3) = 100.0
C SET THE VIEW AND CLIPPING PLANES
    PLANE(1) = 0.0
    PLANE(2) = 70.0
    PLANE(3) = -100.0
C SET VIEW MAPPING
    CALL PSVMP3(MKID, VMID, VMNLMLM, PJTYPE, PJRP(1), PJRP(2), PJRP(3),
                PLANE(1), PLANE(2), PLANE(3))
    RETURN
END

C=============================================================================
C MODULE CRVHFM
C=============================================================================
C COMPILER: FORTVS2 (FORTRAN 77)
C=============================================================================
C CALLING SEQUENCE:
C CALL CRVHFM(MKID, VMID, PJVPLM)
C=============================================================================
C INPUT PARAMETERS:
C VMID - INTEGER, WORKSTATION IDENTIFIER
SUBROUTINE CRVFSS(VWID, VMID, PJVPLM)

INTEGER MKID, VMID, RTITM, NITEMS, ROOT, ERRIND
REAL PJVPLM(*)
PARAMETER(RTITM = 6, NITEMS = 1)
INTEGER SSITM, SUBSTR
PARAMETER(SSITM = 7)
INTEGER VIEMI, PICKID
PARAMETER(VIEMI = 0, PICKID = 5)

ROOT = 25
SUBSTR = 26

C CREATE VIEW FRAME MENU ROOT
CALL CRINRT(ROOT, VIEMI, PICKID, SUBSTR)

C CREATE VIEW FRAME SUB-STRUCTURE
CALL CRVFSS(VMID, SUBSTR, PJVPLM)

RETURN
END

MODULE CRVFSS

CALL CRVFSS(VMID, SUBSTR, PJVPLM(*))
C NAMSET() - INTEGER, LIST OF CLASS NAMES
C FUNCTIONAL DESCRIPTION:
C CREATES A VIEM FRAME SUB-STRUCTURE
C
C MODULES CALLED:
C PEMST - PHIGS EMPTY STRUCTURE
C POPST - PHIGS OPEN STRUCTURE
C PADS - PHIGS ADD NAMES TO SET
C PSHPID - PHIGS SET PICK IDENTIFIER
C DRVMFM - DRAMS VIEM FRAME ELEMENTS
C DRVMPK - DRAMS VIEM PICK ELEMENTS
C PCLST - PHIGS CLOSE STRUCTURE
C
C CODED BY: SANKAR JAYARAM
C DATE ADAPTED FROM ACSYNT/VPI CODE WRITTEN BY STEVE MANNELER
C
SUBROUTINE CRVFSS(VMID, SUBSTR, PJVPLM)
INTEGER VMID, SUBSTR
REAL PJVPLM(*)

INTEGER N
PARAMETER(N = 1)
INTEGER NAMSET(N)

DATA NAMSET / 255 /

CALL PEMST(SUBSTR)

CALL POPST(SUBSTR)

CALL PADS(N, NAMSET)

CALL PSHPID(VMID)

CALL DRVMFM(PJVPLM)

CALL DRVMPK(VMID, PJVPLM)

CALL PCLST

RETURN
END

C============================================================================
C MODULE DRVMFM
C
C CALLING SEQUENCE:
C CALL DRVMFM(PJVPLM)
C
C INPUT PARAMETERS:
C PJVPLM() - REAL; PROJECTION VIEW PORT LIMITS
C
C OUTPUT PARAMETERS:
C NOLINE
C
C COMMON INPUTS:
C NOLINE
C
C COMMON OUTPUTS:
C NOLINE
C
C LOCAL VARIABLES:
C NPNTS - INTEGER, NUMBER OF POLYLINE POINTS
C FRAME - INTEGER, VIEM FRAME COLOR IDENTIFIER
C FRAMEX() - REAL, VIEM FRAME VERTICE X COORDINATES
C FRAMEY() - REAL, VIEM FRAME VERTICE Y COORDINATES
C
C FUNCTIONAL DESCRIPTION:
C DRAMS A VIEM FRAME AROUND THE SPECIFIED VIEW PORT
C
C MODULES CALLED:
SUBROUTINE DRVMPK(PVPLM)

REAL PVPLM(*)

INTEGER NPNTS, FRAME
PARAMETER(NPNTS = 5, FRAME = 15)
REAL FRAMEX(NPNTS), FRAMEY(NPNTS)

C SET POLYLINE COLOR INDEX
CALL PSPLCI(FRAME)

C CALCULATE VIEW FRAME X COORDINATES
FRAMEX(1) = PVPLM(1)*2.0 - 1.0
FRAMEX(2) = PVPLM(2)*2.0 - 1.0
FRAMEX(3) = PVPLM(2)*2.0 - 1.0
FRAMEX(4) = PVPLM(1)*2.0 - 1.0
FRAMEX(5) = PVPLM(1)*2.0 - 1.0

C CALCULATE VIEW FRAME Y COORDINATES
FRAMEY(1) = PVPLM(5)*2.0 - 1.0
FRAMEY(2) = PVPLM(5)*2.0 - 1.0
FRAMEY(3) = PVPLM(6)*2.0 - 1.0
FRAMEY(4) = PVPLM(6)*2.0 - 1.0
FRAMEY(5) = PVPLM(3)*2.0 - 1.0

C DRAW POLYLINE
CALL PPL(NPNTS, FRAMEX, FRAMEY)

RETURN
END

MODULE DRVMPK

CALLING SEQUENCE:
CALL DRVMPK(VRID, PVPLM)

INPUT PARAMETERS:
VRID - INTEGER, VIEW IDENTIFIER
PVPLM() - REAL, PROJECTION VIEW PORT LIMIT

OUTPUT PARAMETERS:
NONE

COMMON INPUTS:
NONE

COMMON OUTPUTS:
NONE

LOCAL VARIABLES:
TTLIM - INTEGER, DATABASE ITEM NUMBER FOR VIEW TITLE TEXT
ERRIND - INTEGER, ERROR INDICATOR RETURNED BY DATABASE ROUTINE
TEXT - CHARACTER, VIEW TITLE TEXT
PX - REAL, VIEW TITLE TEXT X POSITION
PY - REAL, VIEW TITLE TEXT Y POSITION
XSHIFT - REAL, VIEW TITLE TEXT X POSITION SHIFT VALUE
YSHIFT - REAL, VIEW TITLE TEXT Y POSITION SHIFT VALUE

FUNCTIONAL DESCRIPTION:
DRAWS A VIEW PICK ELEMENT

MODULES CALLED:
PHGATB - INSERTS ELEMENT ATTRIBUTES INTO THE OPEN STRUCTURE
PTX - PHIGS TEXT

CODED BY: SANKAR JAYARAM
DATE: 02/18/88 (10:26:12)
REAL PJVPLM(1)
INTEGER TTLITM, ERRIND
PARAMETER(TTLITM = 1)
CHARACTER*20 TEXT
REAL PX, PY, XSHFT, YSHFT
PARAMETER(XSHFT = 0.025, YSHFT = 0.025)

C SET THE VIEW IDENTIFIER TEXT
TEXT = 'GEOMETRY VIEW'

C CALCULATE TEXT POSITION
PX = (PJVPLM(1) + XSHFT)*2.0 - 1.0
PY = (PJVPLM(3) + YSHFT)*2.0 - 1.0

C DRAW TEXT
CALL PTX(PX, PY, TEXT)
RETURN

END

C MODULE STVHFM
C COMPILER: FORTRAN 77
C CALLING SEQUENCE:
C CALL STVHFM(VMID, VNID)
C INPUT PARAMETERS:
C VMID - INTEGER, WORKSTATION IDENTIFIER
C VNID - INTEGER, VIEW IDENTIFIER
C OUTPUT PARAMETERS:
C NONE
C COMMON INPUTS:
C NONE
C COMMON OUTPUTS:
C NONE
C LOCAL VARIABLES:
C VFITM - INTEGER, DATABASE ITEM NUMBER FOR VIEW FRAME ROOT
C mmc - INTEGER, DATABASE ITEM NUMBER TO BE RETRIEVED
C ROOT - INTEGER, VIEW FRAME ROOT STRUCTURE IDENTIFIER
C ERRIND - INTEGER, ERROR INDICATOR RETURNED BY DATABASE ROUTINES
C ACTITM - INTEGER, DATABASE ITEM NUMBER FOR VIEW ACTIVE FLAG
C PRIORT - REAL, STRUCTURE DISPLAY PRIORITY
C ACTIVE - INTEGER, VIEW ACTIVE FLAG (0 = INACTIVE, 1 = ACTIVE)
C COMMON CALLED:
C PPort - PHIGS POST ROOT
C PUpport - PHIGS UNPOST ROOT
C CODED BY: SANKAR JAYARAM
C ADAPTED FROM ACSYM/VPY CODE WRITTEN BY STEVE NAMPLER
C DATE: 02/18/88 (10:33:04)
C
SUBROUTINE STVHFM(VMID, VNID)
INTEGER VMID, VNID

INTEGER VFITM, mmc, ROOT, ERRIND
PARAMETER(VFITM = 6, mmc = 1)
INTEGER ACTITM, active
PARAMETER(PRIORT = 1.0)

ROOT = 25

ACTIVE = 1
IF (ACTIVE .EQ. 1) THEN
   POST ROOT
   CALL PPort(VMID, ROOT, PRIORT)
ELSE
   UNPOST ROOT
   CALL PUpport(VMID, ROOT)
END
EPIIIIF

RETURN
END

C**************************************************************
C MODULE CRMNRT
C**************************************************************
C COMPILER: FORTVS2 (FORTRAN 77)
C**************************************************************
C CALLING SEQUENCE:
C CALL CRMNRT(ROOT, VIENID, PICKID, SUBSTR)
C**************************************************************
C INPUT PARAMETERS:
C ROOT - INTEGER, MENU ROOT STRUCTURE IDENTIFIER
C VIEMID - INTEGER, MENU VIEW IDENTIFIER
C PICKID - INTEGER, MENU TYPE PICK IDENTIFIER
C SUBSTR - INTEGER, MENU SUB-STRUCTURE IDENTIFIER
C**************************************************************
C OUTPUT PARAMETERS:
C NONE
C**************************************************************
C COMMON INPUTS:
C NONE
C**************************************************************
C COMMON OUTPUTS:
C NONE
C**************************************************************
C LOCAL VARIABLES:
C NONE
C**************************************************************
C FUNCTIONAL DESCRIPTION:
C CREATES A MENU ROOT STRUCTURE WHICH EXECUTES A SUB-STRUCTURE
C CONTAINING THE ACTUAL MENU ITEMS.
C**************************************************************
C MODULES CALLED:
C PEMST - PHIGS EMPTY STRUCTURE
C POPST - PHIGS OPEN STRUCTURE
C PSVMI - PHIGS SET VIEW IDENTIFIER
C PSHPID - PHIGS SET PICK ID
C PE)GT - PHIGS EXECUTE STRUCTURE
C PCLST - PHIGS CLOSE STRUCTURE
C**************************************************************
C CODED BY: SANKAR JAYARAM
C ADAPTED FROM ACSYNT/VPI CODE WRITTEN BY STEVE HAMPLER
C DATE: 02/18/88

SUBROUTINE CRMNRT(ROOT, VIEMID, PICKID, SUBSTR)
INTEGER ROOT, VIEMID, PICKID, SUBSTR
C EMPTY ROOT STRUCTURE
CALL PEMST(ROOT)
C OPEN ROOT STRUCTURE
CALL POPST(ROOT)
C SET VIEW INDEX
CALL PSVMI(VIEMID)
C SET PICK IDENTIFIER
CALL PSHPID(PICKID)
C EXECUTE SUB-STRUCTURE
CALL PE)GT(SUBSTR)
C CLOSE ROOT
CALL PCLST
RETURN
END

C**************************************************************
C MODULE GTXFMT
C**************************************************************
C COMPILER: FORTVS2 (FORTRAN 77)
C**************************************************************
C CALLING SEQUENCE:
C CALL GTXFMT(VALUE, MAT)
C**************************************************************
C INPUT PARAMETERS:
C VALUE(N) - REAL, TRANSFORMATION VALUES AS FOLLOWS:
C VALUE(1) = X AXIS ROTATION IN DEGREES
C VALUE(2) = Y AXIS ROTATION IN DEGREES
C VALUE(3) = Z AXIS ROTATION IN DEGREES
C**************************************************************
C VALUE(4) = TRANSLATION ALONG X AXIS
C VALUE(5) = TRANSLATION ALONG Y AXIS
C VALUE(6) = TRANSLATION ALONG Z AXIS
C VALUE(7) = SCALE FACTOR
C
C OUTPUT PARAMETERS:
C MAT(4,4) - REAL, TRANSFORMATION MATRIX
C
C COMMON INPUTS:
C NONE
C
C COMMON OUTPUTS:
C NONE
C
C LOCAL VARIABLES:
C GAMMA - REAL, X AXIS ROTATION ANGLE IN RADIANS
C BETA - REAL, Y AXIS ROTATION ANGLE IN RADIANS
C ALPHA - REAL, Z AXIS ROTATION ANGLE IN RADIANS
C CA - REAL, COSINE OF ALPHA
C CB - REAL, COSINE OF BETA
C CS - REAL, COSINE OF GAMMA
C SA - REAL, SINE OF ALPHA
C SB - REAL, SINE OF BETA
C SS - REAL, SINE OF GAMMA
C
C FUNCTIONAL DESCRIPTION:
C SUBROUTINE GTXFMT(VALUE, MAT) C CALCULATES A 3D TRANSFORMATION MATRIX
C C CODED BY: SANKAR JAYARAM C DATE: 09/22/87
C
C REAL VALUE(4), MAT(4,4)
C REAL PI, DG2RD
C PARAMETER(PI = 3.14159, DG2RD = PI / 180.0)
C REAL GAMMA, BETA, ALPHA
C REAL CA, CB, CG, SA, SB, SG

C CONVERT DEGREES TO RADIANS
C GAMMA = VALUE(1) * DG2RD
C BETA = VALUE(2) * DG2RD
C ALPHA = VALUE(3) * DG2RD

C CALCULATE SINES AND COSINES
C CA = COS(ALPHA)
C CB = COS(BETA)
C CG = COS(GAMMA)
C SA = SIN(ALPHA)
C SB = SIN(BETA)
C SG = SIN(GAMMA)

C CALCULATE XFORM MATRIX
C MAT(1,1) = (CA * CB) * VALUE(7)
C MAT(1,2) = (CA * SB * SG - SA * CG) * VALUE(7)
C MAT(1,3) = (CA * SB * CG + SA * SG) * VALUE(7)
C MAT(1,4) = VALUE(4) * VALUE(7)
C MAT(2,1) = (SA * CB) * VALUE(7)
C MAT(2,2) = (SA * SB * SG + CA * CG) * VALUE(7)
C MAT(2,3) = (SA * SB * CG - CA * SG) * VALUE(7)
C MAT(2,4) = VALUE(5) * VALUE(7)
C MAT(3,1) = (-SB) * VALUE(7)
C MAT(3,2) = (CB * SG) * VALUE(7)
C MAT(3,3) = (CB * CG) * VALUE(7)
C MAT(3,4) = VALUE(6) * VALUE(7)
C MAT(4,1) = 0.0
C MAT(4,2) = 0.0
C MAT(4,3) = 0.0
C MAT(4,4) = 1.0

RETURN
END
SUBROUTINE CKVLVL(MID, VLDNR, VLVAL)
INTEGER MID, VLDNR
REAL VLVAL

COMMON /VALUAT/ VALUE
REAL VALUE(8)

IF (VLVAL .EQ. 0.0 .OR. VLVAL .EQ. 1.0) THEN

INITIALIZE VALUATOR
CALL AINVL(MID, VLDNR)
ELSE

STORE CURRENT VALUATOR VALUE IN COMMON BLOCK
VALUE(VLDNR) = VLVAL
ENDIF

RETURN
END

MODULE AGTVL

COMPILE: FORVSZ (FORTRAN 77)

CALLING SEQUENCE:
CALL AGTVL(VLVAL)

INPUT PARAMETERS:
NONE

OUTPUT PARAMETERS:
VLVAL - REAL, VALUATOR VALUE

COMMON INPUTS:
NONE

COMMON OUTPUTS:
NONE

LOCAL VARIABLES:
IMID - REAL, INPUT WORKSTATION IDENTIFIER (RETURNED BY PHAIFT)
IDNR - REAL, INPUT DEVICES NUMBER (RETURNED BY PHAIFT)

FUNCTIONAL DESCRIPTION:
RETURNS LAST VALUATOR VALUE FROM SEQUENCES OF VALUATOR EVENTS

MODULES CALLED:
PGLVL - PHIGS GET VALUATOR
PHAIFT - PHIGS WAIT EVENT

CODED BY: SANKAR JAYARAM
SUBROUTINE AGTVL(VLVAL)

REAL VLVAL

REAL TOUT
PARAMETER(TOUT = 0.2)
INTEGER IMKID, ICL, IDNR

INTEGER PVALUA
PARAMETER(PVALUA = 3)

C SET VALUATOR EVENT
100 CALL PGTVL(VLVAL)

C WAIT EVENT TO SEE IF ANY MORE EVENTS A QUEUED
CALL PHAIT(T(1|T• I|I(ID) ID))

IF (ICL .EQ. PVALUA) THEN
C LOOP BACK TO SEE IF THERE ARE MORE EVENTS
GOTO 100
ENDIF
RETURN
END
The vita has been removed from the scanned document